SHORT-TERM PRODUCTION SCHEDULING AND EQUIPMENT DISPATCHING FOR UNDERGROUND METAL MINES

CHABVA M. TSOMONDO
BSc. Hons (UZ), M.Eng. (McGill)

Department of Mining and Metallurgical Engineering
McGill University, Montreal
Quebec, Canada

A Thesis Submitted to the Faculty of Graduate Studies and Research
in Partial Fulfilment for the Degree of Doctor of Philosophy

January 1996

Abstract

An integrated approach for short-term production planning and equipment dispatching for underground metal mines is proposed in this thesis. The salient factors influencing a shift production schedule are controlled by both management and the mine environment. The mine system constraints reflect the inter-play of geological, geomechanical and economic factors. The management goals are considered with respect to the operating policies of draw-point and ore-bin control, and ground control, equipment allocation and ventilation systems. The integration of these various issues constitutes the mining system, which requires flexibility to achieve the production goals under changing mining environments.

The geological and geomechanical factors are known only as estimates, often only fully determined at the functional stage of planning. The mining process causes complex dynamic changes in these factors with regard to their interaction. Each factor has a vaguely defined effect on the mined ore grades because the planning is performed under conditions of limited information. Therefore, a fuzzy logic modelling approach is applied in this thesis to integrate the various dynamic variables in the assessment of the uncertainty in the mined ore grades. A fuzzy logic analysis of typical stope descriptions indicates the successful implementation of the method in giving specific and unambiguous solutions of ranking ore sources and the reliability in the mined ore grades.

A goal programming technique is implemented to determine work shift production schedules based on multiple management goals. The stope rankings are achieved through fuzzy logic modelling incorporated into the goal programming optimization. This technique provides a fast and efficient method of producing work shift schedules. Two functions for determining the effectiveness of a goal programming schedule for a multi-objective problem have been developed for specific shift budgets and production targets.

A dispatch model consisting of an admission control and six routing policies is developed and tested through simulations for underground trackless mining systems to determine their impact on product quality, cost and productivity. The effects of fleet size and number of fixed facilities within a system is investigated. The three policies designed to maximize the mined ore quality successfully model the product quality goal. The remainder of the policies meet their goal of maximizing productivity during a production shift. An interactive program was developed to allow dynamic changes of the policies and goal priorities to minimize deviations from schedule targets. The results of the simulation studies lead to a proposal for a re-engineering of the design of underground trackless mining systems and a requirement for the integration of information systems to maximize the benefits of mine automation and control.

Résumé

Cette thèse propose une méthode d'intégrer la planification à court terme avec l'affectation des équipements miniers souterrains. Les principaux facteurs qui influencent l'échéancier de production du quart de travail sont contrôlés à la fois par la gestion et l'environnement minier. Les contraintes du système minier réflètent l'interaction des facteurs géologiques, géomécaniques et d'évaluation du gisement. Les objectifs de la gestion sont considérés par rapport aux politiques d'opération des points de soutirage et le contrôle des réserves de minerai foudroyé, de la distribution des équipements et du système de ventilation. L'intégration de ces facteurs affecte la flexibilité du système minier qui est requise pour atteindre les objectifs de production, pour des conditions de minage changeantes.

Les variables géologiques et géomécaniques sont des estimés qui sont souvent connus uniquement durant la planification de la production. Le minage cause des changements dynamiques complexes de l'interaction de ces facteurs. Chacun de ces facteurs a un effet plus ou moins bien connu parce que la planification se fait à l'aide d'informations limitées. Une technique de modélisation logique floue (fuzzy logic) est utilisée pour intégrer les diverses variables dynamiques qui influence l'incertitude des teneurs minées. L'analyse par logique floue pour des descripteurs de chantier prouve que la technique réussit à hiérarchiser les sources de minerai en fonction de la fiabilité des teneurs effectivement minées.

Une technique de programmation par buts (goal programming) est développée pour déterminer les plans de production des quarts de travail qui sont basés sur plusieurs objectifs visés par la gestion. L'hiérarchie des chantiers est obtenue par la modélisation logique floue qui est incorporée dans le processus d'optimisation par programmation par buts. Cette technique est rapide et efficace pour produire des plans de production des quarts de travail. Deux fonctions furent développées pour mesurer l'efficacité du plan obtenu à l'aide de la programmation par buts pour un problème ayant plusieurs objectifs avec des budgets fixes et diverses cibles de production.

Un modèle d'affectation qui consiste d'un contrôle d'accès et six procédures d'affectation fut développé et simulé pour des systèmes de manutention souterrains sans rails. Les effets de la taille de la flotte et du nombre d'installations fixes sont analysés. Les résultats démontrent que trois procédures réussissent à contrôler la qualité du produit tandis que les autres procédures maximisent la productivité. Un logiciel interactif fut développé et permet des changements dynamiques des procédures et de la priorité des objectifs afin de minimiser les écarts avec les objectifs planifiées. La thèse est conclue en proposant la ré-ingénierie de la planification des exploitations minières souterraines et en indiquant les conditions requises pour intégrer les systèmes d'information visant à maximiser les bénéfices du contrôle et de l'automatisation minière.

Acknowledgements

I wish to express my gratitude to my supervisors, Professor F. Hassani and Professor Y. Lizotte for having taken keen interest in my work and providing me with invaluable guidance and positive criticisms. Besides my supervisors, I wish to thank all the members of staff in the Department of Mining and Metallurgical Engineering at McGill University who made me welcome and always had their doors open for discussions and moral support.

I thank Inco Ltd. (Sudbury Division) for allowing me to visit their mining operations during my field work. I am indebted to Mr. E. Hinton for his assistance in a data survey conducted for this thesis.

My deep appreciation goes to the Zimbabwe-Canada Commonwealth Scholarship Fund, without which I could not have undertaken this work.

The fulfilment of this work has been made possible by the great support I got from my family, in particular, Mr. and Mrs. Muzvanya. Last but not least, I thank my dear wife Inzanisai, for solely persuading me into undertaking this work. With a heavy heart, I posthumously applaud her for the kindness, love and strength she gave me, and I dedicate this work to her soul.

Statement of Originality

This research identifies the complexities of short term production planning in underground metal mining. Based on these complexities, a basic plan evaluation guideline of the important variables or parameters for short term underground planning decisions has been created. The guideline assures a systematic and fundamental parameter evaluation at the production planning stage. The guideline concepts are also applicable to open-pit mine production planning.

A fuzzy logic model is developed to integrate the fuzzy information bases used in short term planning to generate unique expectancies of ore quality as a dynamic function of progressive mining. The problem of ore dilution estimation and/or prediction is therefore approached from a novel idea of fuzzy parameters such as drill-hole information and rock mass strengths. This approach is a departure from the traditional single attribute analysis, where one variable is investigated with all other variables fixed. While the problem of ore dilution is not eliminated, the approach adopted here increases the quality of decision-making through full assessment of impacting parameters. This enables responsive actions to be taken to minimize the effects of stope dilution on the ore streams reaching the process plant. The developed system also has a positive contribution to offer to operating mines in ensuring consistency in decision-making and reduced training costs of planning recruits.

While other researchers have applied goal programming to coal mining for medium term planning [Barbaro and Mutmansky, 1983, Jawed, 1993 and Youdi et al, 1992], this work introduces the mathematical technique to solve multiple, and sometimes conflicting, objectives at the work shift production scheduling level in underground metal mining. For the first time, the study identifies the multi-objective function relationships with production level and budget sizes for given mine layouts. These relationships are identified as measures of effectiveness of decision-making under the mine layout's constraints and restrictions for the scheduled shift. The trade-offs between the different

objectives are then easily assessed which is crucial for the application of the technique to real-time production control systems.

A new multi-dispatch policies model has been developed and validated for the underground mining environment that successfully implements a schedule in a dynamic fashion. It achieves the shift goals of minimizing both the unit production cost and product quality deviations, while maximizing both productivity and fleet utilization. With the advent of underground communication systems, the model can be adapted for real-time production equipment dispatching on a multi-level, multi-work section using different equipment types. The dispatch model is also applicable to medium and long-term planning decisions for mine layout design and equipment selection, by assessing through stochastic simulations the impact of changes in policies or responses associated with the decision-making.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

DM = decision maker
DP = dynamic programming
GP = goal programming (model)
LP = linear programming
IP = integer programming
OR = operations research

FMS = flexible manufacturing systems
FLSM = fuzzy logic stope model
UADM = underground active dispatch model
LHD(s) = load-haul-dump equipment
JIT = just-in-time systems
AR = artificial reasoning
CAD = computer aided design
CQN = closed loop queuing model
COG or COA = centre of gravity or area method
MOM = mean of maxima method
MISO = multiple input single output model

CRatio = maximum critical ratio of unfinished work selected first.
EEST = prioritized earliest expected service time destination first
MaxQ = maximum quality and/or maximum contained product first
MaxQ/MinS = combined policy of MaxQ and MinS selected first
MinS = minimum slack time between an LHD arrival and it being serviced first
STT = prioritized shortest expected travel time to a destination first

Q = quality control specification limit
T = production specification limit
U = utilization specification limit

μ = sample mean
σ_μ= variance of the sample mean (confidence measure)
e_t= error from target value in time interval t
λ = mean rate of equipment failure per unit time
λ_r = mean rate of equipment repair per unit time
α = Weibull distribution scale parameter
β = Weibull distribution shape parameter
γ = Weibull distribution location parameter

Chapter 1
Research Objectives and Scope

1.1 Introduction

A mining activity constitutes a production system marked by the hierarchial levels of decision making, implementation and control. A production system is defined here as a conversion process of resources into the extraction of a sought mineral. This definition implies three building blocks of a production system, namely:

1. the technology utilized to convert inputs to products.
2. the organization of the production system, i.e. the allocation criterion of resources to meet goals and,
3 the management techniques used to control the system.

There are basically three levels of planning, each distinct in its frequency, time consumption and impact on the project outcome. Decision making in mining as well as in other industries can be categorized into long, medium and short term planning stages. The long term planning takes considerable time, effort and money. It is concerned usually with the identification of star projects and giving the green light to develop them. At a shorter time span, the plans constitute annual production plans. The solution space is normally small and decision options are few. The impact of the long term planning decisions on projects tend to be large because they normally determine the project technology. Equipment selection, mining method selection and major capital expenditures fall within this category. The medium term planning spans a couple of years to as short as monthly production plans. Their purpose is to focus on the near future needs and how those needs would affect the evolution of the long term plans. The number of decisions is increased, and their impact is not as significant. Both long and medium term planning fall in the technology and organisational blocks of the production system, but they lack the information base to perform the system control.

At the bottom of this three tier sequence is operational planning and implementation. This is the stage of actualization of long and medium term plans. It is characterized by repetitious decision making. Both the decision variables and the solution space are large, making it cumbersome to enumerate and identify the optimal set. In addition, operational decisions frequently tend to be fairly complex and unique for a point in time, thus indicating the futility of the process. While the impact of individual decisions on a project may be minimal, their accumulated effect can be significant. Therefore, persistent poor decision-making at the operational phase leads to cost escalation. In addition to decision making, this stage is concerned with implementation, monitoring and reconciliation. The short time spans of these plans makes it crucial to be able to quickly observe target deviations in time and allow feed-back instructions to nullify such problems. The hierarchy of decision making is illustrated in Figure 1.1.

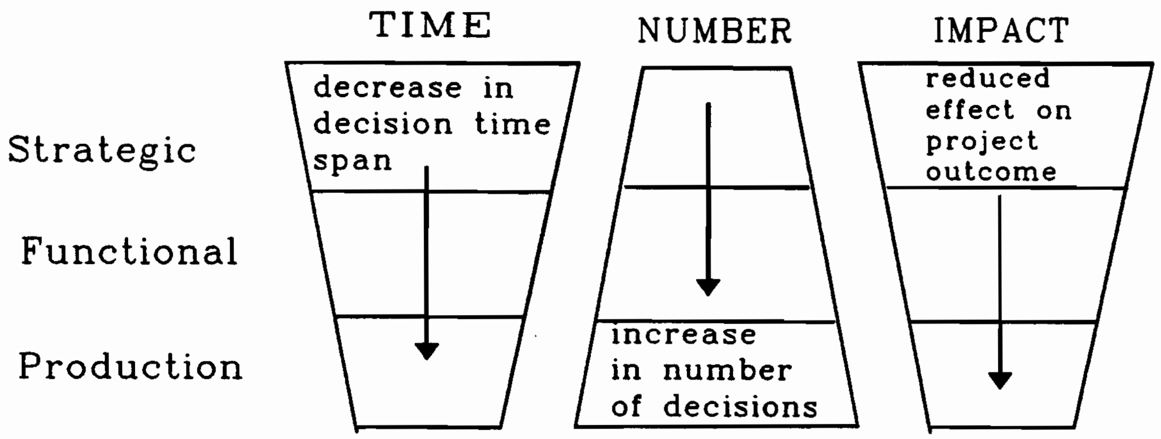

TIME
NUMBER
IMPACT
Strategic
Functional
Production
decrease in decision time span
increase in number of decisions
reduced effect on project outcome

Figure 1.1 Hierarchy and impacts of mine planning decisions

Constant production and/or demand rates rarely exist in real situations. The chances of such rates varying is a function of the planning period. A short planning review period is less likely to have major deviations from pre-set targets occurring. This statement highlights the weakness of medium to long term planning criteria where whole operations are considered in an aggregate form. The information is based on global averages and estimates, as is typical of feasibility reports. In real situations, however, the forecasts (i.e. long term plans) are only point estimates of an uncertain state. If they were to be used to direct short term production over the entire project life, it is unlikely that the plans would be appropriate after the first few production shifts. The whole plan would likely become unworkable. For this reason, it is suggested that the implementation phase be treated as a dynamic process rather than a static one. The long term plans establish the production and resources required through time. The short term plans establish the feasible production levels based on the current information. The dynamic approach enables the production process to continue with modifications.

1.2 Problem Definition

In an overview, Kim [1979] concludes that though mine production scheduling comes at the bottom of mine planning hierarchy, it is the most difficult and demanding task to achieve. The production schedule has to conform with the medium to long term plans as well as with the practical aspects of day-to-day operations.

In this study, the problems of underground mine production planning are identified as follows:

1. A production shift is concerned with the allocation of resources to maximize production (grade and tonnage), and minimize costs. In addition, some special site objectives may be set, such as adhering to some draw schedule to minimize ground control problems. The choice of sites and the allocation of equipment and labour is under the control of the decision maker and can be modified by changing policies or strategies. The ore quality or some deleterious material

concentration are estimated from some form of sampling and, at scheduling, their values are expected to be within reasonable limits about the forecast values.

However, it is known that grade streams can be highly erratic depending on the geology, structure and sampling intensity. Equipment is usually run on preventive maintenance but there still exist unscheduled, random breakdowns. Draw-point and ore-pass blockages may also be random and little information is available to indicate the possibility of such occurrences. All these variables are uncontrollable by the decision maker. They appear in the scheduling process only as mean values or most likely values of their respective distributions.

In the process of scheduling resources, the system is assumed to be static for a certain planning period, namely the shift length. This description of a production schedule implies that an optimization mode could be applied to the decision process. Several objectives are usually simultaneously sought; therefore the solution requires a multi-objective optimization approach. The solution is subject to uncertainty because of uncontrollable variables.

2. The implementation of a shift schedule is a dynamic process affected by variable situations such as equipment breakdowns and mining rates differing from targets. Traffic congestion develops due to irregularities in the system such as unexpected length of time required to service individual load-haul-dump vehicles or trucks, or their irregular arrival at draw-points. The solution of such problems considers future events and requires forecasting. These decision variables can be dependent or independent of some events pertaining to the mining problem. For example, the loss of a scheduled site through blockage would affect the resources allocated to that site.

3. The implementation of a static plan under dynamic conditions leads to:

   (i) inherent process errors typical of the static model, and

(ii) errors due to changes in the system inputs at implementation. Therefore, for a mine to implement, for example, product quality control, it has to be able to identify the true cause of a process error during the time frame of a production shift. The industry currently lacks this ability.

4. The stochasticity in the implementation phase requires that production information be continually recorded and used to evaluate the optimization schedule. This information indicates what material has been mined, from which sites, of what quality and most importantly, whether the schedule targets can be attained under current system conditions. Therefore, the underground production phase has to include an inventory control routine. Currently, some mines record production statistics but no analysis is performed until the end of the shift. In this mode, these mines are conducting shift reconciliation and not shift control as the functional benefits of the information is lost.

5. The underground mine environment may be constrained by access between work areas, ground conditions, auxiliary functions such as support and ventilation systems, and the number of available working areas at any one time. The lack of flexibility in the system prevents higher resource utilization and improved process control. The mine design therefore, may either improve or constrain the shift production schedule.

6. Global competition exists among the mines in developed nations (e.g. Canada and South Africa) that are usually deep and mature, and the relatively inexpensive open-pit mines (e.g. Chile, Papua New Guinea, British Columbia) and the usually richer new deposits in developing nations. This depresses the commodity prices and may make the deep underground mines marginal. An operating mine's profitability may be affected by these external factors, causing large deviations from the project feasibility studies' forecasts. It is therefore essential to minimize strategic planning errors. Recently, the international standards of quality control

3. In the last decade, several open-pit mines have adopted new technologies of equipment dispatch in conjunction with geographic positioning systems resulting in leaner fleets, and higher flexibility. Similar technologies for underground mines are just now breaking ground with the first prototype fixed dispatch system reported at the Finsch underground mine in South Africa [Luke, 1993]. Therefore, there is still much work that needs to be done to develop appropriate systems as the underground environment is much more constrained than in open-pits.

Since the industry is at a cross-roads, from the traditional rule of thumb planning schemes to the modern computer assisted mode, this work recognizes the shortcoming of "black-box" models towards management acceptance. Consequently, the methods developed here are tools to facilitate decision making without replacing the human beings. This allows the decision maker to up-date production plans as seen fit to reflect the dynamic nature of the operational environment.

4. The production planning stage is identified as critical to any concept of quality control in a mining environment. It is noted that while the impact of individual production decisions is generally low, their large number may have a major effect on the project success. Therefore, this study proposes a methodology to enhance the quality of these decisions using a multi-objective based approach. Such an approach captures the circumstances of practical application, more than the traditional use of single objective criterion.

   Multiple goals generally occur at the production phase and they are sometimes in conflict. As a result, some difficulties exist in attempts to reconcile these multi-objectives problems into single objective problems. This work proposes a scheme of that keep the goals separate while searching for a solution that is optimal or near-optimal. It is argued that such a solution is superior to one given by a single objective mathematical optimization approach.

5. Whilst other industries such as manufacturing and construction have benefited from the advent of computer integrated planning and processing, it is still at the infancy stage in mining. Computers have been used more in functional planning such as mine design, economic models and off-line data storage and manipulation. As a cardinal part of this project, an underground materials handling dispatch system is developed to implement the production schedules. An active dispatch model that maximizes system flexibility under the constrained underground environment is required. The model has to allow automatic re-distribution of resources, e.g. equipment in the event of breakdown, as a strategy of minimizing the deviation from the shift schedule.

6. The problem of process control during the production shift has to be resolved through real-time data acquisition, interpretation and feedback to the respective resources. The interpretation has to be based on site specific norms contained in some rule-based algorithm.

The process control requirement of the underground system is effected through a rule-based control procedure. Production statistics are accumulated in real time on a wide variety of variables such as machinery status, tonnages, grades and material source depletions. At defined time intervals, the operation supervisor reviews the work progress within the whole mine. The process controller determines the performance measures at that time and suggests to the supervisor whether the process is under control or not. The supervisor then decides to accept the current status, change the operating rules or terminate the process for a re-schedule. This activity constitutes process control as feedback information and is communicated either as a reward or a mitigation. A reward is communicated when the process is allowed to continue because it is under control, and a mitigation occurs when decisions are made to improve the system performance so as to return control.

The process monitors the material flow from the source through the intermediary sub-system. The objective is to keep tally of the various conveyed material and identify where in the system it is. This leads to better control of the product quality as schedules can be drawn for ores located not only in the primary stopes but also in secondary holding facilities. The process control objective relates to the recent concept of 'lean' mining where a mine capitalization, material re-handling, throughput time and stockpiles are minimized through an efficient use of an integrated mining information system [Knights and Scoble, 1995; Scoble, 1995]. A 'just-in-time' mining has been reported at the Garson Mine where the throughput time is minimized by use of a quick setting paste fill that allows immediate ore extraction in adjacent stopes. The method also allows the control of wall damage, fragmentation and dilution through the use of smaller diameter holes [Whiteway, 1993].

1.4 Research Scope

The scope of this research is to provide a rationale for production planning and scheduling in underground metal mines. It recognizes that the production objectives are numerous and varied and, due to system dynamics, are only transitory. A strategy is developed to assist in the determination of decision variable coefficients since these values are usually not known precisely. The decision variables have inter-dependence, for example, it is not unusual that a high grade stope may have either a large or a small ore tonnage such that the objective of a specified grade is in conflict with the tonnage objective from the same area.

Underground production shift simulations are carried out for a typical mine layout with connected level work sections and internal ramp system that permits maximum equipment routing flexibility. Incidentally, as few mines currently have this design flexibility, it is studied here to illustrate the potential benefits of re-engineering underground operations. At present, no mines are fitted with the communication system that is necessary for the implementation of this model. Therefore, the model performance cannot be fully field tested with a real world example. Instead, the model is validated by comparison to an analytical solution based on an identical mine layout using an identical haulage fleet.

In traditional process control systems, planned performance measures are set and the actual performance is compared to the former. When the two are incongruent, a control input is introduced in the actual system to promote it towards the planned targets. A similar approach has problems when applied to mine production due to the uncertainty in some of the production schedule variables. Typically, the planned grade variable is uncertain and is influenced by such factors as dilution, local geology and sample intensity. If a systematic trend in the actual grade variation with respect to the planned estimate exists, then a feedback signal is necessary to determine the factor(s) responsible for that trend. This approach basically accepts the output as a fact and queries the input such that a re-schedule would have to update its confidence of the initial grade estimates for a certain stope.

Through grab sampling of the work areas, a reconciliation of the estimated and the mined grades is achieved with present grade control techniques. However, this process is slow and is an off-line control which fails to control grade fluctuations within a shift period. Since one of the study objectives is effective control throughout the shift period, a 'futuristic' real-time quality scanner at the material source is proposed. Such an instrument can be likened to a Geiger counter that can measure the radioactivity of a load, or an X-ray fluorescence device which does an elemental spectral analysis of the load. The information is then relayed in real time to a central computer which collects production statistics. This enables the mined grades to be continuously determined and recorded. The mined grades are then compared to the initial schedule targets.

The real-time quality scanner is simulated through random sampling of either a sine or cosine waveform and then adding the result to the scheduled target grades as follows:

$$g_{actual} = g_{target} + \sin(\lambda T) \tag{1.1a}$$

or

$$g_{actual} = g_{target} + \cos(\lambda T) \tag{1.1b}$$

where g_{target} = target grade expected at scheduling

g_{actual} = actual grade obtained at extraction

T = cumulative tonnage from a particular draw-point

λ = amplitude of the waveforms (measure of variability about the targets).

The simulated grade variations are necessary to determine the effects of the dispatch rules on quality, i.e. it may not be necessary to re-schedule operations if some dispatch rule can be used to discriminate on the basis of observed stope qualities.

1.5 Research Methodology

The research methodology consists of three aspects of production planning, implementation and control. The relationship between these aspects is illustrated in Figure 1.2. Crisp data describes those variables that can be uniquely defined by a binary system, for example, TRUE or FALSE; or by cardinal numbers as in *ten* stopes. The fuzzy data represents gradational information that are not binary as in SLIGHTLY TRUE or VERY FALSE. Also, ordinal numbers describe fuzzy data as in *close to ten* stopes.

1.5.1 Scheduling Aspects

The research undertaken is aimed at operational planning which is the day-to-day planning and allocation of resources. A technique for solving multi-criteria problems is proposed, namely goal programming (GP). The need to satisfy multiple goals simultaneously generally over-rides the single objective optimization techniques such as linear, dynamic and mixed integer programming or simulation. Such objectives as

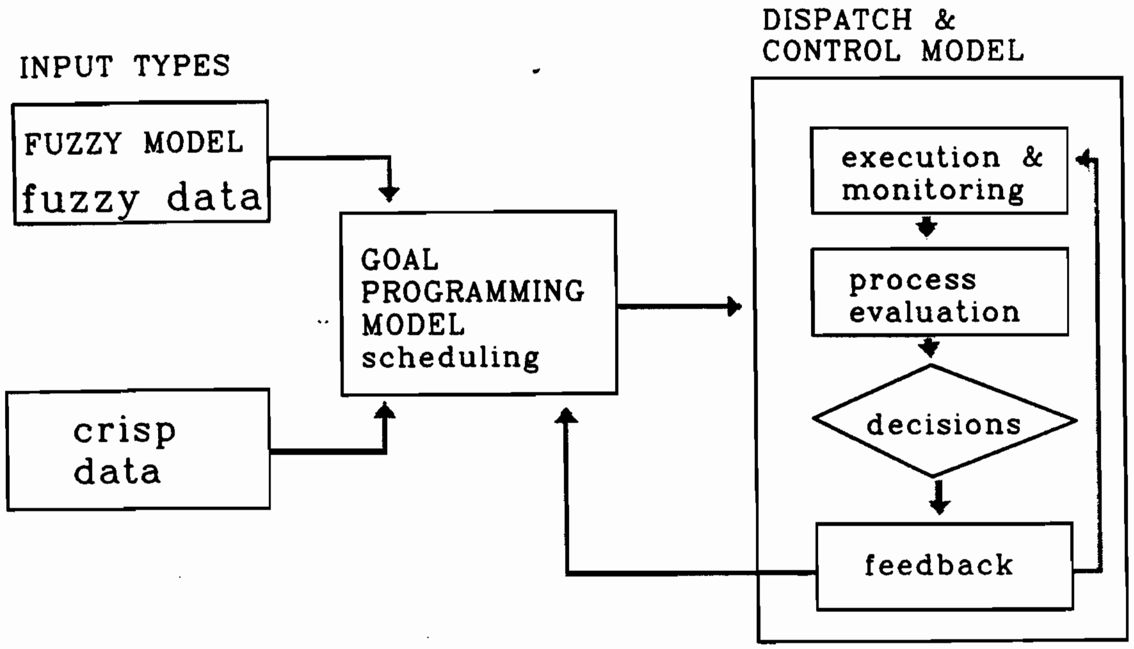


Figure 1.2 Research methodology on production and control

minimization of costs, least equipment movements, high productivity at certain ore qualities are common features of the realities of underground metal mining. These goals may be complementary but often they can also be conflicting.

Basic multiple objective solutions are 'satisfying' or 'best' compromises rather than mathematically optimal. The later is usually typical of single objective problems which generally ignore the interdependence of the goals. A hierarchy of priorities on the objectives and differential weights on the goals with the same priority are used to identify the preference of the decision maker. Goal programming satisfies the objectives in the priority order specified in the objective function. When it becomes impossible to improve the solution any further, the goal programming model terminates.

The use of linear programming is well accepted and several commercial programs are available, for example, in Lotus 123©, Excel© and Quatrro Pro© spreadsheets. The method proposed here requires a prior knowledge of targets to be met. These targets can be established by solving linear programs (LP), noting that although optimal, the LP solutions have single objectives. The goal program then determines the extent to which each goal is met when all goals are combined into the same model for simultaneous solution. This ability to use the information from LP in the GP enhances the flexibility and adaptability of the method in solving real world problems.

Production scheduling is dynamic, consisting of an inter-play of several parameters, most of which exist over short time periods. Therefore, there exists a need for providing unique solutions in the shortest possible time if decisions are to be made in real time. The proposed method succeeds in this aspect.

A generalized goal programming model is stated as:

$$\textit{minimize}: \sum_{i=1}^{m} |F_i(x) - T_i| \qquad (1.2)$$

subject to:

$$x \in X \qquad (1.3)$$

where T_i = target or goal set by the decision maker for the objective function, $F_i(\mathbf{x})$ and

i = ith objective.

X represents the feasible set from which to choose the decision variables, **x**.

The objective function expressed by equation 1.2 is a minimization problem. The absolute value of the difference between the target or goal and its actual achieved value is determined. This procedure is repeated for each of the m objectives. The sum of these m absolute values is a feasible solution. The goal programming procedure aim is to effect choices of the vector $\mathbf{x} \in \mathbf{X}$ to minimize the sum of the absolute deviational slack variables.

The objective function in equation 1.2 is non-linear but it can be transformed into a linear function by introducing slack variables, an over-achievement, δ^+, and under-achievement, δ^-, with respect to the objective targets as follows [Charnes and Cooper, 1961]:

$$\delta_i^+ = 0.5(|F_i(x) - T_i| + (F_i(x) - T_i)) \tag{1.4}$$

$$\delta_i^- = 0.5(|F_i(x) - T_i| - (F_i(x) - T_i)) \tag{1.5}$$

Summing equations 1.4 and 1.5 yields:

$$\delta_i^+ + \delta_i^- = |F_i(x) - T_i| \tag{1.6}$$

which upon substitution in equation 1.2 gives:

$$\min \sum_{i=1}^{m} (\delta_i^+ + \delta_i^-) \tag{1.7}$$

subject to $\delta^+, \delta^- \geq 0$.

Subtracting equation 1.5 from 1.4 yields the goal constraint resulting from the transformation of equation 1.2, i.e.

$$F_i(x) + \delta_i^- - \delta_i^+ = T_i \tag{1.8}$$

The slack variables, δ^+ and δ^- are either both zero, or one has a positive value with the other zero. Hence the product of slack variables is zero and this property is used by the Simplex algorithm to drive the slack variables to zero, which in the process minimizes the deviations of the objectives from the set targets.

The generalized goal programming model can be modified by introducing scalar weights to reflect the decision maker's (DM) preference of goal satisfaction. A large weight implies that the DM requires that goal to be achieved strictly before the lower weighted ones. Similarly, the goals can be ranked in the order of importance to the DM. Each deviational slack variable in the objective function is then assigned a non-scalar priority factor which indicates only the sequence/order of goal satisfaction. These steps indicate the GP flexibility and its ability to mimic the DM's needs in a real world problem where goals are not equal both in priority nor weighting.

One problem that appears in the literature is that some mathematically unique solutions determined by LP and dynamic programming are not readily feasible to implement. As timely decision making is essential to this production problem, it is unacceptable to generate solutions that are not implementable. Similarly, single valued optimization is restrictive as it often yields an infeasible solution if any constraint is violated. Given the many variables and constraints faced by the decision maker and the short time available to him/her to evaluate and produce a shift schedule, it can intuitively be realized that several solutions based on single objective optimization will be infeasible because of errors in the model formulation. This would dictate a re-modelling of the problem.

Flexibility, guarantee of a solution and ability to cope with even somewhat conflicting objectives make goal programming the appropriate technique to model the underground production problem as it is based on the principle of satisfying all the objectives. The solution is not necessarily optimal but belongs to the feasible solution space. The short time span over which the solution is implemented minimizes its impact on the global optimality of the process. Indeed, as part of the monitoring and control process of the schedule, a re-schedule is likely if the schedule parameters change significantly.

1.5.2 Implementation Aspects

The implementation and monitoring aspect of shift production is performed by a simulation approach. Simulation is used basically to test the different strategies of resource allocation using active dispatch models. This approach enables testing of different fleet allocations and/or comparison of alternative courses of action. The stochastic events of draw-point blockages, equipment breakdowns, and the dynamic queuing situations of machines can be evaluated under different dispatch policies. This methodology provides the basis for dealing with deviations from targets arising from the dynamic effects occurring during the shift.

During a production shift, all materials handling equipment are under the direct control of a central computer running several dispatch policies. The machines are linked to the controller by some data transmission and communication hardware which allow a two way information flow. For each trammed load, its tonnage and material type (i.e. ore or waste) as well as the visited dump point are recorded in real time for use in the shift control system.

The outcome of the simulation processes are strategies to implement under different scenarios. This information can then be coded into an expert system to run the production system in a mine equipped with the necessary hardware.

The success of the simulation model depends on a sound statistics base of the key simulated activities which identifies the stochastic variables. Time studies are routine methods of data collection for batch production units and are a requirement for this method. In this regard, field work is essential to successfully model a mine's production system. Once the statistical distributions are defined, they can be used until such a time when the data is found to be non-representative of the evolved mine layout or when changes are made in the equipment fleet.

1.5.3 Control Aspects

The control strategy proposed in this work is a pseudo-static model where the decision maker reviews the production status at set key times. A control algorithm compares the production statistics up to the point of review to set targets. If deviations exceed the management set limits for any goal, the algorithm indicates the failed goal(s) and whether the goals are over or under-achieved. The DM is then prompted to accept the process and continue without modification, modify the system or terminate it for re-scheduling.

The underground mine load-haul-dump materials handling system is a batch system. A fully dynamic control model of this system in a constrained environment leads to high instability in the controller and thus results in an impractical solution. Therefore, the pseudo-static mode adopted here is the most appropriate.

1.6 Overview

In this introductory chapter, the problem of underground mine production planning and implementation has been highlighted and shown to emanate from the stochasticity in the decision variables and the highly constrained implementation environment. Multiple objectives or goals are the norm rather than the exception in real world mine production decision making. This calls for flexible methods that can solve such problems. Occasionally the information base upon which the decisions are made is incomplete, bringing in the aspects of fuzzy decision making.

Chapter 2 is a review of decision-making and production control approaches in various industries as well as in mining. Based on this review, the stage is set for describing the models proposed in this study. A rationale for decision-making in underground mining is presented in Chapter 3. This rationale can be used as a planning template for predicting or elucidating stope dilution influencing parameters. This tool allows the decision maker to evaluate the impact of the dilution influencing parameters as single variables. Chapter 4 describes a fuzzy logic model that makes use of this parametric fuzzy information to provide a comprehensive solution to simultaneous multiple variables interaction. A numeric example is presented to highlight the methodology.

Chapter 5 outlines the goal programming model used to schedule the production under multiple goals. The crisp results of Chapter 4's fuzzy logic model are incorporated into the goal programming model, eliminating one of the fundamental problems of most mathematical optimization techniques, i.e. the requirement of crisp input when it does not exist. A numeric example shows the advantages of the model in both flexibility and ability to yield pertinent information at production planning and implementation.

Chapter 6 describes the active dispatch models, the control model and the decision-making criteria used in the system. The basic system assumptions are outlined. The results of the active dispatch models are discussed in Chapter 7 through a simulation study of an underground mine production shift implementation. Finally, in Chapter 8, the key project results and contributions in the area of underground mine production planning, implementation and control are enumerated. Recommended areas of future research in the field of underground production planning are highlighted.

Chapter 2
Literature Review

2.1 Introduction

The aim of this study is to develop a methodology for defining and executing underground mine production schedules at the operational stage. Subsequent to the scheduling of work areas and resources, the next objective is to control the shift schedule. A system control involves two functions, namely, monitoring and corrective feedback. Monitoring involves determining the differences between the actual and planned performance. If during the monitoring of system, a difference exists between the target and actual performances, then corrective measures are implemented to bring the actual performance in agreement with the planned performance. Control is an essential management function that seeks to ensure that plans succeed and it is a necessity if any process is to be performed with maximum effectiveness.

The subject matter of the literature review that has been conducted became simply one of determining four aspects of process control, namely:

1. the rationale of standards or targets establishment against which the performance of the system is measured.
2. determine the performance measures or quantifiable variables.
3. evaluate the process variables and,
4. correct deviations from the standards or targets.

These aspects describe the format of this study as the establishment of production targets is resolved by goal programming and direct fuzzy logic modelling. The other three aspects represent the proposed active dispatch and control models.

A production plan consists of a number of objectives with set bench-marks. In underground mining these objectives are usually productivity, product quality requirements and utilization of the production fleet. The industry practice in the setting of each of these objectives is reviewed below.

2.2 Quality Control and Cut-off Grades

The term 'ore' defines that part of a delineated mineral deposit that can be extracted, processed and marketed at a profit in an ongoing mining operation. The cut-off grade marks the level of mineral concentration at which ore and waste are defined. It enables the calculation of the available ore reserves, average grade above cut-off and the revenues.

Partly due to its role in revenue determination, cut-off grade has received tremendous research. Lane [1988] made significant contributions to the theory of cut-off grades. He proposes the global need to optimize the cut-off grades with respect to the installed facilities capacities, namely, mine-mill-refinery in the case of a vertically integrated system. Such a cut-off grade he terms the optimum cut-off. Dowd [1976] used dynamic programming (DP) to optimize cut-off grades and production rates. This work indicates that the best exploitation strategy is one that involves the mining of high grade at the beginning and low grade at the end of the project life. A similar approach was taken by Elbrond et al [1982] in their study of mining production rates. The approach is limited because it assumes a wholly accessible deposit at any point in time to achieve the progressive selective mining. Ground control problems, limited ore development and spatial distribution of grades are crucial factors to a correct establishment of a production plan.

Taylor [1972] discussed the various cut-off grades at different evaluation stages of a mine project through to the mine-mill grades. His emphasis is limited to the functional planning cut-off grades, though he noted the following:

1. the likely most important cut-offs are those at the point of mining and,
2. bulky mining methods have two cut-off grades in series, namely an in-situ pre-

development cut-off and a draw cut-off grade of broken ore.

However, Taylor does not elaborate on the problems caused by the dual cut-off grades. This subject has virtually remained unexplored by researchers in the last two decades despite its significance in correlating pre-development data and production phase assays, wall dilution and mixing function of broken material as it is drawn.

Billette and Elbrond [1986] discussed the question of cut-off grades in the context of production planning. They argue that cut-off grades should vary with respect to the mineralogical, mechanical and locational properties. These production cut-off grades are imposed by quality control considerations, e.g. a smelter contract agreement and/or the incremental development cost. With respect to this issue, we expound that despite having production cut-off grades, the production activity is dynamic and open to both internal and external influences of the environment. Internal effects arise from the potential existence of waste or very rich pods within a stope which could be missed in the ore definition programme. External factors such as wall dilution and market price impact on the cut-off grades in the production phase.

The use of linear programming in ore quality control parameters has been conducted by several authors such as Johnson [1969] and Fytas [1986] to determine the optimal strategy such as in scheduling of open pits. Johnson [1969] used dynamic cut-offs for short to long term planning in which the cut-off grade changes with time to reflect the state of the system and the future cost/price forecasts. Mirani [1969] applied a critical path method to both long and short-term underground mine planning in which the difference between the two plans was only the length of activity times. A cost function was determined to relate the mining sequence to the mineral sales (revenues). A critical path method has also been applied to quality control of mine-mill ore. In this model, the impurities are set as resources that do not exceed certain levels and those ore sources on the critical path are considered scheduled [Russell, 1987].

2.3 Dilution

Ore dilution represents the amount of waste rock in the diluted mined ore. It is commonly between 5% and 30% and is highest in the bulk mining methods such as blasthole and sub-level mining due to lack of selectivity of such methods. O'Hara [1987] estimates that dilution is inversely proportional to the square of the stope width and higher in steeply dipping orebodies. Dilution though, is a production phase problem which is usually factored into the functional stage computation of recoverable reserves. This is an acceptance of the importance of this factor, but is in no way more useful than just an estimate. Dilution changes continuously either during ore drawing, ore blasting or simply from inclusion of waste blocks.

David [1988] describes the primary cause of dilution as the sampling degree and the level of selectivity that is practised. Since the mining blocks are based on grade estimates and not actual values, there are errors in the assignment of material destinations. Some ore is declared waste and some waste declared ore. The intensity of sampling improves these grade estimates and should thus result in less internal dilution. While the effects of this dilution is known, no effort has attempted to tie this to the several "short" term production algorithms and schemes. The review made it clear that several publications on production planning in the mining industry address the aggregate issue of planning in the medium to long term. The realms of short term production have been limited to monthly and weekly planning aspects and not to the daily production. A possible reason of not studying the daily production level is the complex nature of this planning phase characterized by dynamic variables.

The economic effects of ore losses and rock dilution in a mine-mill system indicates that if these effects are greater than predicted they may render a project unfeasible [Elbrond, 1994]. This work does not suggest strategies to deal with these unforeseen problems when they arise at the production stage.

The dilution problem has to be dealt with through a continuous assessment of the drawn tonnage and comparison to the expected stope tonnage giving a continuous reconciliation of production. This procedure highlights potential problems such as excessive dilution, and allows the down grading of the stope ore quality in subsequent scheduling.

2.4 Blending and Product Specifications

Generally, smelter contracts have penalty clauses on the quality of product in terms of its moisture, grade, weight and associated impurities. A range of acceptable deviations in each of these terms is the norm. Similarly, the process plants may require feed of certain specifications, for example, grade, ore moisture and hardness for their efficient extraction. Elbrond [1981] describes the blending function which represents the variation of ore grade from its source to the final destination. The effects of ore breaking through blasting and handling, and the merging and separation of streams of ore are considered through the use of standard deviations and auto-correlation. Unfortunately, this work required too much data acquisition which made it both expensive and obsolete when eventually compiled in the context of a day-to-day operation where variables are very dynamic. Such a problem may now be overcome by modern data acquisition techniques. Unfortunately, no further work has appeared in the mining literature with respect to this topic, resulting in a potential loss of a valuable method of achieving underground ore specifications. Real time monitoring of mined grades should allow equipment to be dispatched in a way that achieves a specified blend. The material mix in the ore holding facilities can then be known both in terms of amounts as well as the stratigraphic levels which is determined by the order the material is dumped into the facility.

A polynomial regression model is described by de Gast and James [1972] to predict the grade distribution of an orebody. The method involves the simulation of an ore block's volume, grade and variance of the grade by triple integration and matrix algebra. The grade at any point within a block is a function of its position in a three dimensional space. By using the exploration drilling data sets, they were able to compute a spatial distribution of grades within stopes and also define any particular block by its grade and standard deviation. The basic assumption of this trend surface analysis is that significant grade trends must occur for the grade at any point in the deposit to be a function of its position. The success of the approach in cyclical thin inter-layering deposits seems impossible. However, the advantage of the method lies in the ability to predict grades at the point of mining without need for grab sampling of the broken ore. The method does not take into consideration the dilution effects that arise through mining.

A geostatistical study of strati-form deposits of Mount Isa was done to determine the effects of varying the exploration drill-hole spacing on the stope grades and tonnages [Dowd, 1986]. The results indicated little difference between the reported tonnage and grades at a tight drill pattern to those at larger spacing which implies, for this particular mine, it was economical to do exploration on wider drill spacing. However, the stope designs based on the different drill intensities showed significant differences suggesting that where quality control consideration is high, it is prudent to reduce the drill spacing. The actual stope depletion should therefore be conducted within a background of the processes and assumptions that went into its definition and design.

Billette and Elbrond [1986] used geostatistical techniques to analyze anisotropic deposits. Anistropy exists in most ore deposits and is an essential factor to consider in the control of ore grade streams. The direction of mining and rate of advance would enable a more accurate prediction of the ore characteristics. The duration of that ore supply would be determined, thus allowing liaison with the process plant to adapt the mill parameters to suit. However, the authors applied their ideas to medium term planning rather than daily production.

The preceeding references indicate that dilution is an important parameter in setting the quality targets. The use of grade estimates modified by a static global factor to reflect the planned dilution is the current industry practice. This approach fails to address the spatial distribution of the material as a function of the material's response to the planned dilution. The quality variable is assumed known crisply which is unlikely as the dilution effects are not correctly tackled in these models of quality optimization.

2.5 Production Scheduling

A production schedule is a plan illustrating the material sources, quantities and quality to be mined at each scheduled source and the equipment assigned to do the job. The material destinations are also included such that the material flow in the handling system is known for future decisions.

An early technical overview on production scheduling in open-pit mines was by Kim [1979]. He concluded that though production scheduling comes at the bottom of a mine planning hierarchy, it is the most difficult and demanding task to achieve. The reason is that production planning has to conform with both the medium and long range plans as well as the practical aspects of day-to-day operations. At the time, Kim attributed the non-use of operations research (OR) techniques at this level of mine planning to the logistical problem of data input when such methods are used. Also, a lack of comprehension in the 1970's and early 1980's by most users did little to promote the use of OR in operational planning, especially in underground mines.

Mutmansky [1979] describes the various OR techniques that have been applied to the mineral industry which include linear, mixed integer and dynamic programming methods, simulations and heuristic algorithms. Of these methods, linear programming has been the most applied technique for both scheduling and blending problems because of the simplicity of the Simplex method. Johnson [1969] applied the method to a multi-period schedule of an open pit to maximize profit. He incorporated the idea of variable cut-off grades, a concept proven to yield the optimum depletion policy. Johnson's multi-period LP gave sub-optimum solutions which were practically acceptable. Parameterized LP was used by Dagdelen and Johnson [1986] for scheduling open pits. The process involved determining nested open pits through variations of Lagrangian multipliers such as price, cost or grade in the maximisation of a profit objective function. Bonates [1992] applies LP to an open pit shift production schedule. He attempted to apply the same approach to underground mining [Bonates, 1995].

A branch-and-bound algorithm and other enumerative methods are used to solve integer and/or mixed integer programming problems (IP). These problems require certain variables to take integral values, for example in allocation of machines to different work areas. Daud and Pariseau [1975] applied the branch-and-bound method for the assignment of trucks to shovels in open-pit mines.

The dynamic programming technique is used to find an optimal sequence of decisions for problems that can be likened to sequential decision processes. Several researchers have applied the technique in the mining industry such as Noren [1969], Dowd [1976], Elbrond et al [1982]. The problems are decomposed into smaller sequential problems for which each sub-problem is solved. The mathematical formulation of DP is based on Bellman's principle of optimality that is expressed as:

$$f^*_n(s) = \min_{x_n}[f_n(s,x_n)] = f_n(s,x^*_n) \qquad (2.1)$$

where s = stage in which the mining process is in, e.g. current operating stopes

n = represents the next exploitation stage e.g. the next shift schedule

m = total number of stages in the system e.g. shifts, weeks, months or quarters, or stope slices, open-pit benches, etc.

x_n= decision variables available for selection, i.e. which stope to mine and how much,

x^*_n = denotes the decision taken at stage s that minimizes the objective function $f_n(s,x_n)$ such that the accumulated cost or quality deviation of the exploitation process at stage n becomes $f^*_n(s)$.

$f_n(s,x_n)$ = immediate cost or quality deviation in the nth mining stage plus the minimum future cost or quality deviations in the remaining stages (m-n).

The listing of the decision variables, x^*_i for $i = 1$ to m, defines the optimal strategy or schedule for production. The optimal strategy depends on the realization of the future minimum costs or quality deviations. Since this information is limited or unavailable for such forward planning, it implies that the method is useful mainly as a guide to production but cannot individually satisfy daily production scheduling.

The extension of forward pass DP used in open pit scheduling is described by Tolwinski and Underwood [1992] in an algorithm that uses heuristic rules of Artificial Intelligence to learn the best extraction sequence. The learning process is essential because an optimal sequence may be difficult, since high grade ore or ore reserves may not be located in readily accessible parts of the mine. The algorithm searches only a limited number of sections of the mine and determines those with the maximum profit, i.e. it maximizes short term gains. By periodically investigating areas of the mine that do not yield the best short term gains, the algorithm consistently updates and ranks the intermediate pits as mining continues. Tolwinski and Underwood's contribution is a radical departure to the traditional use of DP scheduling using cumulative grade-tonnage distributions. The traditional DP is applicable to a medium to long term development and extraction strategy but not to daily production.

In underground mining, DP has been applied to mining sequence optimization in a sub-level open stoping exploitation by Ribeiro [1982] and Dowd and Elvan [1987]. Muge and Santos [1990] applied DP to cut-and-fill mining. In all of these publications, the authors use the same criterion function of minimizing the grade variability of the mined ore.

The models do not address dilution effects from both internal and external waste that occur during blasting and ore drawn as a consequence of following a certain mining sequence. It is therefore argued that the models' objective of minimizing grade variation may not be achievable.

A two module mine planning system for underground mines is described by Mirani

[1969]. The first module forecasts the future market demands using exponential smoothing and Markovian techniques. The results become the inputs of the production planning module implemented on a critical path method to schedule yearly production at minimum cost, on a month by month basis. The recent work of Gillenwater et al [1995] and Wilke et al [1995] has followed Mirani's approach of segmenting the mine production system into hierarchial planning units. Each unit has set objectives and different levels of detail. The outputs of one unit become the input to the next lower unit. In Gillenwàter's model, LP is applied to both the overall mine and mine section production levels with the goal of achieving a unique coal quality. The Wilke et al [1995] model uses a knowledge base to construct logical constraints for the inputs to three sub-systems (network analysis, LP, simulation). The authors emphasize that the traditional methods of simple LP have limitations on the practical aspects of mine planning. A mine planning system has to adapt to the changes in its operating environment.

The mathematical technique of linear goal programming (LGP) has received widespread application in many disciplines such as health, capital budgeting, finance, both public and private management and the military. The reason for this diversity in application is the method's flexibility and ability in solving multi-objective problems which have conflicting goals. Trivedi [1981] describes a mixed integer goal programming model for determining a nursing department's annual budget. The model incorporated several objectives such as cost containment, provision of appropriate services, and minimizing part-time nurses. Trade-off were made on issues of leave, holidays, and overtime.

Lee and Jung [1988] describe a goal programming model in a flexible manufacturing environment. In their model the objectives were to achieve set production targets, machine workload balancing to ensure maximum utilization, and minimization of throughput time. The later two objectives are in conflict as minimization of throughput time could prevent the attainment of equitable or uniform machine utilization.

Linear goal programming has been demonstrated in the optimal allocation of financial assets into portfolios. Lee and Chesser [1980] incorporated beta coefficients (i.e. measures of stock risk) from financial theory into a LGP model to reflect the risk in alternative investments. Following on the same concept, Schniederjans et al [1993] illustrate LGP use as a tool to aid investment advisors who plan investment portfolios for individuals. Their model incorporates the total wealth in the form of financial and non-marketable, illiquid assets. The authors accept that their model is seriously limited by the need of the input parameters to be known accurately. In the real world, the parameters used are based on historical data yet the investments depend on an unknown future.

The goal programming technique has found application in urban planning as described by Miyajima and Nakai [1986] who use econometric estimation techniques to obtain coefficients for both exogenous and endogenous variables. They combine a system of simultaneous equations and goal programming to give a model amenable to iterative search for a solution through a change of goal levels if one run of the GP fails to yield a solution. Soyibo and Lee [1986] applied a GP model to a multi-period planning scheme for a university staff size that involved an implementation of Markovian estimates for the changes in student enrolment and number of professors.

In the mining industry, goal programming use has been limited. Some early work was by Barbaro and Mutmansky [1983]. They developed a goal programming model for coal blending to meet multi-objective contractual requirements. Non-linear functions were solved by piece-wise programming and then used as input into the model. More recent work is by Youdi et al [1992], Jawed [1993] and Tsomondo and Lizotte [1994]. Youdi et al applied goal programming to medium term planning of an open-pit scheduling problem. Jawed [1993] describes a model similar to that of Barbaro and Mutmansky [1983] for coal production with the objectives of maximum production at the least cost. These articles describe crisp goal programming models, i.e. those in which the parameters are specifically known. This is a limitation especially in mining where the information base is small and many decision variables are stochastic. The base tends to grow as more information is gathered through experience on site.

Tsomondo and Lizotte [1994] apply the goal programming concept to daily production planning to provide the inputs for shift to shift schedules. The schedules are reviewed periodically and if deviations from targets exceed set levels, the goal programming model is used to re-optimize the new operational conditions. In their preliminary work they pointed out the need to consider the uncertainty in the production problem through use of fuzzy and/or interval goal programming. This research work is an extension to that earlier reporting.

The mathematical algorithms (LP,LGP,DP) require that input variable coefficients and constants values be known precisely. In some cases these values are vague and treating them as crisp values may lead to unjustified confidence in the input data. This type of data is best treated by fuzzy mathematical theory which allows the determination of the degree of feasibility of the solution with respect to the model definition. Zimmermann [1989] describes an expert system for strategic planning for firms with large portfolios. The system aggregates vague information of different dimensions (type) such as a firm's competitive position, market share, technology position and technology attractiveness to continued growth. The firm's strategy is then made based on its vectorial position in a portfolio matrix defined by the composition of the various dimensions. This approach seems appropriate in the analysis of individual mining stopes with respect to both exogenous and endogenous factors influencing ore quality and production control.

Hintz and Zimmermann [1989] describe a method to control flexible manufacturing systems (FMS) through a fuzzy linear programming and approximate reasoning (AR) hybrid model. A fuzzy linear model solved the master plan. The master plan implementation, i.e. machine and job release scheduling are done through a heuristic AR procedure. The AR procedure uses a hierarchy of decision criteria and fuzzy aggregation concepts of the rule base. The approach performed better than a generally used FMS simulation program based on simple priority rules for release and machine scheduling.

Several workers describe the use of fuzzy logic operators in the solving of multi-attribute objectives problems. The use of triangular fuzzy numbers to describe the range of applicability of a linear programming solution are discussed by Gen et al [1992], Sasaki and Gen [1993], and Nakahara and Gen [1993]. Their work defines the mathematical reasoning in fuzzy multi-objective problem solving. Ward et al [1992] describe a fuzzy logic control of aggregate production planning in a manufacturing system defined by the Holt-Modigliani-Muth-Simon (HMMS) system. They formulate the expected system cost for holding inventories, producing a certain number of units and the labour cost in a time unit. Fuzzy logic controllers are then used to predict the cost at certain times in the future given variable fuzzy inputs.

An et al [1991] apply fuzzy logic theory to mineral exploration for the interpretation of multiple geophysical and geological data. These data sets are complex and ambiguous and may not be represented by classical statistical theory due to their spatial representation. By applying fuzzy logic operators to data sets from the Farley Lake area in Canada, they successfully outlined favourable areas for base metal deposits and iron ore formations. Another use of fuzzy logic techniques in geophysical exploration has been in the interpretation of remote sensing imagery by Wang [1989].

2.6 System Utilization and Control

The concept of control implies a diverse field of applications such as in banking, telecommunications, electronics, manufacturing, transportation and many others. As such, this section of the review covers many areas of production control in which both operational research techniques and computerized production is practised.

2.6.1 Flexible Manufacturing

The state-of-the-art in manufacturing has moved towards flexibility, automation and integration of systems. The incentive is to generate plant floor schedules that honour the factory resource constraints whilst taking advantage of the flexibility of the components. A concept of flexible manufacturing systems (FMS) is one that allows small to medium sized batches of different types of jobs to be performed by the same work station. For a station to be able to process a different job, the station set-up is changed to suit the new product. The flexibility of the plant is measured by the key issue of change-over cost. A FMS is supposed to be one with virtually no change-over costs but in practice these costs exist and can be significant depending on the planning methodology as described by Kusiak [1990]. There are two basic features of FMS, namely the change-over costs and precedence constraints, which determine the sequence in which a product is manufactured. The change-over costs are influenced by parts and tools that are required for the new product. The relative ease of switching these items reduces the lost time due to set-up. The material buffers to hold removed tools and parts can limit a station's flexibility. Precedence constraints in FMS are imposed by market demand which sets due dates. Penalties are imposed on failure to meet deadlines.

Flexible manufacturing systems expanded rapidly in the 1980's as more and more companies adopted the technique. One of the issues that facilitated this adoption was the changing consumer tastes that ceased to be satisfied by a few models. The automotive industry in North America previously held the concept of a focused, single commodity, mass production line. In the 1980's they found this principle no longer profitable and had to adopt the FMS for survival under global competition [Jain et al, 1991]. The issue of global competition is currently gripping the mining industry where countries with mature mining industries such as Canada and the USA are competing against emergent cheaper to mine deposits in developing nations. Therefore, the lessons of FMS could be drawn to the advantage of the former mines to stay competitive.

In a detailed survey on production scheduling, Graves [1981] pointed out that FMS shows both flexibility and inflexibility. The flexibility arises from the generality and adaptability inherent in the processor stations which provides many alternative routing possibilities. Inflexibility is caused by the automated materials handling system as a result of limited capacity of both transport system between stations and the storage buffers at processor centres. Therefore, an FMS must model the transport-storage constraint.

Currently, there are no flexible underground mining systems. A comparison of the FMS production structure to a hypothetical system in underground mining operations indicates both similarities and differences. The current mine layouts and facilities set-up denies flexibility in most mines. Machines are captive in most operating mines. At some mines a load-haul-dump machine re-location can take up to a month, as observed by the author during a survey of mines. The equivalent of a rush job in mining exists when a previously available work face becomes unavailable and management must therefore increase production in other areas to compensate. Under the circumstance of captive equipment, it is impossible to increase production in those areas except through over-time work. A flexible mining environment however could allow the equipment to be quickly dispatched to the critical areas.

An FMS has several work areas and sometimes fairly complex materials handling systems such as conveyors and/or automated guided vehicles (AGV). The AGV direction of movement may be one way, two way or a system with dual pathways as described by Kim and Tanchoco [1993]. At each station there tends to be some limited buffer space. If this space is occupied, then the system is blocked as no new job can be introduced at the next work-station. Different scenarios of FMS blockage include a full input buffer in which an AGV cannot unload its cargo, or an output buffer where the processor station cannot download a finished job. These situations have been found to adversely affect the FMS performance. A trackless underground materials handling fleet is equally complex. It can consist of load-haul-dump machines of different sizes and makes; and/or of trucks and loaders working in tandem. The fleet sizes are variable, generally in the range of 10 to 30 machines. However, some larger mines have more. For example, El Teniente in Chile had a fleet of 286 machines as described by Daniels [1987]. Production performed without monitoring of such large fleets is likely to be ineffective and efficiencies cannot be evaluated in terms of work hours.

Whilst the FMS jobs move from one point to the next during a product manufacture, the jobs are fixed in mining. They are represented by the scheduled tonnages to be drawn from the stopes. The essence of flexible mining then is effected by the machines visiting the various draw-points either randomly or according to some dispatch policy. Congestion in underground mining is not of jobs as in FMS, but of machines arriving at a load or dump point whose machine buffer size is exceeded. This problem can be resolved by implementing a real time control system that keeps tally of the service points' effective buffer spaces and prevents dispatching to capacitated areas. Another constraint in underground operations is ventilation requirements for the horse-power output of site machines. The same control program could be used to automatically direct the ventilation control system to respond to changes in machine numbers in each work area.

2.6.2 Production Process Control

A production process control system is an optimisation that seeks to minimize the cost of producing a sequence of jobs. The cost can be measured in terms of product quality, quantity and/or utilization of the production system. Production processes are implemented in one of two ways, namely as off- or on-line process control. Off-line algorithms operate under predictive information about the future that is available in advance to the decision maker. With knowledge about the future, off-line control is achieved through mathematical algorithms which assure the optimality of the process. Unfortunately, the future is rarely known. Therefore, on-line process control is performed by heuristics that perform immediate actions on job requests. This is also known as real-time or reactive models and is currently the basis of many real world production systems [Kusiak, 1990].

A variant of the on-line method has a limited look-ahead capability that foresees the next few in-coming jobs based on issues such as industry inventory levels, market trend or product saleability.

An on-line control and scheduling scheme of a FMS by Wu and Wysk [1989] implements a tandem simulation-dispatch system. A control mechanism dynamically changes the dispatch policies of FMS jobs based on simulated information. The simulation evaluates a set of sound dispatch rules for a short planning period and selects the best rule under simulated conditions. This process is repeated periodically creating a multi-pass dynamic model. The model shows better performance compared to single-pass, static dispatch rule applications. The basic concept in this model has been investigated by Murotsu et al [1983] who suggested that a control and scheduling strategy should be based on a pull-system rather than a push-system. A push-system is based on the traditional format of globally scheduling jobs ahead of time (i.e. sequencing). This global approach tends to be disturbed by system uncertainties. A pull-system allows jobs to be processed according to the local status of the system (i.e. dynamic state).

The Wu and Wysk [1989] model of a FMS requires a simulation component to generate jobs and their characteristics. But by simulating a certain number of jobs ahead of time and then subjecting them to different dispatch rules, the model uses an off-line control strategy. Whilst some look-ahead feature is feasible in FMS through forecasting, the same cannot be said of implementing a mine shift plan under a similar model structure.

Ishii and Talavage [1994] describe a mixed dispatching rule approach in FMS scheduling where each processor centre can be assigned a different dispatch rule based on discrete event simulation. The current status of the FMS and other pertinent environment information is used by the real time scheduling system to select a dispatching rule that best suits the selected performance criteria in the next short time period. The approach is similar to that described by Wu and Wysk [1989], except that the former model has a fixed schedule interval.

FMS production variability is due to both internal and external factors. The external factors relate to market demands that are uncertain and sometimes seasonal. The internal variability is due to capacity availability and the process yield. The internal factors affect the amount of product available to meet possible demands. Ciarallo et al [1994] developed an aggregate production planning model with both uncertain demand and capacity for a single product. They studied the planning policies over single and multi-period scenarios and concluded that under a one period case there is no incentive to produce more than the optimal amount. Such a situation assumes that no restrictions on production are imposed, i.e. the system is capable of meeting the demand in each period. In a multi-period situation, extended policies are proposed to respond to uncertainty in productive capacity by building up inventories over time to compensate for periods of low capacity. This work provides an approach that may be applied to underground mine planning in that the capacity availability and product quality (grades, contaminants) are uncertain. The output in each planning period (shift) may be variable due to equipment availability. The demand at the process plant however, is known and is essentially fixed in quality and quantity.

Classical closed loop queuing models (CQM) have been used to analytically determine FMS performances but these ignore the blockage or starving of the system. Tempelmeier et al [1989] describe a modified classical CQM that incorporates these dynamic variations through an approximation of the probability that the FMS main buffer system is full. The method allows the determination of the conditions under which the system is full, such that finished products at the stations remain in the output buffers. This method gives only the salient long-run conditions of the system and it ignores the transient effects of blockages. A starved system is one in which the ratio of the number of jobs to the servers is small. This leads to accumulation of idle time by the processors, a typical problem in underground mines that use a loader and truck system. Tempelmeier et al [1989] concede that their method is insufficient for real-time situations. They recommend simulation techniques in this instance.

A problem of system blockage in tandem queuing systems with finite capacity in FMS has been treated both as controlled and uncontrolled. In uncontrolled systems, blockage is only approximated. Under controlled conditions, the probability of finding a work station blocked is near zero, as sufficient storage is allowed and the number of automated guided vehicles is kept low. Pourbabai [1993] used this approach to maximize the throughput of a FMS by appropriately selecting the service rates at assembly and transporter stations using a Poisson arrival process and exponential inter-arrival times of jobs on a single queue (M/M/1). Similar blockage features are present in an underground mine where the material source and dump-points are the tandem systems. Buffer capacities are limited at both areas. However, whilst it may be possible to derive "appropriate" service rates in FMS, the loading and dump times are random and dependent on machine power, material fragmentation and its flow characteristics. The underground mining blockage problem cannot be effectively modelled by analytic methods. This system has uncontrolled blockages.

Mao and Kincaid [1994] investigated the look-ahead control option in which two queues are maintained: one for jobs already waiting and another for jobs next to arrive. The concept is useful for situation in which the next customer has a higher priority than those already in the queue. The scheduler chooses that option with the minimum time based on the assumption that no more jobs will arrive in that future queue, i.e. there may be a case where all incoming customers have increasing priorities, resulting in an "infinite" building of the queue. The possibility of greater than two machine priorities within a mine fleet which would cause a queue build-up is small. However, the space constraint limits the application of this procedure in underground mining. Without the above constraint, the procedure is superior to myopic systems since one can compare the utility or benefits of a current to a future customer in the selection of an appropriate destination. A situation arises when a current customer is sent to a second-best processor if that reduces the overall service times of the two or more next units to be served. Soumis and Elbrond [1990] developed a similar model for open-pit mining equipment dispatch. Since the open-pit mine lacks a space constraint, their approach is appropriate.

Aytung et al [1994] describe the use of intelligent objects for decision making in a simulation environment. Their method is based on classifier systems using a Genetic Algorithm such that the intelligent objects learn to perform within their environment. The conceptual model consists of:

1. intelligent objects represented as a dispatcher or decision making rules,
2. passive objects which behave according to chance or as dictated, and
3. jobs which are given service times, due dates, and a queue according to some discipline.

The model allows the dispatching of customers to destinations through a knowledge base inference engine. The effectiveness of the model depends on the knowledge base.

Certain queues have a two component cost, namely the cost due to a customer waiting in a queue and the cost of an idle processor. In such circumstances, the objective is to minimize the total expected cost of providing the service. Stein and Cote [1994] studied a multi-customer single processor queue in which each customer in a queue has a cost, c, and the processor an idle cost, s, per unit time. The total idle times per shift are calculated. By expressing each unit cost as a ratio of the total unit cost $(c+s)$, the relative importance of the two components is derived. This ratio can then be used to reflect a trade-off between efficient use of the facility and efficient service to the individual customers. This concept is useful in matching the equipment fleet and dispatch rules that minimize total fleet cost.

Yamamoto and Nof [1985] compare three job-shop scheduling procedures when machine breakdowns occur, and show that scheduling/re-scheduling approach improves system performance by 2-7%, compared to dispatch and fixed sequencing procedures. The flexibility of computer manufacturing systems has been attributed to the success of bringing re-scheduling processes into the realms of real time control. Yamamoto and Nof [1985] indicate that a schedule is not necessarily generated in consideration of optimality, but near optimality of the total system. An initial schedule is generated just prior to the start of a new work period, based on all necessary information: quantities, types, etc. The conditions essential to the performing of the jobs in the work period are determined and equipment and resources allocated. The term scheduling, as they use it, is synonymous with sequencing of jobs based on their due dates. A difference exists between this approach and one done under a closed-shop system, i.e. without due dates. In the later case, schedules are based on productivity targets. Such a system has been studied by Bonates [1992] to indicate the performances of active and fixed dispatch modes in open-pit mines.

A combined sequencing, dispatching and switching approach to dynamic FMS scheduling under conditions of breakdowns and specification changes was studied by Matsuura et al [1993]. The sequencing was performed by a branch-and-bound technique under certainty of future information of open jobs. This allows the jobs to be ordered with respect to their due dates. Due to system uncertainties that affects the sequence, they proposed the use of dispatch rules (shortest process time and first-in-first-out) to schedule jobs when the system has significantly shifted from target. The conclusion was that sequencing performs well under static conditions and the dispatch rules are responsive to dynamic conditions especially when breakdown durations and specification changes were large. By combining the two elements into a switching approach, they utilized the strong features of both sequencing and dispatch in FMS. This approach provides some insight into the possibility of implementing a multi-type scheduling approach for underground mining. A number of researchers investigating dispatching policies have recognized that a combination of simple dispatch rules in many cases performs better than an individual rule [Gere, 1966 and Wu and Wysk, 1989]. This result follows from the fact that each dispatch rule is a single objective rule that applies regardless of the obvious system interaction.

The concept of truncation in job-shop scheduling has been investigated by several workers: Eilon and Cotteril [1968]; Eilon et al [1975]. Truncation rules are based on the truncation of jobs with long waiting times or negative slack times in a queue with reassignment to a higher priority queue from which they are dispatched on a first-come-first-served rule. The objective is to expedite the late jobs. Kannan and Ghoshi [1993] studied the interaction between dispatch rules and truncation procedures and conclude that the appropriateness of a truncation-dispatch rule scheme is essential for high system performance. Based on this work, it is concluded that a single objective dispatch rule is incapable of achieving multiple goals. Therefore, within a dynamic mining environment, dispatch rules need to be changed, selecting the most appropriate for the local situation to effectively improve the least achieved production goal.

The loading rates at stope draw-points is random. Therefore, if the rate falls below a certain level or the stope becomes blocked, this causes excessive queuing times at these stopes. In such a situation, invoking truncation of machine queues is the most logical action.

2.6.3 Transportation

Cai and Goh [1994] studied a train scheduling problem to minimize train idling time at passing loops. They present a simple heuristic algorithm for quick generation of feasible solutions for on-line implementation. The model precludes saturation in the passing loops, and assumes that the trains cannot reverse nor over-take. The heuristic procedure schedules trains according to the rule:

$$\textit{IF } C(i) < C(i+1) \textit{ THEN train } C(i) \textit{ stops} \tag{2.2}$$

$$\textit{ELSE train } C(i+1) \textit{ stops} \tag{2.3}$$

where C(i) and C(i+1) are the cost of stopping trains i and i+1 respectively. Stopping the high cost train would increase the total cost, C(i)+C(i+1), more than if the less costly one is stopped. This procedure is rather simple and would fail in the event of delays in the high cost train. Besides, the use of such an algorithm would be suited only to freight trains rather than passenger trains, because passengers would hate being delayed.

The removal of urban snow and disposal operations involve a host of engineering and managerial problems. The main objective in snow management is provision of snow clearance service at a minimum cost. The quality of service is specified by whether deadlines for complete snow removal and disposal are met, subject to economic and political considerations. The issues of concurrent equitable or uniform distribution of snow removal equipment in different sectors of the urban community and the scheduling of disposal should be met. Campbell and Langevin [1993] reviewed these problems for the case of the City of Montreal and conclude that the use of linear programming, or assignment of particular sectors of the city to fixed disposal points is not satisfactory. These techniques are complicated by interdependencies in the problem variables. They suggest that the operational problem of snow removal should involve the routing of trucks between sectors and disposal points in a dynamic fashion to reflect the movement of the snowblower (loader) while a truck travels to and from the disposal site. This author suggests that the interdependencies and multiple objective nature of the snow removal problem could be better treated as a goal programming problem operating in tandem with a dispatch model.

2.6.4 Underground Mining Systems

One of the early applications of computer-assisted planning is by Suboleski and Lucas [1969]. They simulated a room-and-pillar face mining operation to determine bottle-necks in the mine layout. This information was invaluable in medium and long term planning of design layouts, section equipment selection and justification of capital expenditures.

Edlund [1971] reports a computerized traffic control system at the Kirunavaara iron mine in which ore is hauled by trains from 83 ore-boxes on the main level to six underground crushers. A computer simulation technique was used to optimize a proposed new main haulage level layout on the basis of installed equipment. A transport network system was constructed upon which train idle times due to train interferences were determined as more trains were added to the system. An optimal number of trains for the network system was identified. Train dispatching was based on a set of heuristic procedures. The ore dumping destination of each train was determined on the basis of real-time sampling of the train load. Wilke [1971] describes a similar traffic control simulator for large coal mines that use a rail system. The requirements were to ensure minimum production loss at the face, minimum costs and flexibility. Three dispatch rules were tested, namely:

1. Earliest loading destination based on number of cars still to be loaded at a loading point;
2. Earliest loading destination and,
3. Most idle loading destination.

Results indicated that the simple earliest loading destination based on number of cars at the loading site without consideration of the train's travel time was inferior compared to the other two procedures.

A generalized underground transport simulator is discussed by Ryder [1977]. The simulation model features the modelling of a locomotive tramming system involving interferences and queue build-ups. The model was used to establish simple pulling schedules of ore-boxes, and an adequate tip point configuration for a proposed tramming level. The optimal car size to locomotive size was investigated. These simulators are all appropriate for mine design purposes, but not for the production phase.

The effective use of load-haul-dump machines (LHD) remains low, averaging 4 to 5 hours per shift even under favourable conditions, indicating that new methods of increasing productivity are necessary. Matikainen [1991] describes a CECAM system, developed in Finland, that collects and monitors precise information on loaded muck, equipment availability and working time, and is in constant radio contact with the entire fleet. The system revealed that in most cases the actual bucket fill factor was 20 to 30 percent less than the machine's carrying capacity. Therefore, use of LHD load counts may lead to less material trammed than that reported, creating a poor information base for planning.

The benefits of the Modular Mining Systems' DISPATCH^{TM} system in open pit mines has been reported extensively in the mining literature [Arnold and White, 1982, Clevenger, 1983, and Farell, 1988]. White et al [1993] give a detailed description of the optimization algorithms and strategies used in the DISPATCH^{TM} system. These are production maximization, minimization of material re-handling, and meeting ore blending specifications. These strategies are reduced to a single objective by setting the rest of the objectives as system constraints. The model consists of three modules, namely a 'Best Path' module based on Dantzig's shortest path algorithm, a linear programming (LP) module, and a dynamic programming module. The LP module minimizes a single objective function, expressed as the number of trucks required to cover the operating shovels subject to the mining constraints. The LP module variables represent the target flow rates in tonnes per hour for each determined best path from a shovel to either a dump-point or the crusher. The dynamic programming module uses the Best Path module output to assign the trucks to travel routes based on the common rules such as maximizing truck and/or loader productivity, or prioritizing some shovel sites.

Another approach applied in open-pit production planning and dispatching is described by Soumis and Elbrond [1990]. Their model defines initial shovel positions that meet the shift objectives of tonnage and grade by mixed integer programming. A non-linear programming optimization is then performed for a given number of trucks and the set of shovels that are known to generate a solution. The non-linear programming single objective minimizes the cost effects of the following three components:

- the differences between the actual and computed shovel productions,
- the difference between the available and computed truck work hours, and
- a penalty for deviations of computed production blend from the desired grade and other quality properties which is introduced as an increasing cost.

The solution consists of the mining rates and truck paths used in the dispatching procedure. Truck dispatching is based on a mathematical assignment optimization that takes into consideration the destinations of the next 10 to 15 trucks to be dispatched, thereby providing some long-sighted allocations.

Luke [1993] describes the first adaptation of the Modular system to an underground mine at the diamond Finsch Mine in South Africa. The underground communication system consists of Leaky Feeders and infrared beacons. The system allows an LHD unit to transmit its beacon location and data (e.g. its status) via a radio to the Leaky Feeder and up to the main control computer. The main computer can also communicate with the LHD's on-board computer. A fixed dispatching procedure is used in this underground system whereby a machine is assigned to a set of load and tip points at the start of the shift. This procedure has been shown to be sub-optimal for open pit mines by Lizotte and Bonates [1987]. It was used at the Finsch Mine because of its simplicity. Despite the fact that it is based on a fixed dispatch system, Luke states that the system at the Finsch Mine has improved LHD fleet productivity.

A LHD vehicle tele-operation and guidance system was developed and tested at Inco Ltd., [Baiden and Henderson, 1994]. The system indicates the feasibility of multiple LHD operation by one operator from surface, potential improvements in productivity and safety, and reduced operator cost. The LHD has on-board load weighing instrumentation and a programmable logic controller recording the bucket loads, and engine and hydraulic conditions. Knights et al [1995] describe an automated ore tracking system using radio frequency identification techniques developed in conjunction with the Canada Centre for Mineral and Energy Technology (CANMET). The system allows the reconciliation of a machine's activities during a shift, i.e. draw-points and dump-points visited, and in what order. The accumulated shift production figures are also available. This system does not weigh loads as yet. However, such a system would be suitable for integration with a dispatch model to effectively execute a shift schedule.

An underground continuous mining system's operational effectiveness was investigated analytically by Topuz and Duan [1991], using direct reliability analysis and Markovian modelling. Markovian state equations describing the probabilistic transitions from initial to final states were used to derive the probabilities that the system is in various states, such as breakdown, under-capacitated, or fully running. The analysis gives the system's aggregate performance in terms of availability and productivity. Comparison of the direct method to the Markovian model indicates that the Markovian performance measures are lower. This result is important as it shows that with system component interferences, performance is lower than in situations in which component independence exists. As this study was carried out under steady-state conditions, one would expect even greater interference impact on the system performance during transitory conditions. It is proposed that in real-time production monitoring, much wider performance variations are to be expected and these need to be controlled by efficient methods that minimize component interferences. One such technique is by active equipment dispatching.

A conceptual design and an analytical framework for a flexible strip mining system (FSMS) based on hierarchical control in FMS is given by Singh and Skibniewski [1991]. The system allows a bi-directional flow of information between the decision maker and the resources at the face. Sensory feedback information mechanisms allow the decision maker to send a control input to correct for the changed environment. The FSMS concept is envisaged to improve mine optimization and system design.

A knowledge based system for automatic ore blending at the Neves Corvo Mine, Portugal, is described by Caupers et al [1993]. The system consists of a dynamic simulator combined with a knowledge based system of the mining method. The simulator generates the stopes that can feasibly be mined under the current mine status based on a knowledge base of mine sequencing, rock types, stope geometry and production rates. The system is applied to monthly and yearly production schedules at the mine.

2.7 Communication Systems

The feasibility of real-time control requires a means that provides information in real time to the system controller. In the context of underground mining this implies data collection about equipment status, its distribution in the mine layout, and the accumulated production statistics for each scheduled work area. In the past this task had been impossible, explaining why underground mining lags far behind surface mining in computerized dispatch systems.

Beus [1992] describes on-going research work by the US Bureau of Mines for real time monitoring in underground mines. The system allows data collection of different types from different areas using remote sensors. The data is then transmitted to on-line decision-making software via modems. The system has been used remotely in the monitoring of the mine environment, to control ventilation and fan power consumption. There is potential for modification and adaption of this system to collect production data and remotely identify machine locations within the mine.

INCO Limited has developed a broadband communication system and a Distributed Antenna Translator for underground equipment [Baiden, 1993]. The system allows for voice, video and data to operate simultaneously. This system has successfully been field tested [Baiden and Henderson, 1994].

Maenpaa and Saindon [1992] review modern technologies in underground communication. A modified Leaky Feeder multi-channel system called MULTICOM™ allowing voice, video and data transmission has been developed. MULTICOM™ has been commissioned at Exxon's Monterey No. 1 and No. 2 mines. They also describe a Distributed Antenna System (DAS) which uses discrete antennas rather than a leaky cable to broadcast signals. The system allows bi-directional communication of multiple channel voice and data services. The system is suited to mines with long straight drifts and large open areas such as coal and salt mines.

Another development in underground communication is the Bi-Modal Advanced Network for Digitally Integrated Telecommunications (BANDIT^{TM}) which uses the proven concepts of the MULTICOM^{TM} leaky feeder system. BANDIT^{TM} is a specialized process control and information system. It combines a centralized command/control capability via personal computer based software with a unique communication facilities that allows the network to function everywhere throughout the mine. Several equipment types (level sensors, ventilation monitors, pumps and fans) can be linked to the network. The personal computer is the master station from which mine planning and automatic scheduling can be performed.

Radio frequency identification (RFID) systems have been introduced to the British collieries for vehicle monitoring and tracking of inventory underground [Hind, 1994]. The vehicle monitoring system has enabled accurate measurements of the number of loads conveyed as well as the fill-factors. Increased productivity has been reported through the use of the system. Knights et al [1994] also describe the potential uses of RFID in hard rock mining. Some fundamentals of an integrated mining information and control system expected to lead to 'just-in-time' mining or 'lean' mining are described by Knights and Scoble [1995]. The proposed benefits of a 'lean' mining system are the reduction of throughput times, material re-handling and stockpiles which lead to reduced operating costs.

2.8 Summary

This review of production scheduling theory and practice clearly indicates that most research has almost entirely focussed on deterministic and static models. Production schedules are implemented as if the production environment is deterministic and static over a finite period. This may be satisfactory in manufacturing systems where both the internal and external factor inputs can be approximated easily. However, in underground mining the production environment is stochastic and dynamic. Some input variables are not crisply defined, hence posing a problem of ascertaining the degree of variability of such input data.

As a production system configuration becomes complex, the problem of scheduling and schedule implementation requires more than one algorithm. The process is divided into inter-related hierarchial units with information exchange between the different units. The practice in FMS has been to apply several on-line routing policies in the expediting of a generally off-line determined production schedule. The use of analytical solutions to solve these complex problems is limited to functional planning in which the detail of operation is considered in aggregate form. At the operational stage, the analytic methods are inadequate as they fail to effectively account for all the dynamic variables, a phenomenon best addressed by myopic heuristic procedures and simulation techniques.

The performance measures broadly used in all scheduling heuristics are categorized into either schedule cost or schedule performance. In the FMS, the schedule cost includes the fixed costs associated with production change-overs and setups, overtime, inventory holding costs, penalties for failure to meet deliveries, and expediting costs. The schedule performance is measured by utilization of the system, proportion of late jobs, flow times of jobs and average (or maximum) tardiness. The analytical algorithms measure system performances generally in terms of average values of variables such as lost time, waiting time, and unavailability.

The application of dispatching models in several industries including open-pit mining universally report an increased productivity. The timely accumulation of production information through a centrally connected computer control system allows quick responses to production deviations. The communication systems needed for integrated planning systems are now commercially available even for the hostile underground mining environment. This suggests that the time is ripe for such technologies to be adapted or modified for their application in underground mining.

Due to the introduction of ISO 9001 standards, the goals of mine production are likely to change from the traditional single-objective of production maximization. This implies that the traditional use of LP, IP and DP methods for scheduling operations are limited because they are single objective mathematical optimizers. New methods that reflect the multi-objective criteria of the problems have to be embraced, such as the goal programming technique. In the instances when the input variables are vague or imprecise, then either knowledge base or fuzzy logic models must be used in conjunction with the mathematical programming techniques.

Chapter 3
Underground Mine Production Rationale

3.1 Purpose

The constraints affecting production shift planning are investigated in this chapter. The two types of constraints are strategic which are imposed by a long term stope schedule, and tactical constraints determined by the daily variation of planning variables. The investigation leads to a postulation of a new mine design and operating (tactical) criteria for underground mines. Some guidelines to production plan evaluation are determined.

3.2 Introduction

As mines increasingly get deeper, operational problems arise due to high stress regimes and in some instances because of ventilation issues as in the deep South African gold mines. Mine safety increasingly becomes a top priority requiring continuous seismic monitoring of the work areas. These problems impact on the basic mine production bench marks namely, productivity, product quality, cost minimization and human safety. Financial risks increase due to potential loss of reserves in abandoned stopes and the need to leave ore pillars for support. In such instances, a total quality improvement of the production system is essential to stay competitive. A total quality control approach is a global strategy that considers all the bench-marks and their influencing attributes simultaneously, in arriving at a solution.

Productivity is constrained by the mine layout, the mining method, ground response and the fleet size and its availability. The mine layout reflects the flexibility of an operation which directly affects fleet utilization and in some cases the available mineable reserves. Mine design and development are based on estimated parameters such as the rock strength and in-situ grades determined during the medium and long term planning. The limited information used at this functional planning stage poses risks for the production phase. There are many examples of mines that failed due to ground control, metallurgy, or reserve estimation.

In this chapter, we investigate the relationships between the variables that impact on production planning. The objective is to determine a way to reconciliate these variables such that a producing mine can successfully respond to or work around the dynamic and stochastic problems that evolve during the course of mining. It may be possible to use some empirical relationships to set the production goals within specific mining environments.

3.3 Grade Estimation and Stope Design Issues

A stope with variable walls implies an uncertainty and variability of the limits. The error in the stope limits implies greater dilution or ore losses. It is therefore prudent for the production mine planner to be aware of the inherent errors in the reserve estimates and to seek to improve these estimates.

The underground stopes are designed on the basis of a reserve evaluation model and geomechanical assessment. The reserve model input is the field samples. The sample values have errors due to core logging, surveying of their position in the ground, the precision of analysis and sample contamination. This data is used to develop a reserve model using any one of the several reserve estimators such as polygon, triangular, inverse distance and geostatistical methods.

The interpolation of ore grades between the sample points remains highly subjective. This is especially true if the continuity of high grade or waste lenses is less than the sampling grid which result in an over-estimation and under-estimation respectively. Therefore, the sampling grid and number of samples determine the reliability in a reserve estimation. Also, a large disparity exists between the sample size and its support of in-situ material.

For example, one tonne of core sample can represent an in-situ tonnage in excess of ten thousand tonnes. The decision to extract a block of material as ore or waste is based on an estimated value [Figure 3.1]. This estimate may be inaccurate, leading to a situation whereby blocks with actual grades are above cut-off are declared waste and vice versa, some waste is mined mistakenly as ore. The impact of this phenomenon, termed an information effect, is to dilute the in-situ reserves as well as reduce reserves. If the information increases, the quantities of ore losses and dilution decrease. Due to budgetary constraints, the information effects cannot be eliminated, but only reduced by better grade control sampling. At the production planning stage, the information effect has to be quantified for each area about to be scheduled.

The reserve estimators are mostly limited to the estimation error in grade of a given volume. But, there are errors in the volume because the definition of ore limits is ill-defined due to sparse data. Estimation of volumes by traditional methods such as sections and polygon methods causes overestimation when the areas vary significantly from

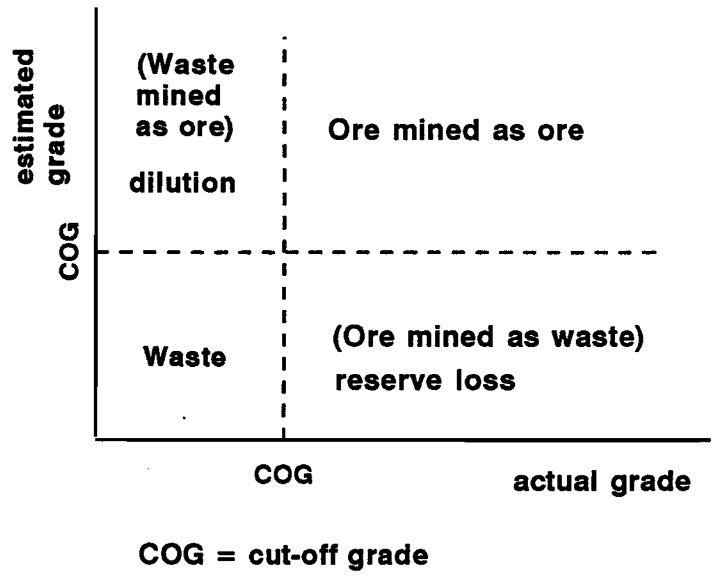


Figure 3.1 Information effect on the classification of material destinations [adapted from Lane [1988]]

section to section. This is due to replacement of oblique contacts by straight line interpolation.

Kriging weights sample values in a way that minimizes the errors of estimation of grades of deposits or mining blocks. However, some kriging methods such as lognormal kriging can generate errors in the estimation of logarithmic variograms. This leads to incorrect ranges of influence which negatively impact the stope limits delineation. Another problem with kriging arises when the estimation block size used is too small compared to the range of influence [Vallée et al, 1992]. As the block size decrease, the estimation grades become less and less related to the sampling information hence physically meaningless. The variance of the grade estimates of these blocks increases. Therefore, it is important to compare estimated grades to the observed mining grades and attempt to rationalize the cause of major differences between the two. This reconciliation process is fundamental to a production planning stage as it filters the sources of errors and allows updates of the reserve estimates.

A comparative back analysis of the various estimation methods on a mined out bench by Bell and Whateley [1994] indicates clear cut zones of high concentrations with or without gradational transitions. In such situations, a strong correlation known as proportional effect exists between the local mean grade and its variance. In a producing mine, the proportional effect can be applied to improve the reliability in the planned grade estimates above those obtained at global estimation. This is achieved by determining the local mean grades and their variances for stope profiles in a moving average fashion. The trends of the two variables are generally similar, since a change in variability reflects a change in the mean. Therefore, a comparison of the running grade standard deviations and the means highlights the areas of high variance and poor accuracy of the mean within the stope profile. These areas coincide with transition from one ore type to another or the mixing of two sample populations.

The daily ore draw is based on local reserves, and not on the average grade of global blocks used in the stope design. The scheduler has to delineate local ore lodes within the stopes, determine their anisotropy, then progressively monitor their depletion or motion towards the draw-points. This approach incorporates both the grade and spatial distribution into production planning.

An environment variable is defined here as the factor evaluation of the reliability on stope design decisions and is illustrated in Figure 3.2. This variable indicates that sample information size in one geological structure is not necessarily adequate in another, in which case the confidence levels on the designed stope layouts are different. The environment variable can also be related to rock types in stope walls as each has

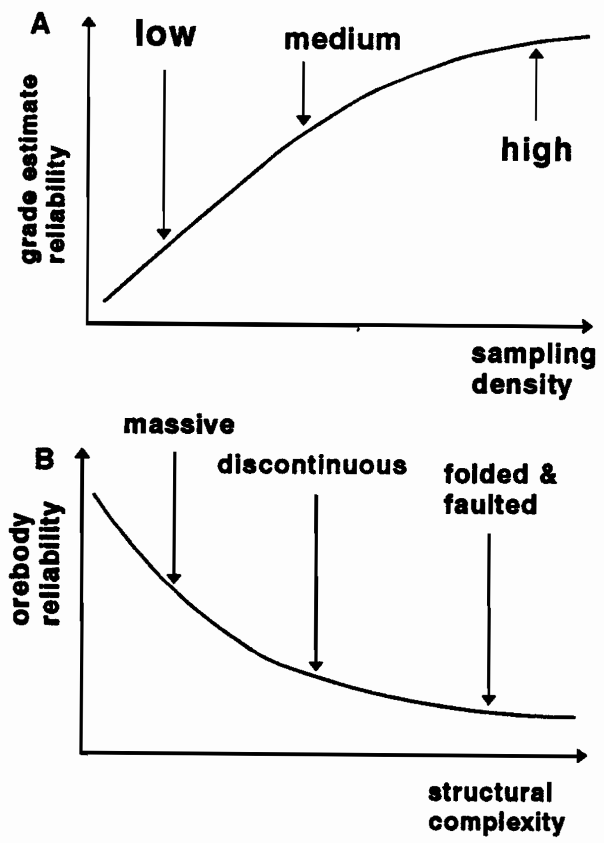


Figure 3.2 Environmental variables effects:
A = Sampling density on same structure
B = Equal sampling on different structures

particular rockmass characteristics, e.g. hardness and strength, which dictate different design approaches. This analysis of information is usually over-looked in mine design.

3.4 Grade Control Issues

Grade control is usually concerned with either

1. keeping the mean grade of the ore delivered to the mill above some required minimum value or
2. reducing the fluctuations in the mill feed grades so as to keep the average grade between some desired limits.

Suppose two stopes, A and B, where A is estimated to have a mean grade of 4% and B of 6%. If equal tonnages are delivered to the mill from A and B it is most unlikely that the mean mill grade will remain constant at 5%. One cause for variation is that A and B's grades are only estimates, and are not in fact the true ones. The possible deviation is measured by the estimation or kriging variance, σ^2_K derived from the use of the geostatistical methods. The kriging variance is the mean squared error of estimation of a variable in the deposit. The greater the sample population the smaller is the kriging variance. However, the magnitude of grade fluctuations is controlled by the variance of the grade distribution. Krige's relationship of variance of distribution of the mean grade of v-sized mining blocks in V-sized estimated blocks is given by [Royle, 1988]:

$$\sigma^2(\frac{v}{V}) = \sigma^2(\frac{s}{V}) - \sigma^2(\frac{s}{v}) \tag{3.1}$$

where s is the sample. This can be expressed in terms of the height and length dimensions of the estimated two dimensional blocks and the mining units, H and L and; h and l respectively as:

$$\sigma^2(\frac{v}{V}) = C.(F(\frac{H}{a},\frac{L}{a}) - F(\frac{h}{a},\frac{l}{a})) \tag{3.2}$$

where a is the range of influence or continuity, C is the variogram sill value and

F(H/a,L/a) and F(h/a, l/a) are functions of the block and mining panel size respectively, and their values are tabulated in geostatistical tables.

The variance in the grade of shift's total production, σ^2_F, is the sum of the kriging and the distribution variances, i.e.

$$\sigma^2_F = \sigma^2_K + \sigma^2(\frac{v}{V}) \tag{3.3}$$

The kriging variance is 'man-made' in that it measures the errors in the grade evaluation process caused by constrained budgets and assaying discrepancies. More sampling and better control of assaying reduces the kriging variance. The distribution variance, on the other hand, is natural and hence, strategies are required to deal with it.

In most instances the distribution variance is considerably higher than the kriging variance and it is the one that contributes more to daily grade variations. A strategy to reduce the distribution variance is to increase the system's flexibility by increasing the number of working stopes. That is, if N stopes are simultaneously mined, the expected variance of the fluctuations is reduced to σ^2_F/N. This has a direct impact on scheduling as it requires more supervision and probably greater ventilation requirement as many areas are concurrently active. However, a dispatching system can effectively reduce supervision and to some extent ventilation requirements through automated ventilation tied to the number of machines allocated to each area.

3.5 Stope Dilution Issues

Stope dilution is material below the marginal grade that is extracted along with the ore. It occurs in three different ways:

1. as planned dilution which occurs at the stope design and is influenced by the mining method and the information available in the definition of mineralized limits. The effects of a generally limited information is negative, because it leads

to excessive extrapolation of sample influences and thus, a poor mineralization delineation.

2. as internal waste due to errors in the geologic modelling of the mineralization. It is also influenced by the sampling density that may result in lodes of waste and/or micro-structures being missed by the definition drilling. The sampling of blastholes identifies these waste lenses and calls for re-evaluation of the initial stope grade estimates.

3. through active mining due to deviations from the mine plan, for example, blasting over-break and ground response to mining that may lead to wall and/or back collapse.

In general, grade control strategies either attempt to stabilize grade over the mine life or more commonly, to extract higher grade ore in the early project life. In the short term, grade fluctuations should be minimized because they cause mill recovery losses with serious revenue implications. In poly-metallic deposits and/or where undesirable elements exist, the strategy may be to stabilize the blends of various ores originating from the different work areas. In order to achieve this objective, it is necessary to determine the wall rock behaviour during mining, the extent of over-blasting, grade variability and spatial distribution at the mining stage, such that appropriate steps are taken in the blending schedules.

The costs of dilution include treatment of lower grade material and handling of waste as well as the opportunity cost incurred through lost ore production. The in-situ grades are only estimates of the unknown actual values. The former grades may be inaccurate hence their use in dilution definition introduces an additional error. The causes of dilution must be known before dilution can be calculated. The limited nature of information in reserve estimation implicitly introduces an error in the planned dilution. A stope design is based on some orebody morphology and as long as the ore limits are unknown, any associated design embodies an inherent error. In order to minimize this error, the mine planner has to conciliate the information base (sample size), the structural geology and stope designs and establish the relationship between these factors. That relationship indicates the reliability in the mining grades. This integrated process of establishing the field relationships of various parameters is superior to the traditional use of planned dilution that considers only the external waste designed into the stope due to the limitation of the mining method.

Planned dilution among different stoping methods commonly ranges between 5 to 30 percent. It is observed that for production planning purposes, planned dilution cannot be uniformly applied to all the stope material. Ore in the centre of a stope is likely to be free of planned dilution compared to ore at the stope margins. Therefore, draw-points must be weighted with respect to their position relative to the ore boundaries. These weights would prove useful in grade control. It is therefore essential at production stage, to control the mining induced dilution and also to know exactly the contributions of the other two diluting processes. This information must be used in assessing the reliability in the mining grades of all the stopes in a mine.

3.5.1 Cumulative Stope Production

A relationship has been established between the age of a mine and dilution [HBM & S in Scoble and Moss, 1994] that indicates a dilution increase during development and the final phases, and a decline during the prime years. The final phase is characterized by pillar recovery and mining in tertiary stopes within backfill sequences. An equivalent relationship can be established between a large open stope as well as block-caving stope in any mining phase and the stope's age as measured by the cumulative drawn tonnage used in production planning. The quarterly production tonnage at Henderson Mine (a block caving operation) was calculated based on the height of column to be drawn [deWolfe, 1981]. The corresponding grade for the drawn height is adjusted for material mixing during draw. The difference between estimated and mined grades plotted against the draw column height indicates negative deviations after 50 percent of the column has been drawn [Figure 3.3].

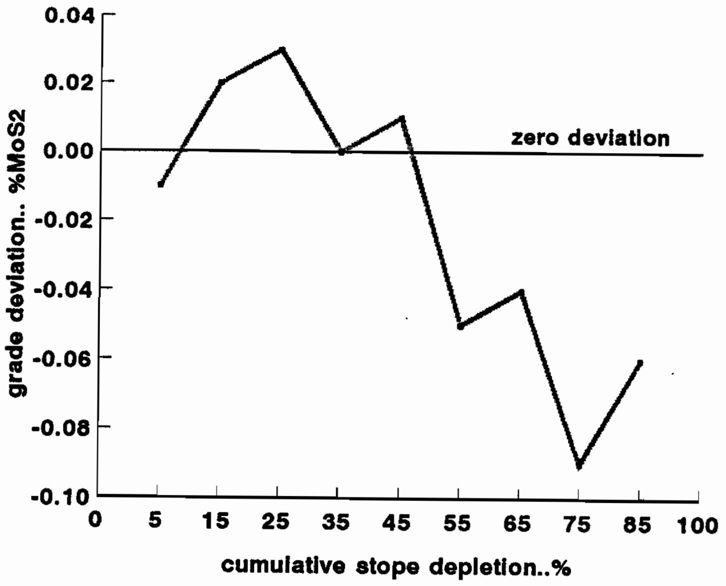


Figure 3.3 Comparison of actual to estimated grade with cumulative stope draw [after deWolfe, 1981]

3.5.2 Orebody Structure

Wall dilution of lenses of mineralization in open-pit mining is related to the perimeter-to-area ratio and the angularity of the ore lens [Tsomondo, 1994]. Angularity affects selective mining in both blasting and mucking operations as material tends to be mixed more in irregularly shaped ore lenses. A similar relationship is expected in underground stopes if the geology is complex.

In the Brunswick Mining and Smelting No. 12 Mine, the mineralization shows zonal anisotropy. The ore characteristics change with depth and show strong banding of galena and sphalerite inter-layered with thick pyrite horizons [Grebenc and Welwood, 1971]. The stopes are defined on strike by diamond drilling on a minimum interval of 23 m. The stopes are 38 m long, and separated by 38 m thick pillars arranged in an alternating sequence along strike. The stope heights vary between 92 m and 153 m. The stope definition in this mine has a minimum interpolation of ore margins of 23 m (distance between two drill holes). The variable stope heights indicate that the stopes are affected to different extends by the surrounding wall rock or backfill. This example serves to indicate the effects of the deposit geology, geometry and stope design parameters on expected stope grades.

3.5.3 Number of Diluting Walls

If a stope is in contact with one or more backfill walls, dilution is expected to be equal or greater than in the case of a primary stope. As the stope depletes, high walls are exposed and stresses increase, possibly causing local failures that increase dilution. Numerical modelling techniques can be used to indicate the progressive stress re-distribution and its potential impact on ore dilution. At the Carolusburg Open Stope Mine, South Africa, ore is kept in the stope, drawing only enough for the next blast slice to be made until the whole stope is blasted [Ross-Watt, 1989]. When the whole stope is fully blasted, the extraction rate is maximized until the stope is empty. This is a way of reducing the ground stress related dilution that would ensue if the stopes were left empty between production slices. Studies at El Salvador Copper Mine in Chile [Marko and Gregorio, 1981] showed that dilution is directly proportional to the number of diluting (waste) faces [Figure 3.4].

Deposits are rarely homogeneous even within massive types. Lodes of poor mineralization constitute internal waste. The lodes may have slight density or hardness differences. Shear zones and faults previously missed by definition drilling may be present. The host rock characteristics may vary from one section to the next, causing problems such as different grindability or mineral recoveries. This host of factors is a reflection of the potential problems that arise at production through use of inadequate sampling density. Unfortunately, the amount of sampling that can be achieved is limited, hence a production plan will always experience these grade variation problems.

Other parameters that affect grade control include the stope sizes, the wall rock or backfill strength and wall over-blasting during production. The bigger the stopes in plan,

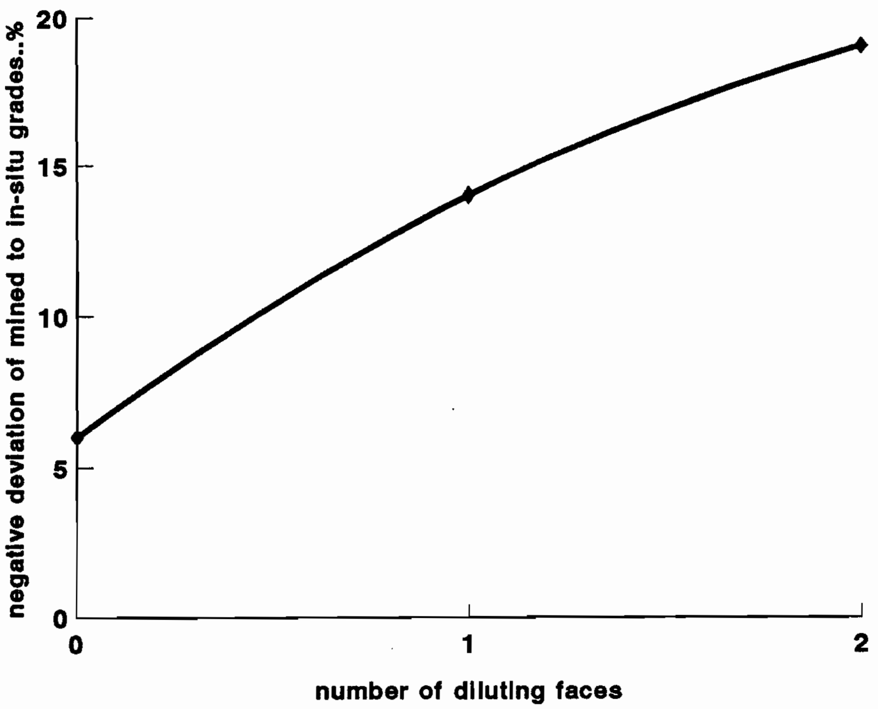


Figure 3.4 Effect of diluting faces on in-situ grades in a backfill copper mine [after Marko & Gregorio, 1981]

the more likely it is that ore is less diluted, all other factors remaining constant. The mechanization of underground mining has lead to increased bulk mining sequences and higher productivity, but the large open stopes cause greater ground control problems requiring the use of backfill support. The use of backfill in mining sequences represents a risk in that the fill may fail prior to the full extraction of adjacent stopes, thus resulting in lost reserves [Pelley, 1994]. Backfill dilution occurs in two ways:

1. over-break into secondary stopes of primary backfill. This is often designed into the system though the problem is accuracy of these estimates and,
2. the failure of free standing fill faces can result in high dilution levels.

Pelley [1994] comments on the effects of sequencing on backfill dilution. At the David Bell Mine, for example, a saw-toothed mining sequence is implemented in which only 40% of the ore is mined from primary stopes, with the remainder being drawn from stopes exposed to two backfill walls. In a 1-5-9 sequence, which follows the numbering as in a checker board, the situation allows 50% of the ore from primary stopes and 50% from stopes with exposed backfill walls to be extracted. In the case where tertiary extraction exists, the ore could be highly diluted due to the number of diluting walls. In cut and fill mining, equipment digging below grade into unconsolidated rockfill or consolidated hydraulic fill cause dilution.

The dilution effect of different mining sequences on mined grades is reported at the El Salvador Mine, where three sequencing modes are investigated. Results are shown in Figure 3.5 indicate that the checkerboard mining sequence has the greatest proportion of dilution compared to the other two. This is due to the greater number of diluting faces (i.e. two to four) exposed to the active stope during the extraction of secondary stopes. In the concentric expansion mine sequencing, the active stope is exposed to one or two backfill faces only. The panel system has only one diluting (backfill) face.

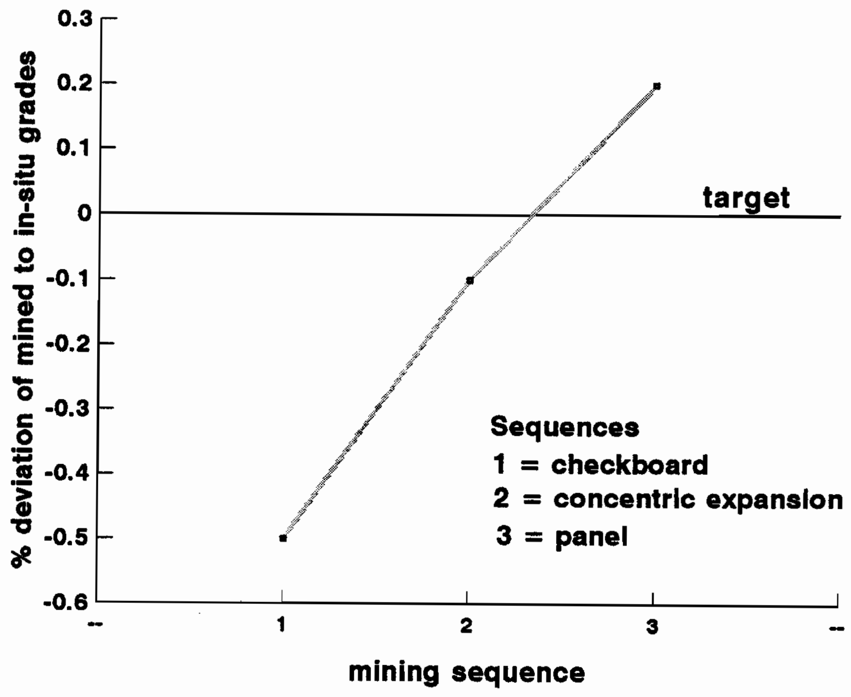


Figure 3.5 Effects of mine sequencing strategy on stope dilution [after Marko and Gregorio, 1981]

3.6 Ore Bin Draw Policy

An ore pass or ore bin draw policy is a production strategy that stipulates the minimum allowed material to remain in the facility at any one time. This policy is practiced mostly in underground mines with ground control problems which cause significant deterioration of the ore-passes if they are kept empty. The effect of the policy on daily production is that it imposes the upper limit of material that can either be dumped or drawn from the pass. A daily schedule must therefore be drawn under this policy constraint. It is also important that a production monitoring system be available to accurately indicate the quantities that can be drawn or dumped at each facility. Two examples are cited where this policy is in operation. At the Bell Asbestos Mine in Quebec the two surge bins feeding the ore hoisting system cannot be drawn more than 50 percent because this would induce ground control problems in the bins. At the Kidd Creek Mine, the policy is to ensure that the surge bin capacities are large to allow independent operation of many various underground systems. Typically, the concrete lined haulage ways are periodically shut down for re-building. This requires adequate time for the concrete to cure after placement, and during this period, the mill is supplied entirely from the ore in the surge bins [Belford, 1981].

3.7 Draw-point Issues

Draw-point availability is a production phase issue that can adversely affect mine productivity. It can be attributed to any of the following factors:

1. poor design, especially if the brows are non-planar,
2. heavy ground, causing brow failure, and
3. poor rock fragmentation in which very coarse material causes hang-ups. The blasting of the boulders weakens the draw-point rock mass. If the material is very fine and/or sticky, it clogs the draw-point.

In heavy ground, a draw-point brow stabilizing crew is required to work ahead of production, installing anchor bolts and concrete support, and rehabilitating failed ones. Production in this case is dependent on the available working draw-points. The loss of some scheduled draw-points may lead to a non-optimal blend obtained from the remaining available draw-points. Therefore, an alternate plan must be determined from the remaining stopes or draw-points.

Rockmass characterization at the San Manuel Mine, Arizona indicated the presence of distinct orebody sections belonging to different rock strength classes based on a rock mass rating procedures [Sandbak, 1987]. However, these different zones exhibited the same fragmentation size [Figure 3.6]. This indicates that the use of empirical rules such as rockmass rating (RMR) does not fully account for the field response of a rock mass. The actual behaviour is vague and there tends to be overlaps between RMR classes at the field value. Therefore, the impact of draw-point failure or hold-up on production can be large when the designs based on laboratory and/or drill core samples exhibit different properties.

In cold climates, some mines using block caving or sub-level blasthole stoping sometimes limit the number of open draw-points to avoid a loss of air pressure underground. Low pressure would cause a downward intake of cold surface air through the broken ore, thus causing freezing and hang-ups. An example of this is the Kidd Creek Mine where, despite the high rate of mining requiring many draw-points, only a few are kept open so as to concentrate the air pressure in the active draw-points [Belford, 1981]. Production scheduling has to comply with this constraint.

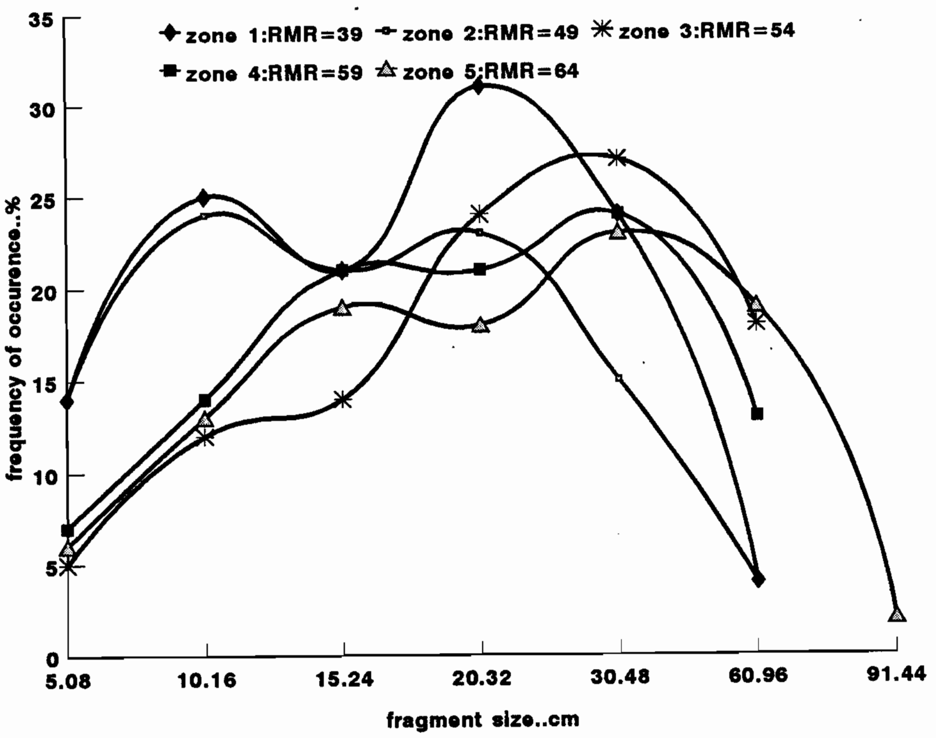


Figure 3.6 Fragmentation differences within the San Manuel Mine, Arizona [after Sandbak, 1988]

3.8 Equipment Policy

Underground materials handling equipment is diverse, and includes belt conveyors, locomotives, slushers, loader/truck systems and load-haul-dump units. The number of operating units have been increasing over the years as more and more mines become mechanized or upgraded to achieve their increased production targets. The objective of a materials handling system is to minimize production cost per tonne. Some mines never achieve this objective because of site operating conditions. The large number of machines at a mine requires an efficient scheduling system that maximizes utilization. The author conducted a survey in which typical examples of trackless fleet sizes and mix from the Canadian mining industry are shown in Table 3.1. The table shows the diversity in equipment make, type and numbers in these mines. The haulage distances vary from 50 m to over 1000 m, with the longer distances exclusively run by trucks. These trackless equipment are the production backbone of all sizes of the Canadian metal mines.

Table 3.1 Trackless haulage in some Canadian Mines

Company, Mine	Production fleet	Tonnes per year	Haulage distance in metres
Brunswick Mining & Smelting, No. 12	10 ST8A, 13 ST8B, 10 Toro 500CD, 5 smaller LHDs & 20 30t-trucks	3 921 000	400
Cominco's Sullivan	19 LHDs & 3 JDT426 & belt	1 000 000	225-480
Corona Corp, Jolu	5 JS & 3 JDT trucks	176 713	730-2700
Falconbridge, Fraser	20 (2-9 cu yd)	957 800	--
Inco, Crean Hill	9 Wagner EST8A & ST8A	695 400	240
Inco, Copper Cliff South	Wagner ST8A, ST8B, ST8AE, ST6, JC JS600	1 500 000	200
Inco, Stobie	22 Wagner EST8A, ST8A &1 MTT 416 , 1 JCI-600	3 500 000	130
Inco, McCreedy West	3 Wagner ST8B, 2 ST8A, 1 ST5 & 4 trucks	690 000	260
American Barrick, Holt McDermott	- Wagner ST2s (diesel & electric)	200 000	50
Aur Resources, Dumont	6 JC JS220	70 000	960
Cambior, Grevet	4 Wagner ST3	840 000	100
Placer Dome, Detour Lake	6 Wagner ST5, 1 ST6, 3 ST3.5, 1 ST8 & 3 trucks	1 145 000	500
Noranda, Gaspé	3 ST8B & 1 30t-Volvo truck	956 000	200

The mucking and haulage costs as a proportion of total mining costs (drilling, blasting, mucking, haulage, and support systems) is relatively high in some mines as illustrated in Table 3.2. The high costs justify the need for more cost effective methods of materials handling and better control of the activity. Such improvements may be realized through equipment monitoring throughout the shift to avoid idle machines, and a more efficient dispatching method. All the mines listed in Table 3.2 use fixed shift allocation of the haulage equipment for different working sections.

Table 3.2 Typical materials handling cost to total mining costs [Canadian Mining Source Book, 1995]

Company. Mine	Mining method	(Haulage cost)/(Total mining cost) ..%
Cominco Ltd, Sullivan	slot & shell	45
Nanisivik Mines	room & pillar	29
Dickenson Mines, AW White	cut & fill	31
Hudson Bay, Ruttan	blasthole	40
Lac Minerals Ltd, Bousquet # 1	blasthole /fill	13
American Barrick, Holt McDermott	longhole	32
Niobec	open stoping	65
Teck-Corona, David Bell	longhole	14
Falconbridge, Kidd Creek	open sub-level	15
Placer Dome, Detour Lake	cut & fill	30

Equipment selection is a strategic planning function performed at mine design stage such that the drift sizes, ventilation requirements and power locations can be determined. The operational planning has its own objectives subsidiary to those at the strategic stage. Typically, the problem of utilization and availability become explicitly apparent only at plan implementation. The shortcomings of the strategic plan could be caused by the mine design; for example, a mine with access problems between sub-levels and levels results in captive equipment. The relocation from one stope to another is difficult such that in one area there may be excess equipment whilst in another there is a shortage. In such instances, production is increased in the work areas with no machine breakdowns. This strategy can be counter-productive to a quality objective, as the scheduled blend is not achieved.

At the North Mine, Sudbury, the relocation of an ST8 Wagner machine from one level to another is reported to take more than twenty-five days. The equipment schedule associated with various work areas is such that crucial stopes are assigned standby machines in case of a breakdown. The limited vehicle space in these stopes prevents the working of multiple machines and hence, the equipment utilization is low.

Some mines establish their equipment populations based on the peak demands rather than long term schedules. This is an expensive proposition as it results in premature capital expenditures. A benefit of having excess equipment on site is realized by allowing the personnel to gain experience on the equipment before it is used for production. The required number of equipment can always be met as fleet utilization if far lower than availability. Such a policy has been reported at the Kidd Creek Mine, where the haulage fleet is sized to operate at 67% availability, with one third in maintenance [Belford, 1981].

Some operations put their equipment on a tight maintenance schedule. The daily plan has to accommodate that schedule as well as be responsive to unplanned breakdowns. In complex equipment such as LHD machines, the failure rate is constant and is defined by a negative exponential distribution [Jardine, 1973]. Failures occur when any one of a number of independent constituent components fails. Paraszczak and Perreault [1994] and Knights [1994] describe studies on the reliability of LHD machines. The reliability of machines declines with age as well as with a time increase between services. This observation is important in production scheduling because the planner has to distribute the fleet such that work areas with high priority receive more reliable machines and/or ensure an equipment mix of both high and low reliability machines working together.

This strategy would minimize the impact of the poor machines' breakdowns. It does not account for the random breakdowns which may affect reliable machines.

To achieve timely responsive decisions requires a total control of equipment status: i.e. the location, whether working, idle or in service. A communication system linking the machines to the decision maker will assist in the implementation of production control. Such production control would additionally aim at meeting all the other objectives discussed above so as to have a safe and low production cost mine.

3.9 Production Flexibility

The production issues discussed above can exist within a single operation resulting in a more complex production system. The decision making in these circumstances requires tools to deal with uncertainties in product, capacity and the environment. Studies in manufacturing systems indicate that firms take either defensive actions to adjust to the uncertainties or proactively control the uncertainty through system flexibility [Gervin, 1993]. Variations in the coefficients of the production function or market price require greater investment in the ability to vary input proportions. As the uncertainty increases, the process flexibility has to correspondingly increase if it has to continue to cope with the problem.

In an underground mining environment, management has to consider flexibility as an adaptive response to both internal and external factors. At the strategic planning stage, the external factors involved are the global trends towards low grade open-pit mines, the relatively low commodity prices and consistent commodity substitution by plastics and ceramics. The strategic response in this respect is to design more flexible underground mine layouts. The basic design has to be one that allows haulage machine flexibility. This is achieved through an inter-connected mine haulage system design with appropriate equipment selection, for example trackless equipment is more flexible than rail or conveyors. This process amounts to a proactive response to the external factors.

At the operational stage, the internal factors that influence production capacity, product quality and local constraints are of concern. Several types of flexibility can be identified at this stage. The equipment flexibility in load-haul-dump systems is high because each unit can handle any job within the system over short to medium haulage distances. As the distance increases (change in environment) loader and truck systems become more suitable. Both systems can move between work areas with minimum time response and effort, provided the mine layout is suitable for such moves. Routing flexibility, which is defined as the ability to send equipment to a server to receive service with a minimum waiting time is necessary. This flexibility is influenced by the number of machines in the system and the connectivity of the mine layout. The fewer the machines working in an area, the higher the routing flexibility for a fixed layout as the servers are under-utilized, i.e. servers have no backlog queues. Therefore, if the planner wants to maximize equipment utilization, it is important that the scheduling process explicitly constrain the number of machines working in a given area.

Ideally, a shift implementation is to be performed without external intervention. The conventional scheduling of fixed mode dispatch is based on this concept. But, unanticipated events occur that disrupt the smooth execution of the shift plan. Therefore, a form of self-control flexibility is essential, which is defined as the ability of the production system to operate for a long time without external intervention. This can be achieved through high routing flexibility and the ability of the system to mine from many areas of materials of varied properties. The later aspect implies the ability of the production process to respond to changes in product quality.

The possibility of increasing production above the scheduled amounts in some areas is typical of captive equipment operations when either draw-point blockages or machine breakdowns occur. This represents a volume flexibility of the system: if there are few dump-points, such a flexibility is likely absent as it results in traffic congestion, long waiting times and correspondingly low production despite an increase in working machines per given area. The volume flexibility is therefore both dependent on the mine design and short term shift scheduling which may create practically infeasible allocations.

In manufacturing systems, two types of operations exist, namely a high volume/low variety system and a low volume/high variety system [Stecke and Raman, 1995]. A review of these operations indicates that the high volume/low variety type closely resembles mining operations. In underground mining there are generally few products, i.e. ore and waste. If product quality is an objective, then an ore source flexibility is required. Ore is an upstream input to a process plant and its characteristics poses processing uncertainty. Therefore, plants usually set a range of acceptable properties within which the process plant can easily adjust and/or cope with. Temporal aspects reflect the length of time it takes to make adjustments to an affected material input.

The key to improving material flexibility is to have many work areas that allow for product blending and compensation when the estimated quality in some area tends to differ significantly from the rest of mined material. The simultaneous working of several areas shortens the temporal requirement to put the process back into control and also reduces the pressure to eliminate product quality problems. Despite the need for several work areas, the number of operating stopes remains low in most operating mines because of the high cost of exhaustive mine development. It is also common to set lower limits on the amount of material that can be scheduled for production from any one stope, a constraint necessary for minimizing supervision cost and excessive equipment movement during the shift.

The high volume/low variety operations require efficient production of few material types according to shift demand requirements. They follow a dedicated mode of operation by ensuring line balancing of material flows between the source and the destinations. This strategy leads to the optimal schedule implementation. Continuous monitoring of quality problems and breakdowns is essential to ensure minimal deviations from targeted production levels. The use of active dispatching has a greater flexibility than the traditional dedicated mode, through rerouting in the event of unanticipated breakdowns and reneging from slow servers. The dedicated mode does not have the control policies to provide alternative actions to follow in such situations except to shut down the area in which a problem has developed. The dedicated mode, therefore, has a low self-control flexibility compared to the active dispatch mode. Another advantage of a highly flexible re-routing procedure is in the reduction of pressure on the maintenance crew on eliminating breakdowns.

3.10 New Design and Operating Criteria

The factors that affect the input and output of a shift production schedule are many and varied. Despite the level of planning (i.e. detailed), there still remain many uncertainties. Two sources of uncertainties exist, namely:

1. those associated with the assumptions made at mine design, e.g. as geological interpretation, sample errors, and estimation errors. These uncertainties can be reduced by systematic reconciliation of mine production figures through which new information is gathered to explain the observed trends. Due care is necessary in the interpretation of the reconciled data as this information is an aggregate of all mining influencing parameters. Dilution has been singled out as one complex mining parameter. These uncertainties are long run and this feature allows their partial solution through the use of fuzzy logic and/or knowledge base techniques.

2. those associated with the environment at the time of schedule implementation, e.g. quality variability, fleet availability and system component problems (blockages and ground control problems for instance). These uncertainties may be successfully represented by statistical data such as machine availability and time between machine failures. However, other variables such as utilization, are directly affected by the current and dynamic conditions in the system. This

involves unanticipated events and as such, requires a system that is reactive to counter their negative effects on the planned.

This thesis postulates a rationale for dealing with the production problems for underground mining as follows:

3.10.1 Design Criteria

An access ramp system is an efficient method of access within different mine levels. This allows a mine to have leaner fleets and greater flexibility in the face of operational uncertainties. Shafts are suited for less mechanized operations where labour can easily be transferred from level to level as well as for fast materials handling to surface. Such ramp systems can be developed in both old and new mines. In old mines, the ramp system needs only be internal and developed in the vicinity of the defined and possible reserves.

Quality control is becoming an issue and mines must be designed to handle quality fluctuations. This is achieved through increasing the number of simultaneous work areas. In mines operating with backfill sequencing, it becomes important to have stopes in all phases of the mining sequence i.e. primary, secondary and tertiary. This is essential to allow the blending of materials from less diluted primary and the usually more diluted tertiary stopes. The operation of many stopes minimizes the variance of fluctuations of the run of mine grade(s).

Equipment selection is important as it directly impacts on the possible system flexibility. For metal mines we suggest the use of diesel-powered LHDs as these machines can be moved from level to level much more easily compared to their electric counter-parts. As the haul distances increase, a truck and loader system is recommended because the trucks become more efficient from an economic point of view. Traffic control systems are required to ensure safety, collision avoidance and minimization of congestion on the haulage ways.

3.10.2 Operating Criteria

Operating experience gained through working in a particular environment should be stored in a retrievable and upgradeable form. Such a format allows the building of an evolving database that is useable as a predictive tool, in the dynamic environment of short-term planning. It is essential to collect the data about all facets of the operation and analyze them in totality because most of the variables are inter-dependent. The following activities constitute some key elements that require monitoring and control at production planning:

- blast design layout, assessed with respect to geological structure and stope limits,
- production drilling monitoring to identify hole deviations from the planned,
- assessment of blast damage, e.g. by remote sensing techniques,
- ground control response monitoring before and during ore draw using sensors and
- keeping track of information on the spatial grade distributions.

For instance, the sampling intensity within a mine is not necessarily the same everywhere, being influenced by the complexity of the geological structure. A simple massive structure requires far less drilling compared to a folded and/or faulted orebody. An assessment of such information at the production phase facilitates both a qualitative and quantitative explanation of observed ore streams. The use of rock mass rating to characterize the in-situ rock mass has its limitations which are apparent at the production phase. However, if a re-evaluation of a stope design is done during the mining stage, it may be necessary to update the support requirements. The result may be a reduction in dilution because of an improved support system (e.g. more cable bolting of the back), or a reduction in the support requirements leading to low support cost. If appropriately used, the data gathered through mining provides a greater insight into the problem and allows a sound rationale for setting targets, especially with respect to quality.

Mining decisions are made on sample estimates and such data is never enough, being limited by exploration budgets. This makes it essential to sample all production drilling so as to increase the sample representation of a particular stope. Equally, the sample data has to be analyzed in a de-aggregate fashion to better represent the mining unit size. This approach creates a series of stratified ore inventories within the respective stopes. The planner is then able to systematically describe the factors that are likely to affect each unit and what the consequence will be. With such information, the stage is set for mathematical optimization of shift schedule.

The production phase becomes essentially a continuous activity of obtaining information, reviewing it, identifying trends, and updating the mine and geological models to reflect the realized output.

The shift schedule implementation and control has to be conducted in real time to allow timely response to unanticipated events. The response success is dependent on the various flexibilities of the system, namely routing, material, process and volume. The aspect of flexibility indicates the potential of computerized materials handling facilities that can efficiently and timely route equipment to the right work centres. The same computer system can be used to record material properties and keep track of material flows through the handling facilities. The relevant information can then be passed to the process plant to adjust its operating parameters in response to the anticipated mill feed.

Therefore, the rationale for underground mining reduces to a theoretical concept of "flexible underground mining", analogous to a flexible manufacturing system previously described. In the following chapters, the benefits of such a system are identified and the necessary research to make it a reality is discussed.

Considering everything stipulated in this chapter and the development in later chapters, leads to a strategic and production plan evaluation guideline presented in Table 3.3. The guideline provides a rational decision-support system for both tactical planning and mine design. The use of this guideline requires an initial identification of the key aspects of control. These aspects can be ranked according to management policy. Then, for an operating mine, the available work sites are identified, e.g. working or bench-plans. The information available on each aspect of a work site is obtained and used to identify the (potential) problems. A strategy is then selected to reduce or eliminate the effect of those problems. The implications of the adopted strategies are then identified. This way, the cost and benefits of an operating plan are known. At strategic planning, the same process is adopted and used to upgrade all aspects with unsatisfactory outcomes. For example, a stope design is upgraded by increasing drill sampling intensity, which leads to a better reserve estimation and stope delineation. The impact of improvements in the stope design at production stage is accurate production drilling and better blast control.

Table 3.3 Strategic and production plan evaluation guideline

Aspect consideration	Problems	Strategy	Implication
grade estimation & stope design	- extrapolated information, variable confidence levels, different stope host environment	identify assumptions in estimation, spatial distribution, delineate & monitor local ore lodes depletion, identify proportional effects	grade-tonnage curves inappropriate for scheduling, equal-sized block modelling has problems
grade control	- grade variability & fluctuations	increase sampling & increase active stopes	better quality control, better utilization, more layout flexibility & development
stope dilution & grade control policy	- estimation of true dilution	assess information base, geological complexity & stress re-distributions	evaluate the confidence levels of information bases
	- progressive increase in dilution with mining	concurrent exploitation of different mining phases (sequencing)	flexibility in system, better mix of materials, development expensive
	- overblast effects due to poor definition, backfill-ore interface complex	ore lode angularity impact on design confidence & planned dilution	reconcile stope geometry with sample information and structural geology
	- wall-rock, backfill strength and size of stopes against ground problems	large size implies less perimeter-to-volume ratio, but ground problems likely high	at design stage, have to optimize between the two; e.g. stronger backfill & large size
equipment policy & productivity	- availability and utilization	over-equipping critical areas, or machine monitoring & dispatch	equipment flexibility, compatibility with mining method, preventive maintenance
draw-point policy	- blockages, failures, ventilation problem of cold air intake, etc - production restriction	system to monitor & record problems, re-allocation of LHDs, re-scheduling of targets	computerized monitoring system, dispatch, access flexibility in system
ore-bin draw policy	- imposes upper limit to production plan	increase flexibility through more facilities, constant monitoring of bins	plan is restricted, set priorities to maximize the important goals, extras = more cost
quality and production policy	required quality, productivity, utilization	internal ramp system, diesel LHDs, more different phase work sites, quality dispatch policies	new design concept of underground mines, aim at flexibility of volume, quality & routing, fleet fault diagnostics

3.11 Summary

The factors present at the short-term production planning stage are of two kinds, namely:

- technological: this includes the mine design and mining method, equipment type and its characteristics, and the quality of the evaluation of key aspects of mineralization continuity, grades, rockmass strengths and stope definition.
- managerial: this determines the objectives of the shift schedule which must be met under both the technological factors and mine policies related to safety, draw-point rates, mining sequence, traffic volumes, time, etc.

This chapter has highlighted the potential problems associated with these factors and suggested methods to better refine the information base to adequately cope with a dynamic production system. Examples have been drawn from different sources to show the real world relevance of these issues, which until now, have not been comprehensively described as pivotal to the success of short-term production. New underground mine design and operating criteria have been proposed to mitigate the currently observed short-term production planning problems. The new criteria emphasize the importance of system flexibility with respect to production rates, fixed facilities and product quality.

A strategic and production plan evaluation guideline has been developed as a decision-support system. The guideline is an invaluable component in an integrated mining information and control system as it allows systematic evaluation of key production factors.

Chapter 4
Fuzzy Logic Modelling of Ore Quality

4.1 Purpose

This chapter introduces a fuzzy logic modelling technique for the assessment of uncertainty in mining grades caused by limited planning information, errors in stope delineation and structural complexity of the mineralization. In addition, dynamic factors such as the interaction of the high stope walls with the wall-rock or backfill, ground control response to ore draw and the location of the draw-points are considered. The result is a comprehensive decision support system useful for the shift production planning.

4.2 Introduction

The definition of production goals influences the results of the work shift. The manufacturing and some service industries have traditionally based their production plans on the maximization of productivity and minimum tardiness. However, in the past two decades, this approach has changed because it does not guarantee minimum production cost nor the right product quality. The production plans are now often based on a just-in-time manufacture or service that has quality as the main focus. This approach minimizes costs by reducing: (a) reworking of products, (b) product returns due to unsatisfactory quality, (c) raw material wastage and (d) inventories. The mining industry has still to adopt similar practices.

One of the greatest challenges in the mining industry is implementing product quality focused production planning because of the limitations in the production plan inputs. The diluted mining grades are currently used as the measure of ore quality within a stope or a draw-point, yet these grades are subject to uncertainty caused by both static and dynamic variables, such as the stope geometry and stand-up time of the stope. While in the manufacturing industry the quality of inputs (i.e. parts and materials) are assured through searching of reputable suppliers, the ore quality in a stope is not guaranteed. The quality may vary through one or more of the following factors:

- ore grade and its variability that affects mill recoveries,
- hardness and fragmentation that affects the comminution rates,
- deleterious material in general,
- mineralogy and textural factors, and
- moisture content of the ore that affects the process plant.

These ore quality problems affect the downstream processing operations. The introduction of the International Standard Organization (ISO 9001) requirement on product quality in some mines demands solution to these problems. The objective is to determine each material source's credibility to satisfy a specified ore quality. By ascertaining the specified quality targets, it becomes possible to schedule the available stopes for quality maximization. The following aspects affecting the stope ore quality and their inter-relationships must be established.

(a) The stope sizes within a mine may be variable leading to different ore volume to surface area ratios. Smaller ratios indicate greater ore-waste contacts and therefore increases the possibility of dilution.

(b) The stope outlines are defined by the geology and mining method. Complex mineralization increases the potential dilution because the drill information may be inadequate for such a structure, whilst production blast design is challenging, leading to potential over-breaks.

(c) The number of diluting faces. The strength and stress response of the backfill material has to be assessed.

(d) Stopes deteriorate with stand-up time leading to greater potential for wall and back failures and hence, to more dilution.

(e) The location of draw-points within a stope leads to different proportions of external dilution, i.e. in thick deposits the perimeter draw-points have a greater likelihood of external dilution than the central ones.

(f) The level of stope depletion may be important, especially in vertical crater retreat, sub-level caving and block caving methods as the ore remaining within the stopes progressively gets more contaminated by wall and/or cap waste rocks.

(g) The ore pulling rates and ground control response to mining are dynamic factors that may lead to a decrease or an increase in the external dilution compared to the planned dilution. A consideration of these dynamic factors should enable the update of planned diluted grades to reflect the true conditions being experienced at the production face.

The switch from the productivity methods such as cut-and-fill or shrinkage stoping to bulky non-entry mining methods has reduced the information level from a production face. Therefore, a procedure is required to indirectly derive the same level of confidence on the mined material for use in the scheduling of quality of a work shift production. The procedure has to integrate the variables listed above in obtaining a single *measure* that would complement the statically determined mining grades in quality establishment for each stope or draw-point. That measure is analogous to the 'reputable supplier' in just-in-time manufacturing systems.

The factors listed above on stope product outcome include both qualitative and quantitative variables. It is therefore required to develop a methodology that handles this qualitative information to establish numeric ratings of the stopes. The ratings are then integrated with the quantitative parameters in the mathematical optimization of a work shift schedule. The mathematical optimization is described in chapter 5. This need has lead to the development of a fuzzy logic modelling approach to evaluate multiple fuzzy (vague) input variables of an extracting stope. The developed model is named Fuzzy Logic Stope Model (FLSM) and is the subject of the rest of this chapter.

4.3 Fundamentals of Fuzzy Modelling

Before describing the fuzzy logic modelling of stope dilution, some basic concepts essential for the understanding of the process are outlined.

4.3.1 Membership Function

Fuzzy subsets are sets defined over a crisp domain or universe of discourse where the characteristic function is non-binary as in ordinary mathematics. The term crisp is defined in this thesis to mean clear-cut or well defined. The characteristic function is an ordered pair which indicates the degree or extent that an element of the universe belongs to the fuzzy subset. Suppose the universal set is X and the fuzzy subset is A, then the characteristic function of an element x in A is defined as:

$$(x|A(x)) = [0-1], \ \forall\ x \in X; \ \ A \subset X \tag{4.1}$$

The characteristic or membership function, A(x) allows fractional values between zero and one inclusively. The concept of fractional membership is due to information which cannot be wholly described by a binary system of true or false. Rather, there is a varying degree of truth such that when an element x approaches the situation of full membership in set A, then its value approaches one and conversely, the membership approaches zero if x does not belong to subset A. In modelling the effects of stope dilution, the information about the effects of the influencing parameters is only an approximation, and possibly vague, hence they exhibit the features of a continuous function of membership.

4.3.2 Support

A support of a fuzzy subset A is defined as the set of elements in A whose membership value is greater than zero. The concept of a fuzzy subset support is important in the determination of the active rules in the fuzzy system knowledge base. Rules are defined as active if the interaction of their premises and conclusions are non-zero.

4.3.3 Possibility

Classical mathematics apply conditional probability theory to determine the chances of occurrence of a certain dependent event given that some other event has already occurred. A similar approach exists in fuzzy mathematics which is termed possibility. Possibility is a measure of confirmation of the truth of some fuzzy proposition. For example, if it is a fact that variable A is known to be a linguistic value V, the objective becomes one of confirming that A is some linguistic value W. The possibility is defined as:

$$Poss(W|V) = \max_x(V(x) \wedge W(x)) \qquad (4.2)$$

where x = element of universe X and

V(x) and W(x) = membership values of x in V and W, respectively.

Possibility measures the upper bound of confirmation of the statement that A is W. A linguistic variable or value is non-numeric and comes from natural language phrases such as high grade where 'high' is a linguistic value describing the universe 'grade'. Possibility is applied in the resolution of fuzzy relationships of the knowledge base of a fuzzy model.

4.3.4 Fuzzy Relationship

Many real world relationships involve some imprecision and a degree of membership. Such relationships are best described through fuzzy relationships which allow the capturing of the imprecision in the relationships. A fuzzy relationship is a fuzzy subset over a base set defined by the product space of the constituent subsets. Suppose a collection of sets X_1, X_2, ... X_n have a fuzzy relation R^n, then R^n is defined over the product space $X_1 \bullet X_2 \bullet \ldots \bullet X_n$. The exponent n is the dimension of the relation. Fuzzy relations are at the core of a linguistic fuzzy model's manipulation of imprecise model inputs to generate a solution.

4.3.5 Power

The power of a fuzzy subset A in a finite domain X is defined as the summation of the membership of each element x in subset A. The fuzzy aggregation algorithm of the de-fuzzification process makes use of the power of subsets. The de-fuzzification process is an integral part of obtaining a crisp output from a fuzzy model.

4.3.6 Probability Distribution

A fuzzy model outputs a fuzzy subset of the output universe of discourse. It is therefore essential to determine the probability of each element in the fuzzy output set F, as a process of adding quantitative information to what is ideally a qualitative analysis. The probability of an element i, P_i in a discrete universe of discourse of size n, associated with a centre of gravity de-fuzzification method, is defined by the following basic de-fuzzification distribution transformation [Filev and Yager, 1991]:

$$P_i = \frac{w_i}{\sum_{j=1}^{n} w_j} = \frac{w_i}{Power(F)} \tag{4.4}$$

where w = membership value of element i or j in the output set F.

4.3.7 Fuzziness

Fuzziness is the indication of the distinction between a fuzzy set A and its complement, A´. It is a valuable measure in the quantifying of the resultant inferred fuzzy set of a fuzzy system analysis. The inferred fuzzy set is the one upon which the final crisp output is calculated. If the inferred set has a high fuzziness value, this implies that the resulting crisp value has a large variance because the set is very fuzzy. If the fuzziness is zero then the set is crisp.

Fuzziness is defined as:

$$Fuzz(A) = 1 - \frac{1}{n}(\sum_{i=1}^{n} |A(x) - A'(x)|) \qquad (4.3)$$

where membership, A′(x) = 1 - A(x) and n is the size of the universe, X, over which the subsets are defined.

4.3.8 Modifiers

Natural languages use words such as 'very good' or 'too small' where the words 'very' and 'too' change the meaning of the base terms. Such linguistic values are called modifiers of the basic fuzzy subset. Only two modifiers are applied in the developed model based on the concentration and dilation of the primary linguistic subsets.

If A is a subset of X and α is a non-negative real number, then A is modified into a new subset, B through concentration if the membership function of B is given by:

$$B(x) = (A(x))^{\alpha} \qquad (4.4)$$

where $\alpha > 1$ and $x \in X$.

If $\alpha < 1$ and $x \in X$ then subset B is a dilation of set A.

A concentration reduces the intensity of the resultant subset B's membership function compared to that of subset A because it takes a positive exponent of A(x) where A(x) $\leq$ 1. On the other hand, a dilation increases the membership values of the resultant subset B. The term 'very' is a concentration whilst the modifier 'slightly' infers a dilation.

A more detailed description of fuzzy mathematics and modelling are provided by Yager and Filev [1994], Zimmerman [1993], and Wang and Chang [1980].

4.4 Theory of Stope Ore Quality Fuzzy Logic Model

The basic concept of fuzzy modelling as formulated by Zadeh [1973] requires an approach that provides an effective, though approximate, means of describing a complex and/or ill-defined system that precludes use of classical mathematics. In this work, the problem of ore dilution is identified as one such complex system because of the inter-dependence of several variables, as described in Chapter 3. A fuzzy linguistic model describes the system by means of a set of rules which have vague predicates. These rules are analogous to a system of equations required in classical mathematical models. The model utilizes these rules to generate a decision for a given fuzzily defined input. The fuzzy input sets are analogous to input parameters in a system of equations in a classical mathematical model.

There are many variables that potentially influence ore dilution in a stope. The objective of the model is to determine the interaction of these variables in such a way that the mine scheduler is able to rank the available work areas in the order they are likely to influence product quality. Consequently, the problem is one of Multiple Inputs and a Single Output (MISO) for each work area. The input variables are considered static in the short term, such as daily and weekly production planning. However, the model requires updates in the long term to reflect major changes in the mine environment.

The approach of solving the MISO model requires the fulfilment of four basic conditions, namely:

- An identification of the universes over which all the subsets are defined;
- A definition of the variables and their subsets;
- An existence of fuzzy relationships between the variables and
- An approximate reasoning algorithm for analyzing the relationships.

4.4.1 Knowledge-Base and Approximate Reasoning

The variables and their universes for the stope ore dilution model are described later in this chapter following a formal description of the MISO model. The MISO model developed here is based on prior knowledge of the functioning of a particular system, namely the effects of different variables on ore dilution within a stope. The modelling is solely built on the experts' knowledge hence the database is static. Besides the knowledge base, each fuzzy model requires a reasoning mechanism to interpret the fuzzy relationships for given fuzzy inputs.

A knowledge base is an exhaustive listing of the relations that interact in a specific way to define a system's behaviour. The MISO knowledge base is typically expressed by **IF** (premises) **THEN** (consequent) relations. For a fuzzy system with m relations or rules and n input variables, the knowledge base is formulated as follows [Yager and Filev, 1994]:

{ **IF** U_1 is A_{11} **AND** U_2 is A_{12} **AND** .... **AND** U_n is A_{1n} **THEN** V is D_1
ALSO
IF U_1 is A_{21} **AND** U_2 is A_{22} **AND** .... **AND** U_n is A_{2n} **THEN** V is D_2
ALSO
......
......
ALSO

IF U_1 is A_{m1} **AND** U_2 is A_{m2} **AND** .... **AND** U_n is A_{mn} **THEN** V is D_m}(4.5)

where U_j = input variables

V = single output variable

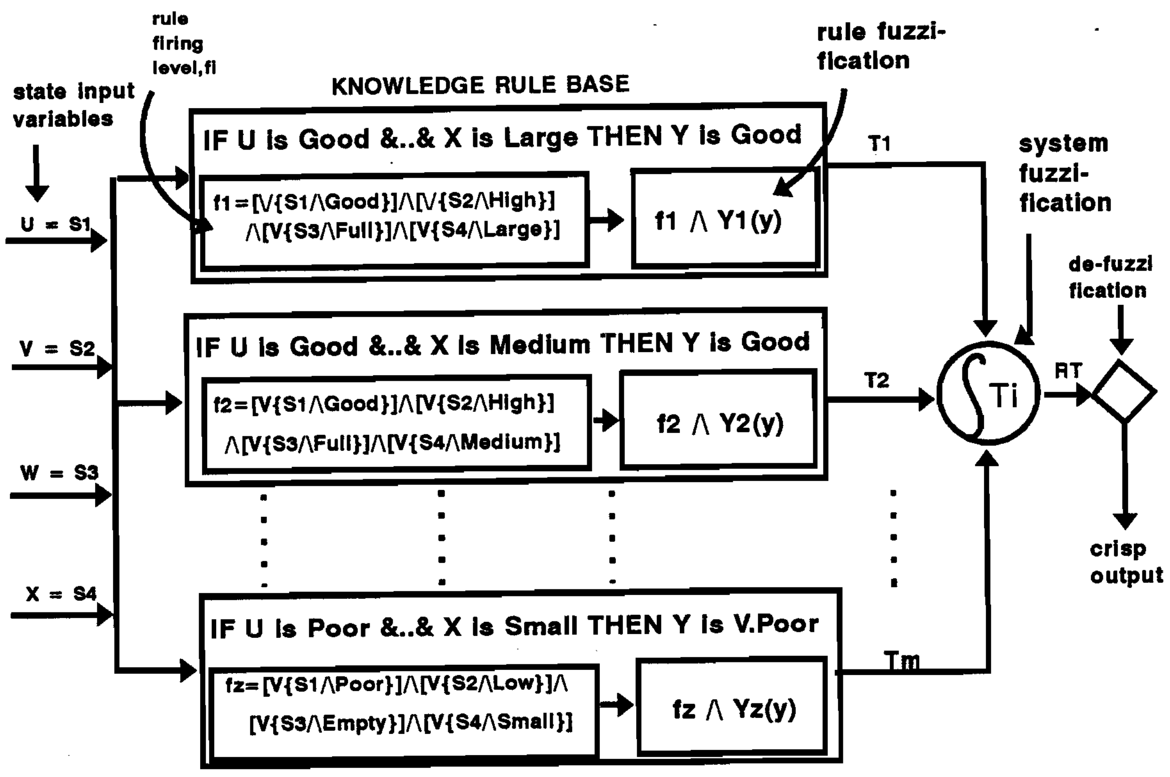
A_{ij} = fuzzy subset of U_j of domain X_j i.e. fuzzy value of U_j

D_i = fuzzy subset of V of domain Y i.e. fuzzy value of V

i = i^{th} rule in the knowledge base for i = 1, m and

j = j^{th} variable in a relation statement for j = 1, n.

The mechanisms of the MISO rule interaction is illustrated in Figure 4.1. Each rule i,



rule firing level,fi
rule fuzzi-fication
state input variables
KNOWLEDGE RULE BASE
IF U is Good &..& X is Large THEN Y is Good
f1=[V{S1∧Good}]∧[V{S2∧High}] ∧[V{S3∧Full}]∧[V{S4∧Large}]
f1 ∧ Y1(y)
T1
system fuzzi-fication
U = S1
de-fuzzi fication
V = S2
IF U is Good &..& X is Medium THEN Y is Good
f2=[V{S1∧Good}]∧[V{S2∧High}] ∧[V{S3∧Full}]∧[V{S4∧Medium}]
f2 ∧ Y2(y)
T2
Ti
RT
W = S3
crisp output
X = S4
IF U is Poor &..& X is Small THEN Y is V.Poor
fz=[V{S1∧Poor}]∧[V{S2∧Low}]∧ [V{S3∧Empty}]∧[V{S4∧Small}]
fz ∧ Yz(y)
Tm

Figure 4.1 Block diagram of the internal structure of a MISO model using constructive reasoning [adapted from Yager and Filev [1994]]

is a relationship defined in the product space of the input variables X_j, (j = 1, n), and the conclusion domain V. A Mamdani fuzzy reasoning is generally used to determine the conjunction of the rule premises and conclusion [Filev and Yager, 1993]. The Mamdani reasoning uses the minimum operator or intersection of sets. A fuzzy relationship, R_i under this reasoning is represented as:

$$R_i = A_{i1} \cap A_{i2} \; \dots \cap A_{in} \cap D_i \qquad (4.6)$$

The minimum operator gives the maximum possible value among the premise variable choices. This value is then used to determine the output of a particular rule through its intersection with the rule's consequent fuzzy set D_i, which is a truncation of the latter at the level of the premise's degree of relevance of a rule i. The degree of relevance of a particular rule's premise is known as the rule's degree of activation or firing under a given set of input values. If the premise membership function differs greatly from that of an input value, then the intersection of these membership functions is small, hence it has a low degree of activation.

The Mamdani reasoning fails to preserve the order of contribution of the consequent membership function [see Figure 4.2]. Due to this problem of Mamdani reasoning, the developed model uses a modified Mamdani reasoning where the conjunction of the premises is done as for the Mamdani case but the conjunction with the consequent is through an algebraic product operator. The equivalent fuzzy relation R_i is expressed as [Mizumoto, 1994; Johnson and Smartt, 1995]:

$$R_i = A_{i1} \cap A_{i2} \cap \dots \cap A_{in} * D_i \tag{4.7}$$

This approach is intuitively appropriate for combining premises and the consequent because it has a trade-off effect in the output, a result consistent with conflicting variables such as illustrated in Figure 4.2.

The fuzzy relation R_i has the joint possibility distribution defined by the conjunction of the membership functions of the respective subsets as:

$$R_i(x_1,\dots,x_n,y) = A_{i1}(x_1) \cap \dots \cap A_{in}(x_n) * D_i(y) \tag{4.8}$$

The **ALSO** in the MISO formulation represents the aggregation of the different rules. A fuzzy solution is therefore based on the entire system interaction and not only a select subset. However, despite this aggregation it is only those rules that achieve a non-zero degree of firing that are of consequence. The relations R_i, ($i=1$, m) are aggregated using the arithmetic mean operator rather than the usual union operator used in the Mamdani reasoning. The arithmetic mean operator is a better representation of information because the output is a weighted value of each rule's contribution to an element in the fuzzy output set. The union operator simply takes the maximum value for each value in the output. The net effect of the later method is that it may assign the same membership to elements of the fuzzy output set which have different contributions based on the

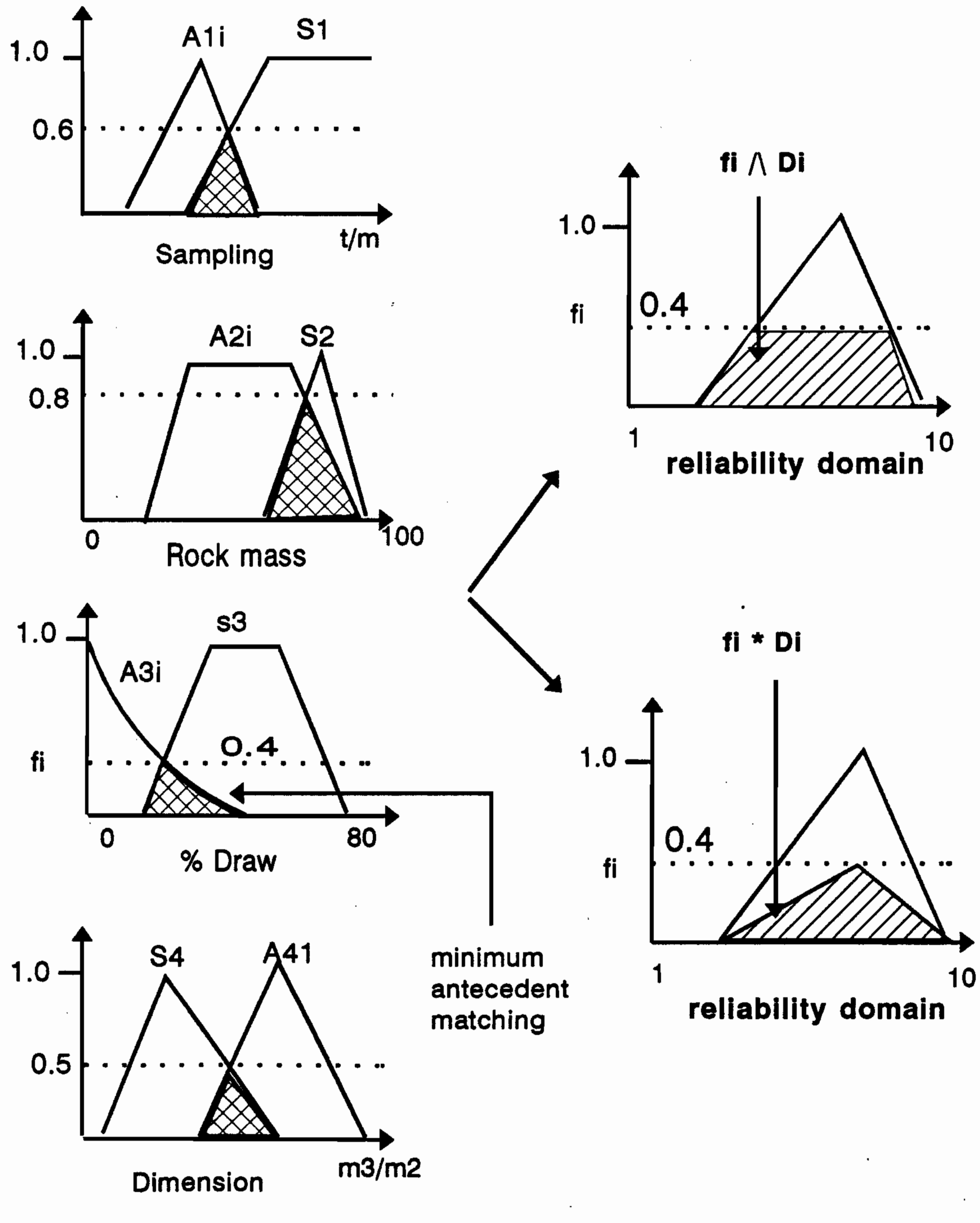


Figure 4.2 Fuzzy conjunction operators of primary variables

individual rules [see Figure 4.3]. The union operator is an optimistic operator compared to the arithmetic mean. The joint possibility distribution of the fuzzy relation R resulting from the aggregation of m individual rules based on the arithmetic mean operator is given by:

$$R(x_1,\dots,x_n,y) = \frac{\sum_{i=1}^{m} R_i(x_1,\dots,x_n,y)}{m} \tag{4.9}$$

This model is then used to infer outputs of user defined fuzzy input sets or linguistic terms. The inputs sets are of the form:

$$U_1 = S_1,\ U_2 = S_2,\ \dots,\ U_n = S_n \tag{4.10}$$

The output of the fuzzy system is a fuzzy output set F obtained by inference of the input sets to the resultant fuzzy relation R:

$$F = (S_1, S_2, \dots S_n) \sum R \tag{4.11}$$

The membership function of the fuzzy output set F is given by:

$$F(y) = \frac{\sum_{i=1}^{m} (\cup_{x_1}[A_{i1}(x_1) \cap S_1(x_1)]) \cap \dots\dots \cap (\cup_{x_n}[A_{in}(x_n) \cap S_n(x_n)]) * D_i(y)}{m} \tag{4.12}$$

$$F(y) = \frac{\sum_{i=1}^{m} Poss[A_{ij}|S_i] \cap \dots \cap Poss[A_{ij}|S_n] * D_i(y)}{m} \tag{4.13}$$

$$F_i(y) = \frac{\sum_{i=1}^{m} \tau_i * D_i(y)}{m} \tag{4.14}$$

$$F(y) = \frac{\sum_{i=1}^{m} F_i(y)}{m}, \; where \; F_i(y) = \tau_i * D_i(y) \qquad (4.15)$$

where τ_i = degree of relevance of the rule i or the confirmation that the input values S_i have an intersection with the variable fuzzy value A_{ij}. The value of F(y) is the arithmetic mean of the membership values of each element in domain V.

The value F(y) of the fuzzy output set F indicates the degree to which each element y in the domain V is valued a good output by the rule-base under the current input $S_1,..,S_n$. In the model, all input variable universes are normalized to be defined between one and ten, and hence, the output domain V also has the same dimension. The objective of the fuzzy modelling procedure is therefore to determine the fuzzy output membership function F(y) for a given input, and subsequently use the output subset F to obtain the best output for decision making or implementation. This aspect is treated by a de-fuzzification algorithm.

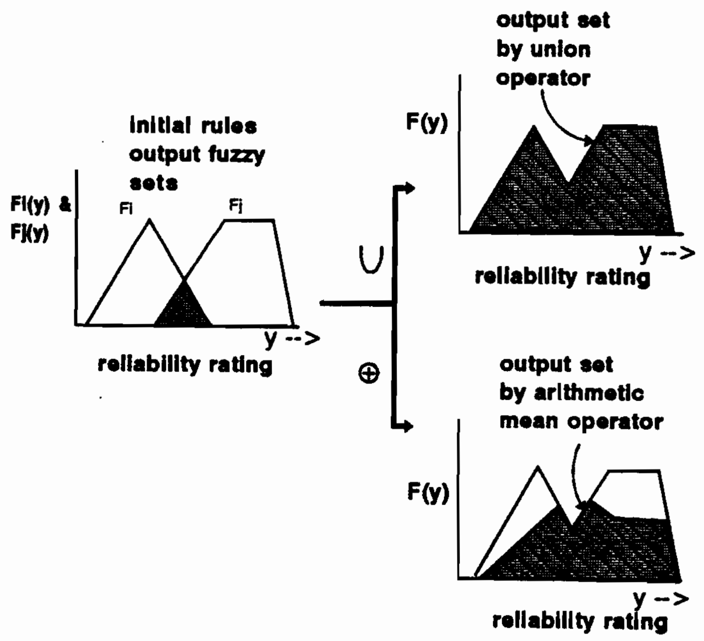

initial rules output fuzzy sets
Fi(y) & Fj(y)
Fi
Fj
y -->
reliability rating
U
⊕
output set by union operator
F(y)
y -->
reliability rating
output set by arithmetic mean operator
F(y)
y -->
reliability rating

Figure 4.3 Fuzzy aggregation operators of fuzzy output subsets

4.4.2 De-fuzzification Algorithm

A number of methods are available for determining the fuzzy model output. This step is known as de-fuzzification and represents the determination of a crisp value from the fuzzy consequent described by equations 4.9a-d. The commonly used methods of de-fuzzification in fuzzy logic applications are the centre of gravity, mean of maxima (MOM) and method of heights [Yager and Eilen, 1994]. The centre of gravity (COG) or centre of area (COA) method calculates the output for all elements in the model consequent that have a non-zero membership. The method of heights is similar to the COG method with the exception that the output is calculated for all elements in the model consequent that attain a certain height (α-level). The α-levels are set as decisions hurdles by management. The MOM calculates the crisp value by averaging the support values whose membership attains the maximum (i.e. the height) for the fuzzy output set. It highlights those elements that are most satisfied for the given input values.

The de-fuzzification problem defines the strategy of using the fuzzy subset F, to guide in the selection of a crisp representative of the set F. The de-fuzzification algorithm uses F to select a best value to be the output of the fuzzy system. In this model, the COG method is used to calculate a crisp output because it takes the information supplied by the rule-base at its face value and its interpretation of an output is built on normal confidence in the model inputs. Other methods, such as the MOM, assume full confidence in the model inputs, which may not be necessarily true. The COG method is defined as:

$$y^* = \frac{\sum_{k=1}^{Z} F(y_k) . y_k}{Power(F)} \tag{4.16}$$

where y_k is the kth element of the consequent universe of discourse and Z is the size of the universe Y for which F is a subset. Power(F) is the summation of the membership values for all Z elements of the universe Y.

The value y^* is the result of fuzzy modelling. It is the sought measure of reliability in the fuzzy inputs based on a knowledge base and a particular reasoning logic. The confidence in the value y^* is determined by the fuzziness measure defined earlier.

4.5 Fuzzy Logic Stope Model Input Variables

There are four basic variables that are attributed to the uncertainty or reliability in the stope mining grade during production. These input variables are:

- the sampling information used in the design and calculation of the stope grade as well as information acquired during development and/or production drilling;
- the wall rock mass strength, the dip, geological structures, general stress distributions around the stope, and the mining sequence;
- the stope layout and dimensions, and,
- the cumulative amount of ore drawn from the stope.

These variables do not represent classical data sets, rather they are fuzzy, having possibly different impacts in different mines. Precise representations of spatially interpolated geological, geomechanical and cumulative production does not appear possible within current mining operations. The above proposed fuzzy logic approach of integrating fuzzy information is used to determine the criterion of stope selection for scheduling. For the case study, these four variables were identified from both the results of a questionnaire survey sent to several mines, and personal communications with mine planners and operators.

4.5.1 Sampling Intensity

The sampling information obtained during the delineation of a stope and subsequent reserve calculation is the direct method of obtaining a grade value of the stope. A review of the estimation process indicated that the reported values are subject to geological interpretation and assumptions made. The accuracy of geological interpretation is directly related to the structure of the orebody and the number of samples taken in a given structure.

The sampling intensity or knowledge is one variable that has to be modelled. It is obvious that the terms intensity and knowledge are fuzzy. They have a range over which an observer can be indifferent. The universe of a sampling strategy varies from no sampling to sampling every tonne of the mineralization. The usual sampling intensity is somewhere in between, and is a function of both the cost and, most certainly, the geology. Budget constraints usually lead to large extrapolation of sample information, a process that reduces the confidence in the reserves. The universe of sampling can be partitioned into fuzzy sets that depend on the intensity envisaged for a particular geological structure. For the same base of ore definition, a tabular coal seam requires fewer samples per tonne of defined ore compared to a folded and faulted seam for the same base of ore definition. Thus, whilst T tonnes per sample is considered good sampling in one geological setting, it may be very good or poor in another setting. The fact that stopes are rarely structurally the same indicates that we are dealing with a fuzzily defined quality of stopes.

It is common to find within the same mine a varying sampling intensity. The sampling may be increased or reduced depending on experience and inference on the continuity of the orebody. In this model, the universe of sampling is partitioned into three fuzzy regions of sampling intensity (SI) defined by three labels:

$$SI = \{GOOD, FAIR, POOR\}$$

An example of the form of the label membership functions are indicated in Figure 4.4. The categories reflect the average stope tonnage attributed to a single sample. An analogy of this concept is that of mineral resource classification, with 'proven', 'probable' and 'possible', ore resource, where each class reflects the information availability. As some operating mines do not practice regular systematic drilling programs for sampling, it

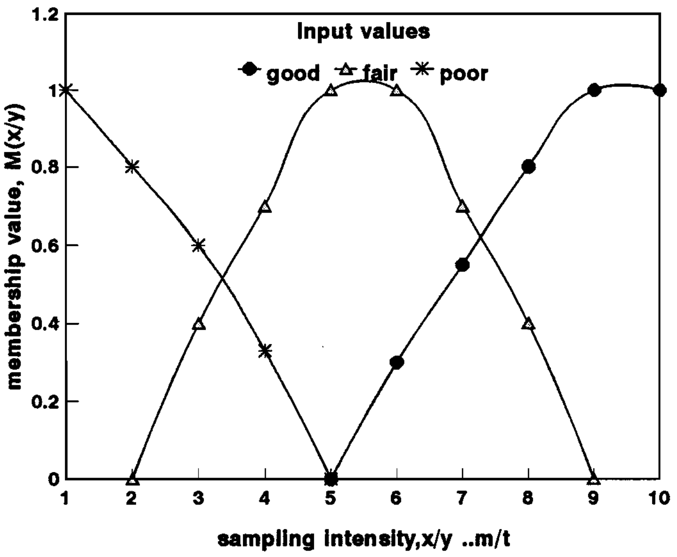


Figure 4.4 Normalized membership functions for sampling intensity values

implies that their stopes are defined on different information levels, hence the importance of this parameter in such circumstances.

4.5.2 Rock mass Characterization

Mines often simultaneously operate stopes at different stages of production. They may also follow a definite mining extraction sequence such as the 1-2-3, 1-3-5, 1-4-7, and checker board patterns. Ground responses to the mining activity under the different sequences is different. In some instances this may be a significant factor that causes the wall rock to break into the open stopes.

In mine backfill sequences, an operating stope can have a variable number of backfill walls. For example, in the 1-4-7 extraction sequence, the primary stopes have no backfill walls, the secondary stopes have one wall, and the tertiary panels (3-6-9) have two. Higher dilution is typically associated with pillar recovery. The backfill material may deteriorate due to blasting effects in adjacent operating stopes, mass movement vibrations during ore pulling, and the progressive transfer of load from the rock (in-situ ore) to the fill. As discussed in Chapter 3, backfill placement is not a perfect operation and may result in segregated material in which the peripheral fill material is weak. It can be inferred that dilution increases as the number of fill walls and/or total time of open standing of an active stope increase.

A mine wide rock mass characterization is likely to represent the distinct regionalized structure and properties. Sandbak [1988] reports on the wide variation in fragmentation characteristics between rock types and different mine sections within the same orebody. Rock mass rating was similar in most of the sections studied, indicating that drill cores and laboratory testing are not accurate in describing an in-situ rock mass. The rock mass characteristics of the stope walls have a direct bearing on stope perimeter dilution, i.e. the weaker the wall rock, the higher the expected dilution. Besides the wall rock mass strengths, the shape of the stope is an influencing parameter on dilution. In a convolute orebody, stope design will invariably include more external waste to make it amenable to the selected mining method. In longhole production it is known that drill holes deviate appreciably from design. This causes a greater possibility of the perimeter production holes deviating into the waste material. A generally unquantifiable amount of waste is therefore introduced through the production blast process.

The rock mass rating is based on point field information, such as rock quality designation, joint-fill, roughness and spacings, and intact rock mass strength measured on sample cores [Brady and Brown, 1985]. The aggregate field rock behaviour is more complex than is modelled by the sample population. Opening shape, dimension, and the sequence of extraction have an aggregate effect on the rock mass behaviour. Experience gained from working in certain sized stopes, using certain mining methods allow the creation of membership functions of rock mass behaviour to mining. Three labels are used to partition the rock mass behaviour (RM):

RM = {LOW, MEDIUM, HIGH}

where HIGH reflects competency and LOW, unfavourable conditions with respect to dilution effects. The base variable of rock mass aggregate behaviour is an ordinal scale of 1 to 10 or any other suitable range for which the observed effects can be categorized.

4.5.3 Stope Geometries

The nature of mineralization dictates the mining method used. Frequently, orebodies consist of several lodes stacked above each other or on strike. Shear zones are other important massive mineral hosts. Both the lodes and shear zones occasionally show wide variations in dimension within the same deposit. Stopes being designed to fit the mineralization, as closely as possible different sized stopes usually exist within the same mine. For a given tonnage, the volume-to-perimeter ratio of a stope increases as its shape becomes regular and the plan section dimensions increase. The implication of this observation is that rectangular stopes approaching a large square format minimize dilution effects of wall rock influences, all other factors being constant.

An operating mine should thus have a list of the working and planned stopes indicating their relative geometries. This information is invaluable in mine production scheduling as it contains some unquantifiable effects on the realized mined grades. The universe of geometry, G can be in terms of shape, volume-to-area ratio or length (e.g. stope height and width). The three labels used to describe this input are as follows:

$$G = \{\text{SMALL, AVERAGE, LARGE}\}$$

The stope width may vary from two to over twenty metres depending on the mineralization type and mining method used. If a stope is assigned values LARGE and LOW for its dimension or regularity, and rock mass rating, respectively, the two variables are in conflict. That is, a large stope with the least perimeter dilution effect based on maximum volume-to-area and a poor rock mass rating is geotechnically unstable with a potential consequent of massive wall failure. The measure of this conflict with respect to the schedule goal can be obtained through a fuzzy aggregation algorithm scheme.

4.5.4 Cumulative Stope Draw

The cumulative ore drawn from a given stope has a bearing on the grades pulled. Usually, if a stope designed on sufficient sample data is new and is mined by either vertical crater retreat or block caving methods, the pulled grades indicate less departure from the estimated mineable grades. This comparison is based on consistent sampling intensity and is generally the case in bulk mining methods, where the orebody is massive and tends to be modestly regular. As production progresses however, the influence of high walls in the case of open stoping increases the wall stresses that result in parts of the walls sloughing into the stope, thus causing additional dilution.

In sub-level and block caving methods, the mixing of mineralization originating from different locations in the stope is common. Fragmented rock is heterogeneous, consisting of both small and large blocks. This precludes a bulk flow and more often exhibits a hybrid of bulk and funnel flows. Friction between the blasted muck and the wall causes such material to lag behind the core material. In the presence of backfill walls, some of the backfill deteriorates and becomes an additional dilutent. As these stopes reach 70 to 80 % extraction, the incremental dilution has been reported to rise rapidly rendering, in some cases, the remainder of the material sub-economic.

The cumulative ore drawn variable (CD) is defined over the universe of 0 to 100 percent, indicating no draw to complete draw, respectively. The linguistic terms used to describe these situations are FULL, HALF and EMPTY, defining the set:

CD = {EMPTY, HALF, FULL}

4.5.5 State (User Input) Variables

The state variables are the required user input values. Four state variables are needed for each stope description, with each variable representing a subset of the input variable sets. The same terms as those used to describe the four input variables are used with two modifiers. The membership functions of the state and input variables are identical if the same linguistic terms describe both the state and input variables. The state inputs are the values which the model tests to confirm that they belong to the system input variables (N.B. the input variables are defined in the knowledge base).

The two modifiers used in the model are "very" and "slightly" and they are defined as a concentration (α = 2) and a dilation (α= 0.5), respectively. The effect of the "very" modifier is to increase the confidence on the modified linguistic term or set. The modifier "slightly" describes the lower end of a base label. For example, "slightly good" implies that the base value 'good' is met only to a limited extent. The state variables are all fuzzy subsets of their respective universes of discourse. For example, the sampling intensity output variables are defined in the set:

sampling state variables = {very poor, poor, slightly poor, slightly fair, fair, very fair, slightly good, good, very good}

The scheduler is expected to be able to qualitatively describe the factors of rockmass characteristics with respect to fill walls, geometry, stress distribution, wall rock response and geometries of the planned stopes. Such a description constitutes the model state variable data. Such a description can be facilitated by use of a check list or matrix.

4.5.6 Output Variables

The output variables represent the consequent of the interaction of the antecedent relations in the knowledge base. Thus, if a stope has a weak wall rock mass, is well advanced in its exploitation, was designed on fairly sparse information and is large in extent, we conclude that it is a "poor" stope with respect to ore quality. This is due to expected high dilution effects. The descriptor "poor" is the output variable for that stope. The output variable (Y) set is partitioned into five subsets defined as:

Y = {VERY POOR, POOR, FAIR, GOOD, VERY GOOD}

The universe of discourse for the variable is any ordinal range, for example 1 to 10, with 10 indicating the output subset in the good to very good category.

The relationship of the input, state and output variables is illustrated in Figure 4.5. The model has four input variables each described by three label fuzzy subsets, which gives a total of 81 rules (i.e. 3^4) in the knowledge base. Each of the 81 rules is combined with one of the output variables to generate 81 inferred output fuzzy subsets. The later 81 subsets are aggregated using an arithmetic mean operator to obtain the fuzzy output set of the system (model). The decision-making criterion is based on a crisp value obtained through a COG de-fuzzification process for each stope fuzzy output set. As the number of rules increase the system becomes complex, recalling that all the relations are based on human experience and may also be site specific. The coding of a problem with numerous rules requires large amounts of computer memory, and computation time increases accordingly.

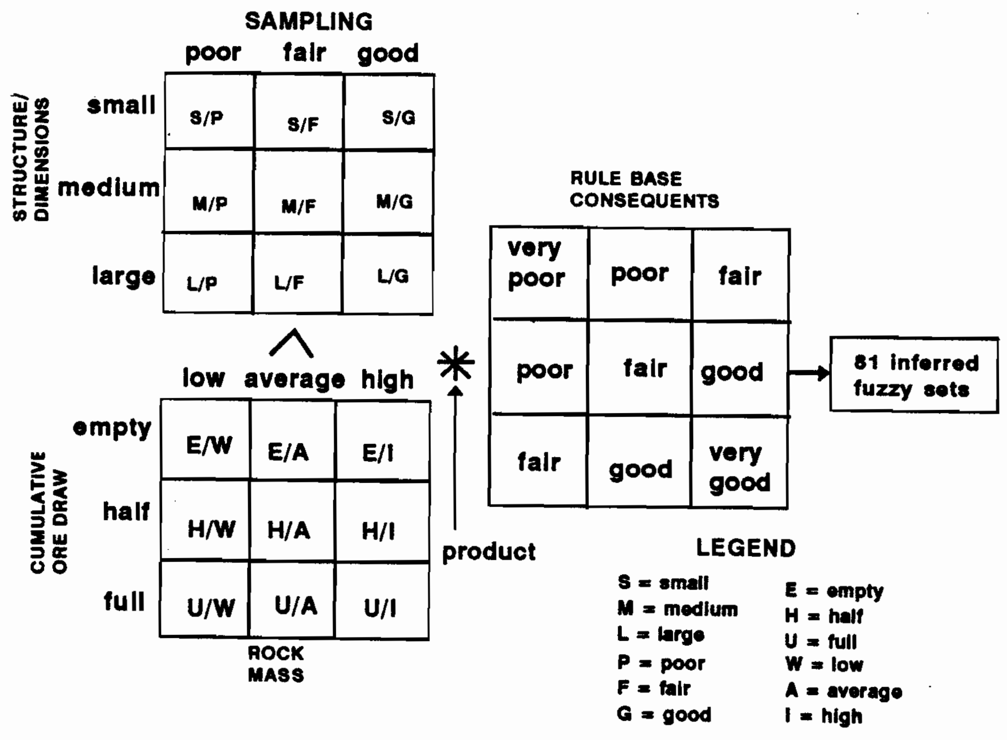


Figure 4.5 FLSM knowledge base variables

4.6 Construction of Fuzzy Sets

The fuzzy input parameters of a fuzzy system are first identified. In this model these are the sampling information, cumulative stope draw, rock mass characteristics and stope layout and/or dimensions. A modeller then selects a number of linguistic labels which allows a full description of all possible values of each parameter. For sampling information, labels such as "very poor" to "very good" can be used. These labels are then arranged in the hierarchial order that describes the model goal. The purpose of this model is to maximize the stope grade information and therefore, the sampling intensity parameter is represented by the following fuzzy set in which the labels are ordered:

{very poor, poor, fair, good,very good}

A set of examples containing a variety of each label is presented to the modeller who is then asked to classify them by label. If an example is fully described by a particular label, then that label is assigned a value of one; if the label does not describe the example at all, a value of zero is assigned. If in some cases the modeller cannot decide whether to assign zero or one, a value somewhere in between can be assigned. For the sampling intensity variable, geological structure directly affects what sample density is deemed sufficient for a particular stope. This leads to the gradational variation ('grey area') from one label to the next.

Suppose the modeller is given the following two stopes which have relatively 'good' sampling intensities:

Stope 1 is defined on 1000 tonnes per metre sampled and,

Stope 2 in similar geological setting is defined on 2000 tonnes per metre sampled (i.e. not as intense).

The modeller may assign a value of one to the 'good' label for Stope 1, and some value less than one for Stope 2. However, he/she may also decide to assign a value of one to the 'fair' label for stope 2, and some value less than one for Stope 1. A value of zero would naturally be assigned to the 'very poor' label for both stopes. A possible result is indicated in Table 4.1, where fractional values indicate situations when the modeller cannot fully decide on zero or one.

Table 4.1 Basic sampling variable acceptance values

Label->	very poor	poor	fair	good	very good
Stope 1	0	0	.5	1	.7
Stope 2	0	0.5	1	.6	0

This means that a sample intensity of 1000 to 2000 tonnes per metre sampled cannot possibly be regarded as 'very poor' nor as 'very good', but is in the fair to good range.

The labels are assigned a membership of one if they have been assigned a one value and those with a zero value have a zero membership. The membership between the zero and one is defined by connecting the zero and one memberships by a continuous monotonic function to indicate that the membership increases as one gets closer to the limit of one [see Figure 4.4]. This procedure is repeated for each parameter and for each parameter label, thus generating the system membership functions.

4.7 Fuzzy Logic Stope Model Program

The above procedure is coded into a C language program called Fuzzy Logic Stope Model (FLSM). A listing of the program along with the input files are given in Appendix A. The implementation of the program requires two data files: (a) a file containing the numeric description of each of the linguistic terms for each of the four variable fuzzy input sets, namely, sampling, rock mass, geometry and amount of ore draw of stopes (see Table 4.3), and (b) a file containing the numeric description of the fuzzy output sets (see Table 4.4). A user is prompted to input the four state variables for each stope being scheduled. This is simply a linguistic description of the stope at a point in time during its extraction.

The output consists of the following:

- the aggregation and fuzziness values for each stope. These values are crisply defined (i.e. not fuzzy). They are the relative measures of reliability and level of confidence on mining grades of specific stopes that result from the multiple variable input assessment.
- the membership values of the aggregate output fuzzy set that is defined over a ten point rating domain (i.e. 1 to 10). This domain represents the reliability on mining grades during extraction of a given stope. The low and high value elements imply a low and high reliability, respectively, in the mining grade.
- the probability distribution of the fuzzy output set of the system, i.e. the distribution indicates the probability of a membership value of each domain element, and
- the joint possibility distributions that indicate the extent of the different variable interactions. The joint possibility values reflect the level of intersection (commonality) of the input variables towards meeting the objective of maximum reliability on stope mining grades at extraction. The location and area of the intersection of a joint possibility distribution provides information on the confidence levels of the aggregation and fuzziness measures.

4.7 An Example of Fuzzy Logic Stope Model Analysis

Suppose the hypothetical Inza Mine has three active stopes, A, B and C. Stope B is secondary whilst the other two are primary. The user linguistic state variables of the stopes are indicated in Table 4.2. The objective consists of ranking the stopes in order of least quality distortion. In addition, the scheduler needs information about the blending strategy for the coming shift. Accordingly, the process plant has expressed concern about wide variation in run-of-mine ore. Therefore, the scheduler has to achieve a certain minimum quality in the subsequent schedules.

4.7.1 Input Data

The state variables defined in Table 4.2 have the membership functions listed in Table 4.3. The membership functions represent numeric description of fuzzy variables and were obtained following the method discussed above, in section 4.6. Similarly, Table 4.4 illustrates the membership values of the output variable term set in the consequent of the knowledge rule-base. All the linguistic labels for the input, state and output variables were mapped over a ten point rating, where one marks the lower end and ten the upper. The choice of the rating span (i.e. width) for each variable is arbitrary. Ten was selected for all the variables only because it simplified the multi-variable multi-dimensional problem computation.

Table 4.2 FLSM Linguistic input values for the Inza Mine

Stope	Sampling data: tonnes/m	Stope shape factor	Rock mass behaviour	Relative ore drawn
A	good: 150	large	medium	full <30%
B	fair: 200	average	medium	half: 30-50%
C	good: 130	small	very high	half: 40-70%

Table 4.3 Fuzzy input sets membership values for the Inza Mine

	Domain element ratings (1-10)									
Subset	1	2	3	4	5	6	7	8	9	10
good	0.0	0.0	0.0	0.1	0.3	0.4	0.6	0.9	1.0	1.0
fair	0.0	0.2	0.6	1.0	1.0	0.9	0.8	0.5	0.2	0.0
full	0.0	0.0	0.0	0.3	0.4	0.6	0.7	0.8	0.9	1.0
half	0.1	0.3	0.6	0.8	1.0	0.7	0.6	0.5	0.1	0.0
medium	0.0	0.2	0.4	0.6	0.9	1.0	0.7	0.4	0.1	0.0
large	0.0	0.0	0.2	0.2	0.4	0.5	0.7	0.8	1.0	1.0
average	0.0	0.0	0.3	0.6	0.8	1.0	0.9	0.6	0.3	0.0
small	1.0	1.0	0.8	0.6	0.5	0.4	0.3	0.2	0.1	0.0

Table 4.4 Fuzzy output sets membership values at the Inza Mine

	Domain element ratings (1-10)									
Subset	1	2	3	4	5	6	7	8	9	10
v.good	0.0	0.0	0.0	0.0	0.0	0.1	0.4	0.8	0.9	1.0
good	0.0	0.0	0.0	0.1	0.2	0.4	0.7	0.9	1.0	1.0
fair	0.0	0.2	0.6	0.9	1.0	1.0	0.7	0.5	0.2	0.0
poor	1.0	0.9	0.7	0.5	0.2	0.1	0.1	0.0	0.0	0.0
v.poor	1.0	0.9	0.6	0.3	0.1	0.0	0.0	0.0	0.0	0.0

4.8 Fuzzy Logic Stope Model Results

The results obtained by executing the program for the input values given in Table 4.2 and system membership values defined in Table 4.3 and 4.4 are indicated in Table 4.5. The aggregation values are the crisp output from a centre of gravity de-fuzzification of each stope variable. In Table 4.5, stope C has the highest aggregation followed by stope A, hence the stope ranking is C, A and B. This conclusion is not obvious, especially between stopes A and C, when one simply considers the input values from Table 4.2. This illustrates the lack of ambiguity when all influencing parameters are considered simultaneously.

Table 4.5 FLSM stope ranking for Inza Mine

Stope	Aggregation value	Fuzziness measure	Stope rank
A	4.878	0.325	2
B	4.670	0.485	3
C	5.042	0.295	1

The difference in aggregation values between the stopes is a measure of the difference in differential weights that may be placed on the stopes. Stope A and C have the least difference for the three possible, two by two stope combinations whilst stope B and C have the largest difference. This result reflects the intuitive problem of ranking A and C, and the straight forwardness of ranking B and C.

The fuzziness measure strengthens the conclusions based on the aggregation values of the stope fuzzy output sets. Stope C has the lowest fuzziness and would therefore contribute the least to grade variability. The use of fuzziness is essential for breaking ties in stopes that have the same aggregation values. For instance, suppose that two probability distributions have the same mean but different variances. The distribution with the smaller variance is of higher value to a decision maker whose intention is to minimize deviations from a specified target.

Recalling that an aggregation value is obtained by de-fuzzification methods such as COG and MOM, the following comments are made about this value and fuzziness:

- if the aggregation value of a stope is high, it means the stope's membership function has its highest values in the region of high ratings, which in this case is closer to ten, and
- if the stope membership functions peak in the high rating range, it follows that fuzziness is small by definition (see equation 4.3). Fuzziness has a high value only if the membership function is characterized by a relatively flat and low membership value function. Such a function only results in medium to low aggregation values.
- It has to be noted that FLSM gives the reliability factors of mining grades based on the qualitative information available at the stopes' extraction. If the stope mining grade is above or below target, that problem is solved through a tandem goal programming blending model that is described in chapter 5. The goal programming model utilizes these reliability factors of minimum additional stope dilution.

The output membership functions for the three stopes are illustrated in Figure 4.6. The graph indicates a similar form for all three stopes but all are distinct and unambiguous. Stope C has the highest membership, especially in the centre interval of the output domain of outcome desirability. The membership of stope A is everywhere significantly below that of stope C and marginally above that of stope B. These membership functions confirm the results described by the aggregation and fuzziness of the output subsets for the individual stopes.

The probability distributions for the stopes obtained through the basic de-fuzzification distribution transformation are illustrated in Figure 4.7. The distribution for stope C is

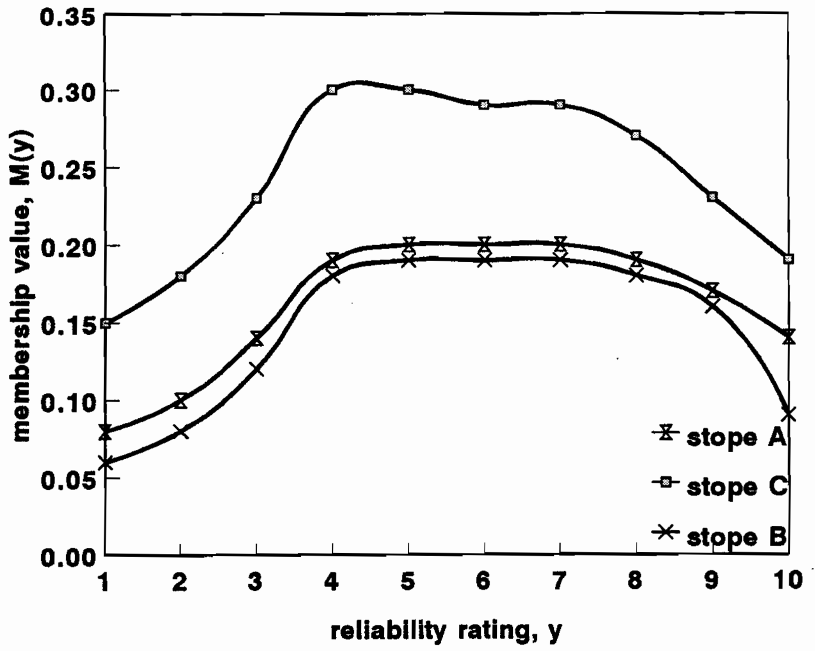


Figure 4.6 Output membership functions for the Inza Mine stopes

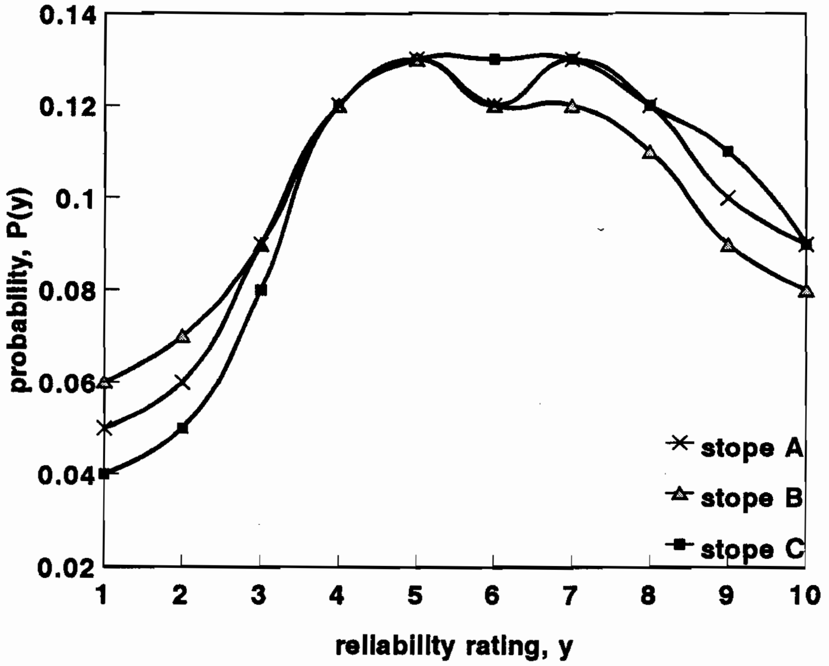
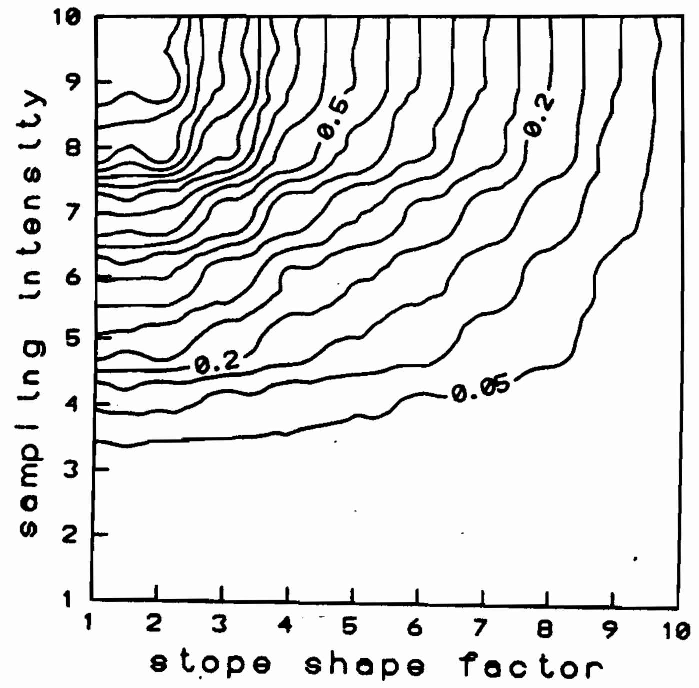


Figure 4.7 Probability distributions for the Inza Mine stopes

such that it has a low probability in the low output levels of less than a rating of four. Above this value, it dominates the other two stopes' probability distribution functions. As with the membership functions, the probability distribution function of stope A is located between that of stopes B and C. It is superior to stope B and inferior to stope C in the output levels with ratings of less than 3 and greater than 6. In the centre ratings of the output of 3 to 6, the relationship is not as evident.

An insight into the problem is provided by joint possibility analysis of two parameter accumulation matrices. The FLSM generates (1) a sampling intensity - stope shape (or dimension) factor, and (2) a cumulative draw - rockmass characterization (or mining sequence) joint possibility matrices. By contouring the joint possibilities, two dimensional maps are obtained. The joint possibilities for stope C are illustrated in Figure 4.8. The equivalent three dimensional graphs of the joint possibilities for stope C are illustrated in Figure 4.9. These maps enable the decision maker to assess the regime in which the output (i.e. a decision) is made. If a decision maker requires that a certain minimum joint possibility be met, then this hurdle contour is easily determined on the map. The area with joint possibilities higher than the hurdle contour reflects the elements that meet the objective prior to complete fuzzification of the problem.



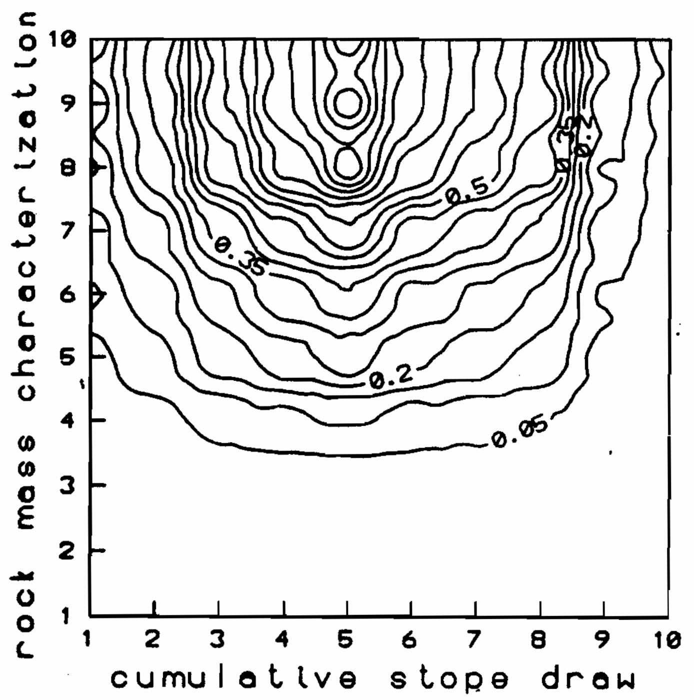
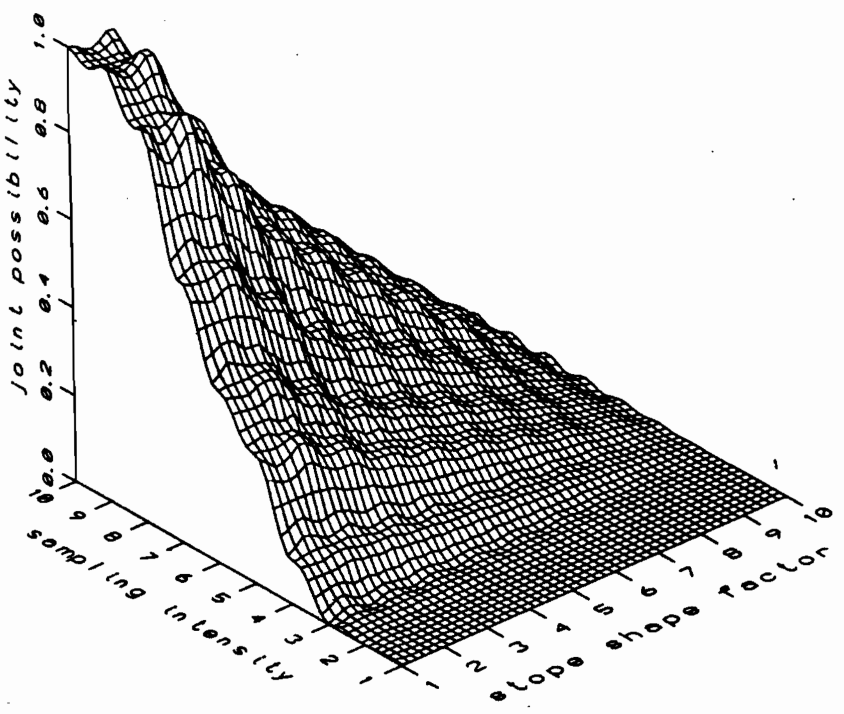


Figure 4.8 Contour maps of joint possibility distributions for stope C



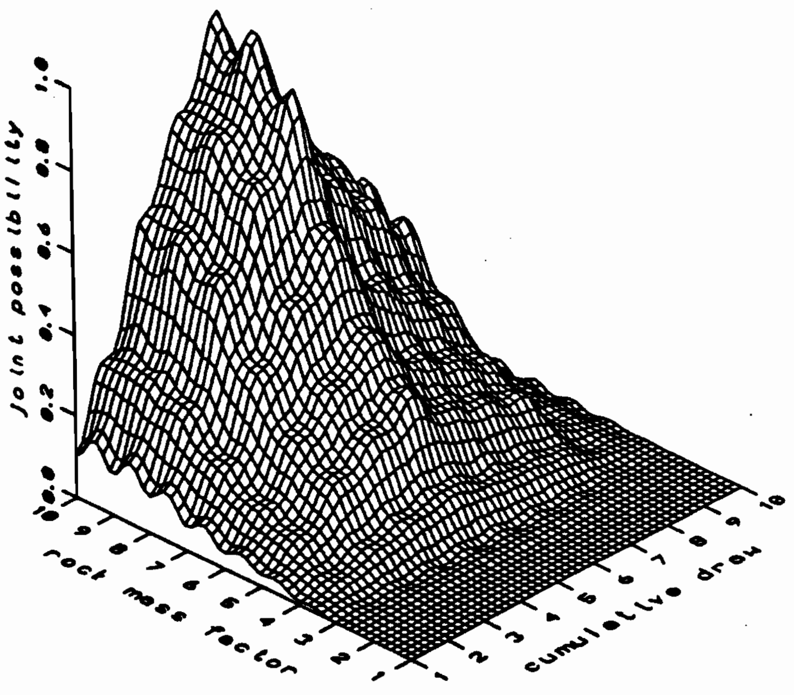


Figure 4.9 Three dimensional joint possibility distributions for stope C

4.9 Parametric Analysis

A parametric study was done on the FLSM to determine the following:

- its ability to rank multiple stopes described by diverse variable values and the consistency of those ranking, and
- to determine the sensitivity of the model to dynamic variables during a progression in a stope depletion.

4.9.1 Ranking of Multiple Stopes

Table 4.6 is a listing of the different input values used to define nine different stope conditions for a given database of input memberships (same databases as that used for the Inza Mine). It should be noted that Table 4.6 represents nine stopes of different geometries, rockmass characterization, sampling intensities and in different stages of depletion.

Table 4.6 Input linguistic values of stopes for FLSM sensitivity

Analysis number,N	Sampling	Dimension &/or structure	Rock mass factor	Cumulative stope draw
1	very poor	very small	very low	very empty
2	poor	small	low	empty
3	fair	average	medium	half
4	good	large	high	full
5	very good	very large	very high	very full
6	slightly good	slightly large	slightly high	slightly full
7	fair	slightly small	very high	full
8	poor	average	high	half
9	poor	large	medium	half

The FLSM program was executed and the resultant membership values are listed in Table 4.7, where: **A** represents the aggregation value,

F represents the <u>fuzziness</u> of the output membership functions, and

x represents an element in the output domain of the output membership functions

Table 4.7 Output membership functions for input value combinations in Table 4.6

	Output domain element ratings (1-10)										Measures	
N	x=1	2	3	4	5	6	7	8	9	10	**A**	**F**
1	.10	.11	.09	.10	.10	.08	.06	.05	.04	.04	**3.64**	**.155**
2	.14	.15	.14	.16	.16	.14	.11	.09	.08	.08	3.66	.249
3	.16	.19	.19	.25	.29	.27	.23	.20	.19	.19	4.55	.434
4	.08	.10	.11	.15	.18	.17	.16	.15	.15	.15	4.94	.279
5	.03	.05	.05	.08	.10	.11	.11	.11	.11	.11	**5.36**	**.173**
6	.12	.14	.15	.20	.23	.23	.20	.19	.19	.19	4.82	.366
7	.12	.14	.14	.17	.19	.18	.14	.12	.11	.11	4.38	.285
8	.13	.16	.16	.21	.23	.21	.16	.14	.12	.12	4.28	.325
9	.13	.16	.16	.20	.22	.20	.16	.13	.12	.12	4.29	.319

These membership functions and their corresponding probability functions are shown in Figures 4.10 and 4.11 respectively, where the following legend applies:

A = average, E= empty, F = full, G = good

H = high, L = large, M = medium, P = poor

W = low, X = small, Y = fair, Z = half

s = slightly and v = very are modifiers, e.g. v(P) means very poor.

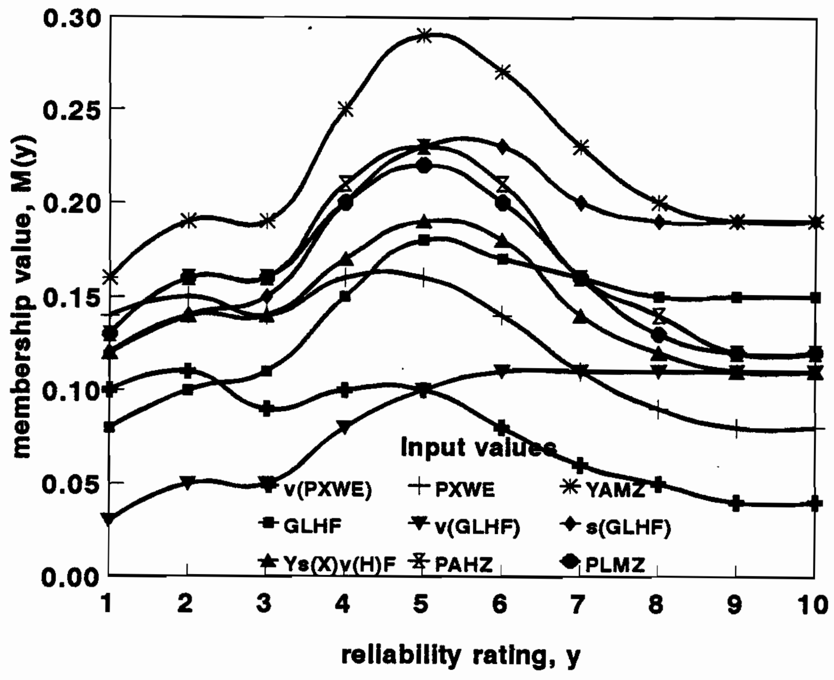


Figure 4.10 Membership functions of multiple stopes

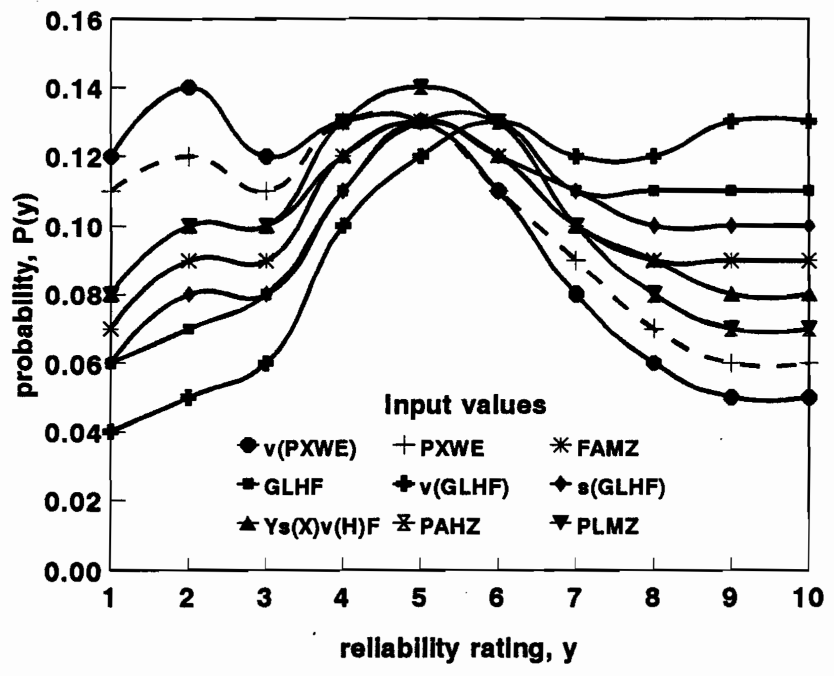


Figure 4.11 Probability distributions of multiple stopes

Both the analysis of these graphs and the aggregation values indicate the successful description of a stope's additional dilution effects during its depletion. The effect of moving from a poorly sampled, nearly empty stope with a poor rockmass factor, to one that is well sampled, structurally massive and possesses competent wall rock increases the aggregation value and decreases the fuzziness. These results are exactly those required by the scheduler to correctly assign priorities and differential weights to the stopes in production scheduling.

4.9.2 Stope Progressive Extraction

The sampling and dimension variables are determined at strategic planning and therefore are fixed for a given stope. The sampling base, however, may be updated through sampling of production drilling in which case, it is a pseudo-static variable. The rockmass factor and cumulative ore draw are dynamic variables and hence, all their labels are subject to change during a stope extraction. These variables are determined at the tactical planning stage. It is also noted that the rockmass factor has some components derived from strategic planning such as a mining sequence and the sizing of openings during mine design.

The dynamic changes that occur during the life-time of a stope were analyzed with the FLSM to determine the sensitivity of the membership functions to those changes. The progressive depletion of a typical stope of the Inza Mine was simulated for four different mining stages as indicated in Table 4.8. The sampling and stope structure variables are constant, whilst the rockmass characteristics deteriorate due to stand-up time and stress response to continued mining. The proportion of ore in the stope decreases as mining continues.

Table 4.8 Input linguistic values for a stope during its progressive extraction

Mining stage, N	Sampling information	Dimension &/or structure	Rockmass factor	Cumulative stope draw
1	good	large	high	full
2	good	large	medium	half
3	good	large	medium	empty
4	good	large	low	empty

The results of executing the program are the membership, aggregation and fuzziness values listed in Table 4.9. The membership and probability distributions are illustrated in Figures 4.12 and 4.13 respectively. The highest and lowest aggregation values are obtained for the initial mining stage (N=1) and the last (N=4). This is the expected result for a stope showing increased wall-rock instability due to progressive mining and long stand-up time. Fuzziness numbers are low for the first and fourth mining stages, indicating greater certainty on the behaviour of the stope than in the intermediate stages with respect to additional dilution.

Table 4.9 Output membership functions for a stope during its progressive extraction

	Domain elements of output set (1-10)										Measures	
N	**x**=1	2	3	4	5	6	7	8	9	10	**A**	**F**
1	.08	.10	.11	.15	.18	.17	.16	.15	.15	.15	**4.94**	**.279**
2	.09	.12	.13	.18	.21	.21	.19	.17	.17	.17	4.88	.326
3	.09	.12	.12	.17	.20	.19	.16	.14	.13	.13	4.69	.291
4	.10	.12	.12	.16	.18	.16	.13	.11	.10	.10	**4.39**	**.257**

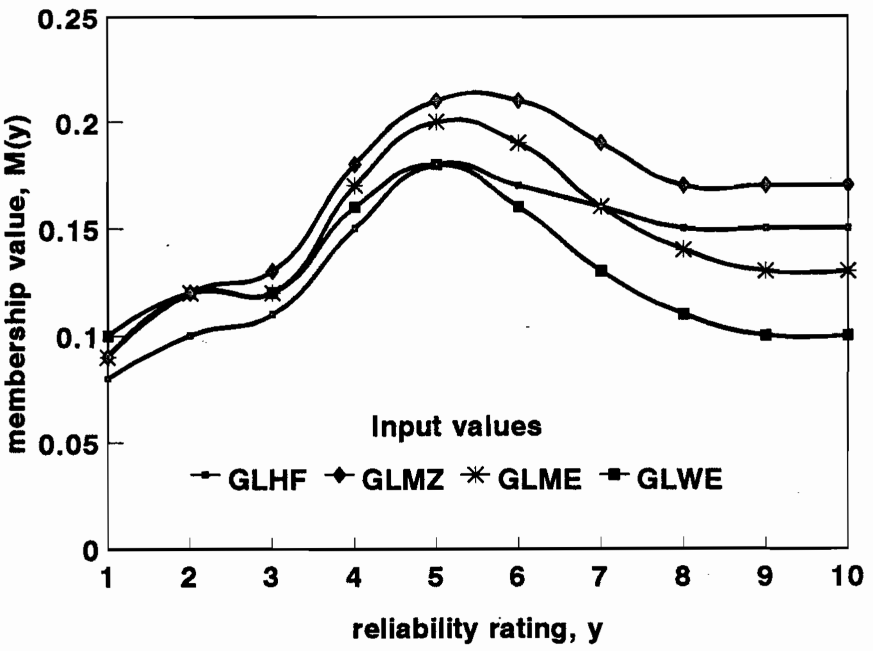


Figure 4.12 Systematic membership functions of stope phases during extraction

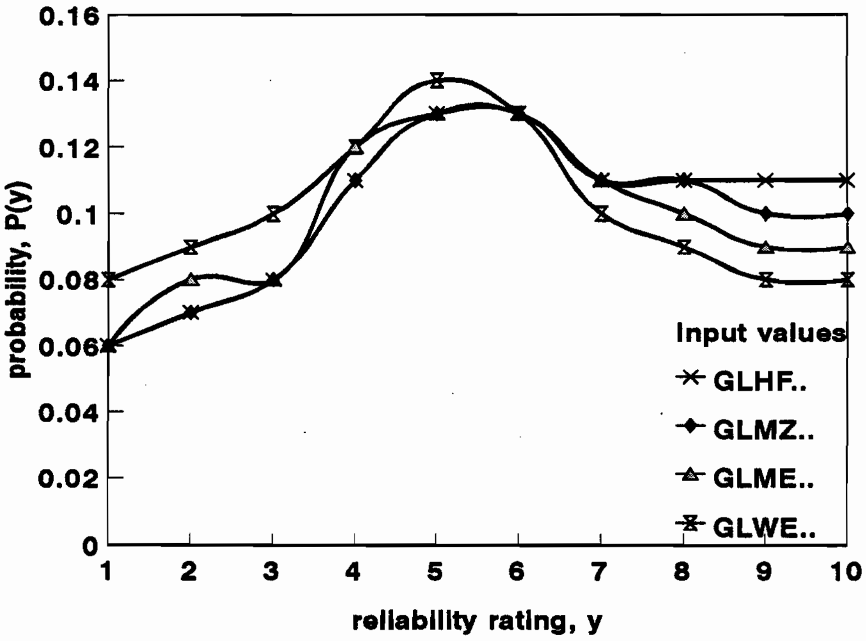


Figure 4.13 Systematic probability distribution of stope phases during extraction

4.10 Model Limitations

The MISO linguistic model described here allows the subjective assignment of the linguistic terms to the fuzzy set elements and through approximate reasoning logic, it infers the output. A limitation of this approach is the impossibility of including quantitative data that may be available. Despite this shortcoming, the approach is a significant improvement on current industry practice in mine production planning and scheduling.

Since the knowledge base is static, it requires re-coding of the model if some changes are required. This makes the model less flexible. However, by appropriately modifying the membership functions of the various variables, e.g. by using the complement membership when the observed variable characteristic is the opposite of the coded relationship, correct results are obtained.

The membership functions are precise yet the rules are imprecise. Therefore, the successful use of this or any other fuzzy system depends on how successful the membership functions relate to the linguistic variables. If the site experts cannot agree on a set of fuzzy rules, linguistic values and membership functions, then a fuzzy logic system cannot be applied in decision support systems.

4.11 Summary

A fuzzy MISO model for evaluating stopes has been developed for use as a decision support tool at the production scheduling stage. The approach recognizes the many factors that influence stope grades. The decision maker(s)' description of each of the four input variables, namely:

- cumulative ore drawn from each stope at point of decision-making;
- the sampling information knowledge-base used in each stope design;
- the structural geology and geometric features of each stope, and,

- the wall rock mass characterization and/or backfill sequence and mining induced stress distributions;

are the input required to determine an integrated net dilution effect expected for a stope being considered for scheduling. Post mining back-analysis of such a data-base in an operating mine would lead to fine-tuning of the variables through better modelling of the variable membership functions.

The result of the MISO model is the ranking of the stopes according to their ability to satisfy a quality constraint. The higher the aggregation value, the more acceptable is the stope with respect to meeting the quality constraint. In addition, the probability distribution and fuzziness of the results are determined. This allows the decision maker to assess the risk associated with meeting his/her objectives by using a particular stope.

A major contribution of the FLSM in production planning is in its consistency in decision-making. Therefore, the model is a useful decision-support tool for mines that muck from several stopes. The model can also be applied to a group of mines that supply a common processing plant. The levels of reliability of the mining grades of different stopes in one or more mines are applied to a goal programming model as coefficients of the quality constraint. The goal programming model is described next, in chapter 5.

The model is also a useful approach for training new recruits for production planning at an operation because it allows the harmonisation of decision making. The fundamental requirement is the description of the variable membership functions by experienced personnel. This information can be used repeatedly until such a time when the behaviour of the variables change. Variables are typically expected to change in the medium to long term.

Chapter 5
Goal Programming Modelling

5.1 Purpose

The objective of this chapter is to generate work shift schedules using a multiple objective goal programming approach. The reliability or certainty in the diluted stope grades as assessed by the fuzzy logic stope model is applied to the ore sources quality coefficients in a goal programming model. The credibility of the schedules is assessed by the magnitude of the goal programming objective function. In general, the smaller the objective function value the more acceptable is the schedule because this implies greater total resource utilization in the mine production system.

5.2 Scope and Limitations

A linear multiple objective goal programming model that allows the setting of the production goals at either the same or different priority levels as well as the same or different differential weights for those goals at the same priority is described.

The limitations imposed by the linear nature of the model are shown to be negligible because the decision variables are formulated as real numbers and not integers. The schedules are to be implemented through dispatch systems that allow fractional utilization of equipment in different work areas.

5.3 Introduction

Real world problems are generally constrained by input variables. For example, a firm can have many investment opportunities at one point in time but its financial and human resources are usually limited to undertake all the options. Similarly, a public school may wish to raise its funds by enrolling many students but the school has a capacity it cannot exceed without engaging in capital expansion. Even simple decisions like going from one place to another are constrained in that there is a critical time requirement typified by the shortest route between the two points. Therefore, in most instances both the lower and upper bounds are placed on the decision maker(s). These constraints limit the number of feasible solutions to a finite set. The quest of decision makers is therefore one of identifying the best and hopefully an optimal solution from the feasible set.

The production planning stage whether in mining, FMS, agriculture, or another industry has basic features that distinguishes it from the long and medium term planning. These are the relatively large number of decisions that have to be made in very short time spans. The lack of time implies little or no revision of decisions is made. As described above, if any decision is selected from the feasible set, the question is whether that decision was the best. This is a fundamental problem in operational planning. Also, due to the large number of decisions that are made over short time spans, the aggregate effects of these decisions can have a global influence. This cumulative behaviour is contrary to the idea that short term decisions have little impact on a system performance.

This chapter outlines the mining production scheduling problem. A goal programming approach was selected because:

1. It allows simultaneous solution to multiple objectives which are generally found in production planning.
2. It is more flexible than linear programming in that it always generates the best solution for either single or multiple objective problems under a given set of constraints whereas linear programming could generate an infeasible solution for the single objective problem. Goal programming is better than dynamic programming if multiple goals are involved. The latter requires the division of the problem into smaller units that are then solved in a sequential manner. The shift production schedule is the smallest time unit of interest hence a further division is not useful as it generates a sequence of mining that in practice cannot be

followed systematically. Such a sequence is violated at implementation due to the system dynamics as well as by the need to satisfy those goals not considered in the dynamic programming model.

A goal programming algorithm is described for solving multiple objective scheduling problems. A computer code based on the goal programming algorithm written in C language is used to solve a typical mine shift production schedule. The results of the model execution are illustrated and their significance discussed. The basic assumptions for the application of the method are presented.

5.4 Goal Programming Algorithm

The algorithm for solving problems with linear constraints is a modification of the Simplex procedure for solving either maximization or minimization linear problems. The theoretical aspects of goal programming consist of a two step procedure that progressively searches for a solution that minimizes the *total absolute deviation* of multiple objectives from their targets. The two steps are the formulation of a problem into a base matrix of decision variables and an iterative computation of the algorithm to obtain the optimal solution. The matrix is referred to as the initial tableau.

5.4.1 Initial Tableau

A generalized goal programming problem is outlined below to show the procedure of generating the initial tableau. Supposing that two objectives exist as:

$$\textit{Maximize}: \; z_1 = c_1 x_1 \tag{5.1}$$

$$\textit{Minimize}: \; z_2 = c_2 x_2 \tag{5.2}$$

$$\textit{subject to}: \quad a_{11} x_1 + a_{12} x_2 \leq b_1 \tag{5.3}$$

$$a_{21}x_1 + a_{22}x_2 \geq b_2 \tag{5.4}$$

$$x_j \geq 0 \tag{5.5}$$

where x_i = decision variables, i = 1, 2

c_j = cost coefficients of decision variable in an objective j. j = 1, 2

b_i = available resource for use in the system constraint i, and is the right-hand side term (RHS); and

a_{ij} = rate of consumption of the resource by variable x_i in constraint j. a_{ij} is also known as technological coefficient of the variable x_i in constraint j.

This problem is re-formulated using deviation variables δ^+ and δ^- to make all system constraints (equations 5.3 and 5.4) equalities and the two objectives (equations 5.1 and 5.2) into goal constraints. The targets of the initial two objectives become their right-hand values V and W respectively. The summation of the product of weights, w_i and deviation variables at their appropriate priority P_i, become the objective function Z, which has to be minimized. If the deviation objective function is minimized, then the original objectives would have been attained with minimum sacrifice, hence the satisfying approach. The re-formulated problem becomes:

$$\textit{Minimize:}\ Z = P_1 w_1(\delta^+_3 + \delta^-_3) + P_2 w_2(\delta^+_4 + \delta^-_4) \tag{5.6}$$

$$\textit{subject to:}\ a_{11}x_1 + a_{12}x_2 + \delta^-_1 = b_1 \tag{5.7}$$

$$a_{21}x_1 + a_{22}x_2 - \delta^+_2 = b_2 \tag{5.8}$$

$$c_1 x_1 + \delta^-_3 - \delta^+_3 = V \tag{5.9}$$

$$c_2x_2 + \delta^-_4 - \delta^+_4 = W \tag{5.10}$$

$$x_j, \delta^+_j, \delta^-_j \geq 0 \tag{5.11}$$

Equations 5.7 and 5.8 represent system constraints. Such constraints are inflexible and cannot be violated in the problem solving. They are assigned the highest priority P_0, in the initial tableau. Equations 5.9 and 5.10 are the goal constraints derived from the previous problem objectives, z_1 and z_2 respectively. In some cases, only one deviation type is of interest, that is, positive or negative deviation of a decision variable appears in the new objective function, Z (equation 5.6). For example, if one objective was to minimize cost, then it is the positive deviation above the budget that is of interest and is the one that enters the objective function. The algorithm applied here always requires a positive deviation to be present in the initial formulation of the constraints because the method approach is based on the elimination of positive deviations, hence minimizing the objective function, Z.

An initial basic feasible solution is obtained by making the positive deviations the subjects of the constraints in the re-formatted system of equations. Zero values are then assigned to all decision variables (x's). The result is the positive deviations equal to the right-hand side values. The negative deviations are zero, since if one deviation type exists then its opposite signed deviation is zero.

An initial tableau for the above example is illustrated in Table 5.1 using the approach by Schnierderjans and Kwak [1982].

Table 5.1 Initial goal programming tableau

			NON-BASIC VARIABLES						
	PRIORITY ROW ->		0	0	0	0	w_1P_1	w_2P_2	Rows 1
PRIORITY	SOLUTION (read off these two column)		x_1	x_2	δ^-_1	δ^-_2	δ^-_3	δ^-_4	2
COLUMN	BASIS Column	Z_value $\Sigma \mid\delta^+ + \delta^-\mid$	0	0	0	0	w_1	w_2	3
$0P_0$	δ^+_1	b_1	a_{11}	a_{12}	0	0	0	0	4
$0P_0$	δ^+_2	$-b_2$	a_{21}	a_{22}	0	0	0	0	5
w_1P_1	δ^+_3	V	$-c_1$	0	0	0	1	0	6
w_2P_2	δ^+_4	W	0	$-c_2$	0	0	0	1	7
Columns 1	2	3	4	5	6	7	8	9	

5.4.2 Iterative Step

The iterative steps are performed on the initial tableau and its successors until a solution is obtained. The procedure consists of selecting the highest priority variable from the basic solution of the tableau and interchanging it with a non-basic variable with least contribution to the minimized objective. The elements in the resulting tableau are computed as described below. The following iterative steps use Table 5.1 to illustrate the column and row elements of the algorithm.

1. Goal programming technique aims to satisfy a multi-objective problem in the order which the objectives are prioritized. If the priorities are the same then the differential weights are used to order the variables. Therefore, the procedure removes the variable in the solution basis (column 2) with the highest priority and

re-locates that variable in the non-basic variable columns (columns 4 to 9). If the priority and weights are equal, selection is based on the largest value of the right-hand side constant. The selected basic variable is the "out-going" basic variable and the row which contains it is the pivot row. This step reduces the deviation of the highest ranked, most weighted and largest right-hand valued goal.

2. The non-basic variable that is exchanged with the "out-going" basic variable into the solution basis is determined through:

   (a) Selection of columns with priorities, P_{next} lower or equal to that of the pivot row, i.e. $P_{next} \leq P_{current}$. A lower priority cannot substitute for a higher priority goal.

   (b) Dividing the positive technological coefficients (elements in the array row 4/ column 4 to row 7/column 9 inclusive) into their objective function weighting coefficients in Row 3 , i.e. w_j/a_{ij} and,

   (c) Selecting the smallest resulting ratio of the division in step (b) above, i.e. $\min\{w_j/+a_{ij}\}$. The variable in this column (i.e. one of the terms in row 3) is the next "in-coming" variable of the basis solution. The column with the "in-coming" variable becomes the pivot column. This step has the least impact on the objective function since the selected variable has the least contribution to Z, since Z is being minimized.

Exchanging the "out-going" and "in-coming" variables in steps 1 and 2 implies that the greatest deviation contributing variable is eliminated from the basis and is replaced by a least contributing one from the non-basic variable columns. This is the fundamental technique of goal programming which has to be repeated until no further improvements (i.e. minimization of the total deviation) can be achieved at which time the algorithm terminates. The priorities and weights of the inter-changed variables are moved to the appropriate positions in the tableau.

3. The intersection of the pivot row and pivot column is the new pivot element

position. The reciprocal of the current element at the new pivot element position is the new pivot element. The rest of the pivot row elements are a result of pivot element division of the old tableau elements and changing their signs. The new elements in the pivot column are the result of pivot column elements division by the pivot element. The rest of the tableau elements are determined by subtracting from the old elements the product of the pivot row, pivot column corner elements and the reciprocal of the pivot element.

4. The total absolute deviation value (Z_value) of the variables in the basic solution is the summation of the product of differential weights (in column 1) and the corresponding right-hand side values (in column 3) for all of the solution variables. This value represents the goal programming objective of deviation variables which has to be minimized.

5. An optimum solution is obtained if all the basic variables in the basis solution are positive (i.e. column 3) and at least one value in the objective function weights (i.e. row 3) is negative. Also, if the next variable to enter the basis (i.e. column 2) is at a higher priority than those already in the basis, then the solution is optimal. Returning this variable to the basis would increase the deviation and fail to satisfy a higher priority. If the basic variables are negative, the procedure is repeated from 1 through 5.

6. If no negative value exists in the objective function weights (row 3) while all basic variables are positive, a solution is still not achieved. The next "out-going" variable from the basis is based on the highest weight or largest right-hand value to break the impasse in priority. The row from which this variable is derived becomes the pivot row as in step 1.

7. To obtain the entering non-basic variable, the negative coefficients in the pivot row are divided into their respective positive elements in the row vector of

differential weights. Then the column with the smallest resulting ratio has the "in-coming" non-basic variable i.e. min $\{w_j/-a_{ij}\}$. The column is the new pivot column. The procedure is repeated from steps 3 to 7 inclusive until a solution is obtained.

5.5 Problem Definition

In order to model the goal programming problem, it is essential to first describe the short-term production system, its objectives and restrictions and/or limitations. The model is then formulated to address these issues in a realistic way.

There are generally four bench mark objectives in production systems. They are meeting the production volume targets, achieving maximum process utilization, minimization of production costs and delivering a quality product within the schedule span.

A mine production schedule has to meet the client demands. The client could be a concentrator or a mill. The client demands usually include some or all of the following items:

- ore quality as expressed in the content of the desired metal
- ore quality as expressed in lack of impurities
- ore moisture content within specified limits
- degree of variability in grindability and fragmentation of material so as to optimize the comminution plant
- flow of ore types with different characteristics into the process plant in a situation where ore from different mine sections have different ore chemistry. The greatest problem found in process plants, which have not been fully overcome in flotation control, is the developing of algorithms which accommodate changes in ore types and can define flexible limits to the maximum and minimum amounts of reagents used. Therefore, if a mine can effectively supply a homogeneous

blend of mill-feed, this leads to better downstream plant control. These product quality issues are typical in several Noranda Inc. mines [Larsen, 1995].

The supply of a process facility with a given quality of material that optimizes the downstream functions such as comminution, moisture content and recovery is complex. The main problem is the grade and composition of the mill-feed. Ore streams are defined at the material sources but they undergo transformations during their extraction and transport to the process plant (see Elbrond [1994]). Mineable ore reserves include a planned dilution given as a fraction of the in-situ reserves. At production, the dilution occurrence is not random. It is a function of the mining phase, location of the draw-points and the cumulative extraction from a given draw-point. This results in production periods of relatively low dilution and periods of high dilution, for example from primary to secondary to tertiary stopes.

Besides satisfying the quality requirement, the mine has to meet certain production rates at costs reflected in the evolving medium and long term plans. This issue is important in that the cumulative production effects could significantly alter the strategic plans.

The use of single or multiple objective decision-making methods in the industry has been entirely deterministic. Deterministic production level and resources are used, yet in practical applications the forecasted production and resources are imprecise. Usually the inputs are represented by known subjective or objective distributions which leads to several possible outcomes according to the distribution pattern. In some instances, the probability distributions are unknown such that decisions are made under uncertainty conditions. A method for incorporating some of this imprecision in decision-making was modelled and evaluated in Chapter 4.

5.6 Goal Programming Modelling Prelude

Two plans exist at the production planning stage, one is a ratio of the average day's production to that of a week or month, and the other is the daily plan. The daily plan reflects the actual current mining environment in terms of the availability of resources, ground conditions and the prevailing management objectives of the shift. These two plans have significant differences as the average daily plan is not responsive to dynamic events that exist at implementation. The two plans always co-exist together in an operation with the long-term based plan acting as the theoretical basis for production rationale while the other represents reality.

The targets set by a medium term plan require transformation to give discrete quantities for each scheduled work area. At the production stage, generalized estimates which are a function of drill spacing, assay accuracy, ore continuity, etc., are inadequate unless used in conjunction with the most recent information obtained from a work face. The cumulative tonnages drawn from a stope do in some instances reflect a progression in the dilution or a reduction in quality with depth as reported in the Brunswick Mining & Smelting No. 12 Mine by Grebenc and Welwood [1971]. This implies that grade estimates continuously change due either to geological updates or changes in the ground conditions. If quality is to be met, it is imperative not to ignore these additional influences on the decision variables.

5.6.1 Resources

A scheduler has to be aware of the accumulated work hours of each machine, know its recent performance, and the last time it was out for service. This allows objective decisions on the assigning of different machines, especially to critical areas of production. Paraszczak and Perreault [1994] show that for some LHDs, the failure rate increases with machine age.

Rock fragmentation is non-uniform within a stope hence the ore draw characteristics is affected by over-sizes and choking of draw-points. Hill [1987] simulated the effects of over-sizes on ore production rate and indicates that production drops if the frequency of over-sizes is high. Such information lacks in long term average daily plans but is available to the decision maker on the current status plan.

5.6.2 Layout

The stope layout (sizes) and sequencing in practice have a direct relationship to the expected dilution. The size of the stope determines the volume and perimeter surface in contact with the wall rock and/or backfill material. Using acceptably (for stability) large stopes gives large ore volume to ore surface ratio, hence a lesser wall dilution factor. This dilution is expected to increase towards the stope perimeter. Therefore, if a daily production is coming from the centre draw-points, intuitively little or no dilution is expected upon which the estimated grades are permissible to apply without factoring. Primary stopes in 1-3-5, and checker board sequences are devoid of rockfill dilution. But the secondary stopes are exposed to two or more diluting sides in 1-3-5 sequence and checker board sequences.

Similarly, perimeter ore in the upper levels of high VCR stopes is exposed to a wall contact the height of the stope. If we assume a unit dilution per unit length of stope height, a linear function is obtained. In practice, this relationship is concave upwards indicating increasing dilution effects with stope height.

As part of mine automation, we propose that a database be kept of the relationships between rock dilution and stope planar size, height and sequence. Sampling intensity distribution in each work place is also an important parameter as it measures the accuracy of the estimated grades and tonnages. Supposing, lower grades are being drawn from an area with low sample intensity, it can be inferred that some internal dilution is occurring from a waste lode within the stope. This situation is not atypical, as many mine operators have had the bad experience of discovering previously unknown faults truncating the orebodies. The old stope design becomes inapplicable.

One advantage of decision making at the production level is the availability of information on ground conditions. If the ground is deteriorating more than predicted, these effects need to be factored into each affected work area. Conversely, if the conditions are stable, planned dilution may require revision to reflect the prevailing situation. A ground control check-table is proposed in each mine with potential instability problems. The relative importance of these factors is site-specific and empirical.

Despite the planned dilution at medium-long term planning, dilution is only an estimate that is subject to change during operations. At the operational level, such estimates become irrelevant if they fail to predict the on-going process. Since quality of the product is perhaps the most important aspect in mining, the quick response to its changes and readily determining the possible causes of the deviations from targets is required. This objective has been dealt with in Chapter 4 under fuzzy modelling of the material sources' potential to dilution at mining stage.

5.7 Model Assumptions

The goal programming model is based on the following assumptions about the operation setup:

- The average equipment productivity per hour is known and it includes the times lost due to breaks and minor delays within an hour. If two or more different machines are available, then their weighted mean productivity is used. The consideration of idle times is important when it comes to the reconciliation of this model output with the implementing dispatch model output at each time window.

- The system congestion within the mine is prevented by routing procedures and/or because few mines have over capacity. The goal programming cannot solve system congestion, therefore it is important for the modeller to fix boundaries in the problem formulation to prevent unrealistic results. For example, a stope production may be set so high that it cannot be practically achieved due to machine congestion even though cost or quality objectives are optimized at such an unrealistic stope production level.

- The fixed material handling facilities, namely the ore-passes holding capacities are known. Equally, the expected available tonnages at each stope or draw-point have to be known. Rather than using the global stope or draw-point tonnages and grades, it is only a fraction of this material that is feasibly accessible and can be considered for scheduling.

- Material types and qualities are known according to grade control information. This may not be entirely true as discussed in Chapter 4. The results of Chapter 4 allow ordinal ranking of work areas according to the reliability of reporting the mining grades. Through this rational process, it is possible to prioritize and assign weights to the different stopes.

- The destination of materials from a given source is not restricted to one dump point. There is flexibility for a scheduled stope tonnage to be dumped in any number of dump points as long as the dump points are designated for that material type, i.e. ore or waste. However, different routes are associated with different costs due to the haulage distances involved and road suitability. This aspect reduces flexibility if cost minimization is an explicit goal.

This model is general such that it can be used to schedule production work in a multi-level, multi-source and multi-dump point mine layout. The model generates a schedule which is a snap shot of these static conditions under the system constraints and the various set objectives.

5.8 Goal Programming Model Formulation

The goal programming procedure is illustrated in Figure 5.1. The process requires a listing of the objectives and goals as well as the resources available and the system constraints. This information is then re-arranged into an initial tableau as discussed earlier. The problem is then solved by the described iterative process of objective function minimization. A computer code in C language was developed and is used to produce typical schedules, described in section 5.10 of this chapter. The program listed in Appendix B, runs under disk operating system (DOS). The solution of the goal programming model is used to allocate production equipment for a shift. If during the shift sudden changes occur, such as machine breakdowns, heavy oversize frequency or large quality deviations, then the scheduler has the option to re-schedule the remainder of the shift. At re-schedule, the scheduler takes into consideration the production to date and the current system configuration to create a new shift scheduling formulation.

The decision variables for a production plan are basically the tonnages (or volumes) that need to be mined from each material source within a planning period (shift). Since multi-

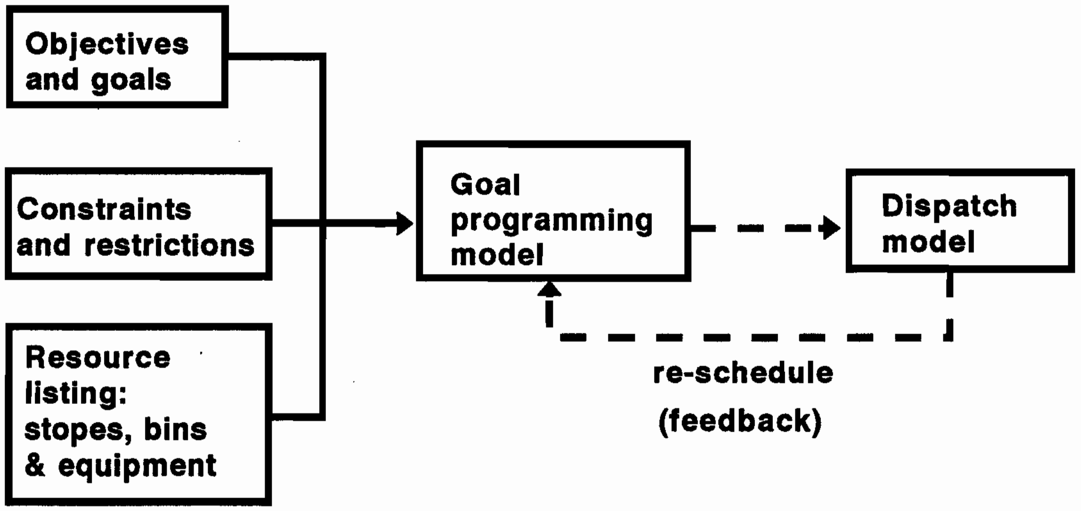


Figure 5.1 The goal programming procedure

point dumping is allowed from one source, each source-to-dump route flow can be calculated, noting that the total flow from one source should equal that stope's scheduled quantity.

The decision variables are denoted X_{kij} and W_{kij}, for ore and waste material respectively. Subscript i represents the stope or draw-point number, j represents the dump point and k is the k^{th} level or work section. X_{kij} or W_{kij} represents material flows between a source and a dump-point. The scheduled stope tonnage is easily obtained by summation of these flows for each i^{th} stope. Similarly, the total scheduled tonnage to be dumped in a particular dump-point is the summation of all the flows into that dump-point.

The deviation variables are the under- and over-achievement of the goals. They are denoted by δ^- and δ^+ respectively. The concept of deviation variables implies that the goals are known. The level of certainty of a goal is emphasized as the need to represent the current mining situation with the most updated information available and use of trend analysis to make forecasts of probable future outcomes. The goal constraints are formulated based on system performance evaluation criteria: the relaxation of these constraints allows for the analysis of their effect on overall system performance.

5.8.1 Production Bench-marks and Objectives

Typical bench-mark performance criteria for underground mining are listed below:

- Production

The trammed material is either ore, waste or both. This work models the later situation which wholly describes the production activities. The objective is to maximize ore production and fully achieve waste tramming. Full achievement of waste mucking is due to the assumption that all waste is from a critical mine development activity, therefore the headings have to be mucked clean to prevent the development project from falling behind its schedule. This objective can be mathematically represented by the following expression:

$$\max: \sum_{j=1}^{P}\sum_{i=1}^{N}(X_{ij}-X_t)_k + \sum_{j=P+1}^{Q}\sum_{i=N+1}^{M}(W_{ij}-W_t)_k, \quad \forall\ k\ levels \tag{5.12}$$

where: P = number of ore dump points on level k

N = number of ore draw-points (or stopes) on level k

Q = total number of dump-points on level k, i.e. both ore and waste

M = total number of material sources on level k, i.e. both ore and waste

X_t = actual ore production achieved up to a time t and

W_t = actual waste production up to a time t of the shift.

This objective function is then transformed into a goal constraint by introducing deviation variables as surplus and slack, and setting the objective to some target production level, T. The resulting production goal constraint is given by:

$$\sum_{j=1}^{P}\sum_{i=1}^{N}(X_{ij}-X_t)_k + \sum_{j=P+1}^{Q}\sum_{i=N+1}^{M}(W_{ij}-W_t)_k + \delta_1^- - \delta_1^+ = T \tag{5.13}$$

The negative deviation variable enters the goal program objective function since by minimizing the under-achievement, the objective of maximizing production to a desired level (T) is achieved. The negative deviation variable appears in the objective at a priority and weighting commensurate to the management's requirements.

The quantities X_t and W_t are cumulative production of the current shift up to a time t. Such data is required on-line for re-scheduling of the goal programming model should the control algorithm, described in Chapter 6, find the shift goals unattainable.

● Production Cost

Excessive tramming distances are costly. It is therefore usual to have an objective to minimize such travel distances especially when loaded. Mine information indicates that haulage cost variation with distance is non-linear, generally concave upwards. An appropriate objective links the tonnage trammed over each distance to the cost of a haulage section. Basically, this would require the creation of a mine roadway cost network where sources and dump points are nodes and the roadways are the branches. We term this objective a work-distance-cost function and express it algebraically as:

$$\textbf{min:} \sum_{j=1}^{P}\sum_{i=1}^{N} c_{ij}d_{ij}(X_{ij} \ - X_t)_k + \sum_{j=P+1}^{Q}\ \sum_{i=N+1}^{M} c_{ij}d_{ij}(W_{ij} \ - W_t)_k, \ \forall\ k \ levels \quad (5.14)$$

where c_{ij} = cost coefficient per unit distance on route ij.

c_{ij} takes into account the road grade, and possible traffic congestion.

d_{ij} = distance between source i and destination j

Short and medium term plans indicate the expected mining cost, namely the mucking and haulage costs per tonne. By applying that cost to the required shift production, a target shift budget is obtained. A cost goal constraint is formulated from the equation (5.14) objective with a resource of value C and is defined by:

$$\sum_{j=1}^{P}\sum_{i=1}^{N} (c_{ij}d_{ij}(X_{ij}-X_t))_k + \sum_{j=1}^{P}\sum_{i=1}^{N} (c_{ij}d_{ij}(W_{ij}-W_t))_k + \delta_2^- - \delta_2^+ = C \qquad (5.15)$$

The over-budget deviation variable, $\delta^+{}_2$ at set management priority and weight enters the goal objective function since by minimizing this variable the cost objective is achieved.

● Product Quality

The quality objective is perhaps the most difficult to formulate as it is made under uncertainty. We propose to transform the stope i grade estimates, g_i by a factor Ψ_i. The factor represents the evolution of the real situation and is objectively (if historical data is available) or subjectively determined from the Fuzzy Logic Stope Model aggregation and fuzziness measures described in Chapter 4. The psi factor can be defined in one of many ways as:

$$\Psi_i = \frac{\textit{aggregation of stope } i}{\textit{aggregation of best of } N \textit{ stopes}} \tag{5.16}$$

where N = total number of available ore sources

Supposing the objective is to continue to supply the client with a continuous feed, the goal is not entirely a maximization. Rather, a quality is required that is within the tolerance levels of the process plant. This leads to a strictly bounded objective mathematically expressed as:

$$\textit{Optimize}: \ \Omega \leq \sum_{j=1}^{P}\sum_{i=1}^{N} (\Psi_i g_i(X_{ij}-X_t))_k \leq \Phi, \quad \forall\ k \textit{ levels} \tag{5.17}$$

where: Ω = lower limit of the quality consideration , and

Φ = upper limit of the quality objective.

We note that these limits are in units of contained mineral, e.g. grams in the desired production tonnage target, T mentioned above. The modeller has to multiply say a grade with the required tonnage to obtain the right-hand values in (5.17) re-formulation. This constraint can be transformed into either two goal constraints as:

$$\sum_{j=1}^{P}\sum_{i=1}^{N} (\Psi_i g_i(X_{ij}-X_t))_k + \delta_{3a}^- - \delta_{3a}^+ = \Phi \tag{5.18}$$

and

$$\sum_{j=1}^{P}\sum_{i=1}^{N} (\Psi_i g_i(X_{ij}-X_t))_k + \delta_{3b}^- - \delta_{3b}^+ = \Omega \tag{5.19}$$

or as a single constraint represented by:

where: g_p= planned average grade in the short term.

$$\sum_{j=1}^{P}\sum_{i=1}^{N}[(\Psi_i g_i - g_p)(X_{ij}-X_t)]_k + \delta_{3c}^{-} - \delta_{3c}^{+} = 0 \tag{5.20}$$

where: g_p = planned average grade in the short term.

In the first format, the deviation variables δ^{+}_{3a} and δ^{-}_{3b} enter the objective function as they each make the upper and lower limits of quality deviations respectively. In the later format both the under-, δ^{-}_{3c} and over-achievement, δ^{+}_{3c} deviation variables enter the objective function. These two formats are not equivalent. Equation (5.20) attempts to minimize deviations from the mean grade and a situation arises in which the optimized deviations may still be large, perhaps outside the limits of Ω and Φ. Equations (5.18) and (5.19) minimize deviations of quality variations at the management limits. In this regard, management is indifferent of variations within the set limits. Therefore, selection of one format over another is entirely site specific with respect to what is a better approach as it is management objective dependent.

A productivity objective can be described following the same procedure as the other three goals above. However, we implicitly consider productivity as a system constraint in the setup of possible amounts of work in the active work areas.

5.8.2 System Constraints

The system constraints represent the environment under which the schedule is made. It is a listing of the conditions that have to be honoured. The available resources and limitations such as prohibited draw strategies or setup of equipment in any one place are typical examples of system constraints. In the following paragraphs, we describe the features of these constraints.

● Minimum Production Requirement

Work area supervision requirement limits the possible number of work areas operating concurrently. The optimal goal programming solution may have schedules of a few loads from some stopes which would necessarily increase supervision costs. Management therefore usually set minimum quantities L_i, that it deems acceptable for resource allocation on each kth level. This strategy constrains the model leading to a compromised sub-optimal solution. The constraint is stated as:

$$\sum_{i=1}^{N} X_{ik} \geq L_{ik}, \quad \forall\ k\ levels \tag{5.21}$$

where: L_{ik} = minimum acceptable stope or draw-point tonnage that has to be scheduled for the work area, i on level k to be active.
If the tonnage is less than L_{ik} then the stope is not scheduled for production in that shift.

The waste W_i, has been assumed to come from development activities and is critical that it is mucked irrespective of the quantity involved. If this condition is invalid, then similar constraints of minimum permissible production as for ore sources are set.

● Maximum Equipment Allocation

The underground mining environment is tightly constrained by space limitations. This limits the number of machines that can be allowed to operate in any one section simultaneously for safety reasons as well as lost production through excessive machine interferences. The constraint is algebraically expressed as:

$$\sum_{i=1}^{N} \frac{X_i}{Z_i} \leq J_k, \ \forall\ k\ levels \tag{5.22}$$

where: J_k = maximum number of machines allowed on level k, and

Z_i= machine productivity working in stope i of level k.

The value of Z_i is calculated on the assumption of dedicated equipment allocation to work between one source and one dump-point. The number of loads per shift for each branch of the work-distance-cost network is determined. If two or more dump-points are possible for a single source, the arithmetic mean of loads per shift is taken as the Z_i value. For different size machines, the average productivity of the fleet is used in the determination of the Z_i.

- Dump-point Capacity

The scheduled tonnage per level has to comply with the available dump space D_j, on that level. This condition may be necessary where an ore or waste pass draw policy is practised for ground control. In such circumstances, the passes are maintained with a small dump space. If material is pulled from holding facilities, the total scheduled quantities cannot exceed the current quantity in the facilities. The ore and waste dump-point constraints are respectively expressed as:

$$\sum_{j=1}^{P}\sum_{i=1}^{N} X_{kij} \leq D_{kj},\ \forall\ k\ levels \qquad (5.23.a)$$

$$\sum_{j=P+1}^{Q}\sum_{i=N+1}^{M} W_{kij} \leq D_{kj},\ \forall\ k\ levels \qquad (5.23.b)$$

The system constraints are transformed into equalities through the introduction of deviation variables similar to the goal constraints. However, the priorities of all system constraints are set at a higher level than any goal constraint(s). This is necessary to prevent their violation during the algorithmic computation. The system deviation variables are not part of the objective function of the goal model.

5.8.3 Objective Function

The goal objective function based on the above three objectives is expressed as:

$$\min: P_1 w_1 \delta_1^- + P_2 w_2 \delta_2^+ + P_3 w_3(\delta_{3c}^- + \delta_{3c}^+) \qquad (5.24)$$

where P_i = priority ranking of objective i

w_i = differential weight placed on objective i

If all objectives have the same priority, then the P_i's are equal to one and similarly for the differential weights w_i's. It should be noted that in the third objective both the priority and differential weight are not necessarily equal. An option exists to differentiate between deviations of a particular objective. For example, whilst not very desirable to feed the process plant with very high grade material, it is definitely much preferred to supplying marginal ore to a process plant. In such an instance, the differential weight placed on the negative deviation variable is higher than that on the positive (high grade) deviation variable of the same objective.

5.9 Goal Programming Example

A hypothetical nickel-copper metal mine using an open-stoping mining method with delayed backfill called Inza Mine is considered. The mine is producing from two levels, L100 and L200 respectively, spaced at 100 metres and linked with a 15 percent ramp. L100 has three active stopes, two primary and one secondary stope. L200 has two active stopes also a primary and secondary and one development section. The mineralization in all the stopes is heterogeneous and grades vary significantly between stopes as shown in Table 5.2. A tight geological grade control is kept based on exploration and stope definition drilling as well as production drilling chip samples. The mine routinely takes grab samples at the active draw-points for comparison with the grade database based on drilling. Grab sampling is also used to see dilution effects of mining secondary and tertiary stopes. The detailed information on the available ore and its grade for the coming production shift is shown in Table 5.3. All reserves are in tonnes.

Table 5.2 Inza Mine broken ore reserves for 1995

STOPE	Cu %	Ni %	RESERVES, t
L100 No. 1	2.50	2.75	30000
L100 No. 2	2.75	3.00	75000
L100 No. 3	1.75	1.50	87050
L200 No. 4	3.00	2.50	40000
L200 No. 5	1.50	4.25	15000

Table 5.3 Inza Mine production reserves on DD-MM-1995

Stope	Cu %	Ni %	Accessible reserves, t
L100 No. 1	2.5	1.3	800
L100 No. 2	2.2	2.9	800
L100 No. 3	0.7	2.8	500
L200 No. 4	2.6	3.0	600
L200 No. 5	0.6	2.85	1000
L200 sect F	--	--	100

Ore is trammed from scheduled draw-points to ore-passes on both L100 and L200. The waste material from development areas is utilized as rockfill on the lower levels. Currently, waste from L200 development is trammed to one chute where it is dropped down to L300 for rockfill purpose. Waste in development drifts is always a priority as it delays the development drill and blast activities. Therefore, whenever there is such waste, management attempt to muck it out fully or consider it the highest priority task.

The mine has a fleet of seven load-haul-dump machines. The fleet consists of three relatively new Wagner ST6C and the remainder are old Wagner ST6B. The mine operates two 8-hour shifts. However, the actual operating hours per shift is in the range of four to five hours due to delays in operator transportation, start-up times and incidental time losses.

The ore is bi-metallic and this poses blending problems for the concentrator. The mine is the sole supplier of ore to the concentrator. The mine is in the process of determining a flexible method to schedule its production in order to reduce the variability of the concentrator feed grade. Besides, it requires a method that takes into consideration mining cost minimization whilst at the same time achieving target production levels.

A progressive depletion chart is kept of each draw-point/stope. This allows the estimation of the mineable quantities still held within the stopes. The ore grades are assigned to this ore on the assumption of a mass flow behaviour of the muck within the stope. Some factors are possibly applied to reflect the stope design and some likeliness of funnel flow as the prevalence of erratic flows and fragmentation sizes.

The design mine production capacity is 400K tonnes per year of ore grading at 2.33% copper and 2.43% nickel; and 75K tonnes waste rock. The mine requires the mined ore to have grades as close as possible to these medium term average grades of 2.33 and 2.43% copper and nickel respectively. This implies a daily production of 2250 tonnes for a 210-day operating year. The mine layout is illustrated in Figure 5.2. The cost-distance network for the mine layout is represented by the schematic diagram indicated by Figure 5.3. The cost figures represent the cost per tonne for tramming along each route from a source to a dump-point. A target daily mucking and tramming budget of $6200 is aimed.

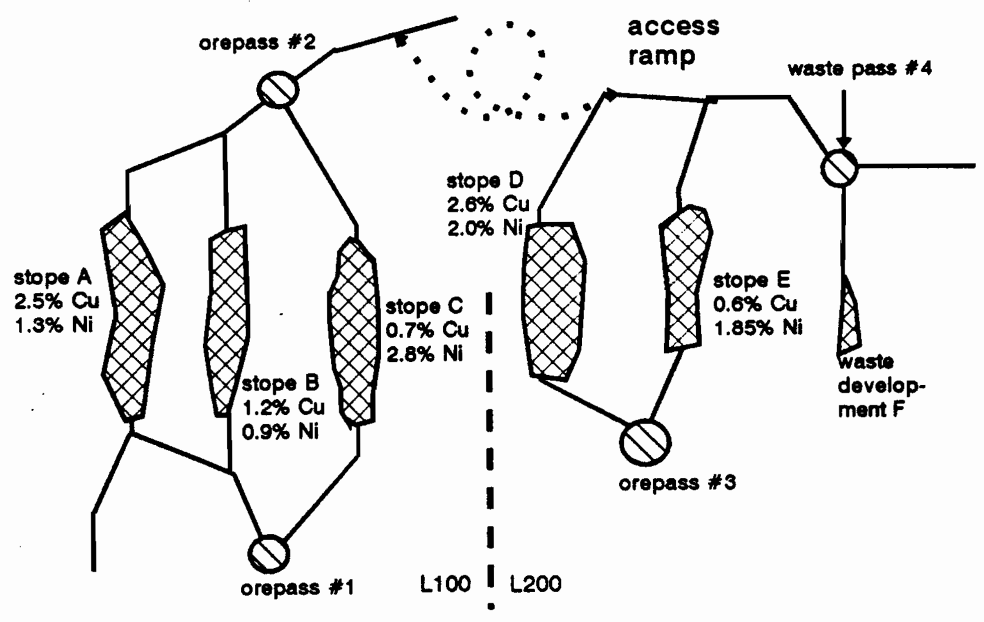


Figure 5.2 Inza Mine layout showing stopes and ore-passes

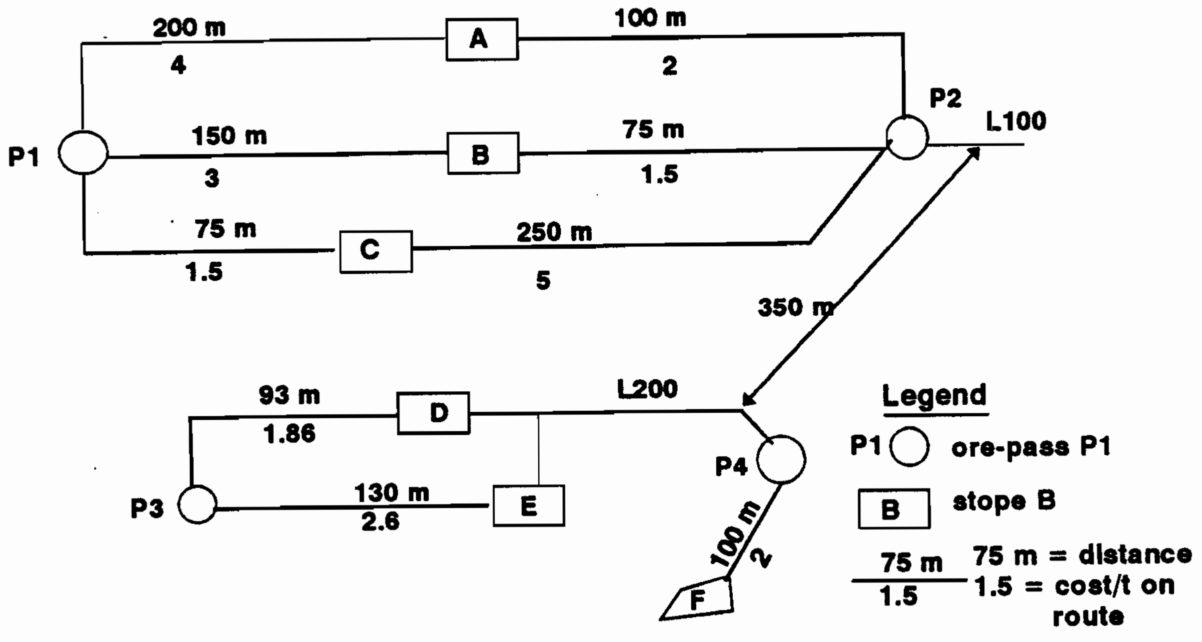


Figure 5.3 Schematic network of the Inza Mine layout

The formulation of this problem is illustrated as follows, in which D_k and Z_k are the negative and positive deviation variables for goal k, and X_{ij} is a flow between source i and destination j:

● Production goal using equation (5.13)

$$X_{11}+X_{12}+X_{21}+X_{22}+X_{31}+X_{32}+X_{43}+X_{53}+X_{64} + D_1 - Z_1 = 2250 \qquad (5.25)$$

● Cost goal using equation (5.15)

$$4X_{11}+2X_{12}+3X_{21}+1.5X_{22}+1.5X_{31}+5X_{32}+1.86X_{43}+2.6X_{53}+2X_{64}+D_2- Z_2 = 6200 \qquad (5.26)$$

● Quality goal using equation (5.20) for copper

$$(2.5-2.33)(X_{11}+X_{12})+(2.2-2.33)(X_{21}+X_{22})+(0.7-2.33)(X_{31}+X_{32}) +$$
$$(2.6-2.33)X_{43}+(0.6-2.33)X_{53} + D_3 - Z_3 = 0 \qquad (5.27)$$

● Quality goal using equation (5.20) for nickel

$$(1.3-2.43)(X_{11}+X_{12})+(2.9-2.43)(X_{21}+X_{22})+(2.8-2.43)(X_{31}+X_{32}) +$$
$$(3.0-2.43)X_{43}+(2.85-2.43)X_{53}+D_4 - Z_4 = 0 \qquad (5.28)$$

● The system constraints are:

$X_{11}+X_{12}+X_{21}+X_{22}+X_{31}+X_{32} \leq 1550$ ... Equipment limit on L100 (5.29)

$X_{43}+X_{53}+X_{64} \leq 710$ ... Equipment limit on L200 (5.30)

$X_{11}+X_{21}+X_{31} \leq 1500$ ... Dump space at ore-pass #1 (5.31)

$X_{12}+X_{22}+X_{32} \leq 2000$ ... Dump space at ore-pass #2 (5.32)

$X_{43}+X_{53} \leq 1000$ ... Dump space at ore-pass #3 (5.33)

$X_{64} \leq 1000$ ... Dump space at ore-pass #4 (waste) (5.34)

$X_{11}+X_{12} \leq 800$ ... Production cannot exceed available ore (Equation 5.23a and b) (5.35)

$X_{21}+X_{22} \leq 800$ ... Production cannot exceed available ore (5.36)

$X_{31}+X_{32} \leq 500$ ... Production cannot exceed available ore (5.37)

$X_{43} \leq 600$ ... Production cannot exceed available ore (5.38)

$X_{53} \leq 1000$ ... Production cannot exceed available ore (5.39)

$X_{64} = 100$ ... Mine all development waste (5.40)

- Permissible number of machines per work section:

$(X_{11}+X_{12})/.003 + (X_{21}+X_{22})/.0023 + (X_{31}+X_{32})/.0026 \leq 6$ ...L100 (5.41)

$X_{43}/.0021 + X_{53}/.0028 + X_{64}/.0023 \leq 4$ ...L200 using equation 5.22 (5.42)

$X_{11} \leq 500$ ... Constraints on possible maximum flows (Equations 5.23a and b) (5.43)

$X_{12} \leq 500$ (5.44)

$X_{21} \leq 500$ (5.45)

$X_{22} \leq 500$ (5.46)

$X_{31} \leq 500$ (5.47)

$X_{32} \leq 500$ (5.48)

The objective function consists of minimizing the total absolute deviation of four goals and is formulated using equation 5.24 as:

$$\text{minimize: } P_1w_1(D_1) + P_2w_2(Z_2) + P_3w_3(D_3+Z_3) + P_4w_4(D_4+Z_4) \quad (5.49)$$

where P_k = priority for goal k

w_k = differential weight for goal k

The priorities P_k can be of different or equal rank for all k (k=4) goals. The weights for goal three and four can be different for the negative and positive deviation variables. Equally, all deviation variables can have the same or different weights in the objective function. We use this aspect in the sensitivity analysis of the case problem to study the effects of different ranks, weights and target goal values (i.e. right hand side) on the output schedule.

The computer code input data file of this model is indicated in Appendix C.

If the quality goal is formulated based on constraints (5.18) and (5.19) such that the management is indifferent if the quality values lie within the range of these two constraints then the copper and nickel goals become:

● Copper grade must be above a minimum value of 1.86% Cu:

$$2.55(X_{11}+X_{12})+ 2.2(X_{21}+X_{22})+ 0.7(X_{31}+X_{32})+ 2.6X_{43}+ 0.6X_{53} + D_5 = (1.86 * 2250) \qquad (5.50)$$

● Copper grade must be less than 2.79%:

$$2.55(X_{11}+X_{12}) + 2.2(X_{21}+X_{22})+ 0.7(X_{31}+X_{32})+ 2.6X_{43} + 0.6X_{53} - Z_6 = (2.79 * 2250) \qquad (5.51)$$

● Similarly, nickel grade must be above 1.94%:

$$1.3(X_{11}+X_{12}) + 2.9(X_{21}+X_{22}) + 2.8(X_{31}+X_{32}) +3.0X_{43}+2.85X_{53}+ D_7 = (1.94 * 2250) \qquad (5.52)$$

● Nickel grade must be below 2.92%:

$$1.3(X_{11}+X_{12})+2.9(X_{21}+X_{22}) + 2.8(X_{31}+X_{32}) + 3.0X_{43} + 2.85X_{53} - Z_8 = (1.94 * 2250) \qquad (5.53)$$

The equivalent goal objective function is of the form:

$$\text{minimize: } P_1w_1(D_1) + P_2w_2(Z_2) + P_3w_3(D_5) + P_4w_4(Z_6) + P_5w_5(D_7) + P_6w_6(Z_8) \qquad (5.54)$$

The system constraints remain the same in both formulations. The later objective function has more goals compared to the former as it has two goals for each product quality rather than one as is the case in the former method.

5.10 Goal Programming Results

A typical computer generated shift production schedule based on the goal programming model and the first data file is illustrated in Figure 5.4. The decision variables (X1 to X9) are the scheduled tonnages from the stopes to the dump points along specific routes and the total scheduled tonnages from each source are indicated in Table 5.4.

Table 5.4 Shift schedule in tonnes of material

Stope	Initial reserves, t	Scheduled tonnage, t
A	800	800
B	800	250
C	500	500
D	600	210
E	400	400
F	100	100

The shift goals are represented by Row 1 to Row 4 inclusive in Figure 5.4. Row 1 is the production goal and it indicates the objective of tramming 2250 t is over-achieved by 10 tonnes. Row 2 represents the shift budget and this is met strictly. Rows 3 and 4 are the copper and nickel grades respectively. By reading off the 'ACHIEVED_VALUE' column the values are negative and each of these values has to be divided by the total scheduled ore tonnage (stopes A to E) which is 2160 t. This results in a deviation in copper and nickel grades of - 0.62% and -0.15% respectively. The negative sign indicates the goals were under-achieved by these values. Since the sought copper and nickel grades were 2.33 and 2.43% respectively, it implies the scheduled tonnages can only achieve copper and nickel grades of 1.71 and 2.28% respectively.

```
THE GOAL PROGRAMMING SCHEDULE OUTPUT
        ======================================

        ***** Morning Shift *****
        **** All objectives are achieved. ****

        DECISION VARIABLES
        -------------------
        Variable            Value
        x1                 367.16
        x2                 432.84
        x3                   0.00
        x4                 250.00
        x5                 182.84
        x6                 317.16
        x7                 210.00
        x8                 400.00
        x9                 100.00


    THE DEVIATIONS FROM SET GOALS
    ==================================

CONSTRAINT  TARGET_VALUE  ACHIEVED_VALUE   GOAL DEVIATION
    ROW 1      2250.00        2260.00         10.00
    ROW 2      6200.00        6200.00          0.00
    ROW 3         0.00       -1346.80      -1346.80
    ROW 4         0.00        -313.80       -313.80
    ROW 5      1550.00        1550.00          0.00
    ROW 6       710.00         710.00          0.00
    ROW 7      1500.00         550.00       -950.00
    ROW 8      1000.00        1000.00          0.00
    ROW 9      1000.00         610.00       -390.00
    ROW 10     1000.00         100.00       -900.00
    ROW 11      800.00         800.00          0.00
    ROW 12      800.00         250.00       -550.00
    ROW 13      500.00         500.00          0.00
    ROW 14      600.00         210.00       -390.00
    ROW 15      400.00         400.00          0.00
    ROW 16      100.00         100.00          0.00
    ROW 17        6.00           4.28         -1.72
    ROW 18        4.00           1.85         -2.15
    ROW 19      500.00         367.16       -132.84
    ROW 20      500.00         432.84        -67.16
    ROW 21      500.00           0.00       -500.00
    ROW 22      500.00         250.00       -250.00
    ROW 23      500.00         182.84       -317.16
    ROW 24      500.00         317.16       -182.84
```

Figure 5.4 Goal Programming program output

```
ANALYSIS OF THE OBJECTIVE FUNCTION
========================================
Priority          Underachievement
----------------------------------------
P5                   0.00
P4                   0.00
P3                 313.80
P2                1346.80
P1                4633.88
Artificial:          0.00

Problem solved in  32 iterations

Process time is  0.439560 sec
```

Figure 5.4 Goal Programming program output (continued)

Rows 5 to 24 represent the system constraints. Where the goal deviation is zero it implies all the available resources have been utilized and if negative, then the absolute value of that quantity still remains after the schedule. It is important to note that no positive deviation is allowed in the system constraints as that has no physical meaning, for example, we cannot tram more than is the total broken material. The model is not feasible if a positive system constraint exists and is a result of a wrong formulation of the problem. A positive system deviation indicates those constraints that are violated under the current formulation.

Row 16 is a goal constraint that represents the requirement that all waste be mined during the shift. By explicitly imposing this condition in the model all waste is mucked. Rows 17 and 18 represent the allowed equipment allocation per level. The results show that the scheduled quantities can be achieved by a fleet of 4.28 and 1.85 LHDs on L100 and L200 respectively. We round this value up to a total of seven machines which is exactly the mine's fleet size. The fractional equipment number implies that the seventh machine will have to do some work on both levels. If there is no access between the levels, this solution would be impossible in which case the machine is expected to work on that level with the higher fractional requirement. This would be a deviation from the goal schedule. Note that integer programming would not provide a better solution since the scheduler has the capacity to use fractions of units on each level or in each stope.

The output entitled "ANALYSIS OF THE OBJECTIVE FUNCTION" indicates the satisfaction of the goals with the aim being to get either a zero or an over-achievement. The priority P1 is the summation of the remainders of resource absolute values of the schedule system constraints. The priorities P2, P3, P4 and P5 represent the absolute value of the summation of the goals' under-achievement at that particular ranking. In this schedule run, the priorities P2, P3, P4 and P5 represent the copper, nickel, production cost and the tonnage respectively. As discussed above, the copper and nickel under-achievement values have to be divided by the total scheduled ore tonnage, i.e. 2160 t to give the actual deviations of 0.62 and 0.15% for copper and nickel respectively.

The schedule solution, i.e. the decision variables, has to be considered in the light of the objective function values for the respective goals as a way of measuring the closeness to the schedule target. Full achievement of the solution occurs when the objective function deviations are all zero. The zero objective function deviation may be achieved due to over-allocation of resources. In this regard the solution does not reflect the most efficient usage of resources. On the other hand, the objective function deviation value may not be zero since some deviations may be actually preferred compared to simply zero deviation values as such values indicate an over achievement of the initial goal(s).

5.11 Sensitivity Analysis

5.11.1 Priority and Differential Weight Changes

The model was tested for the effects of changes of the goal priorities on the scheduled quantities and their sources. In this problem example the production target is always met and occasionally over-achieved by 10 t to give 2260 t irrespective of the priority and differential weight mix used. However, the difference lies in the constituent sources of scheduled material which changes significantly depending on the ranking of the goals. Table 5.5 shows some schedules obtained for the four goals of production, cost, copper and nickel grades where the priority rankings are ordered as 1 > 2 > 3 > 4 > 5. In these schedules the quality goals are met to different extents depending on the goal priority mix as illustrated in Table 5.6. The negative sign in the quality deviation values indicates that the target value was not achieved by the amount equal to this deviation. The cost goal of $6200 is achieved and in some instances savings are realized; for example in the first two rows in Table 5.6 a saving of $1244 which corresponds to the first two schedules in Table 5.5.

5.11.2 Resource Changes

- Budget Size Effect on GP Objective Function

The goal programming model was run for several different budget sizes ranging from $3800 to $7000, keeping all other goals and constraints constant. The objective was to determine the effect of a budget on the optimality of a goal programming solution as measured through the minimization of the objective function value. The results are illustrated in Figure 5.6. For budgets less than $3963, the problem is infeasible. Above a $3963 budget the total objective function deviation progressively decreases until the $4750 budget from which the deviation becomes constant until the budget rises in excess of $6000. Budgets of greater than $6000 are typified by an increase in the total objective function deviation reflecting poor utilization of resources.

Table 5.5 Typical schedules for different priorities

Priorities				Scheduled tonnes on each route, X_{ij}								
T	$	Cu	Ni	X_{11}	X_{12}	X_{21}	X_{22}	X_{31}	X_{32}	X_{43}	X_{53}	X_{64}
1	2	2	2	300	500	0	250	500	210	210	400	100
2	2	2	2	300	500	0	250	500	0	210	400	100
3	3	2	2	367	433	0	250	183	317	210	400	100
2	4	3	5	367	433	0	250	183	317	210	400	100
5	2	3	4	367	433	0	250	183	210	210	400	100
5	4	3	2	367	433	0	250	183	318	210	400	100
5	4	2	3	367	432	0	250	183	316	210	400	100
3	2	1	2	500	300	250	500	0	0	508	102	100
3	2	2	1	104	500	309	137	137	363	210	400	100
1	1	1	1	500	164	500	258	0	129	600	10	100
3	2	1	1	250	416	300	500	0	84	592	18.2	100

where: T = production, $ = cost, Cu = copper, Ni = nickel priority levels,

Priority ranking = 1 > 2 > 3 > 4 > 5

X_{11}, .. , X_{64} are scheduled tonnes per given route, ij.

Table 5.6 Effect of priority changes on multiple goal targets

Priorities				Deviations from targets			
T	$	Cu	Ni	Tons	Cost $	% Cu	% Ni
1	2	2	2	10	-1244	-0.623	-0.145
2	2	2	2	10	-1244	-0.623	-0.145
3	3	2	2	10	0	-0.623	-0.145
2	4	3	5	10	0	-0.623	-0.145
4	2	3	4	10	0	-0.623	-0.145
5	2	3	4	10	0	-0.623	-0.145
5	4	3	2	10	0	-0.623	-0.145
5	4	2	3	10	0	-0.623	-0.145
3	2	1	1	10	-948.89	0	0
3	2	2	1	10	-690.21	0	-0.101
1	1	1	1	10	0	-0.024	0
3	2	2	1	10	0	-0.651	0

where the goal targets are:

Tonnage = 2250 t, Budgeted cost = $6200

Copper grade = 2.33 %, Nickel grade = 2.43 %

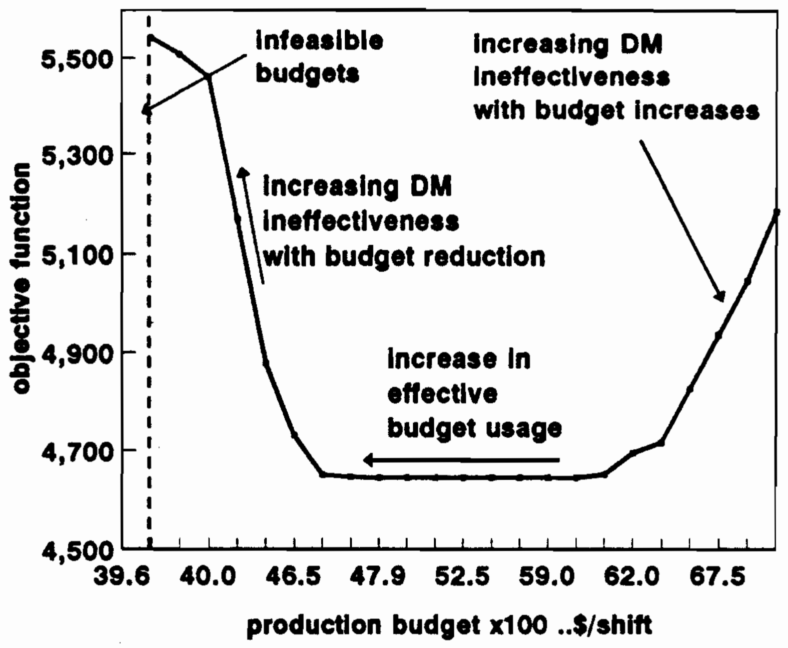


Figure 5.5 Budget size effect on GP objective function value

The model was re-run varying the production budget levels with the only changes between it and the initial model being only in the priority levels of the goals. In the later model, the goals were ranked in the order: copper grade first, nickel grade second, the meeting of the budget cost third and production tonnage last. The summation of the system constraints contribution to the model objective function were plotted for each budget size as indicated in Figure 5.6. The figure indicates that the system constraints' contribution to the objective function value drastically reduce from a maximum value just prior to the model being infeasible due to too small an operating budget. At budgets between $4000 and $4750, the system constraints rate of contribution to the objective function value is very small. At budgets greater than $4750, the contribution is a constant. This region coincides with the region of maximum flexibility and feasibility of the production scheduling. Since goals are achieved more tightly in this budget region, this implies that the production system is operating more efficiently by utilizing the system resources more effectively.

The variation of the objective function value for a model with goals set at different

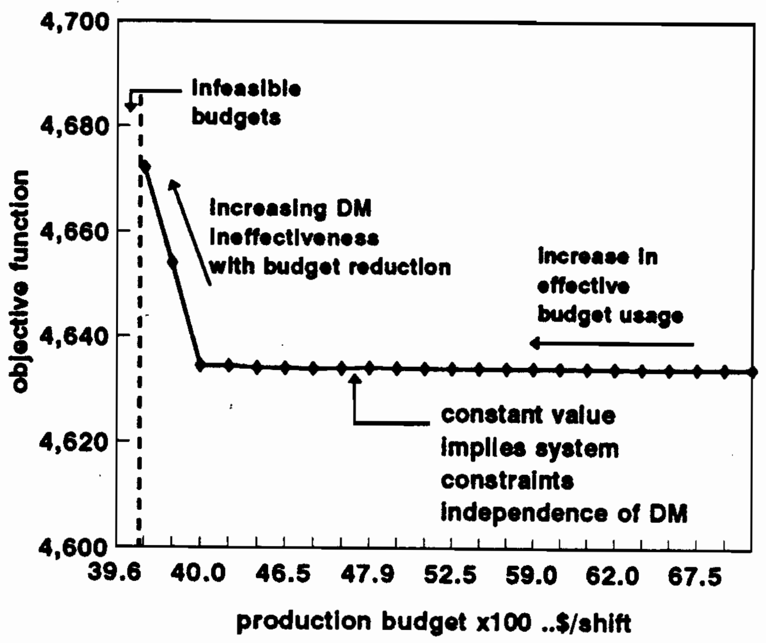


Figure 5.6 Effect of budget size on system constraints' utilization of allocated resources

priority levels is illustrated in Figure 5.7. The plot is a summation of the deviations of all constraints at the different priority levels against an operating budget size. The minimum feasible budget size is the same as that obtained when all the goals are set at the same level (Figure 5.5) which indicates that the feasibility of a model is independent of the goal priorities but rather is governed by the system restrictions. As the budget increases, the objective function value gradually increases, which is the opposite to what happens when the goals are at the same priority level. The objective function value increases until a budget size of $4750 is reached at which point the objective function level is relatively constant till a budget of $5900 is reached.

Higher budgets show some objective function values at higher levels, and in other instances values coinciding with the constant plateau. The higher valued objective function values all correspond to an over-achievement in the cost goal, that is, the schedule is achieved at a lower cost. The difference in cost between the target and the achieved value is what causes the peaks in the graph (Figure 5.7). When the graph is corrected for these values by subtracting them from the objective function value, a stable

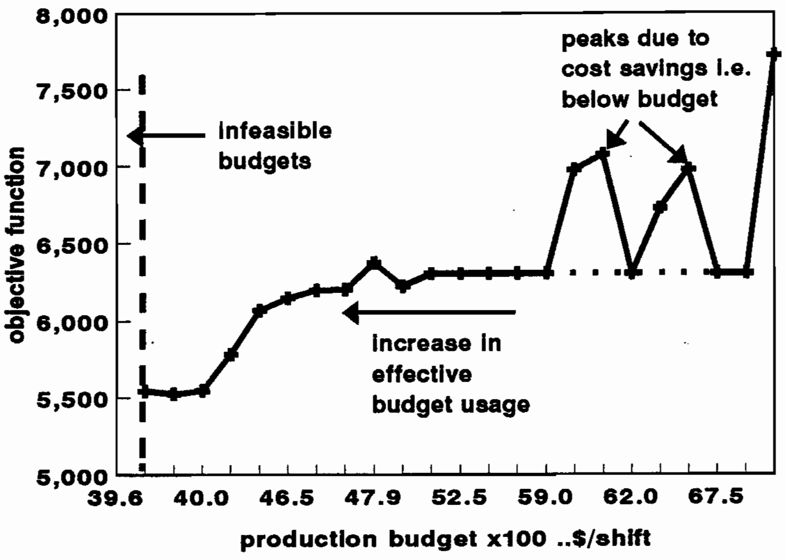


Figure 5.7 Effect of budget size on goal constraints objective function value with different priority goals

plateau is established (see dashed line in Figure 5.7). The plateau in Figure 5.7 coincides with the trough in Figure 5.5 and represents the region of scheduling stability and flexibility, i.e. budgets are not critical to meet the production and quality goals in the particular mine layout. The commencement of the budget over-achievement (causing peaks in the Figure 5.7 plot) also coincides with the incipient of a positive grade limb in Figure 5.5 which marks over-budgeting.

A conclusion is drawn that when goal programming is used with the goals set at the same priority, a 'trough' relationship between the budget size and the goal objective function value is obtained. This function can be used to identify the feasible as well as the optimal budget for a mine layout having specific material flow costs along different routes and specific quality requirements. If the goal programming model is used at different goal priorities, a different function exists where the plateau is at the maximum objective function value indicating that different priority ranking reduces the global objective of minimizing the total deviations. This assertion is in line with the fact that the goal programming solution for multiple priorities and weights is done in the order of the priorities appearance which tend to sacrifice the less important goals or objectives. The minimum objective function value for the given priority ranking exists at \$3995. This value may change with different priority rankings. This implies that, as many curves as there are different permutations of the production goals priorities have to be defined for use in production scheduling of a specific site.

This result is important in that it shows that a specific mine layout has a specific range in which a multiple objective problem can be scheduled with minimum deviations. No single budget value was obtained with a minimum deviation which implies that a reduction in the budget size from the current \$6200 to \$4750 has no effect on the total objective function value. The mine management therefore, can cut the budget without affecting their other goals of production and quality. The result also indicates that the implementation of a particular schedule within the trough of minimum objective function deviation is not critical which justifies the use of on-line dispatch algorithms for production implementation. The on-line dispatch policies are typified by variable operating cost per unit time due to uneven productivity during a production shift.

- Level of Shift Production

A similar approach was used to analyze the effect of changes in the production goal from the current medium term planning requirement of 2250 t per shift. The budget size was fixed at \$6200 and the quality and system constraints were maintained constant, while the production level was systematically varied between 500 and 3000 t per shift. The effects of the changes were measured in terms of the total objective function value and the results are indicated in Figure 5.8. For tonnage goals greater than 2260 t per shift, the problem is infeasible specifically because it violates the dump-space system constraints. As the production is progressively lowered below 2260 t per shift, the objective function value is defined by an inverse linear relationship, namely:

$$y = 7494.42 - x, \ \forall \ x \leq 2250 \qquad (5.55)$$

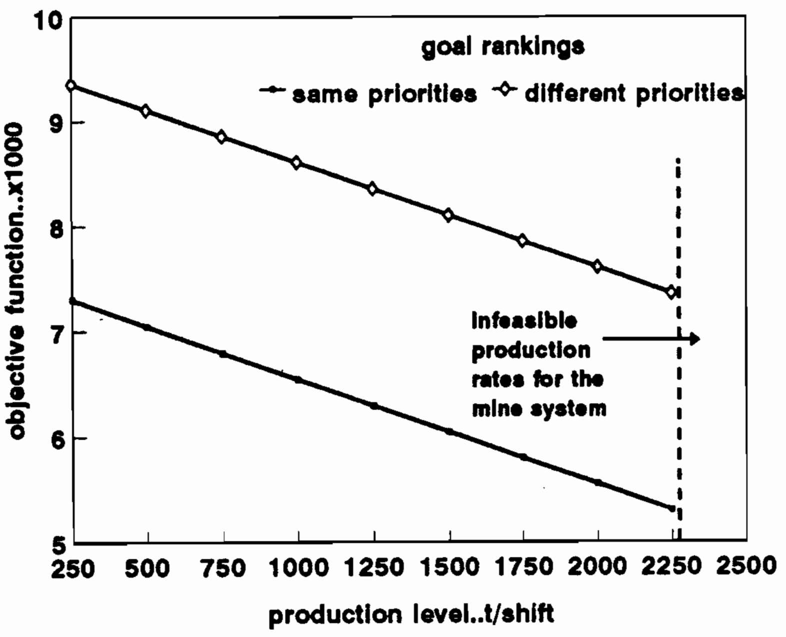


Figure 5.8 Effect of production level on the total absolute deviation of the schedule goals

where y = objective function value (total absolute value of deviation variables) and

x = target production tonnage.

The standard error of correlation for the equation (5.55) is insignificant at an order of magnitude of 10^{-10}.

Changes in the priority levels of the schedule goals result in a series of linear functions of the form y = A - bx; with the value of the intercept A, increasing with the priority level at which the goals are placed, as illustrated in Figure 5.8. The implication of this relationship is that by placing the goals at different levels, those at a higher level are compromised more than if they had been treated at the same ranking. The compromise can therefore be described as inefficiency in the decision making. This however depends on the context of the operations requirement, i.e. if the ranking of goals is to guarantee a specific objective satisfaction rather than simply to achieve a feasible solution then this does not imply inefficiency.

The relationship defined by equation (5.55) allows management to determine its deviation from the optimal use of the specific mine layout resources through settling for lower production targets. The existence of similar linear relations was tested for different mine layouts by changing the haulage distances (i.e. changes to the cost per ton if a certain route is used).

5.11.3 Effects of Budgets on Schedule

The size of the shift budget has an impact on the quantities scheduled from the different material sources. If the budget is large, the model is lax in that once it determines a feasible solution it stops the search for a better solution. However, if the budget is small, the model is more constrained and fewer feasible solutions exist. This leads to solutions very close to optimality or may indeed be optimal. Table 5.7 illustrates the effects of budget changes on the tonnages scheduled for each haulage route. The routes X_{11}, X_{21}, X_{32} and X_{53} are longer and costly. It is noted that the scheduled tonnages along these routes decrease with decrease in materials handling budget. At the same time, quantities along the less expensive routes increase in a trade-off compensation so as to achieve the desired shift production target. All the schedules in Table 5.7 satisfy the quality requirements to the same extent.

Table 5.7 Production schedules at different budget sizes

Budgets	Scheduled tonnes on each route, X_{ij}								
Cost $	X_{11}	X_{12}	X_{21}	X_{22}	X_{31}	X_{32}	X_{43}	X_{53}	X_{64}
3963	0	500	42.50	500	500	0	600	7.50	100
4000	0	500	50	500	500	0	589.2	2.81	100
4400	77.76	500	0	472.2	500	0	210	400	100
4500	103.9	500	0	446.1	490.1	9.92	210	400	100
4800	103.9	500	223.1	222.9	500	0	592	400	100
5200	300	500	0	250	500	0	210	400	100
5600	300	500	0	250	315.9	184.1	210	400	100
6000	300	500	250	250	308.7	191.3	210	400	100
6400	300	500	0	250	500	0	210	400	100
7000	300	500	233.9	16.1	16.1	483.9	210	400	100

5.12 Re-Scheduling

The goal programming re-scheduling of a shift production is necessary if one or more of the system constraints become violated making the attainment of the shift goals impossible. This could happen if the production equipment breaks down or some of the ore sources and/or dump-points become blocked.

At re-scheduling the modeller has to update all the system and goal resources. This is achieved by deducting the cumulative quantities from the original target value, for example in our example problem the target production from stope A is 800 t. If during one of the evaluation time windows, it is found that this stope has become blocked with 400 t still to be mucked, this value is initialized to zero since it becomes impossible to mine it until some future time when the blockage is removed. For the available stopes, the cumulative production is determined and subtracted from the shift target giving the quantities still to be mined within the shift. If no re-scheduling is done, the shift production would be below target by the value of 400 t blocked in stope A unless some arbitrary mining of this shortfall is done from the available draw-points to maximize the fleet utilization. This strategy hurts the quality goals as the extra mining does not consider the extraction as a blending problem for the shift.

In order to minimize both goal deviations, a re-schedule is done in which a re-assessment of the remaining available reserves are considered, not just what remains of the current schedule values. This allows consideration of reserves that had been unscheduled in the current schedule. For example, stope B had 800 t available reserves at the start of the shift and of this, only 250 t had been scheduled. We subtract the production to date of stope B of 150 t from the initial reserves to give 650 t of available reserves for a re-schedule. This procedure is repeated for all work sites and is indicated by the fifth column "Available to re-plan" in Table 5.8, where N/A implies the blockage in the stope A.

The constraints are re-formulated and as an example, the first system constraint in the original model (equation 5.29) representing maximum possible production on level L100 was:

$X_{11} + X_{12} + X_{21} + X_{22} + X_{31} + X_{32} \leq 1550$

becomes:

$$X_{21} + X_{22} + X_{31} + X_{32} \leq (1550 - T) \qquad (5.56)$$

where T is the summation of the productions at all sources on level L100 up to the point of a re-schedule. The values of X_{11} and X_{12} are zero and disappear in all the model constraints.

Table 5.8 Goal programming re-schedules. (Quantities in tonnes)

Stope	Original tonnage present, t	Old schedule tonnage, t	Total mined to date, t	Available to re-plan, t	New schedule, t
A	800	800	400	N/A	0
B	800	250	150	650	500
C	500	500	283	217	217
D	600	200	160	440	50
E	400	400	200	200	200
F	100	100	0	100	100
Total	3200	2260	1193	1607	1067

The computer program is then re-executed to generate a new schedule for the remainder of the shift or up to another 'stochastic' time when some major implementation problem arises. These results are then transferred to the tandem dispatch model for implementation.

The generation of a solution is very short. For example, the computer program finds a solution in less than one second for a model with nine decision variables and 26 constraints when run on an IBM DX50 MHz personal computer. Some significant time may be spent in the re-formulation of the model if one is inexperienced in the use of the developed computer code. The re-formulation involves the addition and/or deletions of some constraints in the model.

5.13 Model Limitations

The described model is a linear goal programming model. Such linear models are built on basic assumptions which must hold if the model is to be successfully utilized. The four basic assumptions are as follows:

5.13.1 Data Certainty

The variable or technological coefficients, the resource parameters, deviation variable weights, and priorities should be known with certainty. The model should therefore be designed to allow for this certainty. But such certainty rarely exists in a real world mining environment. Only estimates are used for ore grades, expected fleet productivity per unit time, and others, hence the model's output is only as good as the input data. This problem has been partly solved for ore grades through fuzzy modelling of stope grade outputs. Equipment, draw- and dump-point availabilities are based on statistical information.

5.13.2 Static Model

The coefficients, parameters, weights and priorities are considered constant over time. This is equally true in other methods such as linear and non-linear programming. In production processes, data tend to be dynamic. However, by narrowing the sampling period, pseudo-static conditions are achieved upon which the problem is resolved through the monitoring of the process deviations (e.g. breakdowns) to prompt a re-evaluation of the input data. This is the method used in this thesis as explained in Chapter 6.

A fully dynamic system in the solving of multi-objective decision making would likely provide impractical solutions with respect to their implementation as each solution set becomes transient. The implementation process would require full automation, removing the human input and judgement. Such a system was not developed here as the aim of this thesis is to provide decision support tools for production planning under a multiple objective scenario. Furthermore, it is doubtful that such an expert system would receive a good reception from the industry because of its 'black box' solutions. Also, like any knowledge-based system available, the system's logic is static and is a function of the model builder's own knowledge.

5.13.3 Additivity and Linearity

Additivity and linearity are required of the objective functions and the constraints. The left and right-hand side of the constraints should equate. But this condition is often violated in real world problems. For example, some variables when placed under competitive environment may be more productive than the same variables under no competition as shown where bonus incentives are involved. This effect can be reduced by applying stochastic and fuzzy numbers to both the coefficients and parameters of the constraints. That is, management expects the production to vary over a range, with the upper limit being reached generally under competitive conditions or certainty. The model can still deal with non-linear functions if they can be piece-wise approximated by linear sub-functions over different ranges.

5.13.4 Divisibility

The models allow divisibility of all variables such that fractional deviation and/or decision variables exists. This may be infeasible where only integral values are expected as in number of mine cars, loads and workers. Under such conditions, integer LGP is used. In the current work, divisibility is assumed because the implementation of the schedule is based on a dynamic system that will effectively limit the impacts of this constraint. Dispatching between work areas implies that haulage equipment works in fractional amounts of work on different work centres.

The solution from the formulated production problem is always divisible because the decision variables are tonnages (or volumes) which do not require integral values. In this respect, the problem of divisibility is not an issue in this model.

The variables and resources must be both finite and tangible. Depending on the time span considered, it can be shown that this is the case in real world problems. The variables are non-negative. Therefore, this restriction does not affect this model.

5.14 Summary

This chapter has detailed the formulation of a production scheduling problem in an underground mine based on a goal programming model. The general constraints and operational goals have been listed. Through a hypothetical case example of Fuzz mine, it has shown how these constraints can be formatted into a computer data file needed to run the goal programming model.

The integration of the results from Chapter 4 of fuzzy logic modelling of the ore sources material grades to the goal programming model has been shown. With this integrated system the mine management can easily obtain feasible solutions to multiple objective production problems within very short time spans. This ability to generate a solution quickly makes the approach appropriate for implementation in tandem with a real time dispatch model.

The use of priorities and weights has been shown to affect the mix of materials in the schedule. In the case example, the priorities had no effect on the tonnage goal which was met in all schedule runs. However, the sources varied with changes in priorities and weights. The minimum cost goal was met and in some instances equivalent schedules were produced at lower cost depending on the priorities mix used. The main point to remember is that the solutions of goal programming models are tailored towards minimizing the deviation from some fixed goals. If the goals are easily achievable, the tendency is that resources are not efficiently used. This apparent weakness of goal programming can be solved by use of single objective linear optimization to set the theoretical goals.

An important relation has been established between the shift budget and the absolute value of the goal objective function. The relationship is described by a 'trough' function that indicates that a given mine layout (fixed system constraints) has a budget range within which the goal programming schedules are met with minimum total deviation. It is possible to reduce the shift budgets to the minimum value that coincides with the minimum objective function deviation. This budget value is the 'hard' goal that is needed to ensure an efficient and superior goal programming solution.

An inverse linear relationship between the production tonnage goal and the absolute value of the objective function has been established. The relation is interpreted as a measure of efficiency in the decision making. A good decision would have a zero objective function deviation representing the full utilization of the available resources. By setting a lower production target for a shift compared to the mine resources and design clearly indicates an under-utilization of the system. The goal programming model also indicates the maximum production that is feasible for a given mine layout. This ability enables decisions to be made in a more precise framework.

The common limitations of linear models have been discussed. It is indicated that these limitations have little effect on daily production schedules due to the short time periods of their implementation. In addition, the proposed use of a goal programming schedule with an active dispatch model eliminates the requirement for integral machine allocation as the machines can be moved around to reflect their proportional work area demands. By appropriately selecting divisible variables (in our case tonnages or volumes of material) the model is and has been able to generate valid schedules that are free of the general limitations. These schedules become the input data to the implementing dispatch and control models described in Chapter 6.

Chapter 6
Underground Dispatch and Control Model

6.1 Purpose

The objective of this chapter is to describe the new dispatch policies and control methods developed for underground metal mines. This work is required for the real time implementation of the schedules generated by goal programming that were discussed in Chapter 5.

6.2 Scope and Limitations

Six dispatch policies are developed. These policies are used in tandem with an admission control algorithm which tests the feasibility of dispatching equipment to the various servers (draw- and dump-points), prior to the machine travel. Work area priorities and multi-level production can be implemented using different types of equipment.

The dispatch model does not model human behaviour such as slow responses to instructions, intentional abuse of equipment, etc. Stochastic variables are sampled from distributions of variables from historical time or work studies. These distributions may be inaccurate in describing the evolving underground mine system.

6.3 Introduction

A goal programming model is implemented to optimally allocate the machines to each work area based on deterministic static operating conditions. However, the goal programming solution is only a guide to the shift supervisor of what the system performance should be. In practice, certain unpredictable events occur such as the breakdown of a machine, blockage in the ore-passes and/or lower than average loading rates. Any one of these phenomena cause a deviation from the goal programming schedule. One logical step is to re-schedule the remainder of the shift production each time a deviation occurs.

Evidence from operating mining environments indicate that the shift schedule of a batch system cannot be implemented entirely on simple linear regression. Bunching problems occur sporadically, especially at the start of a shift. Sufficient time has to be allowed for the materials handling system to get into equilibrium on its own before an external feedback control is introduced. Besides, a continuously re-scheduled process is both impractical and expensive to consider as an option in mine production control.

Therefore, the goal programming model is not executed each time a delay or breakdown occurs. Instead, the study proposes the use of a traffic dispatching model to control and guide the actual production towards the set goals. By routing resources where they are most needed, the strategy minimizes the deviations from plan. In addition, necessary conditional constraints are imposed to prevent excessive machine travel between different work sites and crowding which causes machine interference.

The literature survey of the manufacturing and other industries has revealed a multitude of different dispatch rules ranging from simple to complicated ones. These rules have one thing in common, namely, each rule is based on a single objective performance criterion on the current system situation without consideration of the future. Different rules have different performances under identical layout configurations. Therefore, it is important to identify the proper rule to apply in a given situation to maximize the objective performance. In this chapter, six rules are investigated: three rules are adapted and modified to apply to an underground mining system and the other three are new rules developed to effect a product quality control.

6.4 Principles of Underground Dispatch Model

6.4.1 Comparison of Flow Shop and Underground Mine

A mine production environment is similar to a flow shop in manufacturing systems in the following:

1. A flow shop's production schedule is based on the inventory levels of the various products manufactured, and rarely on external market demands. In mining the production schedule is influenced mainly by the internal mine resources available and material inventories in the system. Development waste production can be prioritized if this is necessary. Occasionally, the process plant could influence the schedule by demanding a certain quality.
2. Parallel jobs are generally performed on the factory floor. In mining this is comparable to ore and waste handling activities as well as other auxiliary activities such as backfilling.
3. Flow shop jobs pass through a series of fixed processors. The underground production operations consist of materials handling of scheduled quantities from pre-defined work areas to some destination. The destinations may be interim or final as in temporary ore-passes and the crusher bins, respectively. The mine jobs are fixed and it is the materials handling equipment (processors) that is mobile.
4. The flow shop jobs and mine handled materials flow in one direction only and do not necessarily have to pass through all the holding points. Some jobs take shorter process times depending on the number of process stations requirement or mine layout.

These similarities strongly suggest that the successful concept of a computerized flexible manufacturing system can be adapted to underground trackless materials handling systems to yield improved productivity and efficiency. It is on this basis that the Underground Active Dispatch System Model (UADM) was created. The basic structure of the UADM is illustrated in Figure 6.1.

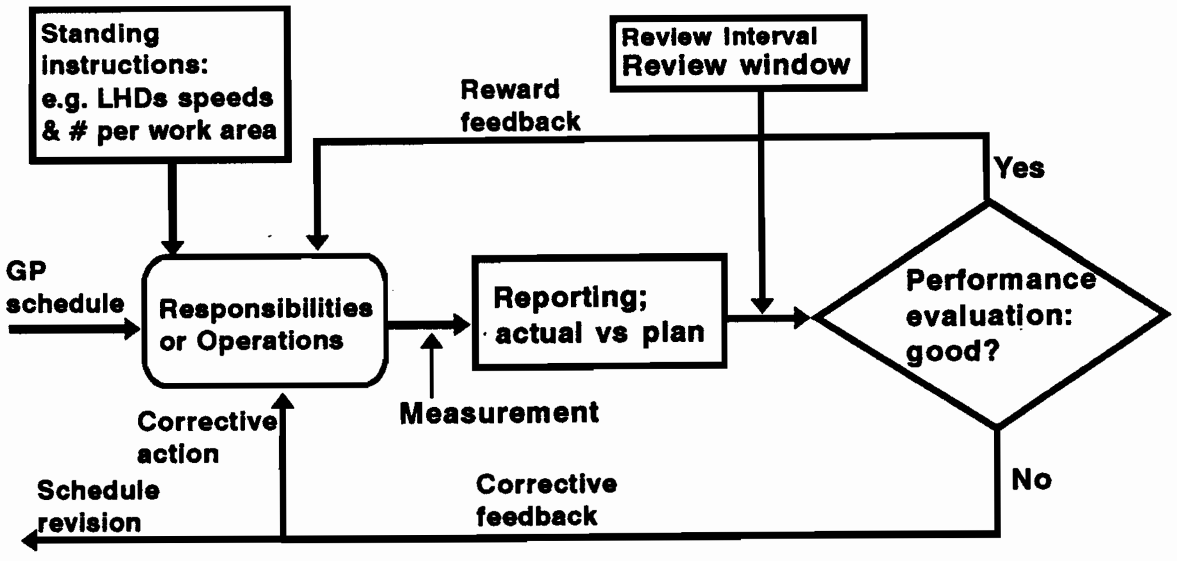

Standing instructions: e.g. LHDs speeds & # per work area
Review Interval
Review window
Reward feedback
Yes
GP schedule
Responsibilities or Operations
Reporting; actual vs plan
Performance evaluation: good?
Measurement
Corrective action
Schedule revision
Corrective feedback
No

Figure 6.1 UADM procedure

A meaningful schedule is one that considers only those operations that are ready for processing and allocates the necessary process time to each work area. Therefore, in mining, the production plan identifies the accessible work areas and assigns the necessary equipment. However, due to the uncertainty in the process, sometimes pre-emptive decisions are needed to drastically re-dress a negative situation. This may be in the form of equipment breakdown or blockage at the draw- and/or dump-points. Such criteria as precedence rules are set to deal with these circumstances to minimize the performance deviations.

The operations consume resources, such as time to execute a task, or occupy a service point. The stochastic resource consumption causes deviations from the initial schedules based on deterministic variables. The real time implementation of a schedule has an associated risk because by dealing with future events there is always uncertainty of the outcome. The risk results in an increase in operating costs, loss of quality control and/or failure to meet targets.

Each scheduled amount of work has some form of 'weight' associated with it because it has to be completed somehow within the schedule duration. As the work shift progresses and the scheduled tonnage is not mucked, the greater the 'weight' that becomes assigned to that work area. In other words, doing nothing leads to higher 'weights' to work areas and usually to higher costs. Costs increase due to lost production and over-time requirements. Inversely, a depletion of a work area leads to a decline in its 'weight'. The value of the 'weight' of a work area is a pointer of how critical that job is in the work shift. This concept is applied in all the dispatch policies developed in this project.

The end of a shift is a form of due date when actual and scheduled production of the last shift are compared. The reconciliation highlights the benefits and/or penalties of the achieved production levels against the plan. Systematic sampling of the shift production at key times indicates the evolution of the process and facilitates a feedback control when major deviations from plan occur.

6.5 Control of Mining Queuing System

There are two types of queuing systems, descriptive and prescriptive. Descriptive evaluates a fixed system and a prescriptive evaluates dynamic systems. In this model the prescriptive system is used because it allows for the specification of the best possible type of queuing system to use for the given operating conditions, i.e. through policy changes.

A queuing system performance is influenced by three factors:
1. the system configuration,
2. the system control parameters or performance measures and
3. the operating policies employed.

The system configuration is defined by the mine layout and consists of the haulage ways, maximum allowed speeds, and the number and size of the facilities. The configuration can be viewed as three sub-systems, namely the face where the material is mucked, the haulage and the dump-points. At the face, loaders are used with truck-loader operation, self-loading LHDs and, less common in mechanized mines, gravity chute systems are implemented. The loading times depend on the fragmentation of the material, material flow and the loader size; they can be short or long depending on conditions.

The material is trammed by trucks or LHDs to the respective dump points. The interaction of this equipment is very important, for unlike surface mining where there is unlimited queuing space and two-way haul roads exist, this is not the case with underground mines. The haulage systems can be mono- or bi-directional, or a third system option under which machines can use the same route but only if none is travelling in the opposite direction at that time. The later case causes substantial production losses as equipment ready to travel in an opposite direction wait until the road section is clear.

As the production progresses, some scheduled material quantities are met. If the equipment in the depleted areas is captive, that equipment becomes idle, thus reducing the fleet's utilization. In practice, the management decision is production based rather than based on overall product blend. Therefore, the captive equipment continue mining thereby exceeding the scheduled work shift tonnage in a particular mine section. A non-captive equipment mode has the flexibility of transferring the mobile resources to areas they are most needed and improve the product quality.

Dump-points are material sinks. Their capacity may be affected by hang-ups or draw policy. The dump-point availability could be affected by blockages at the grizzlies. The space available for machines to queue before dumping is in practice very limited.

The system configuration represents a bipartite graph whose nodes are the material sources and dumping areas. The connecting arcs are the permissible routes and each route is weighted by its distance and maximum allowed speed as shown in Figure 6.2. The configuration is site specific and is static within a shift production period. The system configuration is input to the model through a data file.

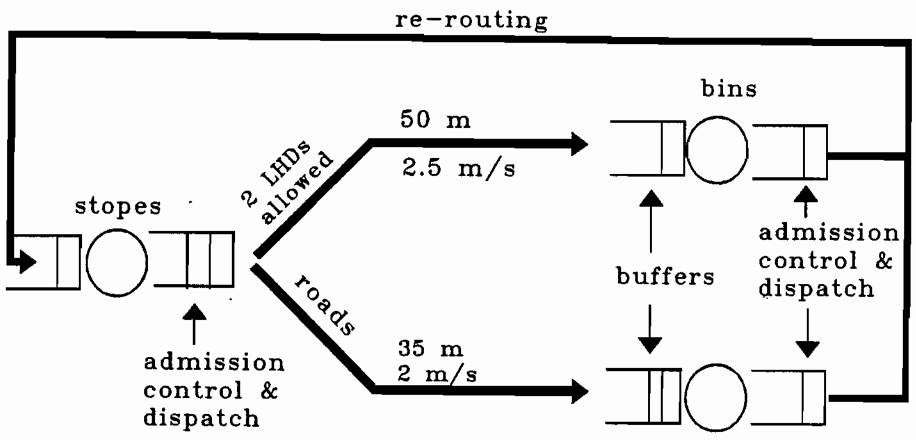


Figure 6.2 Underground mining queuing system graph

The service rates, i.e. loading/dumping times, vehicle speeds, machine availabilities and frequencies of blockages, represent the evaluative parameters of the queue system. These parameters are historical data from the current or some similar mining environment. The implication of this is that the model is only as good as how well this data accurately represent the current and evolving situations. Therefore, the model has uncertainties related to the parameter information base. This problem always exists though it may be resolved through a system sensitivity analysis of these input parameters.

The operating policies are the queuing disciplines of the materials handling equipment. The queue control is related to the issue of customer resource contention such that:

1. all customers receive satisfactory (possibly prioritized) service,
2. all resources are utilized to acceptable levels, and
3. the allocation of service among the clients is based on the management requirements and system constraints.

The queuing control is resolved in this work by one of two dynamic procedures, namely, admission and routing disciplines. The admission control is applied when deciding whether a new client can be allowed or rejected to enter a queue. The reasons for applying this control are the need to ensure that the arrival rate to a server does not exceed the server's service rate. If this condition is violated, then there is a possibility of a theoretical infinite queue build-up at the server. Certain levels of system performance are required and these are compromised through congestion and long queue waiting times resulting in decreased productivity.

The underground system consists of more than one queue, hence a routing control is applied. The criterion of routing is to balance the use of resources shared among customers. To effect routing, information is kept on each queue length and actual sequence of the members in the queue. The required service time at each queue is partly a function of the queue length and partly the type of client as different client types have different service rates. In the developed model, the routing discipline is applied in tandem to the admission control.

6.6 Model Assumptions

The model UADM is based on the following assumptions:

1. Decisions on machine control are made based on the current system status only. The current status implies the queue lengths and service rates at either the draw-points or the dump-points and the current location of the machine requesting a destination. A look-ahead feature that evaluates the next few customers yet to be available, at the current decision making is not applied because it is impractical in the space constrained underground network.
2. Two way travel is allowed in all the mine haulage ways. The provision of fully two-way accesses is expensive, such that in practice, a mine is most likely to have transfer points typified by in-sets in the haulage ways. This modification requires a traffic control system to regulate the flow of equipment between the inter-in-set distances to avoid collisions. UADM models this modification through a time factor which is added to the travel times of all empty LHDs if they are running in a road section occupied by a loaded LHD. A multiple of the time factor is

added for each loaded machine in the common route. UADM therefore, gives precedence to loaded machines to travel without stopping in the in-sets. This is a shortcoming in some instances whereby an empty machine could cover the inter-inset distance faster, an aspect of traffic control that is not covered in this work.

3. Zero or a limited buffer space at both loading and dump points is allowed.
4. Over-taking is allowed.
5. Machines are allowed to travel at highest allowable speed on each route. However, this would be too optimistic given that interference of machines would exist in a two-way haulage system. Delays of random lengths occur and these are additional to the deterministic travel times along each road section.
6. The system is memoryless and greedy. By memoryless we imply that each destination allocation is made independent of past visitations. However, the productivity ratios at each scheduled stope or draw-point modifies this behaviour because the dispatch procedures select the work sections with the least completed job at the given work area priorities. Greedy implies that the policies select the best destination based only on the immediate conditions without assessment of future events that may provide an improved solution.
7. Reneging of an LHD from a queue is permissible if only the server becomes unavailable (i.e. blocked) and the expected time to become available is longer than the time to reach another destination and receive service.
8. Discouraged clients cannot join queues that exceed either the local buffers or the machines on the inter-linking route are at their maximum value. These clients accumulate waiting times whilst at their current locations.
9. Each machine commences work after some random start-up time equivalent to machine warm-up.
10. The initial equipment allocation is based on the goal program requirements. The equipment work in these respective areas until the work is either exhausted, or fixed facilities become unavailable, upon which the equipment is dispatched to

other work sections depending on the requirements in those areas, else the machines become idle until the end of the shift.

6.7 UADM Program Structure

The underground active dispatch model (UADM) has the following fundamental characteristics:

1. It is an active dispatch system rather than fixed which means the entire materials handling equipment is free to move between different work centres. A machine is assigned only a one trip job. This differs from the models described by Gignac [1979] and Hill [1987]. As a result of the active nature, the model is more flexible in dealing with a dynamic production system characterized by random activities.

2. The model is a parallel multi-server system with service rates at each server defined by a three parameter Weibull distribution. Each server has an individual queue built on the basis of its effective buffer space, service rate, priority, haulage constraints and nature of material being handled. The local queues are assigned clients by the admission and routing policies.

3. UADM consists of eight equipment status during production. They are travelling full or empty, receiving service (loading or dumping), waiting for service at the loader or dump-point, breakdown and idle. After receiving service, each machine requests for a destination on a first-to-request-first-served rule.

4. An admission control is then applied in the selection of the destination for the machine that first requested for a server. The admission control used is the "before travel blockage test" which evaluates all servers for consistency in both travel and destination constraints. If the route to a server is blocked or infeasible then that server is discarded as a server candidate for the current client. Equally,

if the client is carrying ore and a particular server is a waste pass, that waste pass is an invalid server. If the destination is blocked or depleted, again a client is said to be blocked. The "before travel blockage test" advantage is that equipment is not routed to infeasible destinations, an important cost servicing routine in real time operations.

5. If admission is acceptable, the machine is allocated a feasible destination based on the operating routing (dispatch) rule. UADM has six dispatch rules which can be used inter-changeably in the simulation process.

6. Historical time study data is used for the parameter selection. The travel time distributions for each machine type are defined by a Weibull distribution.

7. A specified time-unit-advance simulator is used in this model and is characterized by a time interval of y/t for t simulated time periods and time increment of y units. In each interval the simulator checks the current status of each resource variable (equipment and servers). This implies a simulation of length t will involve {kt/y} comparisons, where k is the number of resources. This method of time advance is computationally advantageous compared to an event-to-event advance system if the number of resources in the system is large [Emshoff and Sisson, 1971]. This is due to the high probability of some event occurring in the time increment, y.

One important limitation of time slicing simulators is the loss of accuracy on the exact timing of event occurrences which is a function of the time increment size. A comparison of event and unit increment time advance methods is indicated in Figure 6.3. The use of the UADM therefore requires a time slice to be small to minimize the inaccuracies. Emshoff and Sisson [1971] suggest a choice of a time increment unit to be the largest unit such that the probability that more than one decision will have to be made during that unit time is negligible. Other simulation studies indicate that an interval less than half

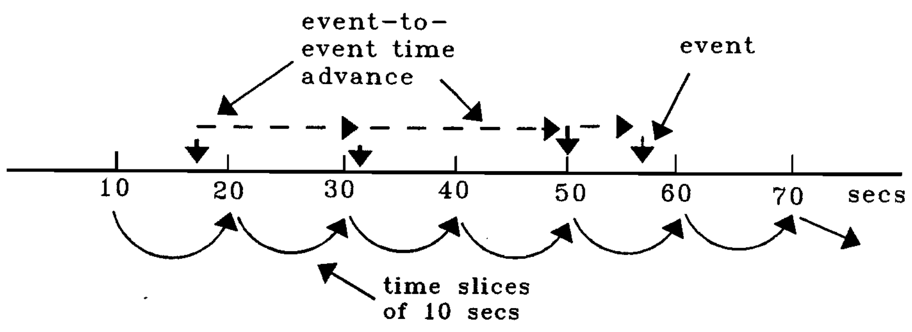


Figure 6.3 Comparison of event and unit increment time advance methods

the smallest activity time is an acceptable rule of thumb. Using this approach for underground mining, the dump activity has the most likely shortest event time therefore a user of UADM should use this value.

8. All events occurring during the simulation are recorded as to when they started and when they ended. Such data include the start and end of a service of a blockage or breakdown, assigned machine destination and sources and quality of each load dumped at each dump-point.

9. Machine breakdowns are random. In UADM the model tests for breakdowns using a random number generator on each machine during a process review interval discussed below. If a random number less than or equal to the statistical probability of breakdown occurrence is generated, then we assume the machine has broken down. The down-time is then modelled using a triangular distribution. The model assumes there are always enough mechanics to immediately attend to a broken down machine and the repair work continues through lunch break. The later condition is reasonable given that the mechanics may be simply on standby when all machines are running. Similarly, draw-point hang-ups that need blasting down are done during any time of the shift including the break periods.

10. Process review and control intervals are available based on the desired times to sample the production process. These could be every half hour, hourly, etc. At

a review time interval (or window), the program evaluates the equipment utilization, production to date and quality of product and compares these values to the expected quantities. Any deviations beyond management set limits are indicated, which identifies the goals that are not satisfied. The model prompts the user either to ignore the out-of-control message, change the dispatch rule or terminate the simulation for a goal programming re-schedule. Graphical outputs of cumulative production and quality deviation from targets are plotted on the screen for the user to visualize the evolving process for possible trend(s) identification. Such trend information is invaluable in a re-schedule process for setting of the priorities of material sources.

6.8 UADM Admission Control

Whenever a mobile machine requests a destination, the UADM admission control polls all the applicable servers on the same work section as the machine for queuing space, job capacity, material compatibility and break-downs. A server is not entirely picked at random as it has to satisfy the structural constraints in the system. First, it is assessed if the destination server is in service, i.e. in a loader-truck operation, the loader may go on breakdown. For LHDs, the draw-points may be blocked. In both instances, no machine would be dispatched to such a server. Also, a machine is dispatched only to those servers that still have enough material to fill the machine, or if it is a dump point, if the dump point has capacity for a load equivalent to the machine's capacity. The servers have limited buffer capacities. Therefore, before assignment it has to assesse *if* by the time the machine arrives at a particular destination there will be space to queue. This feature is a look-ahead strategy where the expected service times for members already queued at each server are estimated. The time necessary for at least one queue member to be serviced so as to create space for a new one by the time it arrives is determined.

Loaded machines have an extra conditionality in their dispatch, of material compatibility between the load material type and the dump point. Ore and waste loads are trammed to ore and waste dump points respectively.

If these constraints are not met, then a number of possibilities exist. If all loading areas are either depleted or blocked, then the current machine is dispatched to other areas depending on the time frame to end of shift and work conditions in those other areas. Similarly, if all dump-points are blocked or full, the machine is dispatched to a new work site. However, a situation may arise that the buffer capacities at each server are full on a level but there is still work to be done. In this case, the system is temporarily blocked and the requesting machine accumulates a waiting time due to a lack of destination. This situation exists if the number of machines per work area is given by:

$$N_{\max_i} \geq S_i + B_i - 1 \tag{6.1}$$

where $N_{\max i}$ = maximum number of machines on level i

S_i = number of servers on level i, and

B_i = number of local buffer capacities on level i.

The appropriate dispatch rule for the work area is applied only within the system constraints as defined by the admission control. Such rules are enumerated below.

6.9 Dispatch Policies

Six dispatching policies are studied in this model. These policies or strategies were selected on the basis of the ones used in surface dispatching models or in an attempt to conciliate a particular strategy with one or several of the goal programming objectives.

6.9.1 Shortest Travel Time (STT)

A 'shortest process time operation next' policy has found intensive use in the FMS to select those jobs in the queue that take least time to complete. It results in short queues in the system. However, jobs with long process times are continually put back and this would result in excessive delays for such jobs and may indeed affect the whole schedule. We have modified this policy to allow the determination of the nearest server for the equipment dispatch. The use of the policy is justifiable in the underground mining environment under these conditions:

- the planning objective is based on quantity rather than quality.
- the quality distribution within the mine is fairly homogeneous, i.e. does not have blending requirements, and
- the mine production (tramming) fleet is below capacity.

Any one of the three conditions is amenable to STT and a review of operating mines confirms the existence of these conditions.

The travel time, t is the expected travel time of a particular client from its current location, i to a certain destination, j. The time t is given by:

$$t = t_d + t_{rand} + N*t_{con} \tag{6.2}$$

where t_d = deterministic time based on the maximum allowed speed on the route ij

t_{rand} = random time delay experienced on route ij, sampled from a Weibull distribution

t_{con} = machine mean interference time when sharing the same road section, i.e. by-passing and over-taking

N = number of machines on the route ij travelling in opposite direction to the current client seeking a destination.

The policy selects the minimum ratio of the travel time to (1) a goal programming ranked stope and (2) the amount of scheduled work still remaining at that stope, j. The policy is expressed algebraically as:

$$\min_j \left(\frac{tT_t}{RT_a}\right)_j, \quad \forall j = 1 - m \qquad (6.3)$$

where T_{tj} = target (scheduled) tonnage for material source j

T_{aj} = actual tonnage still remaining at the source j

T_{tj}/T_{aj} = 'weight' of the source j at that point in the shift production

R_j = rank of the source as defined by the goal programming model

m = number of feasible destinations on a work section.

The priority, R_j is applied such that the dispatch objective respects the shift requirements set in the goal programming schedule.

6.9.2 Earliest Expected Service Time (EEST)

If equipment utilization has highest priority, an appropriate rule for dispatch is the earliest expected service time. This policy attempts to maximize utilization by picking those servers that are likely to go idle first through computation of the total expected service times of the equipment in a current queue. The result is minimum waiting times of the tramming equipment. The policy is both neutral to product quality and scheduled quantities of each work area. It is influenced by the service rate at each destination. The service rate depends on the draw-point ground conditions, fragmentation and material flow. Due to these factors, the policy can effectively lead to poor product quality as the higher service rate stopes are visited more at the expense of low ones. EEST is a suitable policy in homogeneous deposits and when the scheduled tonnages from all stopes have the same priority. Such conditions can be found in some mines, making the policy a valid one to study.

To discriminate between two or more servers that may have the same EEST, the policy is modified to reflect the relative importance of the servers. The selected destination is one that satisfies the algebraic expression:

$$\min_j \left(\frac{st\,T_t}{RT_a}\right)_j , \ \forall\, j = 1 - m \tag{6.4}$$

where st_j = earliest expected service time at server j and

R_j = the priority of the server j among m servers and

T_{tj}, T_{aj} and m have same meaning as described for STT policy in equation 6.3.

The policy is myopic to travel costs as it would simply dispatch a machine to the first idle server irrespective of the distance between the polling machine and that server. Solution of this problem would entail a new algorithm that determines the optimality of each assignment with respect to some near future assignments. The possibility of this look-ahead option is nullified by the highly constrained nature of the underground environment. The lack of buffers at some destinations would lead to a blocked system resulting in potentially more costs than those involved in the longer trip.

6.9.3 Minimum Slack Time (MinS)

In truck and loader operations, each item has an associated operating cost and depending on the mine fleet mix, either could be the bottle-neck to production capacity. Greater emphasis is then placed on the bottle-neck resource. In a well matched operation, the optimal situation involves minimum idle times by both trucks and loaders. Under these conditions, this study proposes a minimum slack time policy defined as the minimum absolute difference between a truck arrival and the time when its service begins. A priority, R_j and the ratio of the remaining work at the destination are applied to direct the dispatch process according to management requirements of each work area. The policy is formulated as:

$$\min_j \left(\frac{|st - t|T_t}{RT_a}\right)_j, \quad \forall\, j = 1 - m \tag{6.5}$$

where st_j = the expected start of service at server, j and

t_j = the expected travel time to server, j.

T_{tj}, T_{aj} and m have same meaning as in equation 6.3.

This means a truck would be dispatched preferably to servers with a slack time of zero which corresponds to a vehicle arriving and receiving service immediately.

The policy, unlike STT does not segregate against the long trips, and is both product quality and production neutral. Unlike EEST, it assesses the travel distances to each server and by aiming at zero slack time eliminates both client and server idleness. The concept of the minimum slack policy is similar to those used in critical path network analysis which lead to an optimal solution. Thus, though myopic due to the type of environment it is being applied to, minimum slack gives a mathematical optimal solution to equipment dispatch.

Travel distances are not optimized. The operating conditions of underground mines tend to differ significantly, for example, poor haul-road maintenance lead to relatively high tire costs. This weakness of the minimum slack rule, tends to be applicable to all other single objective mathematical optimization procedures.

6.9.4 Critical Ratio Policy (CRatio)

The cumulative production in each work section tends to randomly deviate from the planned targets throughout the shift period. This is inevitably caused by unpredictable events such as breakdowns and blockages in the system. The policies described above are based on production time, i.e. earliest service time, shortest travel time, etc. The tonnage and quality outputs are only derivatives of following a certain time based policy.

To appropriately reconcile the actual against the planned production situation, we propose a dispatch rule based on a dynamic critical ratio. The critical ratio technique continually compares the difference between the actual and scheduled stope production to the total scheduled quantity for the respective stope. Ratios of zero and one indicate that scheduled work is finished and that no work has been done, respectively. Therefore, a machine requesting for a server is dispatched to the server with the maximum critical ratio that satisfies the admission control constraints. Algebraically, the critical ratio policy is:

$$\max_j \frac{(T_{tj} - T_{aj})R_j}{T_{tj}}, \quad \forall j = 1 - m \tag{6.6}$$

where T_{tj} = total scheduled tonnage for stope j,

T_{aj} = accumulated production at current time at stope j,

R_j = management priority on the stope j.

m = number of scheduled stopes or draw-points on work area.

The critical ratio policy allows the up-to-date information use on the system. The policy is an expediting procedure since the dispatching is always to execute the job with the highest critical ratio next. If the stope priorities are all equal, the policy is a quantity based technique which closely mimics the goal programming production flows between nodes. In the event that quality is the main shift objective, the policy at equal stope priorities will ensure a homogeneously blended product if the production goes according to plan. Applying different priorities to the stopes results in a priority-production utility measure of the different work areas that reflects the sequence in which management wants the tasks done on a shift schedule. The approach implies that equipment is utilized on the critical scheduled areas first and if breakdowns occur then machinery is moved to these areas. This concept is fundamentally similar to that in the goal programming model where the objective is to minimize the absolute deviations of goals in the order they are ranked.

The critical ratio policy dispatches equipment without consideration of the near-future dispatch events nor the travel distances involved. Therefore, equipment may be expected to have large cumulative travel distances during a shift which would probably increase the operating cost, e.g. tire cost.

6.9.5 Maximum Contained Metal Policy (MaxQ)

Mining is driven by the value and concentration of the product. We therefore propose a dispatch policy based on the contained metal of each scheduled stope. As in the other policies, a priority value is assigned to each work area to reflect aspects not resolved by the contained mineral value. The policy dispatches a customer to a server with the maximum contained metal value defined as:

$$\max_j \left(\frac{gRT_a}{T_t}\right)_j, \ \forall\, j = 1 - m, \quad g_j \neq 0 \tag{6.7}$$

where T_{aj} = the remaining scheduled tonnage in stope j at the present time

T_{tj} = scheduled tonnage in stope j

g_j = the mean grade of the scheduled stope material.

m = number of feasible destinations on the work section.

This policy leads to the preferential mining of scheduled high grade and/or large tonnage stopes first. The tonnage influence gradually decreases through depletion to a stage that another stope becomes the higher utility. Equipment is dispatched to waste stopes depending on the product of the scheduled tonnage and priority of the stope. If a stope grade is less than unity, this results in the waste stope of corresponding rank and tonnage being selected contrary to the policy objective. Therefore, in the UADM this problem is resolved by adding a constant (i.e. 1.0) to all stope grades in the program function that computes the destinations.

The shortcoming of the contained metal policy is that it may lead to excessive traffic jams as all machines are dispatched to the same area. However, this problem is mitigated by the use of the admission controller which initially identifies those stopes that are feasible for an equipment dispatch.

6.9.6 Modified Maximum Contained Metal Policy (MaxQ/MinS)

The effect of increasing haul distances on productivity of load-haul-dump equipment is well known to be negative. Work studies at Creighton Mine, Sudbury on a fixed LHD fleet confirm this impact as shown in Figure 6.4. Besides the low productivity, the haulage cost per ton is expected to increase with distance. If the haul roads are poorly maintained, then tire costs escalate. Table 6.1 illustrates the tire performance for two consecutive years at a number of the Inco Sudbury mines. The table indicates that tire cost is a significant cost item that can be used to infer the different operating conditions within these mines. The incremental cost changes at each mine over the two years points at how a mine can move from being profitable to marginal within a short period depending on changes in operating conditions and policies.

Table 6.1 LHD tire performance cost in Inco Sudbury Mines [Internal Report, May 2 1994]

Mine	$/ton/tire in 1993	$/ton/tire in 1992	Δ$/ton/tire	% Δ in cost
Frood	0.035	0.028	0.007	25.0
Stobie	0.031	0.027	0.005	14.8
L. Stobie	0.036	0.024	0.012	50.0
Coleman	0.036	0.030	0.006	20.0
C.C. South	0.049	0.032	0.017	53.0
C.C. North	0.034	0.024	0.010	42.0
McCreedy W	0.022	0.028	-0.006	-21.0
Crean Hill	0.029	0.030	-0.001	- 3.0
Creighton	0.047	0.046	0.001	2.2

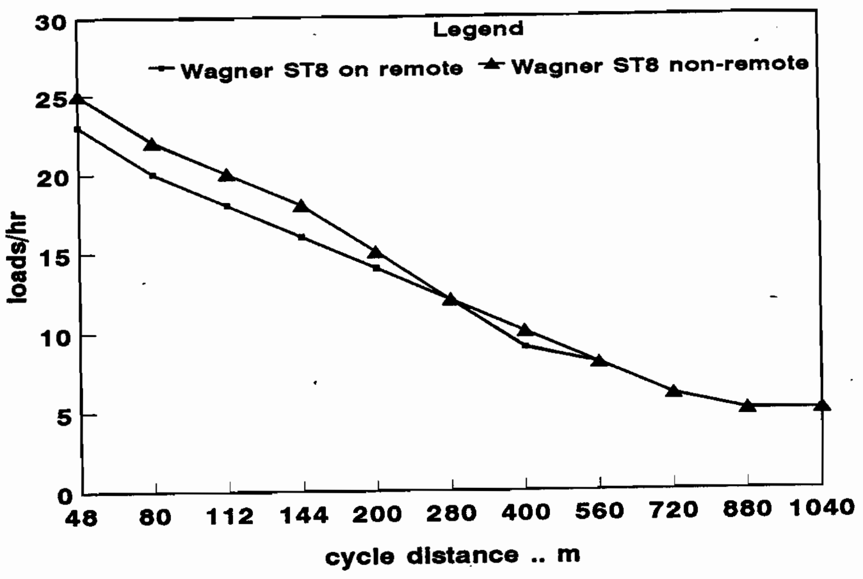


Figure 6.4 LHD productivity profile with haulage distance at the Creighton Mine [after Universal Scheduling Consulting, 1993]

The haulage distance is therefore an important factor that has to be considered in the dispatch rules if they are to be effective for operational use. Consequently, a new policy is proposed that takes into account the travel distance. SST and MinS rules consider the travel times between points based on a calculated linear distance-speed relationship. But the productivity-distance relationship is non-linear hence the introduction of the travel distances into dispatch rules impacts on the productivity differently as compared to the SST and MinS rules. The equipment is dispatched by the modified maximum contained metal policy to the server, j with the maximum product ratio defined as:

$$\max_j \left(\frac{gRT_a s}{|st - t| T_t}\right)_j, \quad \forall j = 1 - m \tag{6.8}$$

where g_j, T_{aj}, T_{tj}, R_j have the same meaning as for contained metal policy,

$|st - t|_j$ = absolute value of waiting time of equipment to serve or receive service

s_j is the reciprocal of the slope of a production cost versus distance function. The shorter the travel distance, the higher is the value of s_j.

The modified maximum contained metal policy has all the characteristics of the maximum contained metal policy with the additional advantage of minimizing travel distances. The policy is ideal for operations with:

1. small fleets and management wants to maximize production,
2. wide variation of travel distances between stopes and dump-points and
3. fleet cost minimization is required.

6.10 Main Program Functions

The program UADM is coded in C language. It consists of a main function that initializes the various variables, opens the input and output files. This function has the simulation clock and monitors and records all the resource events as they occur. The main function calls the other functions that perform specific tasks such as generation of loading time or breakdown of a machine. The main program also performs an evaluation of the actual production statistics against the planned quantities. After process evaluation, the main function indicates whether the process is in control or not and by what percent deviation. At this stage the program prompts the decision maker to either accept, make changes or terminate the simulation. Graphical output facilitates the decision maker in deciding on a feedback mechanism of the process.

The program consists of 40 functions but only the ones fundamental to the program logic are discussed here. The program logic is shown in Figure 6.5. A listing of the program is in Appendix D.

Dispatch functions are *route1()* to *route6()* inclusive and they represent the six described dispatch rules.

Function *checkbuf()* is called by all the dispatch policies. Its purpose is to determine if there is queuing space at a particular server for the current client by the time the client arrives. If the queue is not full, then the client is accepted. If the queue is full, *checkbuf()*

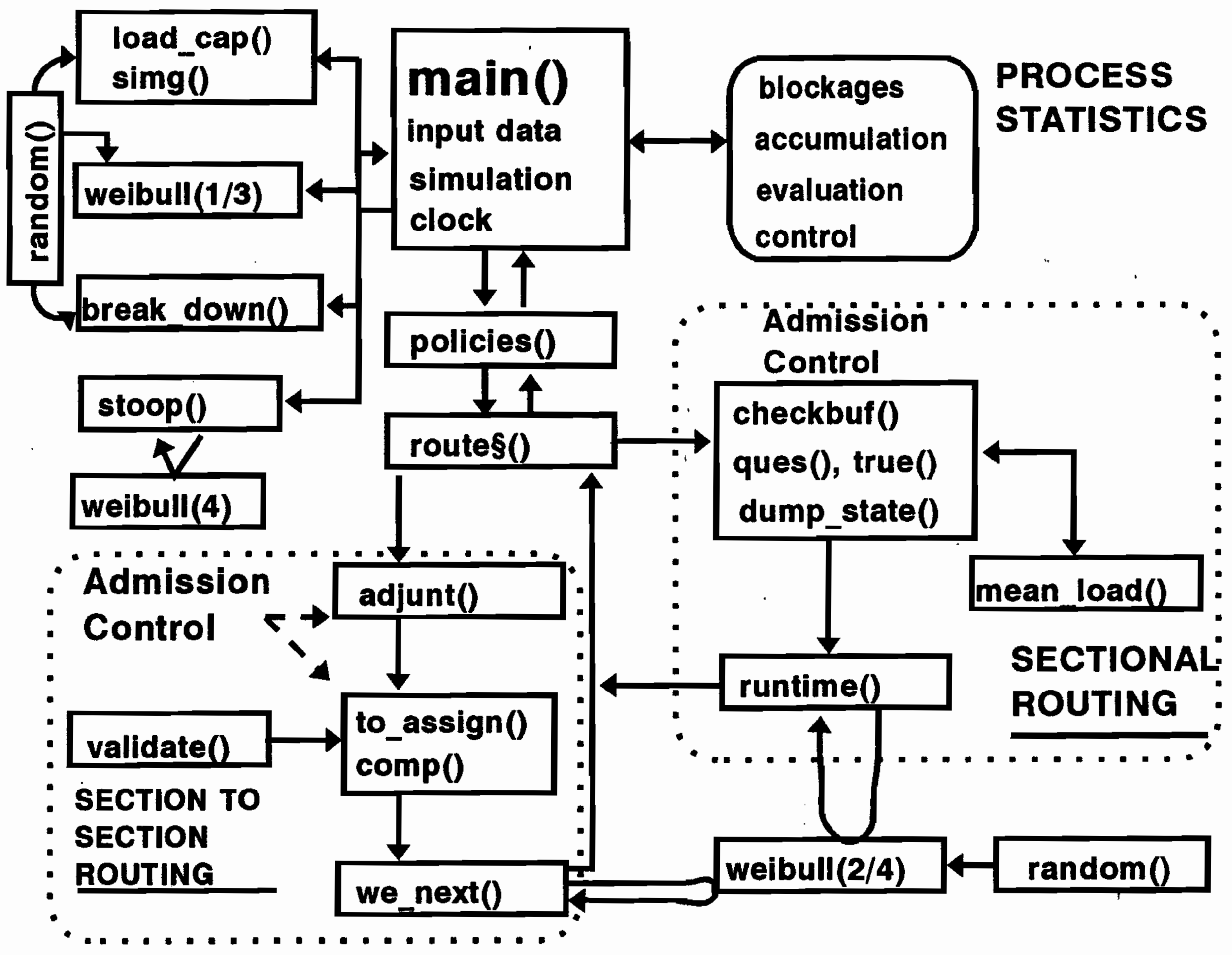


Figure 6.5 UADM logic and functions

computes the service time of the head queue member at the server based on the later's average service time (load or dump time). The travel time of the current client is then determined and compared to the service time of the head of queue. If the service time is less than the travel time, the current client is accepted as this means that when it arrives at the destination the queue there will be shorter by at least one, allowing it to queue. This procedure is done for up to two machines being expected to be serviced before a new client arrives, meaning that even when the assigned machines exceed the server buffer space, new clients are still allowed depending on the service rate.

The function *runtime()* determines the travel times between the mine network. The travel time is a sum of a theoretic minimum time allowed between any two nodes and a positive random delay component. The theoretic time is calculated based on the vehicle speed when empty or loaded as not to exceed the road section maximum allowed speed. Such speed limits are necessary both for safety consideration and the practicality of the model. The random time component is attributed to road conditions, vehicle performance and other unpredictable events. Its value is greater or equal to zero and is sampled from a three parameter Weibull distribution, with a location parameter of zero.

A Weibull distribution is used in the description of the various distributions for equipment loading, dumping, and travel times. The three parameter Weibull density function is given by:

$$f(t) = \frac{\beta}{\alpha}[\frac{t-\gamma}{\alpha}]^{\beta-1} e^{-[\frac{(t-\gamma)}{\alpha}]^{\beta}} \qquad (6.8)$$

where β = shape parameter where $\beta > 0$;

α = scale parameter where $\alpha > 0$, and,

γ = location parameter where $\gamma \geq 0$.

The shape parameter gives the distribution the ability to model various distribution shapes such as normal, exponential, log-normal and intermediate types. The scale parameter serves to compress or extend the distribution along the abscissa hence is a measure of the variability of the distribution. As the value of the scale parameter increases for a fixed shape parameter, the distribution becomes more spread. The location parameter indicates the extreme left value of the distribution along the abscissa and it represents the value at which the probability of an event's occurrence first becomes greater than zero [Mutmansky, 1972].

The cumulative Weibull distribution is given by:

$$F(t) = 1 - e^{-[\frac{(t-\gamma)}{\alpha}]^{\beta}}, \textit{for } t>\gamma \qquad (6.9)$$

The cumulative distributions of the various materials handling activities are sampled using the Monte Carlo simulation to generate the expected activity times and is represented by the function *Weibull()* in the UADM.

The Weibull parameters were determined in this study by first plotting histograms of the various activities frequency-time relationships and then iteratively fitting envelopes to the histograms. The three parameters that best fitted the observed time study data were then used in the activity modelling through function *Weibull()*. A typical example of this process is illustrated in Chapter 7, Figure 7.4.

The random time Weibull distribution is obtained through a time study of a road section where a theoretical time is first calculated. The theoretical value is then deducted from all the time study values. The resulting values represent the delays with respect to the theoretical value. A Weibull curve fitting is performed on this sample data for use in the simulation. The data analysis process is performed for both loaded and empty vehicles for each machine type. When the *runtime()* is called by the routing functions, an additional time component is added depending on the traffic in the assigned route. This time reflects the machine interference at over-taking and by-passing as the haul roads are modelled as bi-directional. The added interference time is a product of a fixed value (i.e. field measured) and the number of machines in the assigned route. The resultant travel time, tt_i is a three component expression given by:

$$tt_i = t_{o_i} + t_{r_i} + t_{v_i} \tag{6.10}$$

where t_{oi} is theoretical (minimum) travel time on route i,

t_{ri} is a random delay experienced on route i and

t_{vi} is a traffic interference on route i for current client.

Historical data of equipment availability is used to determine the fleet failure distribution. LHD machines are complex units comprised of several components. If any one of these components fails, the machine fails. Random events such as sudden excessive loading also cause failure, e.g. bursting of hydraulic hoses. Research in maintenance and machine reliability indicates that such failures can be approximated by a negative exponential distribution [Jardine, 1973].

The model function *breakdown()* determines whether an equipment that has been working for a time length, t since its last service is available or not. The function *breakdown()* uses a reliability function, R(t) to determine the probability that the equipment is still available. The reliability function is defined as:

$$R(t) = \int_t^{\infty} \lambda e^{-\lambda t} d(t) \tag{6.11}$$

where λ = mean rate of failures for the exponential distribution and $1/\lambda$ = mean time between failures for the fleet.

The function *breakdown()* tests the occurrence of a machine breakdown by comparing the reliability function R(t) to a generated random number. If the random number is less or equal to R(t), the machine is assumed still available else it is broken. If the machine is broken and is repaired within the current shift period, it is not expected to fail again before the end of the shift. This assumption is logical with respect to failures not caused by random causes as the maintenance crew checks other most likely failure prone components before returning the machine to work.

The repair time distribution is also a negative exponential because the commonest failure types require short time to repair. Such failures are electrical and hydraulics. In the model, we are interested in determining the repair time of a simulated breakdown. This is achieved by sampling a cumulative probability distribution of a negative exponential density function defined by:

$$RN = 1 - e^{-\lambda_r t_r} \tag{6.12}$$

The repair time, t_r is obtained by taking the natural logarithm of RN as:

$$t_r = -\frac{1}{\lambda_r} \ln(1.0 - RN) \tag{6.13}$$

where RN = random number in the set {0 - 1] and

λ_r= mean rate of repair of a machine per unit time

The determined time includes the response time to a breakdown and the actual time spent on repair. Once a machine is repaired, its accumulated time since the last service is initialized to zero. Typical field data of the repair time is compared to the negative exponential distribution in Chapter 7, Figure 7.3.

The constrained underground mine environment demands that equipment is dispatched only to material sources that can have feasible dump points. For example, if the ore bins are either blocked or full, then a machine requesting a loading point server has to be directed only to waste draw-points. This requirement is controlled by the function *dump_state()* which checks the current status of the dump-points and, if the constraint is not violated allows a machine dispatch.

The functions *ld_time()* and *dp_time()* represent the load and dump times at the draw and dump-points respectively. In both cases, the model samples historical load and dump distributions for each particular machine type. The use of different distributions for each machine type is necessary to correctly model the different service rates which are a function of machine power and/or size. The sampled distributions are all three parameter Weibull distributions.

Three functions deal with the assigning of equipment from one work area to another, e.g. level to level. The functions are *adjunt()*, *to_assign()* and *we_next()*. *Adjunt()* is called by the main program if either all work is complete on one level or all scheduled stopes are blocked or all dump-points are unavailable; equally, if one of an ore stope/bin or waste stope/bin system is not compatible, i.e. one is available whilst the other is not. The function *adjunt()* assesses the adjacent sections for work and selects the nearest section for equipment dispatch. In the event of blockages on the current area, *adjunt()* assesses the time remaining before start of lunch break or end of shift and if this time is small to warrant equipment movement, it sets the machine to the status of idle.

The work assessment by *adjunt()* is not exhaustive therefore *main()* calls the function *to_assign()* which determines if there is need for equipment on the section selected by *adjunt()*. A situation may exist whereby the dump-capacities are met and dump-points are blocked on the next section yet scheduled material is still available. *Adjunt()* fails to resolve this problem but *to_assign()* does, in which case it prevents a dispatch to such a section and assigns the equipment into status of idleness.

In the event of a machine being successfully dispatched to a new area, the function *main()* calls *we_next()* to determine the first loading point for the in-coming machine.

The policy that is applied for this new client is the earliest expected service. After this initial assignment the machine is treated like the rest of the machines on the same work section using the basic routing procedures.

Occasionally, in a truck and loader system the loader goes on breakdown whilst a number of machines are queuing for service. Similarly, the draw-point may be blocked for a LHD system. In such cases, the machines renege from their blocked queue(s) to join active lines. This process is achieved by function *stoop()*. *Stoop()* estimates the time that the current loader breakdown or blockage will last and then compares it to the earliest time a machine currently at the broken server will be served at another server on the same work area. If a machine can be served successfully, before the current server is available, the machine is dispatched on an earliest expected service destination. This rule is used as it mitigates the already lost time waiting in the current position and subsequent travel to a new server.

A futuristic real time grade sampler or sensor operating at the face is simulated by function *simg()*. The sensor scans each LHD or truck load and determines the grade which is then relayed to the central controller via modems. The concept of a grade scanner emanates from the fact that grade is a fuzzy variable and current mining practices simply use sparse data estimates. Magnetism, X-ray fluorescence, reflectance and radioactivity are some mineral properties that could be physically measured in the field. Even though such techniques could be crude, the large sample size that could be obtained is likely to shed more information than few assay values. The highlighting of the need of such instrumentation is one step towards galvanizing research into such an area.

The function *simg()* simulates deviations from the stope target grades as laid out in the goal programming schedule output. A sinusoid variation as a function of cumulative production has been used to mimic bands of mineralized zones of high and low grade ore. The generated grades are then recorded to file by *main()* for future evaluation of the on-going production process at the review time windows.

A function *load()* simulates a weight sensor on a haulage machine. Such sensors are already in use on some operations' LHDs to accurately measure the actual mucked tonnage. Load weighing systems are commercially available from several manufacturers such as Teledyne Specialty Equipment, Rayco Electronic Systems Ltd and Cast Resources Equipment of Sudbury Ontario. The problem of load counts as a measure of tonnage measure is inaccurate due to variable bucket fill factors. It is common knowledge that machines tend to be overloaded when working in well fragmented muck and under-loaded if the muck is coarse. In this model the load factors are simulated by random sampling of a triangular distribution in which the user specifies the lowest and highest likely deviations of a bucket fill from the optimum. A field monitoring study of loading operations enables such values to be determined statistically.

6.11 Control Model

A production system is generally complex and the input data is dynamic. This causes shifts from the set goal or objective. In order to control a situation, a desired state is specified and devices integrated with the control functions are used to try to achieve the objective. Due to deviations that will necessarily occur, a feedback mechanism is essential for controlling or stabilizing the process. It is this aspect of feedback that makes real time systems invaluable as processed output data is quickly fed back into the system to correct a deviation and minimize the error between the desired and actual states. Human feedback control tends to be limited to simple systems where the information amount is small for quick human computation. As the system becomes complex and large amounts of data is present for evaluation, the human control becomes ill-defined, being applied too late in the process and usually over-compensated, which causes system instability. In the UADM, the computer handles the data evaluation and presents a listing of the goals status to the decision maker to effect the feedback by accepting or terminating the system.

In a control system, samples are collected at intervals and these samples are used to evaluate the process status as to whether it is in control or not. Control is effected when the process output exceeds certain set control limits. The control function could be based on the error magnitude, trend analysis or the rate of change of the error. In this model for underground mining the following are defined as the control parameters:

1. mean absolute grade (quality) deviation from the shift target.
2. deviation of actual to planned production at each sample interval, and
3. the fleet utilization.

The mean absolute grade deviation (MAD) is calculated as:

$$MAD = \frac{\sum_{t=1}^{n} |e_t|}{n} \qquad (6.14)$$

where $|e_t|$ is the absolute value of the deviation from target in interval t of the process.

The value of MAD is then used to set the limits beyond which a control intervention is required. The limits are defined as q*MAD where q is a real number representing a shift from the centre of the control line of the control limits. The setting of limits is based on management needs and not related to the process system.

Graphical outputs are generated in the UADM that show the cumulative deviation (error) of the shift grade from the planned as a tracking signal. This cumulative deviation, D_t is defined as:

$$D_t = \sum_{t=0}^{n} e_t = \sum_{t=0}^{n} (g_o - g_t) \qquad (6.15)$$

where g_o is the planned mean grade for the shift,

g_t is the observed grade at interval t of the shift and

n is the number of sample intervals (review windows).

The production and utilization are controlled by management by setting control limits as in the grade control goal. Utilization is influenced by waiting times at service points. Draw-point loading problems cause queue build-up and loss in utilization. Therefore, if this occurs in one area of the mine, the model corrects the situation by routing to other areas. We apply the utilization goal to test the effectiveness of the different dispatch rules in maximizing this goal.

The UADM has a nine rule knowledge base which is used in the evaluation of the three goals of production, quality and utilization illustrated in Chapter 7, Figure 7.14. This 3 x 3 matrix shows the three goals and their possible priorities. The user chooses a priority mix for part or the whole shift. The priority mix is then used in the sequential evaluation of the goals at sample times. The evaluation process indicates the status of all goals under the current dispatch policy and goal priority mix; i.e. which are or not in control.

The control knowledge base is expressed in the form of IF (condition of goals exists) THEN (evaluate as either in control or out of control). A sample of UADM control rules is listed below:

```
    IF (objective is max (quality, production,utilization)) THEN
     IF ((productivity $\geq$ T) AND ($|$quality$|$ $\leq$Q) AND (utilization $>$ U))
       THEN { process in control on all goals }
        ELSE { objective not met by at least one goal }
    ELSE IF (objective is max(quality, production mainly)) THEN
      IF (($|$quality$|$ $\leq$ Q) AND (production $\geq$ T))
          THEN { process in control on the two specified goals }
          ELSE { objective not met due to one or both goals }
        ..........
    ELSE IF (objective is max (production i.e. single objective))
      THEN IF ( production $>$ T ) THEN { process in control }
  ELSE { process is out of control }                                  (6.16)
```

where T,Q,U are management control limits.

The evaluation outcome is provided to the decision maker to accept, reject or modify the current production policy and rule.

6.12 UADM Performance Measures

The UADM is a stochastic model because data changes every time period. This makes the inference of the output of the simulation at the end of the simulation run a more involved process. A measured variable's value at the end of the simulation does not necessarily represent the true model performance. The output values vary. The fluctuations are statistically described by their variance, σ^2 about the true performance (mean) of the variable. The mean of each simulation variable is determined as:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \qquad (6.17)$$

where x_i= the individual simulation variable output value

n = the number of simulation runs representing a sample

μ = the sample mean

The simulation re-runs are independent being seeded with different start numbers for each run. This assertion is correct since the events in each shift production are independent of other shifts. For independent output values x_i, the confidence on the mean, μ is estimated by:

$$\sigma^2_{\mu} = \frac{\sigma^2}{n} = \frac{1}{n} \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n} \qquad (6.18)$$

The different dispatch policies are simulated individually for the same mine layout, resources and production schedule. The policy performance is then measured on the basis of the following measures:

- Productivity: The time taken to complete a given schedule is a measure of the productivity of a policy, as expressed in tonnes per unit time.

- Cost minimization: The running time necessary to complete a set tonnage goal is an indication of operating costs. Besides time elapsed, the total distances travelled during the shift are used to compare the policies, since distances have a direct impact on tire cost and reduction in productivity.

- Equipment utilization is evaluated through total lost time due to queuing process. This is a statistical result because it is affected by random events of server blockages.

- Quality: The main purpose of quality control is to build quality into the product at the first attempt. The mining industry practice is to schedule production based on quality only at the material source. In open-pit mining, a scheduled ore pocket can find its way directly into the crusher from the mine without being stockpiled. Stockpiling is done usually for back-up and not as a routine as it incurs re-handling costs. The consideration of the source quality alone has problems in instances when ore ends up in holding underground facilities for fairly long time. The problem is due to the uneven draw that is subsequently carried on these ore holding facilities that produces streams of ore for the mill widely different from the initial planned grade based on the ore sources.

The UADM successfully tracks the quality of the materials in the holding facilities. The original source, grade, tonnage and most importantly the order in which a particular load was dumped into that facility are recorded. Assuming either a mass-flow or a funnel flow, the management will be able to reconcile the material they pull from the holding facilities at a later date and develop a secondary schedule based on the facilities material mix.

The dispatch rules are compared with respect to the resulting product mix in the ore-bins. The aim is to determine which policy is most suitable for blending ores especially if the material is dumped directly to the crusher ore-bins.

6.13 Limitations

The UADM does not have a traffic control algorithm necessary for safety and collision avoidance. This model assumes bi-directional haulages allowing by-passing and/or over-taking. This is an expensive proposition that may exist probably only on the major haulage levels. To minimize over-productivity due to the two way haulage way, the model has a control conditional that limits the number of machines that can occupy any one road section. In addition, the model accounts for delays due to machines by-passing or over-taking. This aspect in fact defines the haulage ways as restricted as machines have to slow down, or stop at in-bays to allow safe passage of the other machine (usually the loaded machine).

An intelligent traffic control algorithm is excluded because it has to be knowledge-based which is mine layout specific. A simple control such as the highway traffic lights is inappropriate as it is time dependent and not traffic volume related.

6.14 Summary

The UADM is an on-line computer-assisted model in which the computer monitors the production process, evaluates the process and gives specific suggestions to the scheduler for action to be taken. The operating instructions of the process are coded in the computer program logic. The format of the UADM enables the operator to either accept or reject the computer recommendation.

The use of a computer based system in equipment dispatch and production control has several advantages:

1. A computer has a high ability to hold and retrieve information. This data is available at a central point and can be distributed to each mine section. The production plan is run as specified by management unlike in conventional control systems where human interaction can affect delays and poor routing choices.
2. Computers can perform complex calculations on process data they monitor enabling more effective corrective action in real time.
3. Changes to the production system inputs can be performed easily and immediately acted upon according to the dispatching rule. Under conventional methods, much time is lost in getting the new system inputs to the work area and the new instructions are not necessarily acted upon on receivership.
4. The exact times of equipment breakdowns and the machine location is known in UADM. This information can be easily distributed to service crews. If the model is operated in combination with equipment diagnostics, then the service crew will know exactly the problem, enabling it to carry the right spare parts and tools. This results in reduced down-time and increased productivity.

Chapter 7
Dispatching Model Validation and Simulations of Policies

7.1 Purpose

The purpose of this chapter is:

1. to validate the underground active dispatching model using simple mine networks. This is achieved by comparing the production results obtained using a deterministic numerical method to the model's simulated results for identical mine layouts.
2. to evaluate the six developed dispatching policies in terms of the shift production, unit production cost, fleet utilization and product quality in a complex underground mine network, and
3. to identify the best operating conditions for each of the dispatch policies. This leads to the development of operating strategies that best meet the shift objectives.

7.2 Model Validation

A simple one level mine layout is used with two ore sources, A and B and a single ore-pass, C. The scheduled amount of work from each stope is 400 tonnes. The ore grades are 2 and 4 % copper for stope A and B respectively. The distances between the ore-pass and stopes A and B are 50 and 130 m respectively. The maximum loaded LHD speed is 1.26 ms^{-1} and 1.46 ms^{-1} for empty machines. A fleet of three ST8 LHDs is used each with a capacity of 12 tonnes. The routes allow a maximum of two machines at any given time.

The loading, travelling full and empty, and dumping times are described by Weibull distributions with the shape, scale and location parameters indicated in Table 7.1.

Table 7.1 Weibull parameters for the activity distributions

Activity times	Scale factor, α..s	Shape factor,β..s	Location factor, γ..s
Travelling loaded	60	2.0	0
Travelling empty	60	1.5	0
Loading time	16.8	1.42	40.0
Dumping time	14.2	1.39	25.5

The average loading and dumping times are calculated from the mean value of the respective Weibull distributions to be 57 and 39 seconds, respectively. The average delays (t_{rand}) on travelling loaded and empty are 60 and 52 seconds respectively. These delays are also determined from their respective Weibull distributions.

7.2.1 Deterministic Production Statistics

The average LHD cycle times are detailed in Table 7.2.

Table 7.2 Operating cycle times for sample mine layout

Cycle Activities	Stope A to ore-pass C ..s	Stope B to ore-pass C..s
Loading time	57	57
Dumping time	39	39
Travelling loaded, $t_d + t_{rand}$	40 + 60 = 100	106 + 60 = 166
Travelling empty, $t_d + t_{rand}$	34 + 52 = 86	89 + 52 = 141
Average spotting time, by-passing, over-taking	20	20
Total cycle time per load	302	423

Time required to deplete stope A is given by: (302 s/12 t)* 400 t = 10067 s

Time required to deplete stope B is given by: (423 s/12 t) * 400 t = 14100 s

Total production time required is (10067+14100 s) = 24167 s

The average production time for each machine is (24167 s/3 LHD) = 8056 s

Average waiting time (spotting and delays): 20 s/load * 67 loads = 1340 s

The operating cost per hour for each machine is $80.00. Therefore, the total operating production cost is ($80.00 * 24167 s/3600 s/h) = $537.00.

The average operating cost per tonne is $537.00/800 t = $0.671/tonne.

Total travel distance covered during operation:

2 {(50 m * 33 loads) + (130 m*33 loads)} = 11880 m

7.2.2 Simulated Production Statistics

The following synonyms are used for the six developed dispatch policies:

MaxQ = maximum quality and/or maximum contained product first

MinS = minimum slack time between an LHD arrival and it being serviced first

EEST = earliest expected service time destination first

STT = shortest expected travel time to a destination first

MaxQ/MinS = combined policy of MaxQ and MinS selected first, and

CRatio = maximum critical ratio of unfinished work first.

An identical mine layout as that used for the numerical example, was simulated for each of the six dispatch policies. Since the underground active dispatch model (UADM) is a stochastic model, a total of twenty-five simulations were done for each policy and the statistical production results of those simulations are reported in Table 7.3.

The numerical results described above in Section 7.2.1 and the simulation results tabulated in Table 7.3 show a strong similarity indicating that the UADM operates as expected. The simulated unit production costs are within ± 4 % from the theoretical value for all the policies. Similarly, the production time ranges from -2 % to 4 % of the theoretical value. The productivity is always less than the target by no greater than 2 % for all the policies. This is caused by a condition in UADM that restricts machine loading if a certain bucket fill factor is not possible from the remaining scheduled material at a stope. The UADM is operated under a constrained production, therefore the total shift production cannot exceed the scheduled tonnage restriction.

Table 7.3 Simulated production statistics for a sample mine layout

Policy -->	MaxQ	MinS	EEST	STT	CRatio	MaxQ/ MinS	Numeric example
Production, t	790	785	786	789	791	787	800
Cost.. $/t	0.658	0.678	0.661	0.641	0.657	0.699	0.671
Production time..s/LHD	7900	8257	8050	7980	7968	8487	8056
Distance travelled..m	12128	12067	12153	12045	12067	12050	11880
Waiting time..s	1877	3233	2033	2524	2060	3120	1340

7.2.3 Dispatch Policy Effect on Simulated Product Quality

The mean grade, standard deviation of dumped loads and grade profile in the ore-pass, C were determined for two simple mine layouts:

1. A two stope mine layout with one high grade (HG) stope at 4 % copper near ore-pass C and the other stope being low grade (LG) at 2 % copper and far from C.
2. A similar two stope mine layout with the near stope being low grade (2 % Cu) and the further stope, high grade (4 % Cu).

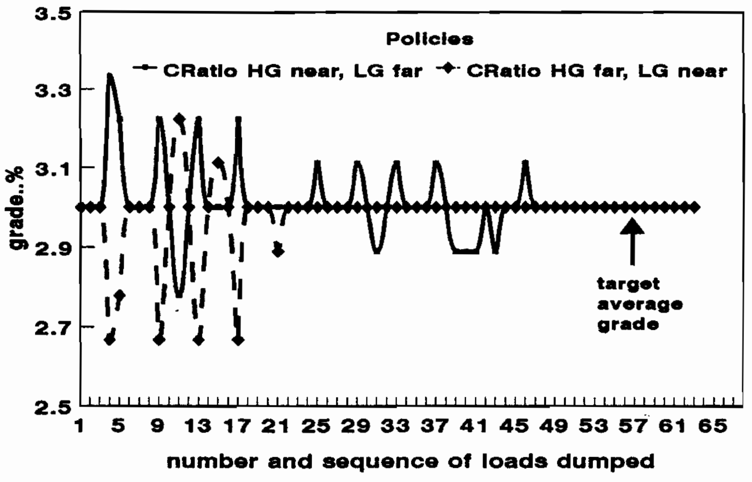
The scheduled amounts in both layouts were identical, namely, 400 t from each stope. The distances between the ore-pass and the two stopes were kept constant at 50 and 130 m. Twenty-five UADM simulations were performed for each mine layout using each dispatch policy in turn. The grade of each load (2 or 4% Cu) and its number in the dumping sequence at the ore-pass is recorded. The grades of the same dumping number (i.e. 1 to 66) in the 25 simulations are averaged. Then, the moving average grades are calculated using these averaged values of the 25 simulations. The moving average is used because (1) it is a forecasting method that establishes trends in a sample population through interval smoothening effects, and (2) the interval reflects the physical blending of material as it is dumped into an ore-pass.

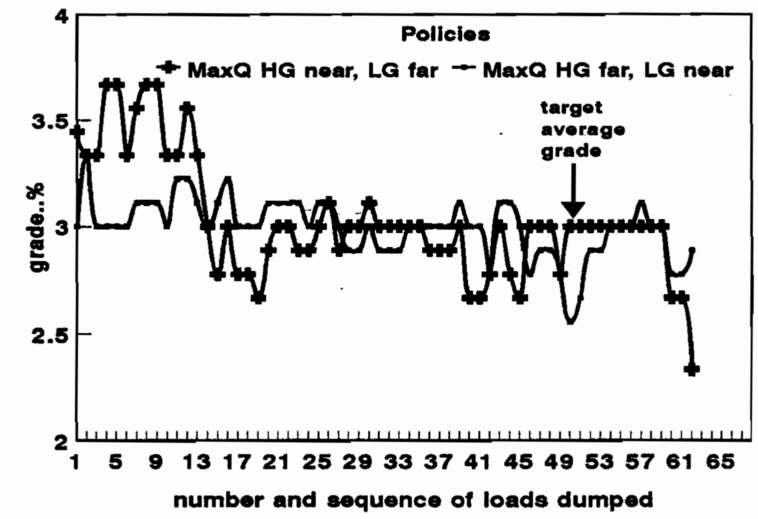
The global mean grade and standard deviation is then calculated based on the moving average grades. The results for the six policies are listed in Table 7.4 for both mine layouts. These results are comparable to a theoretical one based on a consecutive HG-LG load dumping at a moving average interval of six. The later has a mean grade of 3% Cu and a sample standard deviation, based on the moving average grade values, of 0 % Cu. The mean grades in Table 7.4 are not exactly 3% Cu because the production of UADM is not exactly 800 t, caused by a restriction discussed above in section 7.2.2.

Table 7.4 Comparison of policy simulated grades for different layout stope grades

Mine layout --->	HG near and LG far		HG far and LG near	
Policy	Mean grade, % Cu	Standard Deviation, %Cu	Mean grade, % Cu	Std Deviation, %Cu
CRatio	3.02	0.09	2.98	0.10
MaxQ	3.03	0.28	2.99	0.14
MaxQ/MinS	3.02	0.54	2.99	0.40
EEST	3.05	0.40	2.98	0.23
MinS	3.02	0.40	3.00	0.53
STT	3.01	0.50	2.98	0.52

The grade profiles for the two layouts show a pattern where one layout grade profile is a 'mirror reflection' of the other for each policy. In other words, the grade deviations from the mean grade are often on opposite sides of the average. This is shown in Figure 7.1 for CRatio, MaxQ and STT policies. Other policies show similar characteristics. This is the expected result for identical mine layouts differing only in the location of the high grade ore.





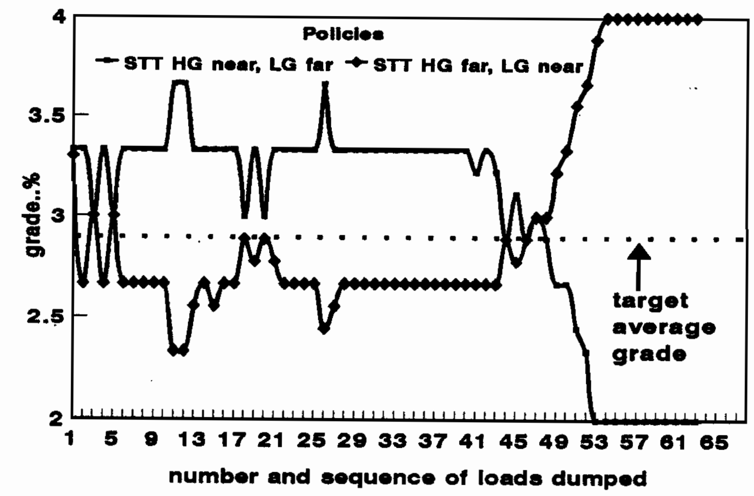


Figure 7.1 Validation of UADM ore-pass quality profiles

The ore-pass quality profiles indicate that the CRatio policy has the minimum variation from the target grade of 3% Cu under both mine layouts. This policy is therefore independent of the initial location of the high or low grade areas relative to the ore-pass. MaxQ policy has a smaller variation when the high grade area is further from the ore-pass. In the case of nearer high grade stopes, MaxQ initially high grades the mucked material, and at some point starts behaving like the CRatio which produces very small variations. The STT, MinS and MaxQ/MinS policies have similar profiles. The STT has the largest cumulative number of loads dumped either consistently above or below target. MaxQ/MinS has larger variations but less consistency compared to STT. The EEST policy shows moderate variations from target and it gives a consistent product profile except near approaching stope depletion.

These simple simulation tests have served to validate the UADM, in terms of producing realistic production levels and illustrating how the dispatching policies achieve various degrees of grade control.

7.3 Introduction to Simulations of Dispatching Policies

The dispatch model implements a production plan. Therefore, it is essential to measure the effectiveness of the various dispatch policies discussed in the preceding chapter under different shift production environments. The objective is to investigate the possibility of of existence of relations or effects of the policies on defined mine environments. If such relations exist, then they can be adopted as strategies to consider and utilize at the shift review intervals, so as to direct the shift production towards meeting the schedule targets.

The UADM policies are based on single objective criteria. This implies that different policies are likely to perform differently under a given set of system constraints such as number of LHDs, material sources and haulage traffic restrictions. The study intends to establish the sequence (if any) of policy changes that must be implemented when machine breakdown or draw- and/or dump-point blockage or significant deviations on product quality occur.

The economic aspects of the dispatch rules must include the shift production benchmarks: high productivity, satisfaction of the product quality and minimizing the shift production cost. By satisfying these goals, the underground dispatch model would indeed have achieved the shift production targets as set through the mathematical optimization of goal programming.

7.4 Description of the Simulated Underground Mine

A shift production process is simulated under different dispatch policies and at varied LHD fleet sizes ranging from four to twelve machines. The make of the LHDs are ST8A and ST8B with bucket capacities of 12 and 13.5 tonnes respectively. The simulations are conducted on a fixed production schedule generated by the goal program outlined in Chapter 5, for a hypothetical Inza Mine. The schedule is shown in Table 7.5 and UADM data file is in Appendix E, Table E1. The stope priority of one implies a higher rank compared to that of two, etc. Such priorities may be present in production schedules where management wants to expedite certain jobs or wants to meet a certain ore quality for the shift.

The simulated mine haulage network as it appears on the computer monitor is illustrated in Figure 7.2 and consists of two production levels (or work sections) labelled Level 1 and Level 2. The network consists of routes, material sources and dump-points marked by lines, circles and ellipses respectively. A ramp connects Level 1 and Level 2. The bins and material sources scheduled capacities on each level are indicated on the right of the screen display as columns labelled B# and S# respectively. The shaded part of the sources indicate the scheduled material remaining to be mucked whilst for the bins, the shaded component represents the dumped material to date. A material source or dump-point blockage is highlighted by change in the colour coding of the respective source or

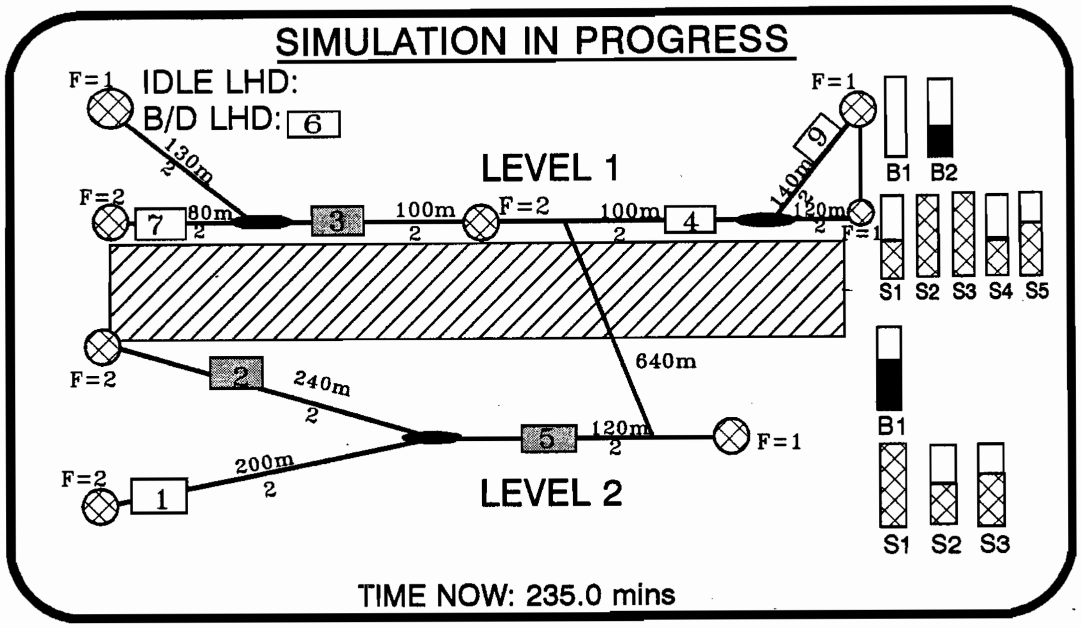


Figure 7.2 Screen display simulated Inza Mine layout

bin shading. Idle machines are posted to the right of "IDLE LHD". Broken machines are posted to the right of "B/D LHD". Active machines are indicated as numbered rectangular icons with a colour coding reflecting whether they are travelling loaded or empty. The position of these vehicle icons marks their position in the mine network at that point in time. The current time is indicated at the bottom of the computer monitor by "TIME NOW".

Level 1 has five material sources (L1_S1 to L1_S5) and two dump-points whilst Level 2 has three material sources (L2_S6, L2_S7 and L2_S8) and one dump-point. On Level 1, four material sources are ore areas and the fifth is a development area in waste. The waste is dumped into the dump-point number B1 whilst all the ore goes to dump-point, B2. All material sources on Level 2 are considered ore and the ore is dumped into a common dump-point. The two levels are connected by a 640 m, 15 percent ramp. The distances between the draw-points and the ore-passes and the permitted maximum number of equipment per road section are indicated in Figure 7.2. The queuing buffers at the draw-points are also indicated by the expression: F= Number, e.g. F=2.

Table 7.5 A typical Inza Mine shift schedule

Stope number	Tonnage.. tonnes	Grade.. %Cu-Ni	Priority of stope	Stope LHD buffer size
L1_S1	400	2.0	2	1
L1_Development	100	0.0	1	2
L1_S3	210	3.0	1	2
L1_S4	200	2.8	1	1
L1_S5	300	3.0	2	1
L2_S6	500	2.5	2	2
L2_S7	250	3.0	1	2
L2_S8	300	2.8	1	1

The haulage routes have restrictions on the number of machines allowed to be on the same section at the same time as this would lead to excessive delays and traffic jam. Each source and dump-point has an upper limit of the number of machines that may queue and this number is generally low, namely one or two, as longer queues imply large excavations for such holding buffers (see Table 7.5). Maximum speed limits for LHDs travelling loaded and empty are imposed on each road section. The mine layout data file required in the UADM program is shown in Table E2 in Appendix E.

The fleet availability was simulated based on a nine-month statistical field data obtained from the McCreedy West Mine that operates an ST8B fleet [Knights, 1993]. The mean time between failures of that fleet was 14.5 hours. The equipment repair times were simulated based on a cumulative distribution of an exponential probability distribution with a mean service time of 1.48 hours which approximated the actual statistical data as shown in Figure 7.3.

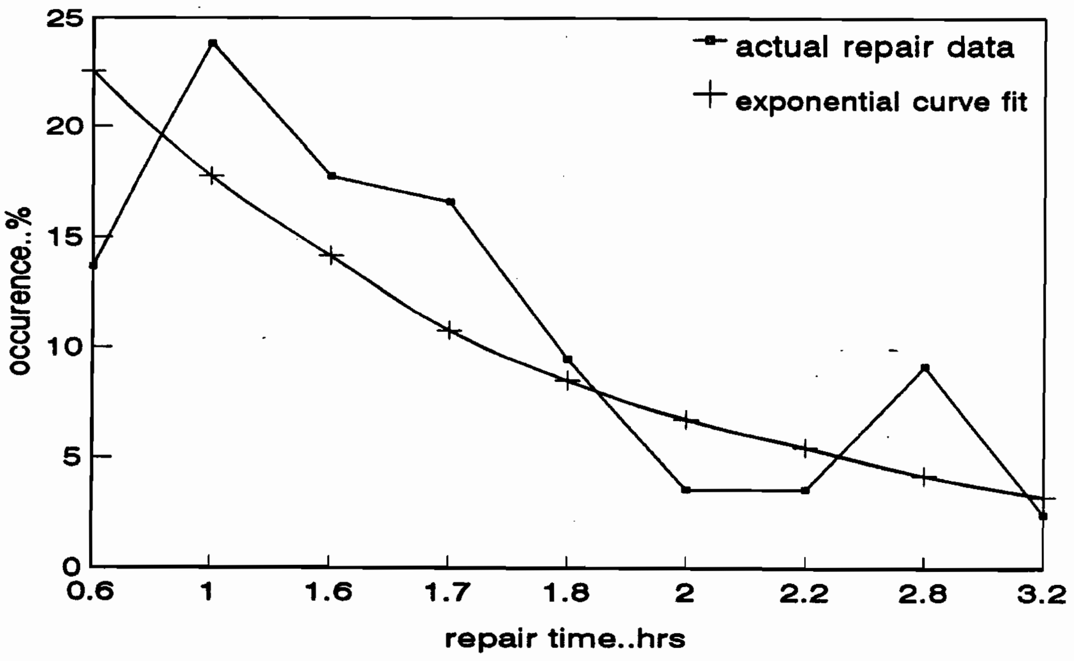


Figure 7.3 Approximation of LHD repair time function based on actual field data

The load, dump and travel time distributions were simulated using the Weibull three parameter distribution which was fitted to published field data [Hill, 1987]. The load and dump time distributions are shown in Figure 7.4 and show the fitted Weibull probability distributions. The parameters of these distributions are indicated in Table 7.1. The format of the equipment input data file of the UADM program is described in Appendix E, Table E3.

The bucket fill factor was simulated based on a triangular distribution where 15 and 5 percent were assumed as the largest negative and positive deviations a bucket fill can have about the optimal load, respectively. These values can be varied depending on the local fragmentation of the ore being mucked. Material sources and dump-point availabilities were simulated based on a 5 and 2 percent probability of blockages during the entire shift. Monte Carlo sampling is performed on all the fixed facilities to test for these blockages which are assumed to occur when a random number less or equal to the mentioned probabilities is generated. In real world practice, the LHD operator would

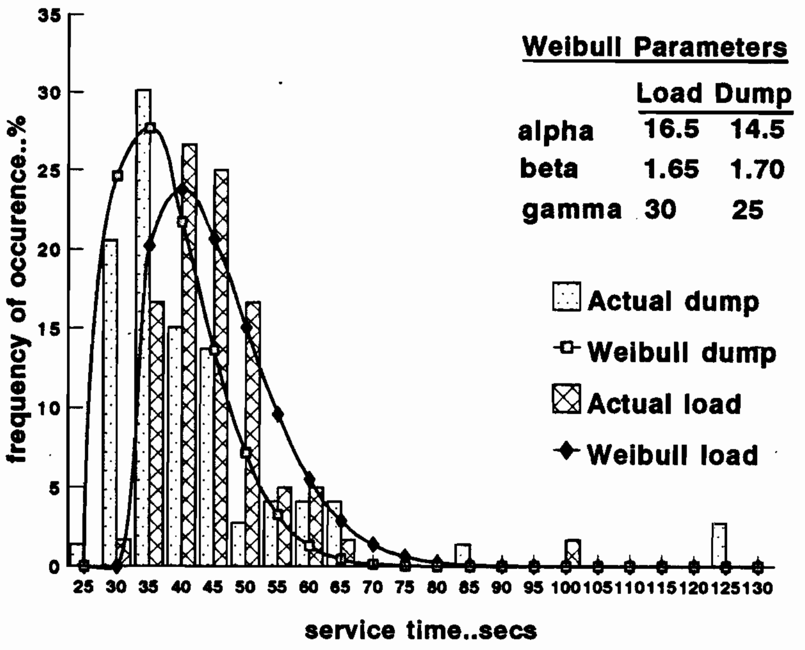


Figure 7.4 Weibull distributions for actual load and dump time activities

directly relay a message of the occurrence of such blockages to the dispatcher in real time.

The LHD operating costs per hour were set at $80 for both the ST8A and ST8B machines. Any available machine is assumed to incur the $80 per hour cost unless the scheduled quantities have been completely depleted (termination of the shift). This approach addresses the worst case scenario compared to situations where idle machines are assigned lower cost values.

7.5 Assessment of the Dispatch Policies

The six new dispatch policies for underground trackless mining were assessed on the basis of the shift production goals of productivity, cost, and fleet and/or system utilization. The shift schedules were all production constrained, i.e. the shift production could not exceed the scheduled quantities. A total of twenty-five simulations were conducted on each policy for a given fleet size for a total of nine fleet sizes. The results of these simulations are reported in the following paragraphs.

7.5.1 Fleet Productivity

The productivity of each individual dispatch policy was measured at the middle of the shift period because at this stage the production system is free or nearly free of the transient effects associated with the shift start. Equally, the end of the shift is marked with uneven distribution of resources. This is due to a decrease in the number of material sources through depletion of the scheduled amounts. The result is that equipment become idle.

The simulated results indicate the STT policy has the highest productivity for LHD fleets less than nine (see Figure 7.5). The next most productive policy is the MinS whose productivity on average is merely 1.5% less than the STT, for fleet sizes smaller than nine. The least productive policy is the MaxQ which is universally lower than all the other five policies at all fleet sizes greater than four. At a fleet of four LHDs, MaxQ is marginally higher than the EEST policy. Between the upper and lower productivity limits, are the CRatio and MaxQ/MinS policies. The productivity of the six policies for fleet sizes less than nine LHDs are ordered as:

$$\{\text{STT} > \text{MinS} > \text{MaxQ/MinS} \geq \text{CRatio} > \text{EEST} > \text{MaxQ}\}$$

The productivity of the six policies at greater or equal to nine LHD fleet sizes are ordered as:

$$\{\text{CRatio} > \text{MaxQ/MinS} \geq \text{MinS} \geq \text{EEST} > \text{STT} > \text{MaxQ}\}$$

The CRatio and MaxQ/MinS policies supersede the STT and MinS at fleet sizes greater than nine LHDs because the former policies have even material draw from all the sources. This maintains the number of available material sources high throughout the work shift. The STT and MinS policies preferentially maximize production from the material sources with short haulage distances over long haulages. Therefore, the short

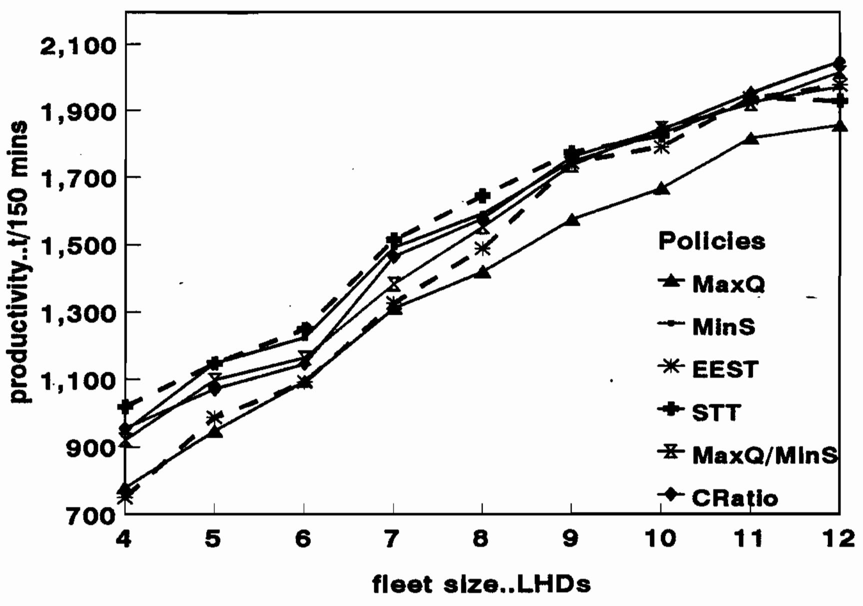


Figure 7.5 Comparison of fleet productivity under different dispatch policies

haulage sources are depleted first. As a result, there is an increase in equipment interference at the remaining work sites because additional equipment is assigned. The results indicate that the CRatio and MaxQ/MinS policies are superior to the STT and MinS policies under conditions of large fleets.

The productivity for each policy increases non-linearly with an increase in the fleet size. Initially, the productivity rate increases at an increasing rate with an increase in the fleet size. Finally, the productivity rate increases at a decreasing rate after a specific fleet size is exceeded for each dispatch policy. This trend of the productivity is illustrated well by MaxQ policy in Figure 7.5.

The implication of these trends is that each policy has its own optimal fleet size for the given mine layout at which productivity is maximized. The observed high productivities of STT and MinS policies at smaller fleet sizes can be inferred directly from these trends. With the exception of MaxQ policy which has a much lower productivity, the rest of the policies show a clustering effect in their productivity at fleets of nine to eleven before fanning out at fleets greater than eleven machines.

7.5.2 Production Cost

One of the shift goals is to meet the production tonnage at a minimum cost. The average unit production cost is calculated as:

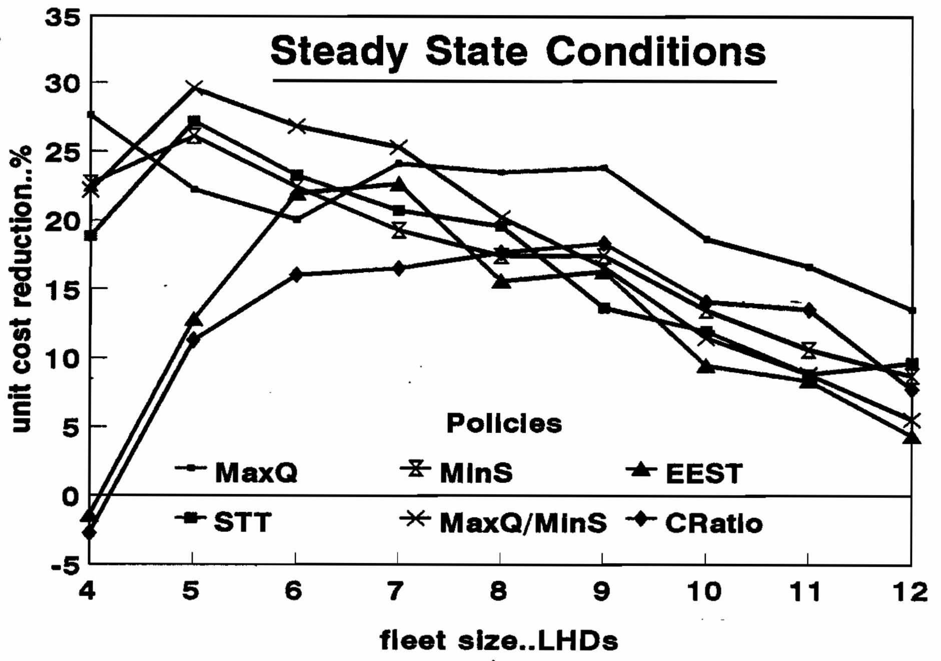
$$ave\ cost_t = \frac{\sum_{i=1}^{n} C_{i_t}}{\sum_{i=1}^{n} P_{i_t}} \tag{7.1}$$

where P_i = cumulative tonnage of machine i at time t, and

C_i = cumulative cost of the machine i by time t (i.e. \$80 hr^{-1} * t hrs).

Therefore, to minimize the average unit production cost requires the denominator in equation 7.1 to be large. This means that policies which have high productivity will also have a low unit production cost since the numerator in equation 7.1 is independent of production.

The unit production costs were investigated at the middle and end of the shift. The middle of shift analysis is essential for study of the shift cost evolution through time and provide guidelines for controls at in-shift review periods. The results of the simulation at the half-way mark of the shift are shown in Figure 7.6 (marked mid) where the high productive policies, STT and MinS define the lower cost envelope, whilst MaxQ represents the top envelope. There is a gradual increase in unit cost with increase in the number of operating machines because these extra machines' contribution to total production is smaller than for fewer machines. The reduced productivity is caused by the system constraints which govern the traffic movement. Whilst the unit costs for EEST, STT, MaxQ/MinS and MinS are very different at fleet sizes less than nine, these values



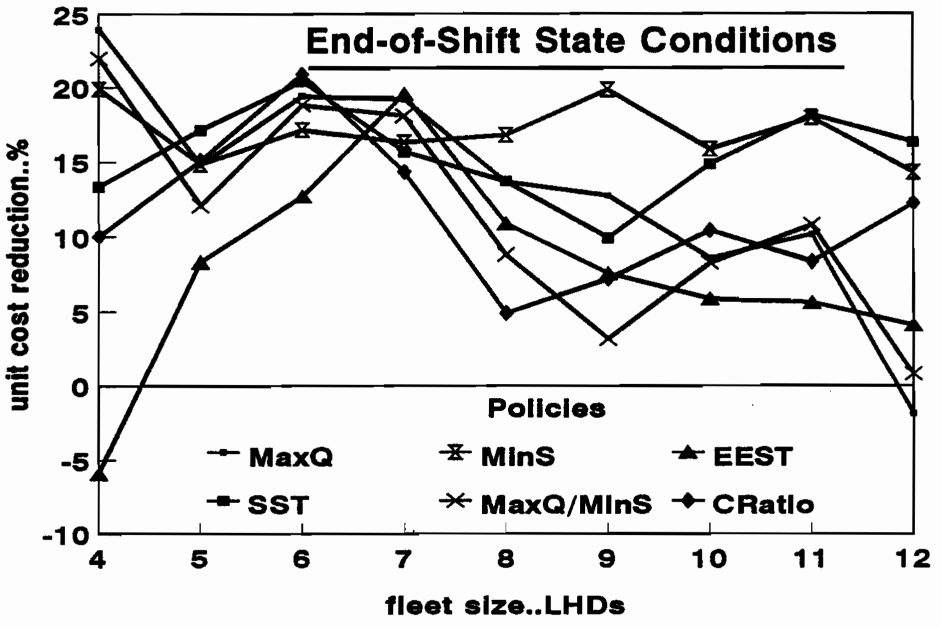


Figure 7.6 Average unit production cost at middle and end of shift under different policies

become very similar at fleet sizes of nine and greater indicating that the efficiencies of these four policies at minimizing cost become comparable under heavy traffic situations.

The situation during the earlier part or middle of the shift differs substantially with that at the end of the shift. The later part of a production shift is marked by depletion of scheduled material sources which reduces the number of equipment destinations. Since the assumption is made that the equipment remains active unless all sources are depleted, there is a reduction in effective machine productivity. The cost curves for the shift ends are also shown in Figure 7.6 (marked full) which shows that both STT and MinS policies have the least cost only at a small fleet of four machines. For fleet sizes greater than four, the STT and MinS policies progressively become the most expensive policies to operate. On the other hand, CRatio, MaxQ and MaxQ/MinS show the lowest average unit cost for fleets of five or greater.

The unit production cost during and at the end of a shift are different and are influenced directly by the policy under application. Policies such as STT and MinS are very effective in the early parts of the shift but lose to the rest of the other policies if considered on the basis of a full shift. Since the objective is to minimize the total shift budget, it is concluded that the end of shift analysis should be used to assess the cost objective in which the appropriate policies are the MaxQ, MaxQ/MinS and CRatio.

The curves in Figure 7.6 identify the appropriate cost minimizing policy to apply for each given fleet size under either steady state (early in the shift) conditions or the end-of-shift state. The fleet size has a direct impact on the unit production cost for a system that has fixed scheduled quantities, traffic volume restrictions on the haul routes as well as at the server destinations. Large traffic has a negative effect on the cost minimization objective under conditions of constrained production.

7.5.3 System Blockages

One measure of the effectiveness of a dispatch policy is determined by how effective it prevents the underground mine network from getting blocked. By measuring the total time per shift that the system is blocked, it is possible to compare the developed policies. The simulated results of this procedure are indicated in Figure 7.7 where the general

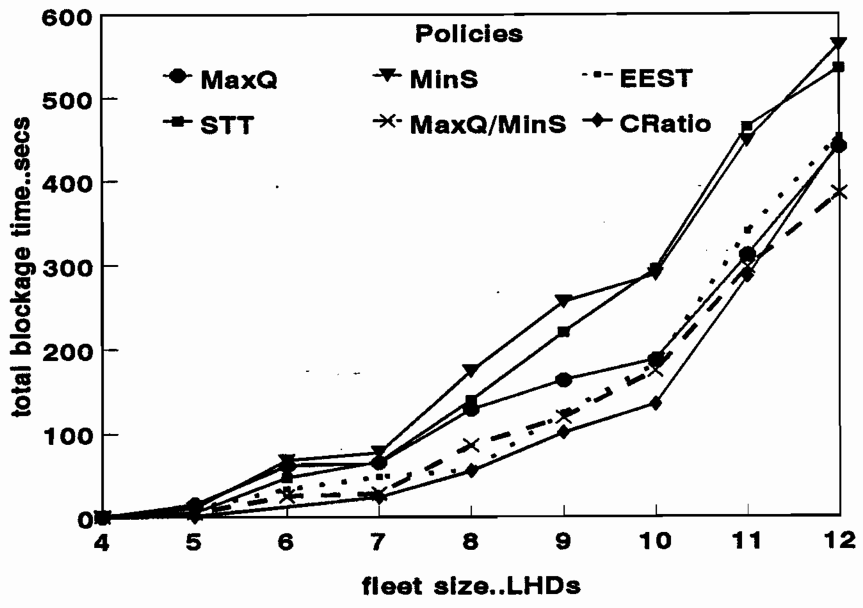


Figure 7.7 Comparison of mine system blockage under different policies

trend for all the policies is a monotonically increasing function with an increasing fleet size. The analysis indicates that the CRatio, followed by the MaxQ/MinS policy are the best policies at reducing system blockage. The worst cases are the STT and MinS which show escalated blockages at fleets greater than seven LHDs.

7.5.4 Fleet Utilization

Equipment utilization decreases with increase in the number of operating units. This relationship reflects the increased system blockages that occur under heavy traffic for a fixed mine layout. The different policies attain different utilization levels with the MaxQ/MinS and CRatio consistently showing the highest utilization. The MinS followed by STT have the least utilization. Note these results were considered on the basis of a full shift length rather than during the shift where different utilization levels exist between the policies. Figure 7.8 illustrates the simulated fleet utilization for different policies and different fleet sizes.

7.6 Effects of Fixed Installations on Productivity

The flexibility of the mine layout was investigated through changes in the number of fixed facilities, namely the material sources and the dump-points. The effective number of ore dump-points was increased by one without making any changes to the mine layout by assuming that the development work on level L1 is in ore. This eliminated the need for a waste dump-point. The only change to the production schedule is the conversion of the development waste to ore which maintains the total material production at 2250 tonnes per shift. The shift process was then simulated using the same equipment types

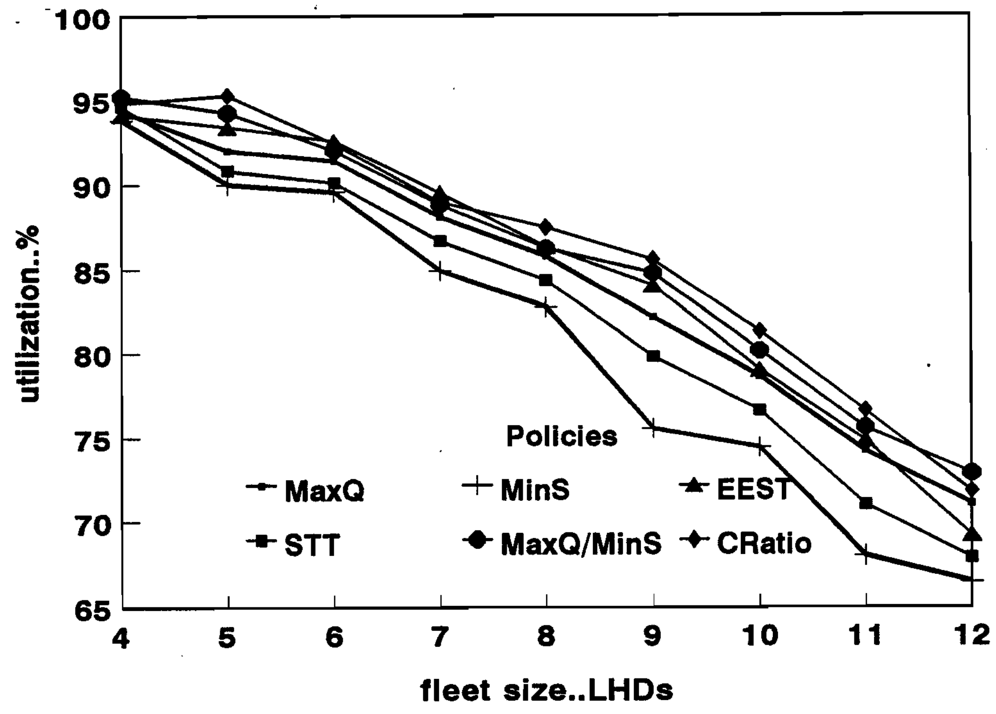


Figure 7.8 Comparison of fleet utilizations under different dispatch policies

and numbers as discussed in Section 7.4 above. The results reported here were determined from a total of 1350 simulations. Each fleet size was simulated twenty-five times in turn, for each dispatch policy.

7.6.1 System Productivity

An additional fixed facility such as a dump-point causes a significant productivity increase among all the developed dispatch rules. The results of the simulation are indicated in Figure 7.9. Figure 7.10 indicates the percentage productivity increases due to the increase in fixed facilities.

The percent productivity increase is defined as:

$$\textit{productivity increase} = \frac{100(P_f - P_i)}{P_i}\% \tag{7.2}$$

where P_i = production under initial (base) mine layout and

P_f = production under increased fixed facilities

The productivity increases of 43 and 49 percent are obtained for the MaxQ and MaxQ/MinS rules respectively. The CRatio rule is the least sensitive to the change in the number of fixed facilities with a maximum productivity increase of 31 percent for a fleet of six LHDs. With the exception of MaxQ, the other policies attain maximum productivity increase at a fleet of six machines as indicated in Table 7.6 by numbers highlighted in bold. MaxQ attains a maximum productivity increase for a 4 LHD fleet because the additional facility is closer to high grade sources. Therefore, the policy exhibits a feature typical to STT of maximizing production from the nearest work centres. The small fleet ensures that the traffic volumes along these short routes are not violated.

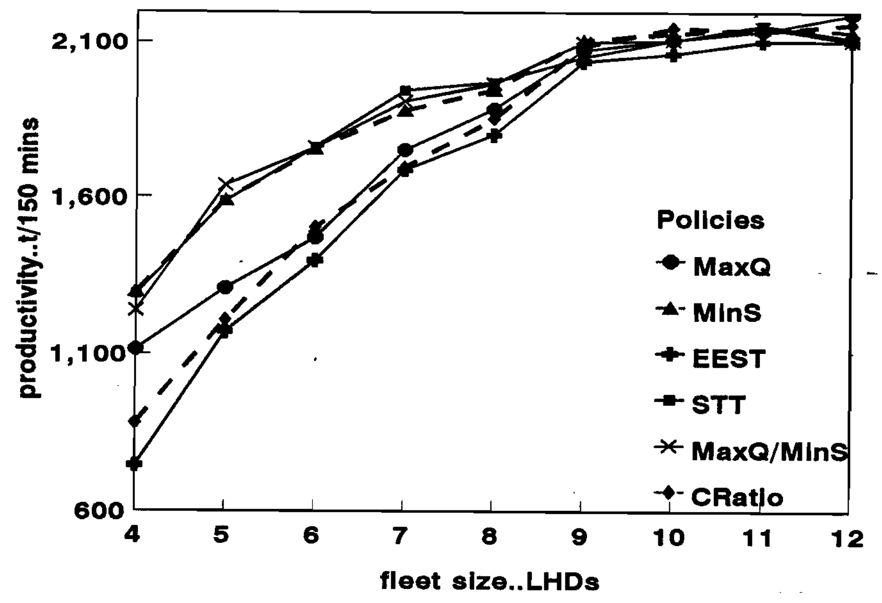


Figure 7.9 Average fleet productivity at increased fixed installations

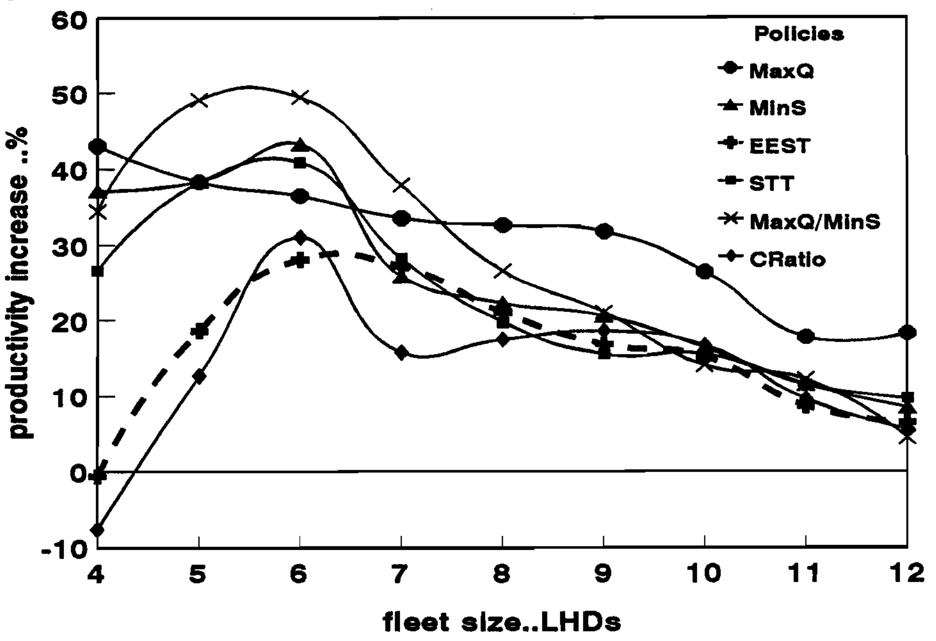


Figure 7.10 Changes in productivity over the base case with facility increase

The productivity increases show mono-modal distributions with change in machine numbers in the system. The results also indicate that lower productivity is realized at large fleet sizes. This is due to lower fleet utilization caused by the constrained environment of the mine network, i.e. on the restriction on the maximum number of machines occupying a road section or a load- or dump-point. More machines in the system results in congestion, and hence low productivity changes compared to leaner fleets.

Table 7.6 Productivity increase due to increase in fixed installations

Policy -> LHD #	MaxQ %	MinS %	EEST %	STT %	MaxQ/ MinS %	CRatio %
4	**43.1**	37.1	-0.6	26.7	34.5	-7.7
5	38.3	38.4	18.6	38.2	49.1	12.7
6	36.5	**43.2**	**28.1**	**40.8**	**49.4**	**31.0**
7	33.6	25.9	27.2	28.3	37.1	15.6
8	32.7	22.3	21.0	19.7	26.6	17.5
9	31.8	20.6	16.8	15.7	21.1	18.6
10	26.5	16.4	15.2	15.5	14.0	16.7
11	17.7	11.5	8.6	11.4	12.1	9.5
12	18.3	8.4	6.4	9.6	4.5	5.5

The EEST policy shows a negative productivity increase at a fleet size of four LHDs. The policy is non-discriminating on travel distances which results in the small fleet being assigned to any free destination identified. The probability of free destinations is high at small fleet sizes. The result is that the LHDs travel unnecessarily longer distances reducing their productivity per unit time. As the fleet size increases, less and less first free destinations are available which forces the rule to consider the waiting times of the machines already assigned to each destination. The reduction in potential destinations results in better controlled dispatching hence the observed increase in productivity with fleet size increase. However, very large fleets have the problem of congestion and this causes a decrease in productivity.

The CRatio policy also has a negative productivity increase for the small fleet of four LHDs because the search for the highest critical ratio causes excessive machine travel. Supposing an LHD has just loaded from one extreme of the mine thereby lowering the ratio of the work remaining in that section, then the next load may be from the other extreme of the mine network. This dictates that the machine traverse the mine thereby spending more time in travelling empty to the next highest critical ratio areas. A fleet increase results in a more even distribution of equipment in all working sections. This reduces the probability of an equipment traversing the entire length of the mine.

The MaxQ and MaxQ/MinS policies predominantly show the largest productivity increases with increase in fixed installation for most of the fleet sizes. This result is possibly due to the distances of the high valued material sources being closer to the dump-points. If high quality ore sources are both close and far from the dump-points, this results in a fair distribution of the equipment to both short and long haulage routes which effectively prevents the onset of traffic congestion.

7.6.2 Fleet Idleness and Utilization

An increase in fixed facilities increases the fleet productivity. Under a fixed production target, this means that a fleet working in a mine environment with more dump-point facilities can finish the scheduled production earlier. As no additional work is assigned to the equipment once the shift production is achieved, this results in increased fleet idle times at the end of the shift. This aspect was investigated for all the dispatch policies by considering the end of shift fleet idleness. The change in fleet idleness between the mine layouts with few and many dump-points was calculated as:

$$\Delta\, idleness = \frac{100(U_f - U_i)}{U_i}\% \qquad (7.3)$$

where U_i = fleet utilization with few dump-points and,

U_f = fleet utilization with many dump-points

The change in idleness is shown in Table 7.7 where all positive values indicate that the increased facilities decreases the fleet idleness, implying that the fleet utilization per shift has increased. In other words, by increasing the facilities, the fleet utilization may be reduced by the respective values indicated in Table 7.7 without affecting the production level compared to layouts with a few dump-points. Figure 7.11 indicates the fleet utilization for selected policies operating under the different number of fixed facilities. It should be noted that if the utilization is considered as a function of shift time, the utilization values for the mine system with more dump-points is higher than that with fewer dump-points.

Table 7.7 Changes in fleet utilization with increased fixed facilities

Policy-> LHD #	MaxQ %	MinS %	EEST %	STT %	MaxQ/MinS %	CRatio %
4	- 0.9	1.4	2.0	1.5	0.5	2.1
5	1.1	3.9	- 0.8	5.8	3.4	2.6
6	2.4	-0.6	4.6	7.1	1.5	0.8
7	-0.5	0.5	0.0	6.9	1.9	- 1.8
8	-2.5	2.2	-1.6	8.5	4.1	-2.1
9	0.7	2.9	10.9	8.9	9.9	6.4
10	- 5.2	-0.9	6.4	7.8	3.5	9.5
11	2.9	- 0.3	11.2	11.7	1.8	11.7
12	- 0.2	- 4.7	10.2	12.1	2.1	5.8

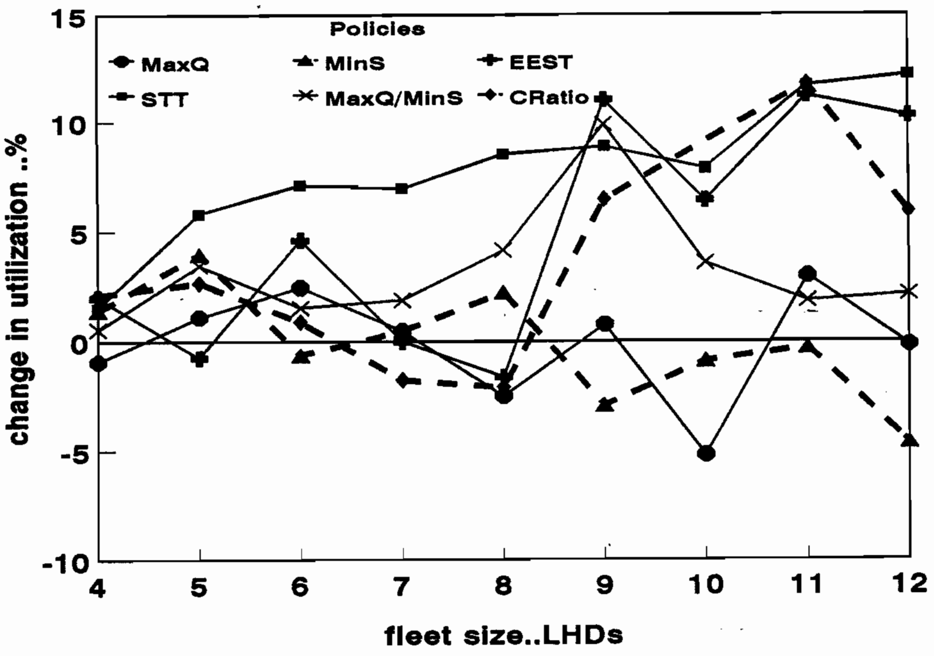


Figure 7.11 Fleet utilization changes for different dispatch policies with increased facilities

The STT and MaxQ/MinS policies indicate improved fleet utilization for all fleet sizes whilst the remainder of the policies have both positive and negative fleet improvements depending on the fleet size. It can be noted from Table 7.7 that statistically, an increase in the fixed facilities generally tends to increase a fleet utilization.

7.6.3 Production Cost

The effects of increasing the dump-points on the unit production cost (variable cost) was studied at two time windows of the shift production duration, namely at the middle and end of the shift. The analysis at the middle of shift was deemed necessary as it represents the steady (or near steady) state conditions of the production phase at which few or no scheduled material has been fully depleted. The changes in production cost are defined as:

$$\Delta\, cost/ton = \frac{100(C_f - C_i)}{C_i}\% \qquad (7.4)$$

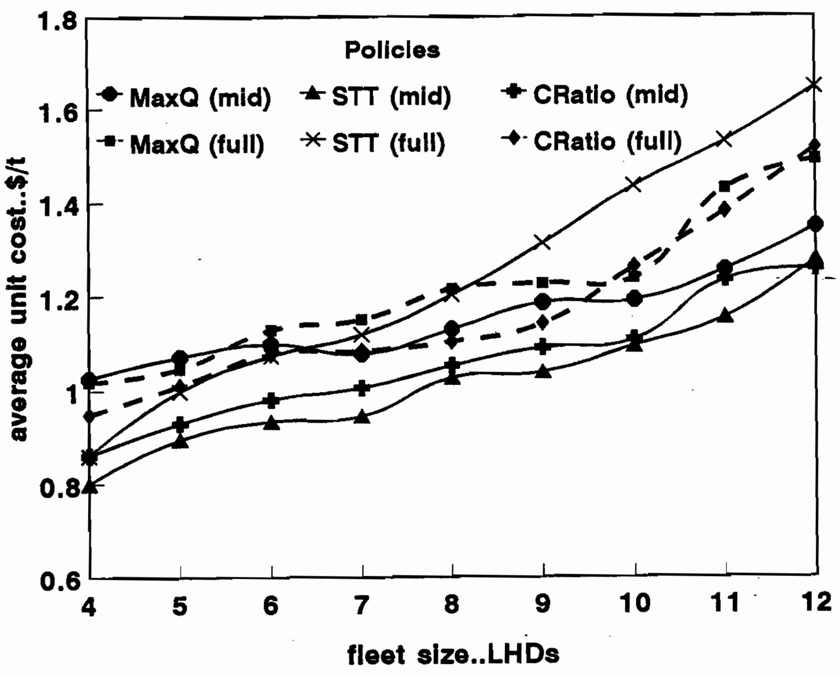
where C_i = average cost per tonne with a base system of dump-points and

C_f = average cost per tonne with a system with more dump-points.

The results are indicated in Tables 7.8 and 7.9. Figure 7.12 shows the comparison between the base and the modified mine system production cost at different fleet sizes for the various dispatch policies. An increase in the dump-points has a net effect of reducing the average unit production cost. The magnitude of the cost reductions depend both on the policy applied and the fleet size available. The results in Table 7.8 obtained with one more dump-point, show maximum cost reduction of 27 and 29 percent for MaxQ and MaxQ/MinS policies respectively, during the steady state conditions. The cost reduction is slightly less if it is considered at the end of the shift (24 and 22 percent respectively). The results obtained at the middle of the shift are a reflection of the productivity increases discussed above in section 7.5.1 (also determined at middle of shift). The unit production cost progressively decreases to a certain minimum value before it starts to rise. The minimum costs are identified with optimal fleet sizes for each dispatch policy.

Table 7.8 Changes in average production cost at middle of production shift

Policy-> LHD #	MaxQ %	MinS %	EEST %	STT %	MaxQ/ MinS %	CRatio %
4	**-27.6**	-22.7	1.4	-18.8	-22.2	2.8
5	-22.2	**-26.1**	-12.8	**-27.2**	**-29.6**	-11.3
6	-20.0	-22.4	-21.9	-23.3	-26.9	-16.0
7	-24.1	-19.3	**-22.6**	-20.7	-25.3	-16.5
8	-23.5	-17.4	-15.6	-19.6	-20.2	-17.6
9	-23.8	-17.4	-16.3	-13.7	-16.6	**-18.3**
10	-18.6	-13.5	-9.5	-11.9	-11.5	-14.1
11	-16.6	-10.6	-8.4	-8.8	-8.8	-15.5
12	-13.5	-8.7	-4.3	-9.7	-5.5	-7.7



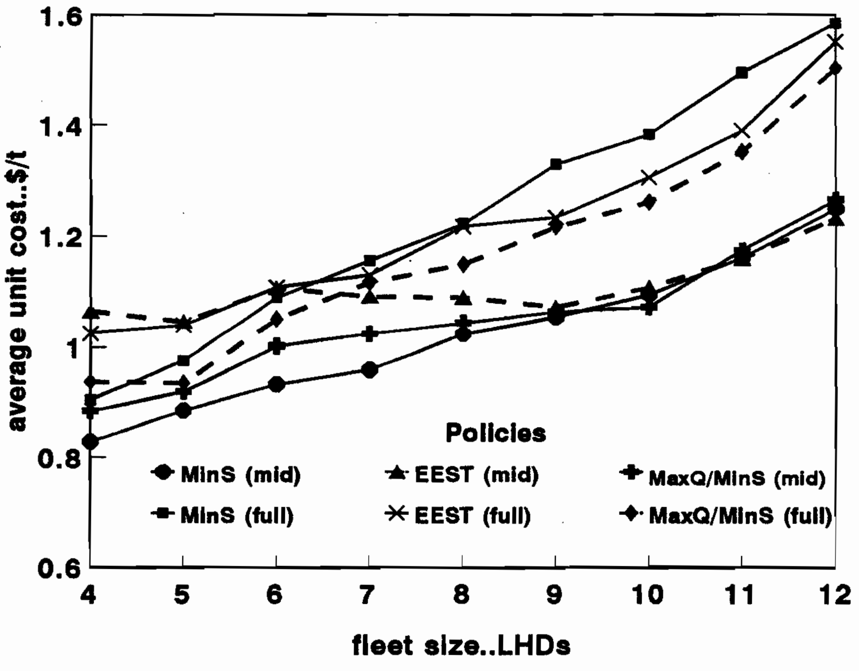


Figure 7.12 Cost reduction effect of increased facilities at different work shift stages and policies

Table 7.9 Changes in the average unit production cost at end of shift

Policy-> LHD #	MaxQ %	MinS %	EEST %	STT %	MaxQ/Min S %	CRatio %
4	**- 23.9**	**- 19.9**	5.9	- 13.4	- 21.9	- 10.0
5	- 14.8	- 14.9	- 12.7	- 17.1	- 12.1	- 15.1
6	-19.4	-17.1	-12.7	**-20.4**	**-26.8**	**-16.0**
7	- 19.2	- 16.3	**- 19.5**	- 15.7	- 18.1	- 14.4
8	-13.7	-16.8	-10.8	-13.7	-8.8	-4.9
9	- 12.7	- 19.8	- 5.6	- 9.9	- 3.2	- 7.2
10	-8.6	-15.8	-9.8	-14.8	-8.3	-10.4
11	- 10.1	- 17.9	- 5.6	- 18.2	- 10.8	- 8.3
12	1.9	- 14.3	- 4.1	- 16.3	- 0.8	- 12.2

The unit production cost curves for the end of the work shift indicate that the increased fixed facilities have a net effect of cost reduction in the order of 5 to 20 percent compared to the base case, depending on the policy and fleet size. The end-of-shift cost reductions are smaller than those obtained under the steady state conditions. The former cost curves show complex multi-modals with a weakly defined trend that indicates a decline in the cost effectiveness with an increase in the fleet size. The complex form of these cost curves is probably due to the stochasticity in the simulation model.

An increase in the number of fixed facilities causes changes in the fleet sizes that yield the maximum cost reduction. The fleet size that minimizes the unit costs varies with the policy and the time interval considered. Table 7.9 indicates the percentage cost reduction compared to the base case (i.e. one with fewer fixed facilities) at the end-of-shift time interval. The size of the most cost effective fleet at the end of shift conditions compared to during the shift, for the MaxQ, STT and MaxQ/MinS increases from five to six. On the other hand, the fleet sizes for MinS and CRatio decrease relative to that required during steady-state conditions of the shift. The EEST policy shows the same size fleet for both the during and end of shift conditions that produces the maximum unit cost savings.

The consequence of the above observation is that the shift equipment dispatcher can:

1. Systematically reduce the number of LHDs in the system as the scheduled material sources are depleted, thereby eliminating system congestion at the remaining scheduled material sites and thus keep the unit production cost down.
2. For a given fleet size operating under a specific policy, the dispatcher can maintain relatively higher productivity by working a **rolling** shift schedule whereby the depleted material sources are replaced by new ones through re-scheduling. This creates a steady state shift implementation and is suitable for application with the newly developed technologies of multiple vehicle tele-operation which allows 'hot' shift changes[1].

7.7 Product Quality Consideration

There are two quality requirements of a shift production, namely meeting the average target quality and how is it achieved, i.e. its distribution with time. The target quality is simply the mean grade of all the ore streams mined over a shift. The achievement of this quality alone does not mitigate the downstream process control problems of poor recoveries that may be present due to erratic ore grade fluctuations. Therefore, the distributions of grades about the target values of the ore streams throughout the shift length are important for feed-forward control of the imminent ore qualities. The distribution information is also required for feedback control of the actual mining process as it can be used to control the equipment dispatching.

1 Operators switch positions at the central control panels hence no time is lost travelling to the work area at the start of a new shift.

These requirements were investigated through a simulation of the L100 production for the schedule indicated in Table 7.5. The target grade is the weighted mean grade for the ore-pass on L100 calculated on the basis of full achievement of the production schedule, i.e. 2.60 % Cu-Ni. The actual grade for the ore-pass was determined at middle of the shift which represents near steady-state conditions and is free of high fleet congestion. The number of loads, N_i from each ore source, i dumped into the ore-pass were determined and the actual grade calculated as:

$$g_{act_t} = \frac{\sum_{i=1}^{4} g_i N_i}{\sum_{i=1}^{4} N_i} \tag{7.5}$$

where g_i = scheduled grade of stope i, and

g_{act} = actual weighted ore-pass grade at time t.

The simulation results were statistical analyzed to give the weighted mean grade and the standard deviations for the various dispatch policies indicated in Table 7.10.

Table 7.10 Ore-bin average quality for different dispatch policies and fleet sizes

Fleet size -->	5 LHDs		7 LHDs	
Policy	Mean Grade % Cu-Ni	Standard Dev % Cu-Ni	Mean Grade % Cu-Ni	Standard Dev % Cu-Ni
CRatio	2.63	0.35	2.60	0.23
MaxQ	2.54	0.40	2.57	0.36
MaxQ/MinS	2.65	0.39	2.65	0.31
MinS	2.72	0.32	2.72	0.28
STT	2.74	0.32	2.73	0.26
EEST	2.50	0.33	2.52	0.31

The CRatio policy achieves the set target of 2.60 % and has the lowest standard deviation for both fleets of five and seven LHDs (Note that the standard deviations in Table 7.10 are based on the sample population used to calculate the indicated mean grades, i.e. they measure the level of confidence in the observed mean values). The next satisfying policy is MaxQ with mean grades of 2.57 and 2.54 % and *deviations* from target grade of -2.42 and -1.18% for five and seven LHDs respectively. The MaxQ/MinS policy shows a mean grade that is higher than the MaxQ. The STT and MinS yield the highest weighted mean grades above the target. The EEST policy under-achieved the quality requirement at both five and seven LHD fleet sizes. These results reflect the same conclusions drawn in the model validation with respect to product quality achievement by the various policies.

The observed results are explained as follows:

- STT, MinS and EEST policies are all quality neutral. They are governed by activity time. In the above schedule example, the shortest activity times were those of travel to a free server and these nearest servers happened to contain high grade ore, hence the higher than average grade obtained by STT and MinS at middle of the shift analysis. Equally, if the nearest ore sources to dump-points were low grade, then the expected mean grade from these policies would be lower than average. This was proven in the model validation in Section 7.2.3.
- EEST policy though quality neutral is based on the earliest expected time a machine receives a service hence is 'blind' to travel time to a destination. Under small fleet sizes, most servers are idle which makes them equally suitable for assignment for an LHD. This has the effect that a destination is decided as if by throwing an N-sided dice where N is the number of available servers free to receive a machine. As the fleet size increases, the policy improves (indirectly) in its quality achievement because there are fewer free servers to provide service. Under these conditions, EEST assigns a destination based on the best earliest service destination determined by analyzing the current server/customer relationship. These conditions on average, approximate equal visitation to all

servers which results in the improved quality profile and smaller grade variability.

- CRatio, MaxQ and MaxQ/MinS are all considerate to quality objectives and they do show weighted mean grades closer to the target grades for the dump-points.

An important result indicated in Table 7.6 is that the ore-bin quality is universally better under a larger fleet, both in mean grade and smaller variability. This is caused by increased efficiency in the dispatch policies that arise from the constraining conditions. That is, the policies under large traffic volume, have a more comprehensive search, through the admission and dispatch control rules, which results in the modification of simple shortest travel, earliest service times, or maximum contained metal destinations. The implication of this result is that the optimal fleet size which maximizes productivity and minimizes unit production cost also leads to a better product quality.

Each time a load is dumped into an ore-pass, its tonnage, grade and source is recorded to a statistical output file of the UADM program. This creates sequential tallies of dumped material for each ore-pass. Quality ore-bin profiles showing the grade variations about the target value can be made based on these tallies. Such profiles were obtained by listing 25 ore-bin sequential tallies (i.e. 25 shifts based on the same schedule) and taking the average grade at each nth load dumped as:

$$g^{n}_{ave} = \frac{\sum_{i=1}^{m} g_i t_i}{\sum_{i=1}^{m} t_i} \qquad (7.6)$$

where t_i = load tonnage

g_i = grade of the load, i

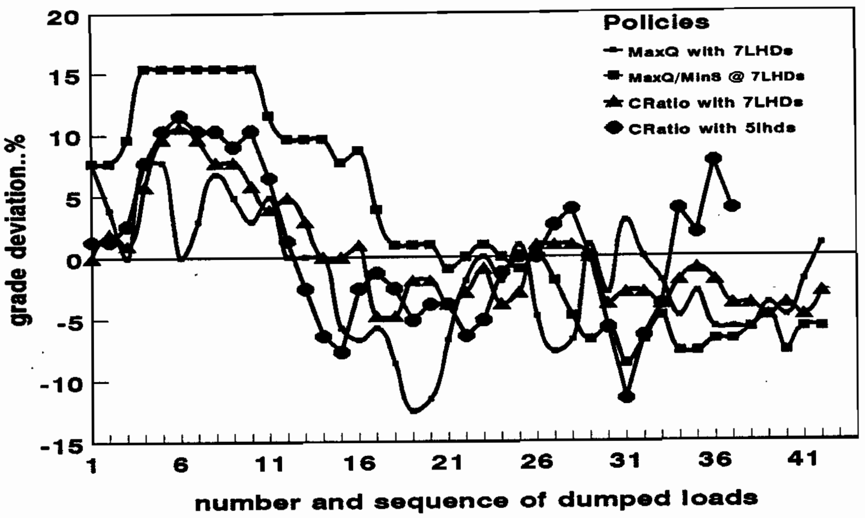
m = number of simulations (= 25)

A moving average of the dumped ore in the ore-pass is then calculated using the grades obtained in equation 7.6. The grade deviation profiles from the target are shown in

Figure 7.13. The STT and MinS show large positive deviations and clearly reflect the lack of quality optimization. The MaxQ/MinS policy is heavily influenced by the minimum slack and haulage distance components which favour the mining of areas closer to the dump-points. Since these areas in this simulated case are also high grade, this enhances the effect of sustained mining of higher grade material more than for the STT and MinS policies. Once the high grade and/or near ore sources are depleted, the policy selects sources with the largest amount of scheduled muck remaining because these sources have higher contained metal (the MaxQ component) though not necessarily high grade. The large tonnage effect is responsible for the earlier departure of the policy from STT and MinS trends at 20 cumulative dumped loads in response of the blend requirement (see Figure 7.13).

In the simulated case, the MaxQ policy defines a sine-wave like profile in grade deviations with an ever decreasing amplitude. This is due to the high grading effect of the policy, i.e. initially mucks the high grade material at the expense of low grade. In addition, the high grade stopes were nearer the ore-passes than lower grade ones. This condition has been concluded in the model validation to cause a wide grade variability. As process continues, the contained metal value in the high grade stopes decrease (i.e. less tonnage) whilst the low grade-large tonnage sources become significant in their contained metal value. Note that the success of this policy at blending requires large scheduled tonnages from the low grade sources and conversely, small scheduled tonnage from the high grade sources. The Max policy has a large negative deviation at 20 dumped loads before the grade improves. This profile is due to the priority set on mining of the stopes, one high grade stope is at a lower priority of two and becomes active only after some 20 loads have been dumped. When this stope becomes active, it accounts for the improvement of the moving average grade deviation (see Figure 7.13).

The CRatio policy has consistently smaller deviations compared to the other policies at all levels of work done. This policy, though not explicitly defined in terms of quality, generates the best quality profile because it systematically draws materials (ore, both high



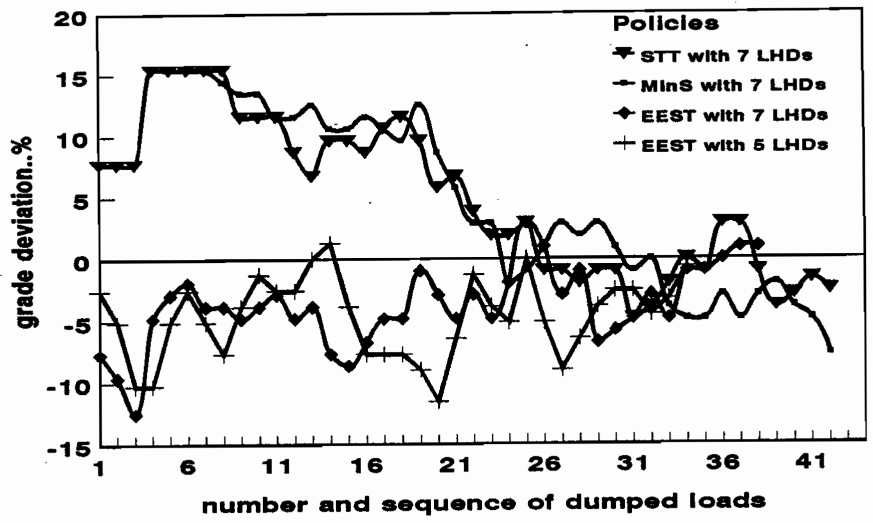


Figure 7.13 Ore-bin quality profiles for different dispatch policies

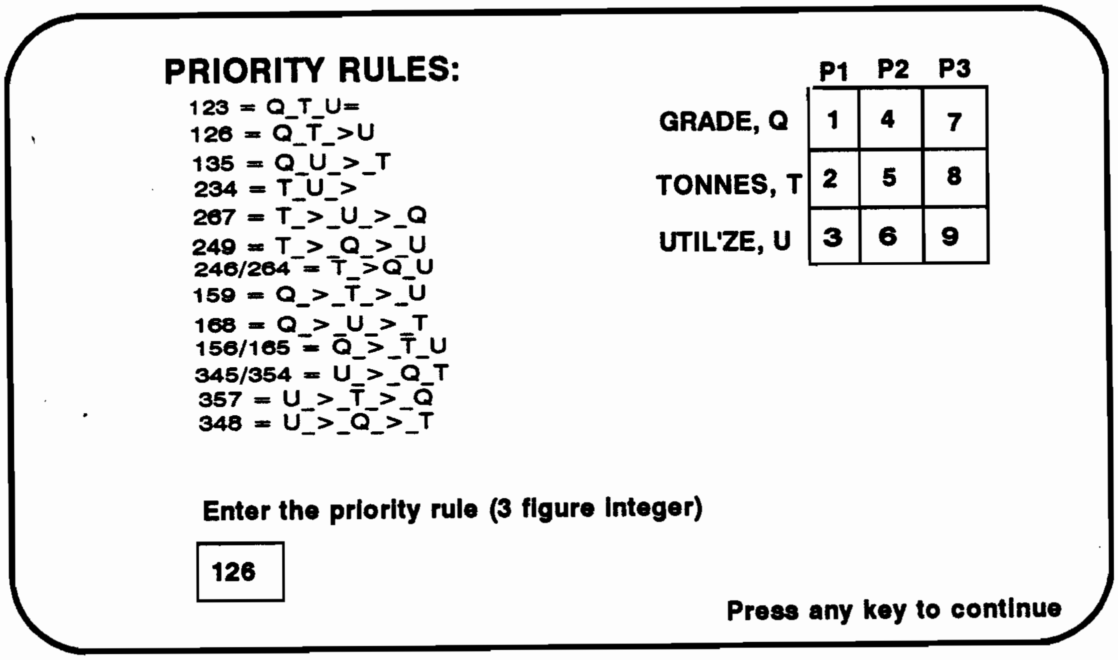
and low grade, and waste) based on the amount of work remaining at a scheduled source. No discrimination of location, distance, etc., is used except that defined in the initial priorities of the schedule. Therefore, the CRatio policy follows the shift schedule closest whether equipment breakdowns occur or not. However, source and dump-point blockages affect this policy in the same manner it affects the other policies.

In the simulated case, the EEST policy showed negative deviations as indicated in Figure 7.13; this is caused by the N-faced dice polling criterion. It is service-time dependent, therefore has no quality consideration which means, like the STT and the MinS policies, it cannot be used when the production quality is a major goal.

7.8 Process Control and Implementation

A shift production process can be implemented with the UADM program. The program requires at least three input files and an optional fourth one. These files are detailed in Appendix E.

At implementation of the program, a graphical screen displays a 3 x 3 matrix which defines the three goals: productivity, (P) quality (or grade, Q) and fleet utilization, (U) and three levels of priorities for these goals ranked as P1 > P2 > P3. The shift scheduler selects a priority combination for the shift based on the desired results. This input panel as it appears on the display screen is illustrated in Figure 7.14.



```
PRIORITY RULES:
 123 = Q_T_U=
 126 = Q_T_>U
 135 = Q_U_>_T
 234 = T_U_>
 267 = T_>_U_>_Q
 249 = T_>_Q_>_U
 246/264 = T_>Q_U
 159 = Q_>_T_>_U
 168 = Q_>_U_>_T
 156/165 = Q_>_T_U
 345/354 = U_>_Q_T
 357 = U_>_T_>_Q
 348 = U_>_Q_>_T
```

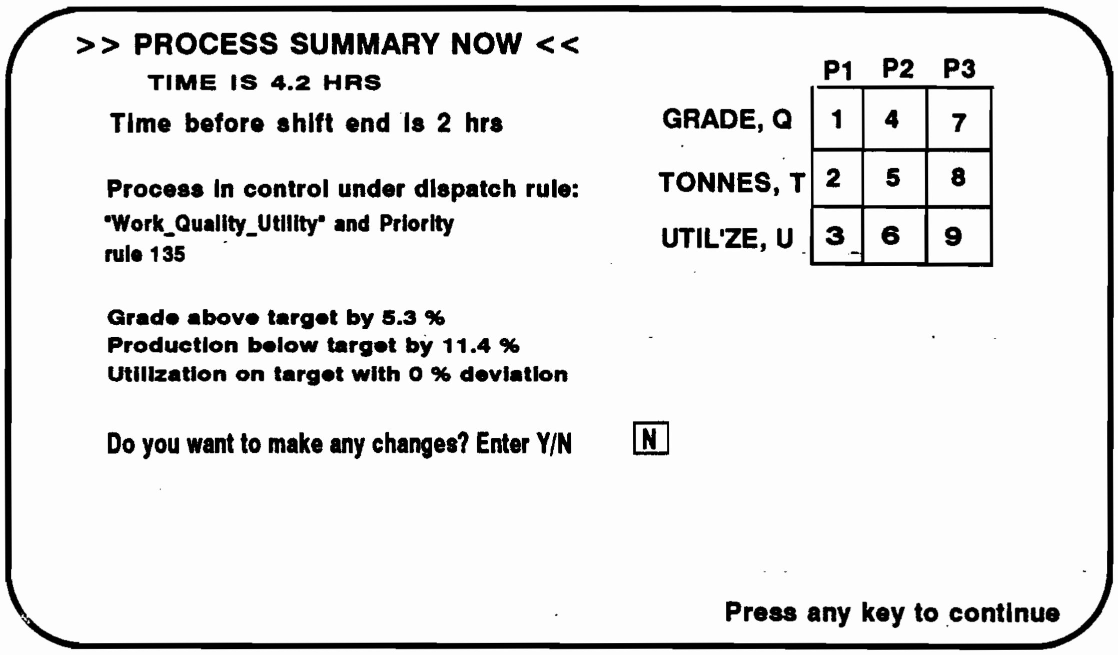
	P1	P2	P3
GRADE, Q	1	4	7
TONNES, T	2	5	8
UTIL'ZE, U	3	6	9

```
Enter the priority rule (3 figure integer)

126

Press any key to continue
```

Figure 7.14 UADM screen display of shift control priority matrix

The program then implements the schedule subject to the operative policy, priorities and stochastic events that may occur during the shift, such as machine breakdowns and blockage of fixed facilities. The program records all events to an output 'event' file. Typical events included are times of blockages, breakdowns and times when they are corrected. Equipment movements are recorded, i.e. the time an LHD is assigned a destination, the name of the destination and when it arrives at destination. If the LHD on arrival finds the destination occupied, it queues and additional information about the waiting time in the queue is also recorded. This shift information may be used in updating the equipment properties such as their efficiency and to update the Weibull distribution parameters for the various activities of loading, dumping and travelling.

The program displays the shift status at set review time windows, on the display screen and prompts the scheduler for a response on what action to take under the given status. This stage marks the interative feed-back control process of the program. Typical display panels for a system in control and out of control are illustrated in Figure 7.15. Besides these text displays, the program outputs the progressive production with time as well as the quality (grade) deviations of the moving average grade about the shift target as indicated in Figures 7.16 and 7.17 respectively. After observing these graphical trends, the scheduler can select from the menu the option that goes back to the "PROCESS SUMMARY NOW" and make the necessary feedback changes.


>> PROCESS SUMMARY NOW <<

TIME IS 4.2 HRS

Time before shift end is 2 hrs

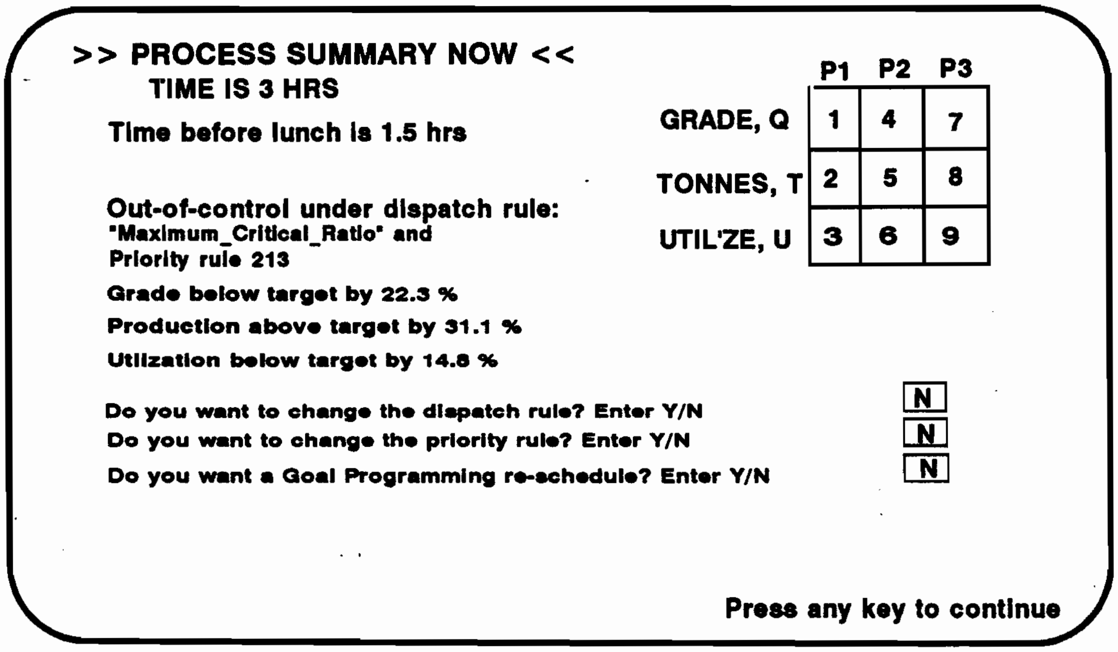
Process in control under dispatch rule:
"Work_Quality_Utility" and Priority
rule 135

Grade above target by 5.3 %
Production below target by 11.4 %
Utilization on target with 0 % deviation

Do you want to make any changes? Enter Y/N [N]

	P1	P2	P3
GRADE, Q	1	4	7
TONNES, T	2	5	8
UTIL'ZE, U	3	6	9

Press any key to continue


>> PROCESS SUMMARY NOW <<

TIME IS 3 HRS

Time before lunch is 1.5 hrs

Out-of-control under dispatch rule:
"Maximum_Critical_Ratio" and
Priority rule 213

Grade below target by 22.3 %
Production above target by 31.1 %
Utilization below target by 14.8 %

Do you want to change the dispatch rule? Enter Y/N [N]
Do you want to change the priority rule? Enter Y/N [N]
Do you want a Goal Programming re-schedule? Enter Y/N [N]

	P1	P2	P3
GRADE, Q	1	4	7
TONNES, T	2	5	8
UTIL'ZE, U	3	6	9

Press any key to continue

Figure 7.15 Screen review windows of systems in control and out-of-control

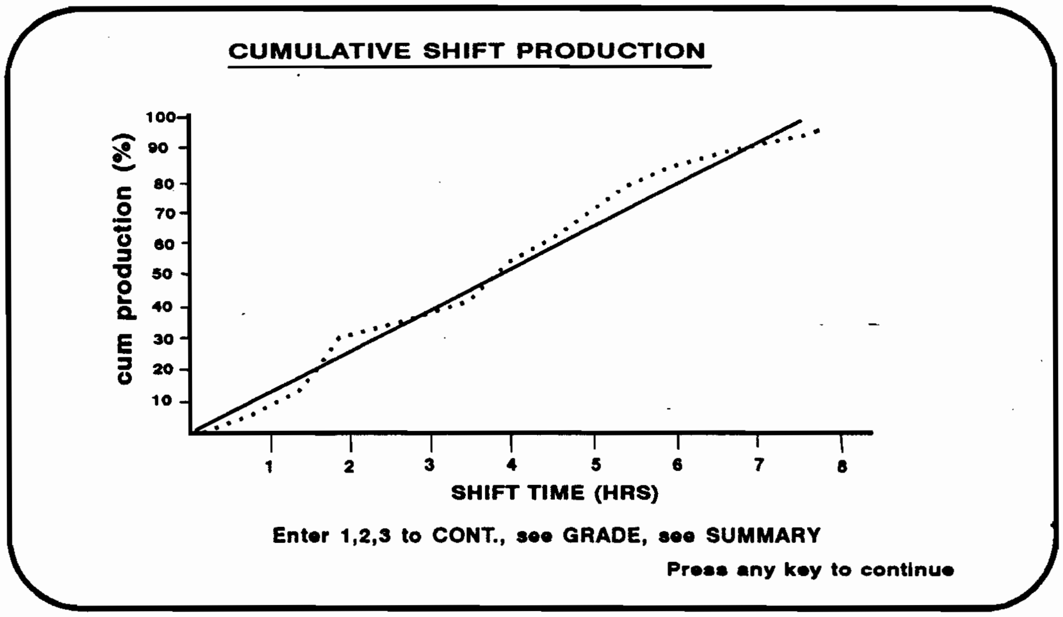


Figure 7.16 Review window graphical output of a shift production with time: solid line = target, dotted line = actual

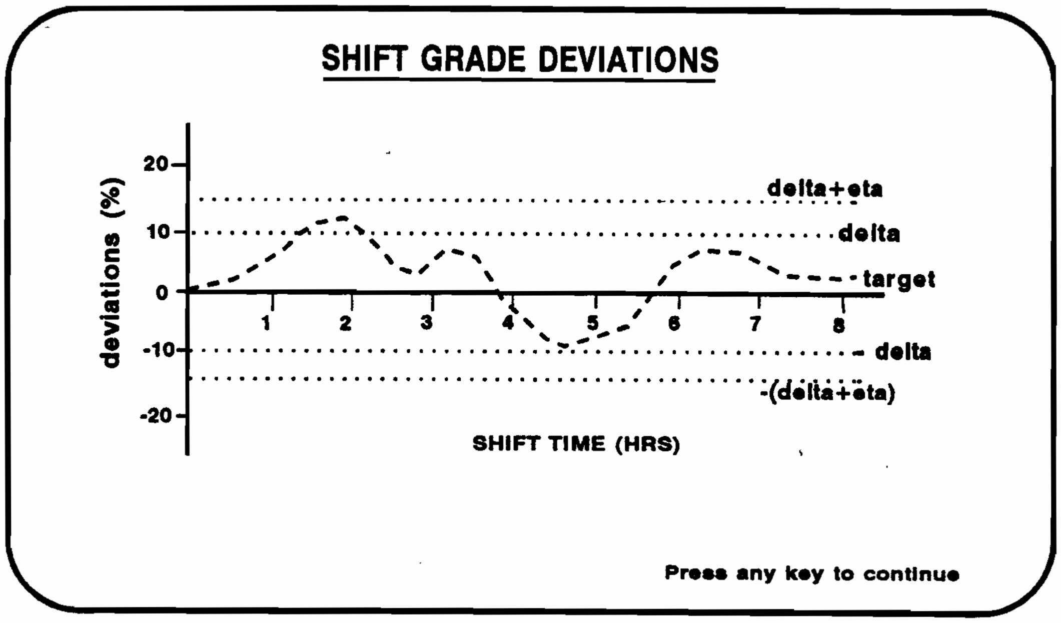


Figure 7.17 Review window graphical output of shift moving average grade deviations: delta and eta = management set limits, dashed line = deviation trend

7.9 UADM and Control: An Interactive Synopsis

The program UADM can be executed in a closed-loop with minor modification to the code. In the closed-loop mode the program performs all the data acquisition, data processing, comparison to shift targets and takes the appropriate control actions.

However, one of the objectives of this research was to develop a system that was interactive, which would allow the human operator to over-ride the computer program. This open-loop mode UADM was executed for an extended ten-hour shift such that the system stayed in the steady state condition for a longer time. During this stage, process deviations were addressed by changing either the production policies, the priorities or both in an attempt to put the production system back into control. The policy changes are based on decision trees shown in Figures 7.18 and 7.19. These trees are derived from the system simulation results discussed in sections 7.3 to 7.6 inclusive.

The cumulative production function for a typical production process under interactive feed-back control is shown in Figure 7.20 where the stepped sections indicate changes of policies. This figure indicates the ability for a shift supervisor to easily and effectively control an underground production process in real time. Equipment breakdowns are reported to the monitoring computer screen as they occur, by icons displaying the machine numbers. Idle machines are also posted to the graphical screen in a similar way to broken equipment. If the material sources and/or dumping points become unavailable, they are automatically flagged onto the screen.

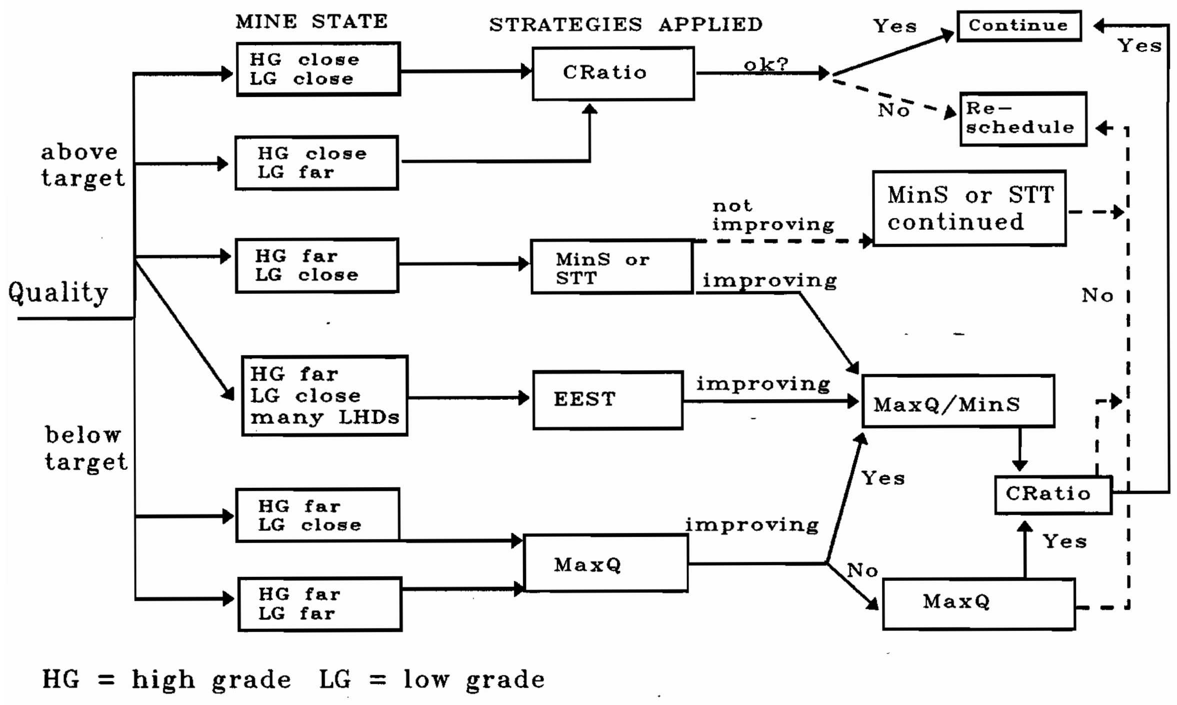


Figure 7.18 Decision tree for grade control at shift review intervals

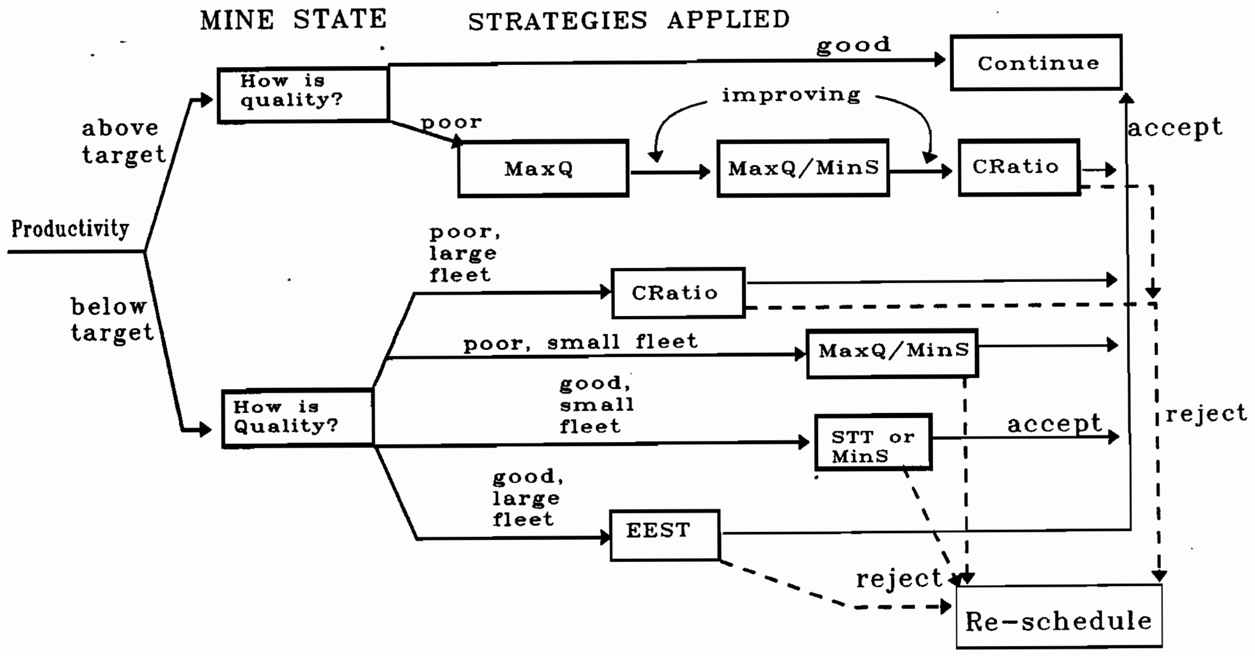


Figure 7.19 Decision tree for production at shift review intervals

The real world application of this program requires a communication system between the central controller running the control dispatch algorithms and micro-computers on-board the mobile machines. Two-way information flow on the communication system is required, i.e. a mobile machine sends a destination request to the controller. The controller computes the best destination and transmits back the optimal destination to the requesting machine. If a machine breaks down, the operator relays a message to the controller. The controller would then dispatch a maintenance crew to the broken machine, giving the maintenance crew the exact machine location and possibly the problems which will be encountered. This would facilitate a faster equipment repair and its earlier return to production. The use of the program in conjunction with equipment diagnostic and monitoring systems would open a new dimension of higher fleet utilization, availability and better product quality control.

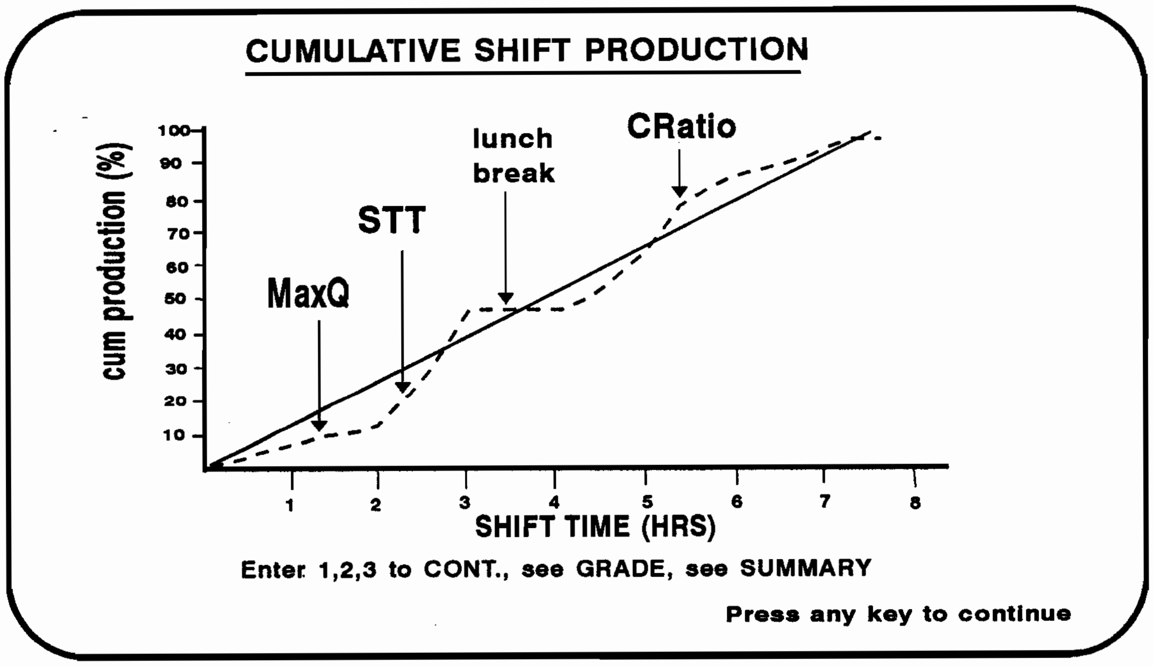


Figure 7.20 Screen display of a production profile under dynamic changes of policies: (N.B. policies indicated on graph for illustration only)

7.9 Concluding Remarks

The conclusions drawn in this chapter are based on simple mine layouts (i.e. for validation purpose) and the hypothetical Inza Mine. Currently, the UADM can be applied to any underground mine layout with a maximum of five active levels, fifteen active stopes, ten ore-passes and twenty LHDs. The program dimension arrays of these variables must be increased for larger mine systems. If there are no access ramps within a mine, the UADM restricts the equipment to its initial assigned level. In this captive mode, it is possible for one level to have idle machines whilst on another level there is a shortage. Such a situation occurs if there is a draw-point or ore-pass blockage and machine breakdowns, respectively. This causes a loss in production compared to a situation where ramps are present that allow inter-level dispatching. Note that this conclusion is based strictly on following an optimized schedule. In practice, its unlikely that equipment is idle because no work is available. The details of modelling different mine layouts are given in Appendix E.

7.10 Summary

In this chapter, the author has validated the UADM and has shown that the six new dispatch policies for underground mining are not equivalent when measured against the common shift goals of productivity, unit production cost, equipment utilization and product quality requirements. The policies have also been shown to perform differently under different conditions of fleet sizes, number of fixed facilities and the time period within the shift.

The following results were obtained:

- The highest productivity is attained during the steady state conditions of a production shift where the number of service points is still high. Under these conditions the best dispatch policies to apply with respect to productivity and fleet utilization are STT and MinS. The worst performing policies are the MaxQ and, if the fleet size is small, EEST. The productivity ranking of the six policies under light traffic is as follows:

{STT > MinS > MaxQ/MinS ≥ CRatio > EEST > MaxQ}

and under heavy traffic:

{CRatio > MaxQ/MinS ≥ MinS ≥ EEST > STT > MaxQ}

- The average unit production cost for a particular dispatch policy is dependent on both the fleet size and the timing during the shift. The large fleets have high unit costs because the incremental machine production is small when the fleet is operated under a constrained production target. Due to the constrained nature of the production, the unit costs determined during the early and middle part of the shift are less than those observed at the end of shift for all policies. The cost efficiency ranking of the policies at the end of the shift are as follows:

  {CRatio ≥ MaxQ ≥ MaxQ/MinS > EEST ≥ STT ≥ MinS}

- The analysis of both the fleet utilization and system total blockage times rank the dispatch policies in an ascending order of increased efficiency as follows:

  {MinS < STT < MaxQ ≤ EEST ≤ MaxQ/MinS ≤ CRatio}

- If the shift is considered in full, the highest productivity is achieved by CRatio and MaxQ/MinS policies because these policies are more effective at spreading out the equipment, minimizing (traffic) system blockage, waiting and idleness. The STT and MinS are the least performers, especially at large fleet sizes.

- Production strategies have been developed to deal with deviations from target which entail the mixing of the dispatch policies, selecting the most appropriate one for the current conditions. This ability to move between policies reduces the frequency of shift re-schedules that may need to be done, as the different policies can operate to return an out-of-control system back into control.

- The product quality is best represented by CRatio followed by MaxQ then MaxQ/MinS policies. The other three policies are neutral to quality which may lead to large positive or negative deviations. However, if the distances between the different ore sources and the dump-points are equal, the target quality profile for the shift may be achieved by the quality neutral policies.

- The two requirements of product quality, i.e. target grade and a stream with minimum deviations from the target, are both satisfied by CRatio and MaxQ policies. The ability to keep track not only of the mean grade of the mined material but its grade distribution provides invaluable information for both feed-back and feed-forward controls of the production process. The feed-forward control is through liaison with the customer (i.e. mill plant), about the ore and the duration of feed to expect as defined by the quality profiles in the different holding bins and ore-passes.

- The simulations have also indicated the benefits in cost reduction, fleet utilization and productivity due to additional fixed facilities, i.e. scheduling of several draw-points or many dump-points or both. These benefits decrease with the size of fleet used as the larger fleets tend to cause system congestion and, consequently reduction in utilization and productivity. The concurrent operation of many sources reduces the product quality distribution variance leading to an improved product. Equally, for a given mine layout, the size of the production fleet has a direct impact on the product quality under all the developed dispatch policies. A large fleet size minimizes both grade variability and the deviation from target.

- The UADM program has the potential use for medium to long term planning as it allows the simulating of different mine layouts. The program can be used for the assessment of additional fixed facilities and/or the inclusion of ramp systems between levels, or drifts linking different sections of the planned mine. A cost/benefit analysis could be carried out to compare the cost of installing these facilities and the benefits accrued, such as increased fleet utilization (which may imply a leaner fleet), unit production cost reduction, improved product quality and the greater flexibility of production operations.

These results establish the potential benefits of using a real time monitoring and control system for underground metal mines. Through simulation studies, the conditions of employing each of the developed dispatching policies have been established. It has also been shown that the UADM successfully implements a schedule generated by a goal program. The priorities and differential weights of the scheduled work sites can be preserved during the production shift.

Chapter 8
Conclusions and Recommendations

8.1 Conclusions

The study objectives were achieved as following:

- Underground mining decision-support tools were developed that enable both qualitative and quantitative information to be truly integrated in the production scheduling of the work shift. This was achieved by the fuzzy logic and the goal programming models that were developed. Cost and production functions were defined that allow the determination of the sub-optimality of the work shift schedules caused by the management priorities and differential weights of the shift objectives.

- Six new dispatching policies were developed or modified from existing manufacturing or surface dispatching policies. The production maximization and unit production cost minimization objectives were achieved through the use of the *maximum critical ratio of remaining scheduled material policy* (CRatio) and to a lesser extent by MaxQ/MinS and MaxQ in that order, on a full work shift basis. Therefore, the policies increase the efficiency of the work shift. This facilitates the competitiveness of underground mine operations by reducing their unit production costs.

- The objective of product quality control is achieved by the CRatio, MaxQ and MaxQ/MinS policies, in that order. These new dispatch policies ensure that both the target grade and minimum grade fluctuations are achieved for a given work shift. In addition, optimal fleet sizes help to achieve product quality through increased mine routing flexibility. This work identified a means of achieving schedules aiming to optimize product quality and represents a significant shift from the traditional schedules for maximum production.

- An interactive decision-support system was developed that enables real-time production and quality control monitoring, evaluation and dynamic control. The six new dispatch policies can be used interchangeably. This takes advantage of the strengths of each policy throughout a work shift.

- Production plan evaluation guidelines were developed that emphasizes the need for greater mine layout flexibility for increased efficiency to meet work shift objectives, especially the product quality goal.

8.2 Summary

The complexity of short-term production planning is due to the inter-play of many variables that influence the decision-making process. The time spans over which these variables are stable is relatively small which leads to a pseudo-dynamic system. These conditions require the ability to correctly infer the effects of each of the variables within the time limitation imposed by production planning and scheduling decision-making. This work has contributed towards solving these problems in a rational way through the development of guidelines of production plan evaluation. The guidelines allow a systematic evaluation of the different factors that are involved in production planning. The use of the guidelines in plan evaluation identifies both the limitations of a plan and the consequences. The guidelines include strategic planning decisions. This indicates the integrated nature of a mine decision system between the strategic and the production stage.

A fuzzy logic model for the assessment of the reliability in the mining grades at the production stage was developed based on both strategic and tactical planning attributes. The strategic planning attributes identified to impact the mining grades are:

- orebody structure and/or continuity which influences the stope designs. The stope shape and size directly determine the surface area to volume ratio of the stope ore. A design may or may not have a negative effect on the mined stope grades.

- sampling intensity upon which the stope design was made, i.e. whether the definition drilling spacing was adequate for a particular orebody structure to allow identification of micro-structures, and stringers of ore or waste within the stope. Poor stope outline definition undermines perimeter production blasting.
- predicted ground control issues defined from rock-mass characteristics of the host and/or wall rock. These issues impact on the mine sequencing and/or backfill decisions.

The production planning stage is characterized by dynamic factors affecting the mined grades. These factors are additional to the strategic attributes listed above. The identified factors are:

- in-situ stresses response to mining. This can be in the form of stress relief or concentration in certain areas of the stope perimeter causing sloughing of wall rock into the stope. The response is both stand-up time and mining phase dependent.
- the rock-mass strength values at the field scale may be very different to the predicted values based on laboratory work, that is used in the stope design. Such differences require re-evaluation of the stope stability leading to either an increase or a decrease in the stope stability. Either outcomes has a bearing on the dilution potential of the stope's back or side walls.
- progressive stope depletion in which the residual ore within the stopes increasingly tend to be more diluted due to the longer stand-up times; and in some mining methods, is due to a greater 'skin' contact with the wall rock during the perimeter ore's movement towards the draw-points.
- production drilling and blasting practice may result in either ore losses or over-break causing dilution.
- the location of a draw-point within a stope, i.e. centrally situated draw-points are less affected by wall rock or backfill dilution. The number of weak diluting faces impacts the perimeter draw-points.

The fuzzy logic method successfully integrates these vague information bases into a form that allows specific solutions of ordinal ranking of ore sources with regards to the reliability and confidence in the pulled ore quality. The fuzzy logic approach improves the quality of decision-making and supersedes the traditional use of single attribute analysis of variables. The improved prediction capabilities in decision-making allows a better scheduling of the work shift production areas. The product quality goal of a shift production is no longer based only on diluted grade estimates, but on an integrated solution of many fuzzy input variables. The diluted grade estimates are only one of such fuzzy input variables.

Two measures are used in the ranking procedure: *aggregation* and *fuzziness* numbers. The larger the *aggregation* number of the level of reliability in the product quality, the less susceptible the reported stope grades will be to further dilution during the dynamics of mucking. *Fuzziness* measures the uncertainty associated with the decision made from a set of input variables. The smaller the *fuzziness* number the less the uncertainty associated with the outcome.

Membership and probability functions of the output fuzzy systems give more insight into the variability of the aggregation values which is useful in deciding between two or more ore sources showing very close aggregation values. *Joint possibilities* between input variables are determined and contoured to illustrate the feasible regions upon which decision-making is made. The smaller the joint possibility area above a given management threshold value, the greater the uncertainty associated with the decisions made because they are selected from a smaller feasible set.

The developed fuzzy logic model utilizes the power of the computer in information storage, retrieval and computation of complex systems. This model creates a harmonized decision-making environment that achieves the following aspects:

- introduces consistency in decision-making, i.e. decisions made under the same linguistic input data are identical,

- updates and refinement of the input variable membership functions can be made and thus better represent the vague and dynamic variables as mining progresses,
- provides a comprehensive analysis of fuzzy multiple variate problems, and
- provides a cost effective tool for training of novice planning personnel.

A multiple objective goal programming technique has been successfully applied to underground mining short-term production planning. The method effectively models the real world environment of short-term decision-making because, unlike single objective optimization, it solves multiple goals and evaluates the trade-offs between the goals at set priorities and differential weights. A work shift re-scheduling is required if large deviations from plan occur. Therefore, a work shift monitoring of the critical parameters is necessary if the process is to be conducted in real-time.

In the process of decision-making, it is important not only to obtain the optimal solution, but also to know the extent of failure to reach that optimal solution. In this thesis, the effects of assigning different priorities and differential weights to different goals were investigated. The results show that the sources of material and the amounts drawn from each of the areas change to reflect the changes in ranking of the goals. This generates different deviations from targets of the respective goals, for each goal ranking combination. An inverse linear function has been formulated that determines the extent of simultaneous satisfaction of the multiple objectives at specified levels of production for a given mine system, i.e. for a given number of equipment, restrictions on the number of machines in each working district, material sources and dump-points. A series of these inverse linear relations can be determined for any mine, at different priorities and differential weights of goals. The inverse linear functions are a quick method of the goal programming objective function at a specified priority, where the smaller the goal programming objective function the more acceptable is the decision made. This concept is schematized in Figure 8.1.

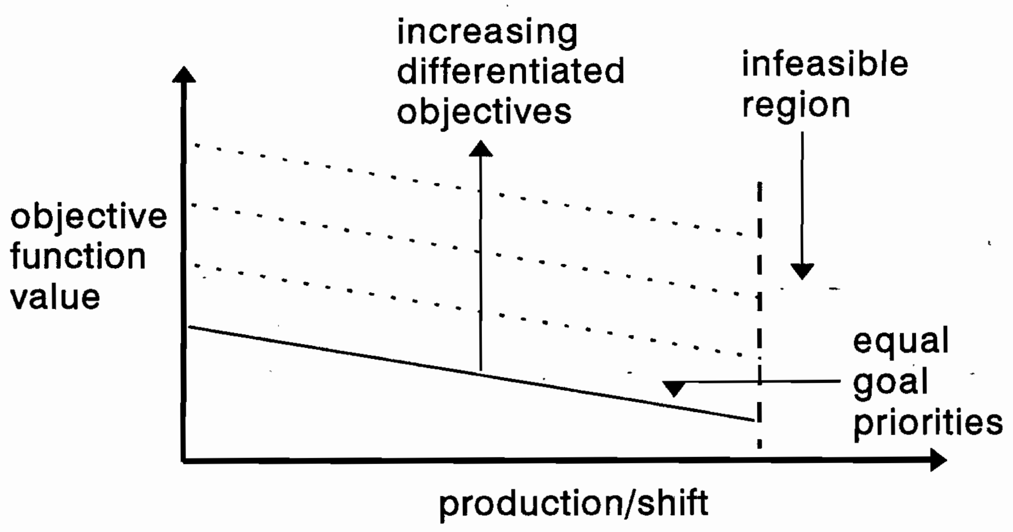


Figure 8.1 Assessment of the optimality of production schedule goals at given production levels

Real world production problems have cost minimization goals; however, there is no rigid cost constraint as the minimization is essentially an average one. Single objective cost minimization problems do not recognize this situation. In this work, we have identified the range of feasible budgets for a given mine system's production shift that minimizes the total goal programming objective function. The relationship between the objective function and the budget size at equal goal priorities is a flat bottomed U-shaped function where the left-side limb highlights under-budgeting and the right-side limb, resource under-utilization.

When the goal priorities are set at different levels, the relationship of the objective function to the budget size is a positive 'ramp' function. The lowest value of this 'ramp' function coincides with the critical budget below which the problem is infeasible. These functions indicate the sacrifice or compromise made in the decision-making by differentiating the priorities of the objectives. It is therefore invaluable for a given mine to determine these functions for all permutations of priorities and differential weights. The results are used to evaluate the global criteria of the shift objectives, as illustrated in Figure 8.2.

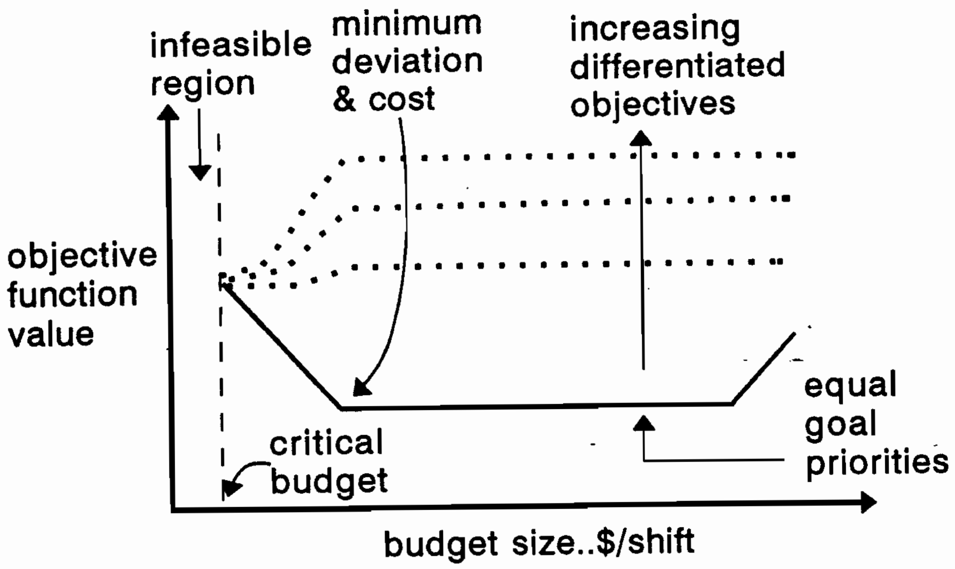


Figure 8.2 Assessment of the optimality of production schedule goals at given shift budgets

An underground active dispatch model (UADM) consisting of six new dispatch policies was developed and validated by an analytical solution of a production shift. The policies implement a production shift schedule in a real-time or a simulated system. The following conclusions on the shift bench-marks are made:

- - Productivity - Equipment dispatching between different work sections increases productivity through an increased utilization. The productivity ranking of the six new policies are traffic volume dependent, i.e. under light traffic the ranking is: {STT > MinS > MaxQ/MinS ≥ CRatio > EEST > MaxQ} and under heavy traffic: {CRatio > MaxQ/MinS ≥ MinS ≥ EEST > STT > MaxQ}.
- Production unit cost for each policy is dependent on both the fleet size and the timing within the shift. The dispatch policies effectiveness at minimizing the unit production cost per work shift is as follows: {CRatio ≥ MaxQ ≥ MaxQ/MinS > EEST > STT ≥ MinS}, where CRatio is the best performer and STT and MinS are the least.
- Product quality control is best produced by the CRatio, then MaxQ and MaxQ/MinS policies. The other three policies (STT, MinS and EEST) are not

product quality based. The later policies have inconsistent quality achievements dependent on the mine layout.

- Fleet utilization is achieved by the different policies as follows: {CRatio $\geq$ MaxQ/MinS $\geq$ EEST $\geq$ MaxQ > STT > MinS}, where CRatio has the best fleet utilization and MinS the least. This conclusion is based on a full work shift basis.

The different policies achieve maximum productivity at different LHD fleet sizes for a given mine layout. This allows the mine operator to always select the best policy under given dynamic conditions to maximize production and minimize unit production cost.

The UADM operates according to the specified job priorities in the schedule. These priorities are the same management priorities used in the goal programming model. Therefore, UADM attempts to execute a production schedule based on the optimized objectives.

A large number of ore sources and ore-passes reduces the ore quality variability within the handling facilities. Equally, a large material handling equipment fleet reduces the ore quality variability and the mean grades of the ore-passes approach their targets. The increased unit production cost due to a large fleet size has to be weighted against the improvement in quality.

The problem of poor resource utilization can be illustrated through UADM simulation. The implementation of UADM shows that the equipment resources are poorly utilized if the fleet size is kept constant throughout the work shift. The number of equipment working in a work section increases with depletion of other scheduled work areas. This causes congestion at the remaining work sections. This problem is solved as follows:

- create rolling schedules such that the depleted scheduled material is replaced at the process review intervals. This leads to pseudo-steady state conditions where the depletion rate is equal to the additional re-scheduled material. Such a condition has been shown to minimize the unit production cost for all of the new policies.

- a progressive fleet size reduction as more work areas are depleted, so as to maintain a constant ratio of the number of machines per section. This strategy equally reduces the unit production cost.

Besides these tactical considerations, the routing flexibility within the mine is critical. Routing flexibility requires access ramps and connectivity between mining districts. This requirement is at the strategic planning level.

UADM is an interactive model that allows changes to be made to the operating criteria should the shift production process go out of control. It is a first in the area of continuous monitoring of ore streams in underground mines in the following ways:

- The source of each load dumped at an ore-pass is recorded to a file at the time of dumping. These records represent stratigraphic sequences of the ore in each ore-pass. The ore-pass stratigraphic sequences can be used for subsequent scheduling of material in these holding facilities to achieve a required ore blend without a need for re-assaying of the ore-streams.
- Many holding facilities have large capacities and are unlikely to be drawn at the initial rates determined by mathematical optimization. Suppose a schedule required ore dumping in three ore-passes such that equal draws from the three ore-passes ensured a perfect blend. This blend is likely to be lost as future draws from these holding facilities is done independent of the initial schedule. The developed model creates a file where all dump-point profiles are recorded which can be used to re-enact the quality-optimized schedules.
- A real-time simulator for measuring grades at the draw-point was developed and it has been shown that such a device is an effective solution to product quality control. This approach was adopted because:

  (a) the number of grab samples collected from one draw-point is generally small and the assay turnaround too long to effect any quality control, e.g. 24 hours, and

(b) the stope design errors, blasting over-breaks, internal waste, etc., affect the quality of ore streams reaching the draw-points in a sometimes unpredictable way.

This study has highlighted the need for a real-time grade measuring device similar to the automatic weighing devices now fitted to some operating LHD fleets.

An increase in facilities (fixed or mobile) increase the mine layout flexibility to cope with both production and quality requirements at a minimum operating cost.

The concepts described in this thesis indicate the great potential of using the computational power of computers to develop a tandem system for short-term production planning and the subsequent implementation of the schedules using computerized dispatch systems. Process monitoring and control allows timely feed-back in the production process and feed-forward to the mill-plant on the product quality. This system should logically lead to the flexible, quality-conscious underground metal mine of the 21st Century.

8.3 Limitations

The fuzzy logic stope model is a bipartite model consisting of one level of input variables and the root representing the output variable. The input level has four fuzzy variables (or dimensions). There is room for improvement of this model through the expansion of each of the four input variables to levels two or three. For example, the rock-mass characterization of the host rock considers the number of diluting faces, stage of a mining sequence, backfill and in-situ stress responses all in one. The rock-mass characterization in this instance, is an output of four independent fuzzy variables. Therefore, an analysis performed at this level refines the global model output. Also, the current number of modifiers (two) used in this model should be increased as required. The model knowledge base is static and hence, it requires the re-coding of the program to incorporate new rules.

The goal programming model does not model integral variables. For the purpose of this study, this is not a limitation because fractional variables are made possible by the process flexibility that is assumed. The daily schedules produced by the program are a compromise solution to a multiple and perhaps conflicting objectives problem. If the resource values of goal constraints are optimized, for example, in a single objective optimization, then the goal program solution is both efficient and near-/or optimal.

The dispatch model is based on some assumptions that may not be true in many mine layouts, notably, equipment over-taking is allowed and by-passing is possible. It is more likely that by-passing or over-taking occurs only at certain points (in-sets) on the haul roads. This requires the coding of these specific sites on the mine road network, resulting in a program that is site specific. The UADM accounts for traffic volume in a specific road section by giving a time allowance for by-passing and over-taking on the road section. The UADM is a general model that can be used in any underground metal mine.

8.4 Discussion

This work has applied a fuzzy logic approach to determine the level of reliability in mining grades of different stopes and/or draw-points. This approach was adopted because of the fuzzy variables that influence the mining grades. Currently, there is research interest by financial industry to use fuzzy logic modelling in forecasting the stock market trends. Fuzzy logic is considered a most plausible approach to solving this complex problem. This is because fuzzy logic is powerful in integrating many fuzzy variables, in which multiple-variate analyses are inadequate.

The use of a goal programming technique in production scheduling enables multiple shift goals to be dealt with in a way representative of a real world problem. Besides the generation of the shift schedules, the technique allows the assessment of the opportunity cost of each schedule. The opportunity cost is measured as the sum of the absolute deviation variables in the goal program objective function. The decision-maker is therefore aware of the extent of sub-optimality or optimality of a shift schedule.

This work has described only one component of production control in the inter-related underground metal mines operations. Other functions that influence product quality include the stope blast designs, production drilling accuracy, blasting efficiency, and blasthole surveys. The monitoring of these functions will enable the assessment of their performances. Stope blast designs are now routinely done in CAD systems. The use of electronic survey field books and down-the-hole cameras allow the survey and drilling accuracies to be quickly checked against the CAD designs which facilitates the designs of the blast explosives powder factors and initiation. These production functions are upstream to the materials handling activity. Any one of them can lead to poor productivity and product quality through poor ore recovery within the stopes, waste over-blast and/or poor fragmentation which affects draw rates. The UADM attempts to optimize production and quality subject to these upstream influences through grade sensing, the use of appropriate loading rates distributions, equipment dispatching and process control.

The concept of just-in-time (JIT) systems closely resemble the UADM. JIT systems focus on *adding value* to products through increased quality and a reduction in scrap material, rehandling and reworking of products [Cheng and Podolsky, 1993]. At the Garson Mine, JIT mining is done to minimize the throughput time in a heavy ground [Whiteway, 1993]. The net effect is reduction in costs and increased productivity levels. UADM describes heuristic dispatch policies that allow quality to be built into the work shift production.

Quick response and/or small set-up times are essential features of JIT systems. The use of an active dispatching and process control models in this work enables timely response to disturbances in the production system. The dependence of JIT systems on their suppliers, requires forging of good relationships. Similarly, the mine materials handling system has to be backward linked to the drilling and blasting crews, and mine design and geology departments. Forward linkage is provided through liaison with the process plant on the expected ore stream qualities. This will make ore quality a universal concern of all functions in the mine organization.

The lack of large product inventories in JIT systems requires that they be flexible to volume changes and an ability to have a modest product mix. These requirements are met through the use of flexible equipment to cope with changes in job types and breakdowns. In this work, it has been emphasized that a new mine design rationale is necessary to provide a flexible underground mine system similar to JIT systems.

8.5 Recommendations for Future Work

The author recommends the following issues as important areas of further research and enhancement of the models developed in this thesis:

1. The benefits of a fully automated mine of the future can only be realized through the coalescence of ideas and efforts from the present stand-alone departments. This will require research in continuous monitoring of mine ventilation systems to be tied into the equipment dispatch systems, such that whenever a dispatch algorithm assigns a machine to a certain area, it automatically communicates this decision to the ventilation control algorithm. The later then adjusts the airflows in the current and the new assigned work areas of the requesting machine.

2. The modelling of the reliability of stope mining grades has been performed on a fuzzy logic model that incorporates only four input variables. The potential of including other variables into the model such a stope ore mineralogy, should be considered especially, in the development of a specific mine site dilution assessment model.

3. The dispatch model described in this thesis requires the development of an efficient traffic control system to ensure mine safety, collision avoidance and/or situations where machines meet in sections of the drifts where no insets are present to allow by-passing. The use of traffic lights alone to indicate whether a section of a mine drift or ramp is occupied is not adequate as it does not consider the different travel speeds between machine types nor does it differentiate between the loaded or empty machines. This aspect is site-specific, which is the reason why it was not addressed in this work.

4. A 'futuristic' device has been proposed for measuring the ore quality as it is mucked. The measured quality is then automatically transmitted to the central controller which performs the dispatching and product quality control. Such information would then be compared with the planned targets and the appropriate control taken quickly in the form of re-scheduling and re-allocation of the equipment. Therefore, it is strongly recommended that research into the feasibility and development of such a tool be done as it will eliminate the current time lag (e.g. 24 hours in some mines) between draw-point grab sampling and the reporting of chemical assays which obviously nullifies any real-time quality control.

5. The industry is challenged to re-consider the traditional mine design philosophy of designing mine sections without common access except by shaft or raise systems. The maximum benefits of mine automation are unlikely to be realized as these mine layouts are highly inflexible to both quality and production optimization.

BIBLIOGRAPHY

Alford, C.G. and Whittle, J., 1986; *Application of Lerchs-Grossman pit optimization to the design of open pit mines*, AusIMM/IE, Large Open Pit Mining Conference. 201-207, Oct.

An, P., Moon, W.M. and Rencz, A., 1991; *Application of fuzzy set theory to integrated mineral exploration,* Canadian J. of Exploration Geophysics, Vol. 27, No. 1, 1-10, Dec.

Aytung, H., Koehler, G.J. and Snowdon, J.L., 1994; *Genetic learning of dynamic scheduling within a simulation environment*, Computers Ops Res., Vol. 21, No. 8, 909-925.

Baiden, G.R., 1988; *Load-Haul-Dump vehicle monitoring system*, Computer Applications in the Mineral Industry, First Canadian Conference, Quebec. 453-458, Mar.

Baiden, G.R., 1992; *Automatic haulage truck - design, development and mine implementation at Inco Limited.* CIM Bul. Vol. 85, No. 964, 41-46, Oct.

Baiden, G.R., 1993; *Combining Teleoperation with Vehicle Guidance for Improving LHD Productivity at Inco Limited*, International Congress on Mine Design, Kingston Ontario, 527-532, Aug. 23-26.

Baiden, G. and Henderson, E., 1994; *LHD Teleoperation and Guidance; Proven productivity improvement tools*, 6th Canadian Symposium on Mining Automation, Montreal, 77-86.

Belford, J.E., 1981; *Sublevel stoping at Kidd Creek mines*, Design and Operation of Caving and Sublevel Stoping Mines, AIME, New York, Chapter 42, 577-584.

Bell, T.M. and Whateley, M.K.G., 1994; *Evaluation of grade estimation techniques*, Mineral Resource Evaluation II: Methods and Case Histories, Geological Society Special Publ. No. 79, Whateley, M.K.G. and Harvey, P.K. (eds), 67-86.

Beus, M.J. and Orr, T.J. 1992; *Applications of Real-Time Monitoring to an Underground Mining Environment*, 5th Canadian Symposium of Mining Automation, 68-76, Sept.

Billette, N. and Elbrond, J., 1986; *Correlation and blending in a mining productive system*, Proceedings, 18th APCOM. IMM. London,UK. 39-49.

Bonates, E. and Lizotte, Y., 1988; *A combined approach to solve truck dispatching problems, Computer Applications in the Mineral Industry*, First Canadian Conference, Quebec. 403-412, Mar.

Bonates, E.; 1992; *The Development of Assignment Procedures for Semi-Automated Truck/Shovel Systems*. PhD Thesis, McGill University.

Bonates, E.J.L., 1995; *Mathematical programming and trackless scheduling*, 3rd Int. Symposium on Mine Mechanization and Automation, Colorado, Session 22, 31-42.

Bozer, Y.A., Chao, M. and Srinivasan, M.M., 1994; *Expected waiting times in single-device trip-based materials handling systems*, European J. of Operations Research, Vol. 75, 200-216.

Brady, B.H.G. and Brown, E.T., 1985; *Rock Mechanics for Underground Mining*, George Allen & Unwin Publishers, London, UK.

Cai, X. and Goh, H., 1994; *Train scheduling*, Computers Ops Res., Vol. 21, No. 5, 499-510.

Campbell, J.F. and Langevin, A., 1993; *Operations management for urban snow removal and disposal*, GERAD Publication #G-93-34, Ecole Polytechnique & McGill University, Oct.

Carter, A. and Pratt, A.R., 1993; *Measurement and control of road traffic*, Measurement and Control, Vol. 25, 36-44, Mar.

Caupers, D., Guerreiro, L. and Rodrigues, P., 1993; *Automatic blending system at Neves Corvo mine*, Proceedings, 24th APCOM, Montreal, Canada, 27-34, Nov.

Ciarallo, F.W., Akella, R. and Morton, T.E., 1994; *A periodic review, production planning with uncertain capacity and uncertain demand - Optimality of extended myopic policies*, Management Sci., Vol. 40, No. 3, 320-332, Mar.

Chanda, E.K.C and Wilke,F.L., 1992; *An EDP-Model of open pit short term production scheduling optimisation for stratiform orebodies*, Proceedings, 23rd APCOM. University of Arizona, Arizona. 759-768.

Charnes, A . and Cooper, W.W., 1961; *Management models and industrial applications of linear programming*, Wiley Publ., New York.

Chatterjee, P.K. and Just, G.D., 1981; *Cost analysis for the design and operation of sublevel open stoping*, Mining Engineering, Vol.33, No.10, 1445-1449, Oct.

Cheng, T.C.E. and Podolsky, S., 1993; *Just-in-Time Manufacturing: An Introduction*, Publ. Chapman and Hall, London.

Chryssolouris, G., Dickie, K. and Lee, M., 1994; *An approach to real-time flexible* *scheduling*, Int. J. of Flexible Manufacturing Systems, Vol. 6, No. 3, 235-253.

Cromb, R., Kotak, D., Stoakes, M. and Whale. K., 1992; *A New Approach To Mine Planning and Scheduling*, 5th Canadian Symposium of Mining Automation, 6 pp, Sept.

Cross, B.K. and Williamson, G.A., 1969; *Digital Simulation of an Open-pit Haulage System*, 8th APCOM, A Decade of Digital Computing in the Mineral Industry, SME-AIME, New York, 385-400.

Dagdelen, K. and Johnson, T.B., 1986; *Optimum open pit mine production scheduling by Lagrangian Parameterization*, Proceedings, 19th APCOM. Pennsylvania State University, PA. 127-141.

David, M., 1988; *Dilution and geostatistics*, CIM Bul. Vol.81, No. 914, 29-35, Jun.

De Gast, A.A. and James, R.B., 1972; *Valuation of ore blocks by regression analysis - An aid to the control of mine production*, CIM Trans Vol. 75, 154-159.

De Melo, J.J. and Camara, A.S., 1994; *Models for the optimization of regional wastewater treatment systems*, European J. of Op Res., Vol. 73, No. 1, 1-16.

De Wolfe, V., 1981; Dr*aw control in principle and practice at Henderson mine*, Design and Operation of Caving and Sublevel Stoping Mines, AIME, New York, Chapter 56, 729-735.

Dowd, P., 1976; *Application of dynamic and stochastic programming to optimise cut off grades and production rates*, Trans Inst Min Metall. Section A, Vol. 85, A22-A31, Jan.

Dowd, P., 1986; *Application of geostatistics to mine planning in a complex Pb-Zn-Ag orebody,* Proceedings, 18 th APCOM, IMM, London, UK, 249-264.

Dowd, P. and Elvan, L., 1987; *Use of dynamic programming in sub-level open stoping*, Trans. Instn. Min. Metall. Sect. A, Vol. 96, A171-A177, Oct.

Dowd, P. and Onu, 1991; *Optimising open pit design and sequencing*, Proceedings, 23rd APCOM. University of Arizona, Arizona, 415-422.

Dubois, D. and Prade, H., 1980; *Fuzzy sets and systems theory and applications,* Mathematics in Sci and Eng, Vol. 144, Publ. Academic Press, New York, 393 pp.

Editorial, 1993; *Fuzzy models - What are they, and why?* IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, 1-5, Feb.

Edlund, R., 1971; *On-line computerized traffic control*. C.I.M. Special Volume No. 12.

Decision-Making in the Mineral Industry, 352-360.

Elbrond, J., 1981; *Modelling the grade variation of ore from mine to user*, Bulletin for Sampling Research, Tokyo, No. 221, 9 pp, Jul.

Elbrond, J., Lizotte, Y. and Plasse, C., 1982; *Use of an interactive dynamic program system as an aid to mine valuation*, Proceedings, 17th APCOM. Colorado School of Mines, Golden, CO. 463-474.

Elbrond, J., 1994; *Economic effects of ore losses and rock dilution*, CIM Bul. Vol. 87, No. 978, 131-134, Mar.

Emshoff J.R. and Sisson, R.L., 1971; *Design and use of computer simulation models*, MacMillan Company, London.

Farrell, T.R., 1988; *Computerized truck dispatching at Quintette Coal Limited*, Computer Applications in the Mineral Industry, First Canadian Conference, Quebec, 375-381.

Filev, D.P. and Yager, R.R., 1993; *Three models of fuzzy logic controllers*, Cybernetics and Systems, Vol. 24, 91-114.

Forsman, B., Vagenas, N. and Lindstrom, C., 1992; *Using METAFORA to Evaluate the Transport System in a Swedish Open Pit Mine*, 23rd APCOM, 734-738.

Fytas, K. and Calder, P.N., 1986; *A computerized model of open pit short and long range production scheduling*, Proceedings, 19th APCOM. Pennsylvania State University, PA. 109-118.

Gen, M., Tsujimura, Y. and Ida, K., 1992; *Method for solving multi objective aggregate production planning problem with fuzzy parameters*, Computers and Industrial Engineering Vol. 23, Nos 1-4, 117-120.

Gere, W.S. Jr, 1966; *Heuristics in job shop scheduling*, Management Sci., Vol. 13, 167-190.

Gershon, M.E., 1987; *An open pit production scheduler: algorithm and implementation*, Mining Engineering, Vol.39, No.8, 793-796, Aug.

Gerwin, D., 1993; *Manufacturing flexibility: A strategic perspective*, Management Sci. Vol. 39, No. 4, 395-409, Apr.

Gignac, L.P, 1979; *Hybrid simulations of underground trackless equipment*, D.Eng Thesis, University of Missouri-Rolla.

Gillenwater, E.L., Bernardo, J.J., Lineberry, G.T. and Yama, B.R., 1995; *A hierarchial production planning approach to work unit scheduling in the coal industry*, AIME Annual Meeting, Denver, Colorado, 1-15, Mar. 6-9.

Graves, S.C., 1981; *A review of production scheduling*, Operations Research, Vol. 29, No. 4, 646-667, Jul-Aug.

Grebenc, F.J. and Welwood, R.J.R., 1971; *Brunswick Mining Operations*, CIM Bul. Vol. 64, 37-44, Sept.

Hagenbush, L.G., 1986; *An Integrated Truck Management Information System (Truck MIS) Concept*, CIM Bulletin Vol. 79, No. 892, 62-69, Aug.

Harvey, C.M., 1993; *Multiattribute risk linearity*, Management Science, Vol. 39, No. 3, 389-393, Mar.

Hauck, R.F., 1979; *Computer-Controlled Truck Dispatching in Open-Pit Mines*, Computer Methods for the 80's In the Mineral Industry, SME-AIME, New York, 735-742.

Hill, R.D., 1987, *A micro-computer based underground mine haulage simulation program*, Masters Thesis, McGill University.

Hind D., 1994; *Applications of radio frequency identification systems in the mining industry*, 6th Canadian Symposium on Mining Automation, 34-42, Oct. 16-19.

Hintz, G.W. and Zimmermann, H.J., 1989; *A method to control flexible manufacturing systems*, Europ. J. Op. Res. Vol. 41, 321-334.

Hitt, D., 1985; *Management in Action*, Publ. Battelle Press, Columbus Ohio.

Horn, J. and Nafpliotis, N., 1993; *Multiobjective optimization using the Niched Pareto Genetic algorithm*, University of Illinois at Urbana-Champaign, IlliGAL Report No. 93005, 32 pp, Jul.

Houlding, S.W., 1992; *Real time grade control in mine-planning & production*, Proceedings, 23rd APCOM. University of Arizona, Arizona, 747-755.

INCO Limited Internal Report, 1994; *1993 LHD Cost and Performance Report*, May 2.

Ishii, N. and Talawage, J.J., 1994; *A mixed dispatching rule approach in FMS scheduling*, Int. J. of Flexible Manufacturing Systems, Vol. 6, No. 1, 69-87.

Jain, S., Barber, K. and Osterfeld, D., 1991; *Expert simulation for on-line scheduling*, Handbook of Expert Systems in Manufacturing, ed. Maus, R. and Keyes, J., McGraw-Hill, Inc., Chapter 15, 209-223.

Jardine, A.K.S., 1973; *Maintenance, replacement and reliability*, Pitman Publ. UK.

Jawed, M., 1993; *Optimal production planning in underground coal mines through goal programming - a case study from an Indian mine*, Proceedings, 24th APCOM. Montreal, Canada. 43-50.

Johnson, T.B., 1969; *Optimum open pit mine production scheduling*, Proceedings, 8th APCOM. SME. Littleton, CO. 539-561.

Johnson, A. and Smartt, H.B., 1995; *Advantages of an alternative form of fuzzy logic*, IEEE Trans on Fuzzy Systems, Vol. 3, No. 2, 149-157, May.

Kannan, V.R. and Ghoshi, S., 1993; *An evaluation of the interaction between dispatching rules and truncation procedures in a job-shop scheduling*, Int. J. Prod. Res., Vol. 31, No. 7, 1637-1654.

Kim, C.W. and Tanchoco, J.M.A., 1993; *Operational control of a bi-directional automated guided vehicle system*, Prod. Res, Vol. 31, No. 9, 2123-2138.

Kim, Y.C., 1979; *Production scheduling*, In Mine Production Planning and Control. Computer Methods for the 80's in the Mineral Industry, SME-AIME, New York. 610-14..

Khorramshahgol, R. and Okoruwa, A.A., 1994; A *goal programming approach to investment decisions: A case study of fund allocation among different shopping malls*, European J. of Operations Research, Vol. 73, 17-22.

Knights, P.F., 1993; *Analysis of ST8B scoop downtime at McCreedy West Mine*, Inco Sudbury Division, Internal Report, 12 pp.

Knights, P., Kairouz, J., Daneshmend, L. and Pathak, J., 1994; *Applications of radio frequency identification systems in underground mining*, 6th Canadian Symposium on Mining Automation, 28-33, Oct .16-19.

Knights, P.F. and Scoble, M.J., 1995; *Integrated mining information and control systems; towards the digital mine*, SME Annual Meeting, Denver, Co., 8 pp, Mar 6-9.

Knights, P.F., Henderson, E., Pathak, J. and Daneshmend, L.K., 1995; *Automated ore tonnage tracking using RFID systems*, 3rd Int. Symposium on Mine Mechanization and Automation, Colorado, Session 5, 33-48.

Kusiak, A., 1990; *Production scheduling: Classical and knowledge-based approach*, The Automated Factory Handbook, Technology and Management, ed. Cleland, D.I. and Bidanda, B., 652-679.

Lane, K.F., 1988; *The economic definition of ore*, Publ. Mining Journal Books, London, U.K. 149 pp.

Lee, S.M. and Jung, H., 1988; *A multi-objective production planning model in a flexible manufacturing environment*, J. of Prod. Research, Vol. 1981-1991.

Li, Z. and Topuz, E., 1987; *Optimizing design capacity and field dimensions of underground coal mines*, Proceedings, 20th APCOM, Vol. 1. Mining. Johannesburg, SAIMM. 115-122.

Lindstrom, B.E.A. 1971; *Production scheduling by computer*, C.I.M. Special Volume No. 12. Decision-Making in the Mineral Industry, 368-378.

Lizotte,Y., Scoble,M, and Bonates, E., 1985; *Application of Simulation to Assess Truck Shovel Dispatching Policies*, Mining Equipment Selection Symposium, University of Calgary & CANMET. Paper No. 24.

Lizotte, Y. and Scoble, M., 1986; *Underground mine layout optimization*, 88th Annual General Meeting of the Canadian Institute of Mining and Metallurgy, Montreal, Canada, 1-18, May.

Lizotte, Y. and Bonates, E., 1987; *Truck and Shovel Dispatching Rules Assessment using Simulation*, Mining Science and Technology, Vol 5, No.1, 45-57, May.

Luke, R.A., 1992; *Diamonds with Dispatch*, World Mining Equipment, Vol. 16, No.6, 24-28, Jun.

Luke, R.A., 1993; *Computerized vehicle dispatching goes underground*, CIM Bulletin, Vol 86, No. 967, 49-53, Feb.

Maenpaa, D. and Saindon, J.P., 1992; *Advanced Underground Communication System Development*, 5th Canadian Symposium on Mining Automation, 130-143, Sept.

Mahadevan, B. and Naredran, T.T., 1993; *Estimation of number of AGVs for FMS: an analytical model*, Int. J. Prod. Research, Vol. 31, No. 7, 1655-1670.

Mao, W. and Kincaid, R.K., 1994; *A look-ahead heuristic for scheduling jobs with release dates on a single machine*, Computers Ops Res., Vol. 21, No. 10, 1041-1050.

Marko, D, and Gregorio, V, R., 1981; *Draw behaviour in El Salvador mine*, Design and

Operation of Caving and Sublevel Stoping Mines, AIME, New York, Chapter 57, 737-743.

Matikainen, R., 1991; *How To Increase LHD Productivity in Hard Rock Mining*, Symposium on Reliability Production and Control in Coal Mines. Wollongong. 85-88 , Sept. 2-6.

Matsuura, H., Tsubone, H. and Kanezashi, M., 1993; *Sequencing, dispatching, and switching in a dynamic manufacturing environment*, Int. J. Prod. Res., Vol. 31, No. 7, 1671-1688.

Meyer, H.I., 1979; *Truck Allocation to Shovels in an Open-Pit Mine -A Case Study on the Initial Attempt*, Computer Methods for the 80's In the Mineral Industry, SME-AIME, New York, 637-641.

Mirani, A.M., 1969; *Mine planning system for underground mines*, Proceedings, 8 th APCOM, SME, Littleton, CO., 525-537.

Miyajima, M. and Nakai, M., 1986; *The municipal functional planning model: A simultaneous regression equations and goal programming approach*, Europ. J. OR, Vol. 27, 157-167.

Mizumoto, M., 1994; *Fuzzy controls under product-sum-gravity methods and new fuzzy control methods*, ed. Kandel and Langholz, 276-294.

Modular Mining Systems, 1992; *Dispatch Underground Brochure*, 5 pp, Oct.

Moore, L.J., Davies, J.C., Harvey, A.E., Imholtz, J.J., McIlwain, R.J. and Taylor, B.W., 1992; *Multicriteria models for analysis of the environmental restoration and waste management program*, Computers and Industrial Engineering Vol. 25, Nos 1-4, 21-24.

Mueller, E.R., 1976; *Simplified Dispatching Board Boosts Truck Productivity at Cyprus Pima*, Mining Engineering, Vol. 68, No. 4, 72-76.

Muge, F.H., Santos, N., Viera, J.L. and Cortez, L., 1992; *Dynamic programming in mine planning and production scheduling*, Proceedings, 23rd APCOM. University of Arizona, Arizona. 769-778.

Muge, F.H. and Santos, N.A., 1991; *Application of dynamic programming in a cut-and-fill mining method*, Proceedings, 21st APCOM. SME, Littleton, Colorado. 479-484.

Mukhopadhyay, S.K., Maiti, B. and Garg, S., 1991; *Heuristic solution to the scheduling problems in flexible manufacturing system*, Int. J. Prod. Research, Vol. 29, No. 10, 2003-2024.

Mutmansky, J.M., 1970; *A distribution-fitting method for general use*, CIM Special Vol. 12, Decision-Making in the Mineral Industry, 486-491.

Mutmansky, J.M., 1979; *Computing and Operations Research techniques for production scheduling*, In Mine Production Planning and Control. Computer Methods for the 80's in the Mineral Industry, SME-AIME, New York. 615-622.

Nakahara,Y. and Gen, M., 1993; *A method for solving linear programming problems with triangular fuzzy coefficients using new ranking index*, Computers and Industrial Engineering Vol. 25, Nos 1-4, 1-4.

Nenonen, K.L., 1982; *Interactive computer modelling of truck haulage systems*, CIM Bul. Vol. 75, No. 847, 84-89, Nov.

Noren, N., 1969; *Long-range decision models in mining*, PhD Thesis, The Economic Research Institute, Stockholm School of Economics, Sweden.

Oberholzer, J.W. & Suboleski, S.C.; 1993; *Real time section management: A new approach*, Mining Engineering, Vol.45, No.6, 627-632, Jun.

O'Hara, T.A., 1987; *Quick guides to mine operating costs and revenue*, CIM 89th Annual General Meeting No. 186, 7 pp, May.

Paraszczak, J. and Perreault, J.F., 1994; *Reliability of diesel powered load-haul-dump machines in an underground Quebec mine*, CIM Bulletin, 123-127, Mar.

Pareja, L.D. and Pelley, C.W., 1995; *The effects of mine mechanization and automation on the development of design and planning strategies for underground hard rock-mining,* 3rd Int. Symposium on Mine Mechanization and Automation, Colorado, Session 21, 17-28.

Pelley, C.W., 1994; *A study of sequencing strategy for tabular, hardrock orebodies*, PhD Thesis, McGill University.

Perros, H.G., 1994; *Queuing networks with blocking*, Publ. Oxford University Press.

Pourbabai, B., 1993; *An operational strategy for throughput maximization and bottleneck control in assembly line system: by selection of the processing rates*, Opl. Res. Soc. Vol. 44, No. 10, 1003-1011.

Randhawa, S.U. and West, T.M., 1992; *Incorporating parameter variability in multi-attribute evaluation*, Computers and Industrial Engineering Vol. 23, Nos 1-4, 389-392.

Ravenscroft, P.J., 1992; Re*coverable reserve estimation by conditional simulation*, Case

Histories and Methods in Mineral Resource Evaluation, Geological Society Special Publication No. 63, ed. Annels, A.E., 289-298.

Ravenscroft, P.J., 1992; *Risk analysis for mine scheduling by conditional simulation*, Trans. Instn Min. Metall. Vol. 101, A104-A108, May-Aug.

Ross-Watt, D.A.J., 1979; *Backfilling on the base metal mines of the Gold Fields Group*, 4th Int. Symp. on Mining Backfill Technology. Hassani, F.P., Scoble, M.J. and Yu, T.R. eds., 351-360.

Royle, A.G., 1988; *Workshop in Geostatistics*, Leeds University.

Royle, A.G., 1992; *A personal overview of geostatistics*, Case Histories and Methods in Mineral Resource Evaluation, Geological Society Special Publication No. 63, ed. Annels, A.E., 233-241.

Ribeiro, L.T., 1982; *Dynamic programming applied to the mining sequence optimization in a sublevel stoping exploitation*, Proceedings, 17th APCOM. Colorado School of Mines, Golden, Co., 494-499.

Russel, F.M., 1977; *Application of a PC-based network analysis program to mine scheduling*, Proceedings, 20th APCOM, Vol 1. Mining. Johannesburg, SAIMM. 123-131.

Ryder, J.A., 1977, *TRANSIM II - A New Generalized Underground Transport Simulator*, 14th APCOM, Pennsylvania State University, 105-111.

Sadler, W.M., 1988; *Practical truck dispatch - A micro computer based approach*, Computer Applications in the Mineral Industry, First Canadian Conference, Quebec. 495-500, Mar.

Sandbak, L., 1988; *Rock mass classification in LHD mining at San Manuel, Arizona*, AIME Annual Meeting, Phoenix, Arizona, 1-19, Jan. 25-28.

Sasaki, M. and Gen, M., 1993; An *extension of interactive method for solving multiple objective linear programming with fuzzy parameters*, Computers and Industrial Engineering Vol. 25, Nos 1-4, 9-12.

Schniederjans, M.J. and Kwak, N.K., 1982; *An alternative solution method for goal programming problems: A tutorial*, J of OR Soc. Vol. 33, 247-254, Jan-Jun.

Scoble, M., 1994; *Competitive at depth: Re-engineering the hardrock mining process*, Proceedings, 1 st North American Rock Mechanics Symposium, Austin, USA, 12 pp, May.

Scoble, M.J. and Moss, A., 1994; *Dilution in underground bulk mining: implications for production management*, Mineral Reserve Evaluation II: Methods and Case Histories, Geological Society Special Publ. No.79, Whateley M.K.G. and Harvey, P.K. (eds), 95-108.

Scoble, M., 1995; *Geological control in the digital mine*, Presented at MRE95, University of Leeds, 14 pp, Apr.

Sides, E.J., 1992; *Reconciliation studies and reserve estimation,* Case Histories and Methods in Mineral Resource Evaluation, Geological Society Special Publication No. 63, Annels, A.E. (ed), 197-218.

Singh, A. and Skiniewski, M.J., 1991; *Development of flexible production system for strip mining*, Mining Sci. and Tech., Vol. 13, 75-88.

Sloan, D.A., 1983; *Mine Management*, Publ. Methuen Publications, Agincourt, Ontario, Canada.

Soumis, F. and Elbrond, J., 1990; *Truck dispatching software using mathematical programming implemented on IBM-PC*, GERAD Publication #G-90-43, Ecole Polytechnique & McGill University, Sept.

Soyibo, A. and Lee, S.M., 1986; *Multiple objective planning model for university resource allocation*, Europ. J. OR, Vol. 27, 168-178.

Stecke, K.E. and Raman, N., 1995; *FMS planning decisions, operating flexibilities, and system performance*, IEEE Transactions of Engineering Management, Vol. 42, No. 1, 82-89.

Stein, W.E. and Cote, M.J., 1994; *Scheduling arrivals to a queue*, Computers Res., Vol. 21, No. 6, 607-614.

Suboleski, S.C. and Lucas, J.R., 1969; *Simulation of Room and Pillar Mining System*, 8th APCOM, A Decade of Digital Computing in the Mineral Industry, SME-AIME, New York, 373-384.

Swain,H.W., 1979; *Bougainville's EDP Technique Up Mine Productivity, Simplify Planning*, Mining Engineering, Vol. 31, No. 3, 74-82, Mar.

Sweigard, R.J., 1992; *Materials handling: Loading and haulage*, SME Mining Engineering Handbook 2nd Ed. Vol. 1, Chapter 9.3, 761-782.

Tan, S. and Ramani, R.V., 1992; *Optimization models for scheduling ore and waste production in open pit mines*, Proceedings, 23rd APCOM. University of Arizona,

Arizona. 781-791.

Tang, X., Xiong, G. and Li, X., 1993; *An integrated approach to underground gold mine planning and scheduling optimisation*, Proceedings, 24th APCOM, Montreal, Canada. 148-15.

Taylor, H.K., 1972; *General background theory of cut-off grade*, Trans Inst Min Metall, Section A, Vol.81, 160-179, Jul.

Technology Development Centre, Finland. 1993 Brochure. *Intelligent mine technology programme 1992-1996.*

Tempelmeier, H., Kuhn, H. and Tetzlaff, U., 1989; *Performance evaluation of flexible manufacturing systems with blocking*, Int. J. Prod. Res., Vol. 27, No. 11, 1963-1978.

Tolwinski, B.and Underwood, R., 1992; *An algorithm to estimate the optimal evolution of an open pit mine*, Proceedings, 23rd APCOM. University of Arizona, Arizona. 399-409.

Topuz, E. and Duan, C., 1991; *An analytical approach to evaluation of the operational effectiveness of continuous mining systems*, Mining Sci and Tech., Vol. 12, 145-155.

Tsomondo, C.M., 1994; *Analysis of ore dilution parameters in open-pit mines*, 3rd Symposium on Science and Technology, Harare, Zimbabwe, Sept. 20-22, 10 pp.

Tsomondo, C.M. and Lizotte, Y., 1994; *Production scheduling and dispatching for underground mines*, 6th Canadian Symposium on Mining Automation, 209-217, Oct. 16-19.

Tu, J.H. and Hucka, V.J., 1985; *Analysis of open-pit truck haulage system by use of a computer model*, CIM Bulletin, Vol 78. No. 879. 53-59, Jul.

Universal Scheduling Consulting, 1993; *Creighton Mine, INCO Ltd.*, Internal Report, June3.

Vagenas, N.; 1988; PROFITIS: *A CAD simulator for automated guided LHD in underground mining*. Computer Applications in the Mineral Industry. Proceed. of 1st Canadian Conf. on Computer Applications in the Mineral Industry. 413 - 422.

Vallee, M., David, M., Dagbert, M. and Desrochers, C., 1992; *Guide to the evaluation of gold deposits*, Special Volume 45, CIM.

Wang, F., 1989; *A fuzzy expert system for remote sensing image analysis*, Proceedings IGARSS, Inst. Electr. Electron. Eng. CH2768-0, 848-851.

Wang, Q. and Sevim, H., 1992; *Enhanced production planning in open pit mining through intelligent dynamic search*, Proceedings, 23rd APCOM. University of Arizona, Arizona, 461-470.

Wang, P.P. and Chang, S.K., 1980; *Fuzzy sets theory and applications to policy analysis and information systems*, Publ. Plenum Press, New York, 400 pp.

Ward, T.L., Ralston, P.A.S. and Davis, J.A., 1992; *Fuzzy logic control of aggregate production planning*; Computers and Industrial Eng. Vol. 23, Nos 1-4, 137-140.

Weakly, L.A., 1993; *Use of diesel equipment in underground mines - the pros and cons*. Mining Engineering. Vol.45, No.8, 1022-1024, Aug.

West-Hansen, J., Sarin, S.C. and Topuz, E., 1986; *Long-term scheduling in underground coal mines - an application of sequencing theory*. Proceedings, 19th APCOM. Pennsylvania State University, PA, 185-195.

White, J.W., Olson, J.P. and Vohnout, S.I., 1993; *On improving truck/shovel productivity in open pit mines*, CIM Bulletin, Vol. 86, No. 973, 43-49, Sept.

Whiteway, P., 1993; *"Just-in-time" mining*, Canadian Mining Journal, 20-23, Aug.

Wilke, F.L., 1971; *Simulation studies of computer-controlled traffic underground in large coal mines*, C.I.M. Special Volume No. 12. Decision-Making in the Mineral Industry, 344-351.

Wilke, F.L., 1971; *Integrated optimisation of production in a German Potash Mine*, C.I.M. Special Volume No. 12. Decision-Making in the Mineral Industry, 361-367.

Wilke, F.L., Mueller, K. and Wright, E., 1984; *Ultimate pit production scheduling optimisation*, Proceedings, 18th APCOM. IMM. London, UK. 29-37.

Wilke, F.L. and Reimer, Th., 1979; *Optimizing the short term production schedule for an open-pit iron ore mining operation*, In Mine Production Planning and Control. Computer Methods for the 80's In the Mineral Industry, SME-AIME, New York, 642-646.

Wilke, F.L., Fabian, J. and Vogt, R., 1995; *Treating of production planning tasks in surface and underground hard-rock mine operations by means of knowledge-based techniques*, 3rd Int. Symposium on Mine Mechanization and Automation, Colorado, Session 18, 39-48.

Wright, E.A., 1990; *A dynamic programming in open pit mining sequence planning*:- a case study, Proceedings, 21st APCOM. SME, Littleton, Colorado. 415-471.

Wu, S.D. and Wysk, R.A., 1989; *An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing*, Int. J. Prod. Vol. 27, No. 9, 1603-1623.

Xiaotian, S., 1992; *Caving process simulation and optimal mining sequence at Tong Kuang Yu mine*, China. Proceedings, 23rd APCOM. SME. Littleton, CO; 386-392.

Xing, L. and Rong, Z.J., 1993; *The application of the dynamic iteractive decision analysis support system in mine planning for underground mine*. Proceedings, 24th APCOM, Montreal,Canada. 5pp.

Yager, R.R. and Filev, D.P., 1994; *Essentials of Fuzzy Modelling and Control*, New York: J. Wiley.

Yamamoto, M. and Nof, N.Y., 1985; *Scheduling/re-scheduling in the manufacturing operating system environment*, Int. J. Prod. Research, Vol. 23, No. 4, 705-722.

Yingling, J.C., 1992; *Cycles and Systems*, SME Mining Engineering Handbook, Chapter 9.4, SME-AIME, ed. Hartmann, H.L., 783-805.

Youdi, Z., Qingxian, C., Lixin, W. and Daxian, Z., 1992; *Combined approach for surface mine short term planning optimization*, Proceedings, 23rd APCOM. University of Arizona, Arizona. 500-506.

Yun, Q.X. and Lu, C.W., 1992; *Optimisation for the determination of transportation system in open-pit mines*, Proceedings, 23rd APCOM. University of Arizona, Arizona. 535-545.

Zadeh, L., 1973; *Outline of a new approach to the analysis of complex systems and decision processes*, IEEE Trans Systems, Man and Cybernetics, SMC 3, 28-44.

Zeng, X. and Singh, M.G., 1995; *Approximation theory of fuzzy systems - MIMO case*, IEEE Transactions on Fuzzy Systems, Vol.3, No. 2, 219-232, May.

Zimmermann, H.J., 1989; *Strategic planning, operations research and knowledge based systems*. In Verdegay and Delgado, 253-274.

Zimmermann, H.J., 1991; *Fuzzy set theory and its application*, Kluwer Academic Publ., 2nd ed.

Appendix A

Fuzzy Logic Stope Model Program Listing

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <math.h>

int no_rules = 81;
int indata(char *); /* input file containing antecedent and input var memb */
int indat(char *);  /* input file containing consequent memberships */
float man[10];
char far input[40], result[40];
FILE *fp, *fp2;

main()
{
int i,j, k;
int width=0;
/* width of fuzzy sets & # of rules */
int flag1 =0, flag2=0, flag3=0, flag4=0;

float maxi[4]={0}, sum[81]={0}, temp=0;

/*   rule base term sets of linguistic variables values        */
float drwv[81][10]={0}, rmv[81][10]={0};
float dim[81][10]={0}, smp[81][10]={0}, y_out[81][10]={0};

/*  definition of input linguistic variables == fuzzy sets values */
float in_dim[10]={0}, in_smp[10]={0};
float in_rmv[10]={0}, in_drwv[10]={0}, rule_out[81][10]={0};

/*  output arrays of fuzzy aggregation and final crisp value      */
float fire_rule[81]={0};
float fuz_out[81]={0}, output=0, out_memship=0;

/*   rule base term sets of linguistic variables == fuzzy sets   */
char DM[81][15], SM[81][15], DRW[81][15], RM[81][15];
char YOUT[81][15];
char zip, c;

/*  input linguistic variables == fuzzy sets of decision under analysis */
char IN_DM[15], IN_SM[15];
char IN_DRW[15], IN_RM[15];
char filename[30];
char RR[15], SS[15], DD[15], DW[15];
FILE *fp1;

/*  variable interaction arrays : FAM analysis */
char  stope[15];
float next = 0, last = 0;
float comp[10][10]  ={0};
float prob_dis[10]={0};
float dcomp[10][10]  ={0};

clrscr();
printf("\t\tSTOPE FUZZY MODELLING
PROGRAM\n\t\t=============================\n\n");
printf("Enter the input file name containing state & input variables \n");
```

```
gets(input);

if ((fp = fopen(input,"r")) = =NULL){
  printf("Fail to open \"%s\"\n", input);
  exit(EXIT_FAILURE);
  }

printf("\nEnter the input file name containing consequent variables \n");
gets(result);

if ((fp2 = fopen(result,"r")) = =NULL){
  printf("Fail to open \"%s\"\n", result);
  exit(EXIT_FAILURE);
  }

printf("\nEnter the output file name \n");
gets(filename);

if ((fp1 = fopen(filename,"w")) = =NULL){
  printf("Fail to create \"%s\"\n", filename);
  exit(EXIT_FAILURE);
  }
clrscr();
re_enter: printf("\nModifiers must be entered with an under score (_) e.g. v_good\n\
or SLIGHTLY_LARGE, V_POOR, etc.\n");
       printf("\nEnter the linguistic variables values describing:\n");
     printf("\nEnter the Stope Label Number = > "); gets(stope);
     printf("\nSampling information = > "); gets(IN_SM);
     printf("\nRock Mass information = > "); gets(IN_RM);
     printf("\nDrawing information = > "); gets(IN_DRW);
     printf("\nDimension information = > "); gets(IN_DM);

    printf("\nYou entered the following values:\n\t%s: for sample info\n\t%s\
    : for rock mass info\n\t%s: for draw info\n\t%s: for dimension info", IN_SM,IN_RM,IN_DRW,IN_DM);
    printf("\nPress Y/y to accept input or N/n to re-input\n");
    scanf("%c",&zip);
    tolower(zip);

    if(zip= ='n'){   /* ascertain the input terms correction */
    clrscr();
    goto re_enter;
   }
    clrscr();
/* ------------------------------------------------------------------- *
 *                converting strings to lower case                  *
 * ------------------------------------------------------------------- */
 strlwr(IN_SM);
 strlwr(IN_RM);
 strlwr(IN_DM);
 strlwr(IN_DRW);
 strcpy(SS,IN_SM);
 strcpy(RR,IN_RM);
 strcpy(DD,IN_DM);
 strcpy(DW,IN_DRW);
/* ------------------------------------------------------------------- */
    textcolor(YELLOW + BLINK);
    gotoxy(20,12); cprintf("Please wait... Still computing\n");
    textcolor(YELLOW);

/* ------------------------------------------------------------------- *
 *          KNOWLEDGE BASE RULES FOR FUZZY REASONING              *
 * ------------------------------------------------------------------- */
```

```
if((strcpy(DM[0],"large"))&&(strcpy(RM[0],"high"))&&(strcpy(SM[0],"good"))&&(strcpy(DRW[0],"full")))
strcpy(YOUT[0],"v_good");

if((strcpy(DM[1],"large"))&&(strcpy(RM[1],"high"))&&(strcpy(SM[1],"good"))&&(strcpy(DRW[1],"half")))
strcpy(YOUT[1],"v_good");

if((strcpy(DM[2],"large"))&&(strcpy(RM[2],"high"))&&(strcpy(SM[2],"good"))&&(strcpy(DRW[2],"empty")))
strcpy(YOUT[2],"v_good");

if((strcpy(DM[3],"large"))&&(strcpy(RM[3],"high"))&&(strcpy(SM[3],"fair"))&&(strcpy(DRW[3],"full")))
strcpy(YOUT[3],"v_good");

if((strcpy(DM[4],"large"))&&(strcpy(RM[4],"high"))&&(strcpy(SM[4],"fair"))&&(strcpy(DRW[4],"half")))
strcpy(YOUT[4],"good");

if((strcpy(DM[5],"large"))&&(strcpy(RM[5],"high"))&&(strcpy(SM[5],"fair"))&&(strcpy(DRW[5],"empty")))
strcpy(YOUT[5],"good");

if((strcpy(DM[6],"large"))&&(strcpy(RM[6],"high"))&&(strcpy(SM[6],"poor"))&&(strcpy(DRW[6],"full")))
strcpy(YOUT[6],"v_good");

if((strcpy(DM[7],"large"))&&(strcpy(RM[7],"high"))&&(strcpy(SM[7],"poor"))&&(strcpy(DRW[7],"half")))
strcpy(YOUT[7],"good");

if((strcpy(DM[8],"large"))&&(strcpy(RM[8],"high"))&&(strcpy(SM[8],"poor"))&&(strcpy(DRW[8],"empty")))
strcpy(YOUT[8],"fair");

//DM at max, RM at medium, SM at various, DRW at various
if((strcpy(DM[9],"large"))&&(strcpy(RM[9],"medium"))&&(strcpy(SM[9],"good"))&&(strcpy(DRW[9],"full")))
strcpy(YOUT[9],"v_good");

if((strcpy(DM[10],"large"))&&(strcpy(RM[10],"medium"))&&(strcpy(SM[10],"good"))&&(strcpy(DRW[10],"half")))
strcpy(YOUT[10],"good");

if((strcpy(DM[11],"large"))&&(strcpy(RM[11],"medium"))&&(strcpy(SM[11],"good"))&&(strcpy(DRW[11],"empty")))
strcpy(YOUT[11],"good");

if((strcpy(DM[12],"large"))&&(strcpy(RM[12],"medium"))&&(strcpy(SM[12],"fair"))&&(strcpy(DRW[12],"full")))
strcpy(YOUT[12],"good");

if((strcpy(DM[13],"large"))&&(strcpy(RM[13],"medium"))&&(strcpy(SM[13],"fair"))&&(strcpy(DRW[13],"half")))
strcpy(YOUT[13],"fair");

if((strcpy(DM[14],"large"))&&(strcpy(RM[14],"medium"))&&(strcpy(SM[14],"fair"))&&(strcpy(DRW[14],"empty")))
strcpy(YOUT[14],"fair");

if((strcpy(DM[15],"large"))&&(strcpy(RM[15],"medium"))&&(strcpy(SM[15],"poor"))&&(strcpy(DRW[15],"full")))
strcpy(YOUT[15],"good");

if((strcpy(DM[16],"large"))&&(strcpy(RM[16],"medium"))&&(strcpy(SM[16],"poor"))&&(strcpy(DRW[16],"half")))
strcpy(YOUT[16],"fair");

if((strcpy(DM[17],"large"))&&(strcpy(RM[17],"medium"))&&(strcpy(SM[17],"poor"))&&(strcpy(DRW[17],"empty")))
strcpy(YOUT[17],"poor");

//DM at max, RM at med, rest at various
if((strcpy(DM[18],"large"))&&(strcpy(RM[18],"low"))&&(strcpy(SM[18],"good"))&&(strcpy(DRW[18],"full")))
strcpy(YOUT[18],"v_good");

if((strcpy(DM[19],"large"))&&(strcpy(RM[19],"low"))&&(strcpy(SM[19],"good"))&&(strcpy(DRW[19],"half")))
strcpy(YOUT[19],"good");
```

```
if((strcpy(DM[20],"large"))&&(strcpy(RM[20],"low"))&&(strcpy(SM[20],"good"))&&(strcpy(DRW[20],"empty")))
strcpy(YOUT[20],"fair");

if((strcpy(DM[21],"large"))&&(strcpy(RM[21],"low"))&&(strcpy(SM[21],"fair"))&&(strcpy(DRW[21],"full")))
strcpy(YOUT[21],"good");

if((strcpy(DM[22],"large"))&&(strcpy(RM[22],"low"))&&(strcpy(SM[22],"fair"))&&(strcpy(DRW[22],"half")))
strcpy(YOUT[22],"fair");

if((strcpy(DM[23],"large"))&&(strcpy(RM[23],"low"))&&(strcpy(SM[23],"fair"))&&(strcpy(DRW[23],"empty")))
strcpy(YOUT[23],"poor");

if((strcpy(DM[24],"large"))&&(strcpy(RM[24],"low"))&&(strcpy(SM[24],"poor"))&&(strcpy(DRW[24],"full")))
strcpy(YOUT[24],"fair");

if((strcpy(DM[25],"large"))&&(strcpy(RM[25],"low"))&&(strcpy(SM[25],"poor"))&&(strcpy(DRW[25],"half")))
strcpy(YOUT[25],"poor");

if((strcpy(DM[26],"large"))&&(strcpy(RM[26],"low"))&&(strcpy(SM[26],"poor"))&&(strcpy(DRW[26],"empty")))
strcpy(YOUT[26],"v_poor");

// DM as average and the rest changing
if((strcpy(DM[27],"average"))&&(strcpy(RM[27],"high"))&&(strcpy(SM[27],"good"))&&(strcpy(DRW[27],"full")))
strcpy(YOUT[27],"v_good");

if((strcpy(DM[28],"average"))&&(strcpy(RM[28],"high"))&&(strcpy(SM[28],"good"))&&(strcpy(DRW[28],"half")))
strcpy(YOUT[28],"good");

if((strcpy(DM[29],"average"))&&(strcpy(RM[29],"high"))&&(strcpy(SM[29],"good"))&&(strcpy(DRW[29],"empty")))
strcpy(YOUT[29],"good");

if((strcpy(DM[30],"average"))&&(strcpy(RM[30],"high"))&&(strcpy(SM[30],"fair"))&&(strcpy(DRW[30],"full")))
strcpy(YOUT[30],"good");

if((strcpy(DM[31],"average"))&&(strcpy(RM[31],"high"))&&(strcpy(SM[31],"fair"))&&(strcpy(DRW[31],"half")))
strcpy(YOUT[31],"fair");

if((strcpy(DM[32],"average"))&&(strcpy(RM[32],"high"))&&(strcpy(SM[32],"fair"))&&(strcpy(DRW[32],"empty")))
strcpy(YOUT[32],"fair");

if((strcpy(DM[33],"average"))&&(strcpy(RM[33],"high"))&&(strcpy(SM[33],"poor"))&&(strcpy(DRW[33],"full")))
strcpy(YOUT[33],"good");

if((strcpy(DM[34],"average"))&&(strcpy(RM[34],"high"))&&(strcpy(SM[34],"poor"))&&(strcpy(DRW[34],"half")))
strcpy(YOUT[34],"fair");

if((strcpy(DM[35],"average"))&&(strcpy(RM[35],"high"))&&(strcpy(SM[35],"poor"))&&(strcpy(DRW[35],"empty")))
strcpy(YOUT[35],"poor");

if((strcpy(DM[36],"average"))&&(strcpy(RM[36],"medium"))&&(strcpy(SM[36],"good"))&&(strcpy(DRW[36],"full")))
strcpy(YOUT[36],"good");

if((strcpy(DM[37],"average"))&&(strcpy(RM[37],"medium"))&&(strcpy(SM[37],"good"))&&(strcpy(DRW[37],"half"))
)
strcpy(YOUT[37],"fair");

if((strcpy(DM[38],"average"))&&(strcpy(RM[38],"medium"))&&(strcpy(SM[38],"good"))&&(strcpy(DRW[38],"empty
")))
strcpy(YOUT[38],"fair");

if((strcpy(DM[39],"average"))&&(strcpy(RM[39],"medium"))&&(strcpy(SM[39],"fair"))&&(strcpy(DRW[39],"full")))
strcpy(YOUT[39],"fair");
```

```
if((strcpy(DM[40],"average"))&&(strcpy(RM[40],"medium"))&&(strcpy(SM[40],"fair"))&&(strcpy(DRW[40],"half")))
strcpy(YOUT[40],"fair");

if((strcpy(DM[41],"average"))&&(strcpy(RM[41],"medium"))&&(strcpy(SM[41],"fair"))&&(strcpy(DRW[41],"empty")
))
strcpy(YOUT[41],"fair");

if((strcpy(DM[42],"average"))&&(strcpy(RM[42],"medium"))&&(strcpy(SM[42],"poor"))&&(strcpy(DRW[42],"full")))
strcpy(YOUT[42],"fair");

if((strcpy(DM[43],"average"))&&(strcpy(RM[43],"medium"))&&(strcpy(SM[43],"poor"))&&(strcpy(DRW[43],"half")))
strcpy(YOUT[43],"fair");

if((strcpy(DM[44],"average"))&&(strcpy(RM[44],"medium"))&&(strcpy(SM[44],"poor"))&&(strcpy(DRW[44],"empty
")))
strcpy(YOUT[44],"poor");

if((strcpy(DM[45],"average"))&&(strcpy(RM[45],"low"))&&(strcpy(SM[45],"good"))&&(strcpy(DRW[45],"full")))
strcpy(YOUT[45],"good");

if((strcpy(DM[46],"average"))&&(strcpy(RM[46],"low"))&&(strcpy(SM[46],"good"))&&(strcpy(DRW[46],"half")))
strcpy(YOUT[46],"fair");

if((strcpy(DM[47],"average"))&&(strcpy(RM[47],"low"))&&(strcpy(SM[47],"good"))&&(strcpy(DRW[47],"empty")))
strcpy(YOUT[47],"poor");

if((strcpy(DM[48],"average"))&&(strcpy(RM[48],"low"))&&(strcpy(SM[48],"fair"))&&(strcpy(DRW[48],"full")))
strcpy(YOUT[48],"fair");

if((strcpy(DM[49],"average"))&&(strcpy(RM[49],"low"))&&(strcpy(SM[49],"fair"))&&(strcpy(DRW[49],"half")))
strcpy(YOUT[49],"fair");

if((strcpy(DM[50],"average"))&&(strcpy(RM[50],"low"))&&(strcpy(SM[50],"fair"))&&(strcpy(DRW[50],"empty")))
strcpy(YOUT[50],"poor");

if((strcpy(DM[51],"average"))&&(strcpy(RM[51],"low"))&&(strcpy(SM[51],"poor"))&&(strcpy(DRW[51],"full")))
strcpy(YOUT[51],"poor");

if((strcpy(DM[52],"average"))&&(strcpy(RM[52],"low"))&&(strcpy(SM[52],"poor"))&&(strcpy(DRW[52],"half")))
strcpy(YOUT[52],"poor");

if((strcpy(DM[53],"average"))&&(strcpy(RM[53],"low"))&&(strcpy(SM[53],"poor"))&&(strcpy(DRW[53],"empty")))
strcpy(YOUT[53],"v_poor");

//minimum DM and various
if((strcpy(DM[54],"small"))&&(strcpy(RM[54],"high"))&&(strcpy(SM[54],"good"))&&(strcpy(DRW[54],"full")))
strcpy(YOUT[54],"v_good");

if((strcpy(DM[55],"small"))&&(strcpy(RM[55],"high"))&&(strcpy(SM[55],"good"))&&(strcpy(DRW[55],"half")))
strcpy(YOUT[55],"good");

if((strcpy(DM[56],"small"))&&(strcpy(RM[56],"high"))&&(strcpy(SM[56],"good"))&&(strcpy(DRW[56],"empty")))
strcpy(YOUT[56],"fair");

if((strcpy(DM[57],"small"))&&(strcpy(RM[57],"high"))&&(strcpy(SM[57],"fair"))&&(strcpy(DRW[57],"full")))
strcpy(YOUT[57],"good");

if((strcpy(DM[58],"small"))&&(strcpy(RM[58],"high"))&&(strcpy(SM[58],"fair"))&&(strcpy(DRW[58],"half")))
strcpy(YOUT[58],"fair");

if((strcpy(DM[59],"small"))&&(strcpy(RM[59],"high"))&&(strcpy(SM[59],"fair"))&&(strcpy(DRW[59],"empty")))
strcpy(YOUT[59],"poor");
```

```
if((strcpy(DM[60],"small"))&&(strcpy(RM[60],"high"))&&(strcpy(SM[60],"poor"))&&(strcpy(DRW[60],"full")))
strcpy(YOUT[60],"fair");

if((strcpy(DM[61],"small"))&&(strcpy(RM[61],"high"))&&(strcpy(SM[61],"poor"))&&(strcpy(DRW[61],"half")))
strcpy(YOUT[61],"poor");

if((strcpy(DM[62],"small"))&&(strcpy(RM[62],"high"))&&(strcpy(SM[62],"poor"))&&(strcpy(DRW[62],"empty")))
strcpy(YOUT[62],"v_poor");

if((strcpy(DM[63],"small"))&&(strcpy(RM[63],"medium"))&&(strcpy(SM[63],"good"))&&(strcpy(DRW[63],"full")))
strcpy(YOUT[63],"good");

if((strcpy(DM[64],"small"))&&(strcpy(RM[64],"medium"))&&(strcpy(SM[64],"good"))&&(strcpy(DRW[64],"half")))
strcpy(YOUT[64],"fair");

if((strcpy(DM[65],"small"))&&(strcpy(RM[65],"medium"))&&(strcpy(SM[65],"good"))&&(strcpy(DRW[65],"empty"))
)
strcpy(YOUT[65],"poor");

if((strcpy(DM[66],"small"))&&(strcpy(RM[66],"medium"))&&(strcpy(SM[66],"fair"))&&(strcpy(DRW[66],"full")))
strcpy(YOUT[66],"fair");

if((strcpy(DM[67],"small"))&&(strcpy(RM[67],"medium"))&&(strcpy(SM[67],"fair"))&&(strcpy(DRW[67],"half")))
strcpy(YOUT[67],"fair");

if((strcpy(DM[68],"small"))&&(strcpy(RM[68],"medium"))&&(strcpy(SM[68],"fair"))&&(strcpy(DRW[68],"empty")))
strcpy(YOUT[68],"poor");

if((strcpy(DM[69],"small"))&&(strcpy(RM[69],"medium"))&&(strcpy(SM[69],"poor"))&&(strcpy(DRW[69],"full")))
strcpy(YOUT[69],"poor");

if((strcpy(DM[70],"small"))&&(strcpy(RM[70],"medium"))&&(strcpy(SM[70],"poor"))&&(strcpy(DRW[70],"half")))
strcpy(YOUT[70],"poor");

if((strcpy(DM[71],"small"))&&(strcpy(RM[71],"medium"))&&(strcpy(SM[71],"poor"))&&(strcpy(DRW[71],"empty")))
strcpy(YOUT[71],"v_poor");

if((strcpy(DM[72],"small"))&&(strcpy(RM[72],"low"))&&(strcpy(SM[72],"good"))&&(strcpy(DRW[72],"full")))
strcpy(YOUT[72],"fair");

if((strcpy(DM[73],"small"))&&(strcpy(RM[73],"low"))&&(strcpy(SM[73],"good"))&&(strcpy(DRW[73],"half")))
strcpy(YOUT[73],"poor");

if((strcpy(DM[74],"small"))&&(strcpy(RM[74],"low"))&&(strcpy(SM[74],"good"))&&(strcpy(DRW[74],"empty")))
strcpy(YOUT[74],"v_poor");

if((strcpy(DM[75],"small"))&&(strcpy(RM[75],"low"))&&(strcpy(SM[75],"fair"))&&(strcpy(DRW[75],"full")))
strcpy(YOUT[75],"poor");

if((strcpy(DM[76],"small"))&&(strcpy(RM[76],"low"))&&(strcpy(SM[76],"fair"))&&(strcpy(DRW[76],"half")))
 strcpy(YOUT[76],"poor");

if((strcpy(DM[77],"small"))&&(strcpy(RM[77],"low"))&&(strcpy(SM[77],"fair"))&&(strcpy(DRW[77],"empty")))
 strcpy(YOUT[77],"v_poor");

if((strcpy(DM[78],"small"))&&(strcpy(RM[78],"low"))&&(strcpy(SM[78],"poor"))&&(strcpy(DRW[78],"full")))
 strcpy(YOUT[78],"v_poor");

if((strcpy(DM[79],"small"))&&(strcpy(RM[79],"low"))&&(strcpy(SM[79],"poor"))&&(strcpy(DRW[79],"half")))
 strcpy(YOUT[79],"v_poor");

if((strcpy(DM[80],"small"))&&(strcpy(RM[80],"low"))&&(strcpy(SM[80],"poor"))&&(strcpy(DRW[80],"empty")))
```

```
 strcpy(YOUT[80],"v_poor");

/*  Initializing the fired rules array to one. i.e. expect <= 1        */
 width = 10;

 for (i = 0; i < no_rules; i++)
     fire_rule[i] = 1.0;

/* ------------------------------------------------------------------- *
 *       Modifiers of input variables if needed                         *
 * ------------------------------------------------------------------- */
  if (strcmp(IN_DM,"v_large")==0){
      strcpy(IN_DM,"large");
      flag1 = 1;
      }

  if (strcmp(IN_DM,"slightly_large")==0){
      strcpy(IN_DM,"large");
      flag1 = 2;
      }

   if (strcmp(IN_DM,"slightly_small")==0){
      strcpy(IN_DM,"small");
      flag1 = 3;
      }

  if (strcmp(IN_DM,"v_small")==0){
      strcpy(IN_DM,"small");
      flag1 = 4;
      }

  if (strcmp(IN_SM,"v_good")==0){
      strcpy(IN_SM,"good");
      flag2 = 1;
      }

  if (strcmp(IN_SM,"slightly_good")==0){
      strcpy(IN_SM,"good");
      flag2 = 2;
      }

   if (strcmp(IN_SM,"slightly_poor")==0){
      strcpy(IN_SM,"poor");
      flag2 = 3;
      }

  if (strcmp(IN_SM,"v_poor")==0){
      strcpy(IN_SM,"poor");
      flag2 = 4;
      }

  if (strcmp(IN_RM,"v_high")==0){
      strcpy(IN_RM,"high");
      flag3 = 1;
      }

  if (strcmp(IN_RM,"slightly_high")==0){
      strcpy(IN_RM,"high");
      flag3 = 2;
      }

   if (strcmp(IN_RM,"slightly_low")==0){
        strcpy(IN_RM,"low");
        flag3 = 3;
        }

    if (strcmp(IN_RM,"v_low") = =0){
        strcpy(IN_RM,"low");
        flag3 = 4;
        }

    if (strcmp(IN_DRW,"v_full") = =0){
        strcpy(IN_DRW,"full");
        flag4 = 1;
        }

    if (strcmp(IN_DRW,"slightly_full") = =0){
        strcpy(IN_DRW,"full");
        flag4 = 2;
        }

     if (strcmp(IN_DRW,"slightly_empty") = =0){
        strcpy(IN_DRW,"empty");
        flag4 = 3;
        }

    if (strcmp(IN_DRW,"v_empty") = =0){
        strcpy(IN_DRW,"empty");
        flag4 = 4;
        }

/* ------------------------------------------------------------------ *
 *  Initializing the input linguistic variables with membership values  *
 * ------------------------------------------------------------------ */

indata(&IN_DM[0]);
for (i =-0; i < width; i++) in_dim[i] = man[i];
switch(flag1){
      case 0: break;
      case 1: ;
      case 4: for (i = 0; i < width; i++) in_dim[i] = pow(in_dim[i],2);
          break;
      case 2: ;
      case 3: for (i = 0; i < width; i++) in_dim[i] = pow(in_dim[i],0.5);
          break;
     }

indata(&IN_SM[0]);
for (i = 0; i < width; i++) in_smp[i] = man[i];
switch(flag2){
      case 0: break;
      case 1: ;
      case 4: for (i = 0; i < width; i++) in_smp[i] = pow(in_smp[i],2);
          break;
      case 2: ;
      case 3: for (i = 0; i < width; i++) in_smp[i] = pow(in_smp[i],0.5);
          break;
     }

indata(&IN_RM[0]);
for (i = 0; i < width; i++) in_rmv[i] = man[i];
switch(flag3){
      case 0: break;
      case 1: ;
     case 4: for (i = 0; i < width; i++) in_rmv[i] = pow(in_rmv[i],2);
        break;
     case 2: ;
     case 3: for (i = 0; i < width; i++) in_rmv[i] = pow(in_rmv[i],0.5);
        break;
    }

indata(&IN_DRW[0]);
for (i = 0; i < width; i++) in_drwv[i] = man[i];
switch(flag4){
     case 0: break;
     case 1: ;
     case 4: for (i = 0; i < width; i++) in_drwv[i] = pow(in_drwv[i],2);
        break;
     case 2: ;
     case 3: for (i = 0; i < width; i++) in_drwv[i] = pow(in_drwv[i];0.5);
        break;
    }

/* ---------------------------------------------------------------------- *
 *          initializing the rule variables with their memberships      *
 * ---------------------------------------------------------------------- */

for(i = 0; i < no_rules; i++){
 indata(&DM[i][0]);                          /* antecedents initials */
 for (j = 0; j < width; j++)  dim[i][j] = man[j];
}

for (i=0; i<no_rules; i++){
 indata(&SM[i][0]);
 for (j = 0; j < width; j++)  smp[i][j] = man[j];
 }

for (i=0; i<no_rules; i++){
 indata(&DRW[i][0]);
 for (j = 0; j < width; j++)  drwv[i][j] = man[j];
}
for (i=0; i<no_rules; i++){
 indata(&RM[i][0]);
 for (j = 0; j < width; j++)  rmv[i][j] = man[j];
}

for (i=0; i<no_rules; i++){
 indat(&YOUT[i][0]);                          /* consequent initials */
  for (j = 0; j < width; j++) y_out[i][j] = man[j];
 }

/* ---------------------------------------------------------------------- *
 *          Finds the degree of firing of the current rule              *
 *          by taking the minimum of the antecedents firing levels      *
 * ---------------------------------------------------------------------- */

for (i = 0; i < no_rules; i++){
  maxi[0] =0;
  maxi[1] =0;
  maxi[2] =0;
  maxi[3] =0;
 for (j = 0; j < width; j++){
  if(dim[i][j] <= in_dim[j]) temp = dim[i][j]; /* intersection of rule & input */
  else temp = in_dim[j];

  if(maxi[0] <= temp) maxi[0] = temp;         /* keep highest membership to date */
 if(smp[i][j] <= in_smp[j]) temp = smp[i][j]; /* intersection of rule & input */
 else temp = in_smp[j];

 if(maxi[1] <= temp) maxi[1] = temp;      /* keep highest membership */

 if(rmv[i][j] <= in_rmv[j]) temp = rmv[i][j]; /* intersection of rule & input */
 else temp = in_rmv[j];

 if(maxi[2] <= temp) maxi[2] = temp;       /* keep highest membership */

 if(drwv[i][j] <= in_drwv[j]) temp = drwv[i][j]; /* intersection of rule & input */
 else temp = in_drwv[j];

 if(maxi[3] <= temp) maxi[3] = temp;      /* keep highest membership */

}

for (k = 0; k < 4; k++)              /* take the minimum of the maxi[] membership */
 if(fire_rule[i] >= maxi[k]) fire_rule[i] = maxi[k]; /* degree of firing of rule i */
}

/* -------------------------------------------------------------------- *
 *    Fuzzy implication: aggregation of DOF & Consequent fuzzy set      *
 * -------------------------------------------------------------------- */

 for (i = 0; i < no_rules; i++){ /* Rule output fuzzy set: Product operator */
  for (j = 0; j < width; j++){
    rule_out[i][j] = y_out[i][j] * fire_rule[i];
    }
  }

/* -------------------------------------------------------------------- *
 *    FUZZY OUTPUT: Aggregation of each rule's  fuzzy output            *
 * -------------------------------------------------------------------- */

 for (j = 0; j < width; j++){
  for (i = 0; i < no_rules; i++){
    sum[j] += rule_out[i][j];
    }
   fuz_out[j] = sum[j]/no_rules;
 }

/* -------------------------------------------------------------------- *
 *     DEFUZZIFICATION OF OUTPUT SET: Centre of Gravity Method          *
 * -------------------------------------------------------------------- */

 for (i = 0; i < width; i++){
    output += fuz_out[i] * i;        /* integral of value * membership */
    out_memship += fuz_out[i];        /* integral of output membership  */
  }

 if(out_memship != 0) output = output/out_memship; /* crisp value of problem */
 else output = 0;
 /* -------------------------------------------------------------------- *
 *                  CALCULATION OF OUTPUT FUZZINESS                     *
 * -------------------------------------------------------------------- */
  float fuzz = 0;  temp =0;
   for (i = 0; i < width; i++){
    temp = fuz_out[i] * 2;
    temp = fabs(temp -1);
     fuzz += temp;
     }
     fuzz = 1 - (fuzz/width);

/* ---------------------------------------------------------------------- *
 *   Calculation of the Output Probability Distribution & Probability   *
 * ---------------------------------------------------------------------- */
 for (i = 0; i < width; i++){
     prob_dis[i] = fuz_out[i]/out_memship;
 }

/* ---------------------------------------------------------------------- *
 *                   FUZZY ASSOCIATIVE MATRICES                        *
 * ---------------------------------------------------------------------- */
/* ---------------------- dim vs sampling ------------------------------ */
 for (i = 0; i < width; i++){
 for (j = 0; j < width; j++){
   next = in_dim[i] * in_smp[j];              /* product operation of inputs */
   if (comp[i][j] <= next ) comp[i][j] = next; /* evaluation of element 9x */
   else continue;
  }
 }

/* --------- draw vs rock mass factor ------------------------------- */
 for (i = 0; i < width; i++){
 for (j = 0; j < width; j++){
     last = in_drwv[i] * in_rmv[j];           /* product operation of inputs */
   if (dcomp[i][j] <= last) dcomp[i][j] = last; /* evaluation of element 9x */
   else continue;
  }
 }
 clrscr();

 fprintf(fp1,"\n\n\t\tFUZZY STOPE MODEL OUTPUT \n\t\t------------------------ \n");
 fprintf(fp1,"\n\t\tSTOPE No:\%s\n",stope);
 printf("\n\t\tFUZZY STOPE MODEL OUTPUT \n\t\t------------------------ \n");
 printf("\n\t\tSTOPE No:\t%s\n",stope);

fprintf(fp1,"\nFAM for Dimension (or Regularity) vs Sampling Intensity [X,Y]\n\n");
printf("\nFAM for Dimension (or Regularity) vs Sampling Intensity [X,Y]\n\n");
for (i = 0; i < width; i++){
 for (j = 0; j < width; j++){
   printf("%5.3f ", comp[i][j]);
   fprintf(fp1,"%5.3f ", comp[i][j]);
   if (j == (width -1)) printf("\n");
   if (j == (width -1)) fprintf(fp1,"\n");
   }
 }
 printf("\nPress any key to continue...\n");
 getch();
fprintf(fp1,"\nFAM for Cumulative draw vs Rock mass characteristics [X,Y]\n\n");
printf("\nFAM for Cumulative draw vs Rock mass characteristics [X,Y]\n\n");
for (i = 0; i < width; i++){
 for (j = 0; j < width; j++){
   printf("%5.3f ", dcomp[i][j]);
   fprintf(fp1,"%5.3f ", dcomp[i][j]);
   if (j == (width -1)) printf("\n");
   if (j == (width -1)) fprintf(fp1,"\n");
   }
 }
 getch();

/* ---------------------------------------------------------------------- *
 *                   PROGRAM OUTPUT                          *
 * ----------------------------------------------------------------------- */

 fprintf(fp1,"\nTotal rule-base membership\n--------------------------- \n");
 fprintf(fp1,"Output membership : ");
 printf("\nTotal rule-base membership\n--------------------------- \n");
 printf("Output membership : ");
  for (j = 0; j <width; j++){
      printf("%3.2f ",fuz_out[j]);
      fprintf(fp1,"%3.2f ",fuz_out[j]);
     }

 printf("\n\nOutput probability distribution :\n----------------------------------\n");
 fprintf(fp1,"\n\nExpected Probability Distribution :\n-------------------------------------\n");
  for (j = 0; j <width; j++){
      printf("%3.2f/%d ",prob_dis[j],j+1);
      fprintf(fp1,"%3.2f/%d ",prob_dis[j],j+1);
    }

 fprintf(fp1,"\n\nTotal aggregation of variables effects:= %f\n\n", output);
 printf("\n\nTotal aggregation of variables effects:= %f\n\n", output);

 if(output >=7.0) printf("The output is strongly positive weighted:= GOOD\n");
 else if ((output >3.5)&&(output<7.0)) printf("The output is neutral:= FAIR\n");
 else printf("The output is strongly negative weighted:= POOR\n");
 printf("\nN.B. Objective is to maximize Decision Output (1 --> 10).\n\n");
 printf("Fuzziness measure of stope:= %6.4f\n",fuzz);
 printf("\nInput variables were: Sampling = %s \t\tRock mass factor = %s\n",SS,RR);
 printf("\t\tCumulative draw = %s \t\tStope dimension = %s\n",DW, DD);
 fprintf(fp1,"Fuzziness measure of stope:= %6.4f\n",fuzz);
 getch();

 printf("\n\tDo you want to model another stope?\n");
 scanf("%c",&c);
 if(tolower(c) == 'y'){
    out_memship = 0.0;
    clrscr();
    goto re_enter;
   }
  else ;
 return (0);
}

/* ----------------------------------------------------------------------- *
 *                          PROGRAM FUNCTIONS                              *
 * ----------------------------------------------------------------------- */

int indata(char *type)
{//function reads in the degree of membership for each input variable
 //searches for appropriate fuzzy term in the data file
    char  *mt = (char*) calloc(12,sizeof(char));
    char  *xx = (char *) calloc(2,sizeof(char));
     xx = "\n";
     if(( fp = fopen(input,"r"))==NULL){
     printf("Failed to open \" %s\" file. \n",input);
     getch();
     exit(EXIT_FAILURE);
      }

    strcat(type,xx);
    strlwr(type);
 while(fgets(mt, 12, fp) != NULL){
      strlwr(mt);
    if(strcmp(mt, type) == 0){
     fscanf(fp, "%f %f %f %f %f %f %f %f %f %f\n", &man[0], &man[1],&man[2],\
        &man[3], &man[4],&man[5],&man[6],&man[7],&man[8],&man[9]);
        free(mt);
        fclose ( fp );
      return ( 1 );
    }
     }
    fclose( fp );
   return ( 0 );
}

int indat(char *type)
{//function reads in the degree of membership for each input variable
 //searches for appropriate fuzzy term in the data file
   char  *mt = (char*) calloc(12,sizeof(char));
   char  *xx = (char *) calloc(2,sizeof(char));
    xx = "\n";
    if(( fp2 = fopen(result,"r")) ==NULL){
    printf("Failed to open \" %s\" \n",result);
    getch();
    exit(EXIT_FAILURE);
     }

   strcat(type,xx);
   strlwr(type);
 while(fgets(mt, 12, fp2) != NULL){
     strlwr(mt);
    if(strcmp(mt, type) == 0){
     fscanf(fp2, "%f %f %f %f %f %f %f %f %f %f\n", &man[0], &man[1],&man[2],\
        &man[3], &man[4],&man[5],&man[6],&man[7],&man[8],&man[9]);
        free(mt);
        fclose ( fp2 );
      return ( 1 );
    }
     }
    fclose( fp2 );
   return ( 0 );
}
```

Fuzzy Logic Stope Model Input Data Files

The Fuzzy Logic Stope Model requires two input database files which contains the description of the various membership functions of the linguistic variables. The first database file contains all the membership functions describing the rock mass factor, stope dimension or structure factor, the sampling information base and the cumulative ore draws. The format of this file is as follows:

1. Linguistic variable value (case insensitive)
   e.g. large >>
2 Membership values describing the linguistic value defined in 1 above.
   e.g. 0.0 0.2 0.4 0.6 0.7 0.9 1.0 0.8 0.5 0.3 >>
   where >> implies a new line or carriage return .

Repeat this two step process for all linguistic variable values that can be encountered at the mine site.

Once this database is created it can be used repeatedly in the Fuzzy Logic Stope Model without modification until some reconciliation of the system parameters indicates a need for revision of the membership functions.

A second input data file has the same format as that described above. This file contains the membership functions of the consequent values.

Typical examples of the two input files are shown in Tables A1 and A2 respectively. These data files were used to generate the results described in Chapter 4.

Table A1 Input variables database file for Inza Mine (ref. Chapter 4)

```
good
0 0 0 .1 .3 .4 .6 .9 1 1
fair
0 .2 .6 1 1 .9 .8 .5 .2 0
full
0 0 0 .3 .4 .6 .7 .8 .9 1
high
0 0 0 .3 .6 .7 .9 .9 1 1
half
.1 .3 .6 .8 1 .7 .6 .5 .1 0
medium
0 .2 .4 .6 .9 1 .7 .4 .1 0
small
1 1 .8 .6 .5 .4 .3 .2 .1 0
large
0 0 .2 .2 .4 .5 .7 .8 1 1
average
0 0 .3 .6 .8 1 .9 .6 .3 .0
poor
1 .9 .7 .5 .2 .1 0 0 0 0
empty
1 .9 .7 .4 .1 0 0 0 0 0
low
1 .9 .8 .4 .1 0 0 0 0 0
```

Table A2 Consequent values input data file

```
v_good
0 0 0 0 .1 .3 .6 .8 1 1
good
0 0 0 .2 .4 .6 .8 .9 1 1
fair
0 .2 .4 .8 1 .8 .4 .2 .0 .0
poor
1 1 .8 .5 .3 .2 .1 .0 .0 .0
v_poor
1 .9 .6 .4 .1 .0 .0 .0 .0 .0
```

The program Fuzzy Logic Stope Model prompts the user to give the two databases described above. Next, the user is prompted to enter a linguistic description of each of the four modelled parameters in turn for a named stope or draw-point. The legal linguistic values are those in Table A1. In addition, the user can use two modifiers, namely **very** and **slightly** to each of those values. For example, the stope sampling information can be described as very_good or slightly_poor, etc.

Appendix B
Goal Programming Model Program Listing

```
#include <stdio.h>
#include <stdlib.h>
#include <fstream.h>
#include <math.h>
#include <conio.h>
#include <string.h>
#include <time.h>

main()
{
//DECLARACTION OF VARIABLES
int nrow=0, ncol=0, nvar=0;  /* nrow = # of goal constraints */
int nprt=0, ktest=0;         /* nprt = # of priorities, iter= iterations */
int iter=0,total=0,kval=0;
int ibasic[61]={0};    /* array containing the basic solution */
int jcol[121]={0};     /* tableau columns for non-basic variables */
int jselect[121]={0};
int jfail[61]={0};

int lest_achv_pt=0;     /* least achieved highest priority */
int kend=0;             // total number of priorities (+ artificial)
int pivt_row=0, pivt_col=0, priority=0; /* pivot row & col and priority number */
int j=0,l=0,i=0, k=0;
int huprio=0, m=0;        //huprio = highest unachieved priority: flag
int bth_dev=0, ind=0;     //bth_dev = # of constraints with both + & - deviations
int row=0, col=0, mix=0;
int index=0, var=0, zano=0, net=0;

float prhs[61]={0};         /* array of initial tableau RHS values */
float rhs[61]={0};          /* array of transformed RHS in problem solving */
float valb[9][61]={0};      /* priority X decision variable matrix */
float valc[9][121]={0};     /* deviational weights matrix   */
float zval[9][121]={0};     /* total absolute deviation values for constraints */
float basis[61][121]={0};  /* initial tableau array  */
float array[61]  ={0};      /* array of output decision variables */
float pos_dv[61]={0};       /* positive deviation from RHS values of goal constr */
float neg_dv[61]={0};       /* negative dev. of RHS values of goal constrans */
float wgt=0.0;             /* deviational weight  */
float piv=0.0,dummy=0;      /* piv = pivot element */
float rmin=0.0, coef=0;     /* coef = technological coefficients of decision var */
float zvalue=0;            /* absolute deviational value for a certain priority */
float theta=0, zeta=0, target=0, diff=0; /* flag values for computational tests:limits */

char finam[20], finam2[20], state[15], sign;  //filenames and deviational var sign
char shift[80];                          //string for shift identification */
FILE *fp1, *fp2;                           // file pointers
time_t start, ended;
```

```
    and 60 decision variables. */

   clrscr();
   window(2,2, 80,45);
   textcolor(YELLOW);
   textbackground(BLUE);
   gotoxy(5,2); printf("\tGOAL PROGRAMMING MODEL
n\t=========================\n ");
   gotoxy(5,9); printf("Please enter the input filename... \n");
   gotoxy(5,11); gets(finam);

  if((fp1 = fopen(finam,"r"))==NULL)
    {
     fprintf(stderr, "Failed to create input file: \"%s\".\n",finam);
     printf("Press any key to halt...");
     getch();
     return 1;
    }

   gotoxy(5,13);  printf("Please enter output filename... \n");
   gotoxy(5,15);  gets(finam2);

  if((fp2 = fopen(finam2,"w"))==NULL)
    {
     fprintf(stderr, "Failed to create output file: \"%s\".\n",finam2);
     printf("Press any key to halt...");
     getch();
     return 1;
    }

     gotoxy(5,17); printf("Press any key to continue \n");
     getch();
     clrscr();

   start = clock();
   // reading input file
   fgets(shift,80, fp1);          //shift number or date
   fscanf(fp1,"%d%d%d%d\n",&nrow,&nvar,&nprt,&bth_dev);  //# of constraints,variables
                                          //# of priorities and both + & - allowed dev.
   mix = (nrow - bth_dev);
   mix = (mix + (bth_dev * 2));

  if ((nrow ==0)||(nvar ==0)||(nprt ==0)){
    printf("Variables, priorities and/or number of \
         constraints MUST be greater than zero\n");
    printf("Press any key to halt...");
    getch();
      goto end;
    }

    ncol = nrow + nvar;           //ncol = sum of constraints and decision vars
  for (row = 1; row <= nrow; row++){
   for (col = 1; col <= ncol; col++){
     basis[row][col] = 0.0;          /* initiallizing the basis array */
     index = col - nvar;
     if (index == row){
         basis[row][col] = 1.0;
         }
      }
     ind = row + ncol;
     ibasic[row] = ind;
  }

 for (col=1; col<=ncol; col++){
     jcol[col] = col;
    }

    kend =nprt + 1;  //indexing for artificial variables, P0

 //setting a kend * ncol array for kend goals
 for (k =1; k<=kend; k++){
   for (col = 1; col<=ncol; col++){
     valc[k][col] = 0.0;
       }
   }

//setting a kend * nrow array for kend goals
for(k=1; k<= kend; k++){
 for (row = 1; row <=nrow; row++){
     valb[k][row] = 0.0;
     }
 }

 ktest = 0;

// reading signs of deviational variables in each constraint
 for( row=1; row<=nrow; row++){
   fscanf(fp1,"%c ",&state[row]);
     }
//assigning allowed dev. signs in the basic and non-basic variable matrices
  for (row = 1; row<= nrow; row++){
     if ((state[row]=='E')||(state[row]=='e')){
       ktest = 1;
       index = row + nvar;
       valb[1][row] = 1.0;
       valc[1][index] = 1.0;    //setting the basic solution
       jcol[index] = 0;
       }
       else if((state[row] =='G')||(state[row]=='g')){
           ktest = 1;
           index = row + nvar;
           valc[1][index] = 1.0;
```

```
          jcol[index] =0;
          }
        else if((state[row] = ='L')||(state[row]= ='l')){
          ktest = 1;
          valb[1][row] = 1.0;
          }
         else if((state[row]= ='B')||(state[row]= ='b')){
          continue;
       }
     }


   if (ktest = = 1) nprt = nprt + 1;

// reading the constraint sign, number, priority and weight of deviational variable
   for (k=1; k< =mix; k++){
            fscanf(fp1,"%c%d%d%f\n",&sign,&row,&priority,&wgt);

       if (ktest = = 1){
        priority = priority + 1;    //add 1 for artificial variable
        }
       if (sign = = '-'){
        index = row  + nvar;
        valc[priority][index] = wgt;  //array of deviational weights
        continue;
       }
       else if(sign = = '+'){
          valb[priority][row] = wgt;  //deviational weights array
          continue;
        }
       else if((sign != '-')&&(sign != '+')){
          printf("Deviational variables must be either positive or negative\n");
          goto end;
        }
     }

// reading the xy co-ordinates and technoligical coefficient of decision variable
rr: do{
      fscanf(fp1,"%d%d%f\n",&row,&col,&coef);
      basis[row][col] = coef;
     } while (row !=0);

//reading the resources (RHS) for each constraints (set goals)

   for (row=1; row< =nrow; row++){
      fscanf(fp1,"%f",&prhs[row]);
      }
/* -------------------------- end of input data reading ------------------- */
// input data checking
 for (row=1; row< =nrow; row++){
      if (prhs[row] < 0){
```

```
     printf(" Enter only positive signed right side values:\
      Multiply expression by minus one \n");
         printf("Press any key to halt...");
         getch();
      goto end;
      }
     else if(prhs[row]>0){
         rhs[row] = - prhs[row];   /* setting all rhs -ve in initial tableau*/
        }
     else{
      rhs[row] = 0.000001;
        }
  }          /* end of loop of testing all RHS values >=0 */

  for (col=1; col<=ncol; col++){
    if( jcol[col] == 0){
      for (row=1; row<=nrow; row++){
      basis[row][col] = 0.0;
      }
     }
    else continue;
  }

/* initially set all the priorities to the lowest level */
        kend = nprt;
    for(j=1; j<=ncol; j++){
     jselect[j] = kend;
     }

/* selection of priorities  with non-zero weights */

    for(j=1; j<=ncol; j++){
     for(k=1; k<=nprt; k++){
        if(valc[k][j]<= 0.0000001) continue;
           else  jselect[j] = k;
     }
    }

  gotoxy(10,8); printf("\tCOMPUTING... Please wait.");
  gotoxy(10,10); printf("\tIteration No:");

begin:    pivt_row=0;         /* pivot row with pivot element */
       pivt_col=0;         /* pivot column with pivot element */
       lest_achv_pt=0;
    gotoxy(40,10);printf(" %d",iter);
    if(iter >4000) goto results;  //termination condition if loop exists

   for(i=1; i<=nrow; i++){
    jfail[i] =1;
      }
```

```
// selection of the highest unachieved priority for positive weights
// then go on to minimize this priority effect
   for(k=1; k<=nprt; k++){
     for (i=1; i<=nrow; i++){
          if(valb[k][i] <= 0.0000001) continue;
              else {
          lest_achv_pt = k;
            goto cont;
          }
     }
    }

/* selection of most negative rhs value and use this for
   exiting basis variable */

cont:  rmin = - 0.0000001;
    for(i=1; i<=nrow; i++){
      if (rhs[i] >= rmin){        /* rhs is > zero */
          continue;
         }
      if (jfail[i] == 0){
          continue;
          }
          pivt_row = i;
          rmin = rhs[i];
     }
/* ----------------------------------------------------------------- */
     if(pivt_row == 0){  /* all rhs values >=0 */
     goto cunt;
       }
/* selection of the pivot column thru identification of column with least
   impact to weighted deviation to Z_value = total absolute deviation */
    coef = 0.0000001;
   for (m=1; m<=kend; m++){
     1 = kend - m + 1;
       for(j=1; j<=ncol; j++){
        if(jcol[j] == 0) continue;
           if(jselect[j] < l) continue;
            if(basis[pivt_row][j] <= coef) continue;
               coef = basis[pivt_row][j];
                  pivt_col = j;
        }
    if (pivt_col > 0) goto simp;
   }
  jfail[pivt_row] = 0;  //initialize jfail to zero if fail to get a pivot col
  goto cont;
//-----------------------------------------------------------------------

cunt:  if (lest_achv_pt == 0){
        zano = 77;         /*flag for all objectives achieved */
        goto results;
        }
      else{
        huprio = lest_achv_pt;     /* huprio is highest unachieved priority */
        }

   for (k=lest_achv_pt; k<=nprt; k++){  //assessment of only yet unresolved priorities
       for (j=1; j<=ncol; j++){
        zval[k][j] = 0.0;

       if(jcol[j] == 0) continue;

       if(jselect[j] < huprio)    //jselect[] holds priorities. Here if < then
          continue;              //already been solved for. Prevents looping

        for(i=1; i<=nrow; i++){
          if(valb[k][i] <= 0.0000001)  //consider non-zero weights
             continue;

        if(fabs(basis[i][j]) <= 0.0000001) //consider non-zero tech. coefficients
           continue;

        // calculating the absolute total deviation element
          zval[k][j] += valb[k][i] * basis[i][j];
        }
      zval[k][j] += valc[k][j];
       }
    }
//---------------------------------------------------------------------{}ok

      zvalue = - 0.0000001;
      for(k = lest_achv_pt; k <= nprt; k++){
      for(j = 1; j <= ncol; j++){
        if(jcol[j] == 0)            //no priority for variable in col j.
          continue;
        if(jselect[j] < huprio)       //already been solved priority j.
           continue;
        if(zval[k][j] >= zvalue)    //Z col coef. >= 0 i.e. positive
          continue;

          if (k <= lest_achv_pt) goto www;
               //selection of a pivot column i.e. pivt_col
www:          zvalue = zval[k][j];
          pivt_col = j;
xx:  continue;
        } // continue to next column
 if(pivt_col > 0){
        goto yy;
      }
    else if(pivt_col <= 0){
        huprio = huprio + 1;
      }
```

```
  } //end of priority loop
//------------------------------------------------------------------------

   if(pivt_col== 0){          //no more pivot variables: solution found
       zano = 77;         // terminate iterations
       goto results;
   }
//--------------------------------------------------------------------
    yy:  theta = 1000000;

// determining which variable to exit the solution basis & find pivot row
   for(i=1; i<=nrow; i++){
    if(basis[i][pivt_col] >= -0.0000001)  //basis positive
        continue;

      if(rhs[i] <= - 0.0000001)  // want positive rhs values
         continue;

      if(rhs[i] <= 0.0000001)  //rhs approx 0, set a v. low value
        rhs[i] = 0.0000001;

        zeta = - rhs[i]/basis[i][pivt_col];   //take -rhs/technological coeff
        if(zeta >= theta)  //identifying non-basic to enter the basic sol.
          continue;
        theta = zeta;
        pivt_row = i;    //identifying the new pivot row for entering var
       }
//--------------------------------------------------------------------

     if(pivt_row <= 0){          //all positive deviations zero or negative
      zano = 77;          //implies solution found, terminates iterations
      goto results;
     }
//--------------------------------------------------------------------
    simp:  piv = basis[pivt_row][pivt_col]; //pivot element definition

   for(i=1; i<=nrow; i++){
     if(i == pivt_row) continue;

         if(fabs(basis[i][pivt_col]) <= 0.0000001) continue;  /* need non-zero basis[i][j] */

       if(fabs(rhs[pivt_row]) <= 0.0000001)  goto zz;     /* need positive rhs[] */
         rhs[i] -= (rhs[pivt_row] * basis[i][pivt_col])/piv;

zz:     for (j=1; j <= ncol; j++){
        if(j == pivt_col){
          continue;
          }
        if(fabs(basis[pivt_row][j]) <= 0.0000001) continue;

       // new elements in the tableau: old - (pd of diagonal)/piv
```

```
        basis[i][j] -= (basis[i][pivt_col] * basis[pivt_row][j])/piv;
       }

   basis[i][pivt_col] = basis[i][pivt_col]/piv; //new element in pivot column [5]
}
     //-----------------------------------------------------------------------
/* determining new elements in the pivot row: pivt_row; is by dividing by
   pivot element (piv) and taking the negative sign                    */

     if(fabs(rhs[pivt_row]) <= 0.0000001) goto pp;
     else  rhs[pivt_row] = - rhs[pivt_row]/piv;

pp: for (j=1; j <= ncol; j++){
     if(j == pivt_col) continue;
      if(fabs(basis[pivt_row][j]) <= 0.0000001) continue;
      basis[pivt_row][j] = - basis[pivt_row][j]/piv; //new pivot row elements
      }
    basis[pivt_row][pivt_col] =  1/piv;  //new  pivot element is reciprocal of old element[4a]
     //---------------------------------------------------------------------
// interchange of 'out-going' and 'in-coming' basic variables
   index = jcol[pivt_col];
    jcol[pivt_col] = ibasic[pivt_row];     //variables swapping positions
     ibasic[pivt_row] = index;

   for(k=1; k <= nprt; k++){                //consider non-zero weights
      dummy = valb[k][pivt_row];
      if(dummy >= 0.000001){
       jselect[pivt_col] = k;
      }
     valb[k][pivt_row] = valc[k][pivt_col]; //technological coeff of 'in' &
      valc[k][pivt_col] = dummy;            //'out' going variables swap positions
   }
//-----------------------------------------------------------------------
// test for termination of iterations
    if (ktest != 1){
     iter += 1;
     goto begin;
       }
    if (valc[1][pivt_col] == 0.0){
     iter += 1;
     goto begin;
       }
     jcol[pivt_col] = 0;          // re-initializing

     for(i=1; i <= nrow; i++){    // re-initializing the pivot column
      basis[i][pivt_col] = 0.0;
     }
     iter += 1;
     goto begin;
     // ---------- initializing arrays for outputs ----------
 results:   ;
```

```
        for (row = 1; row <= nrow; row++){
            pos_dv[row] = 0.0;        //positive deviations initialization
            neg_dv[row] = 0.0;        //negative deviations initialization
            }

        for (j =1; j <= nvar; j++){   // decision variables initialization
            array[j] = 0.0;
            }

        for (row = 1; row <= nrow; row++){
            var = ibasic[row];
            if(var > ncol){
            index = var - ncol;
            pos_dv[index] = rhs[row];  //setting positive deviational values
            continue;                  //for the row th constraint
            }
            if(var > nvar){
              index = var - nvar;
              neg_dv[index] = rhs[row]; //setting neg. deviational values for
              continue;                 //for the row th constraint
              }
            array[var] = rhs[row];
          }
        // ------------------ OUTPUT OF RESULTS TO FILE -------------------------------
          fprintf(fp2,"\t\tTHE GOAL PROGRAMMING SCHEDULE
            OUTPUT\n\t\t=====================================\n");
            fprintf(fp2,"\n\t\t\tShift Number: %s\n",shift);
          if (zano == 77){
              - fprintf(fp2,"\t\t**** All objectives are achieved. ****\n\n");
                }
            else  {
                fprintf(fp2,"\t\t**** Not all objectives were achieved. ****\n\n");
                }
                  fprintf(fp2,"\t\tDECISION VARIABLES\n\t\t-------------------\n");
          fprintf(fp2,"\t\tVariable \t\tValue \n");

          for (j = 1; j <=nvar; j++){        //printing the decision variables solution
              fprintf(fp2,"\t\tx%d \t\t\t%7.2f\n",j, array[j]);
            }

      fprintf(fp2,"\n\n\tTHE DEVIATIONS FROM SET
            GOALS\n\t================================\n");
          fprintf(fp2,"\n\tCONSTRAINT \tTARGET_VALUE \tACHIEVED_VALUE    GOAL DEVIATION
n");
           for (row = 1; row <=nrow; row++){
            if(pos_dv[row]==0){ target = prhs[row] - neg_dv[row];
                              diff = target - prhs[row];}
            else{ target = prhs[row] + pos_dv[row];
                diff = target - prhs[row];}
             fprintf(fp2,"\tROW %d\t\t%7.2f\t\t%7.2f\t\t%7.2f\n",row,prhs[row],target,diff);
             }
```

```
fprintf(fp2,"\n\n\tANALYSIS OF THE OBJECTIVE FUNCTION
      \n\t=====================================\n");
    fprintf(fp2,"\tPriority              Underachievement\n\t--------------------------------------\n");
      total = nprt + 1;
     for (k = 1; k <=nprt; k++){
      kval = total - k;        //selection of priority for deviational analysis
      net  = kval;
       if (ktest == 1){
         net = kval - 1;
         }
         zvalue = 0.0;

      for (row = 1; row <= nrow; row++){
          if (valb[kval][row] <= pow(10,-8)){
            continue;
            }
          if(fabs(rhs[row]) <= pow(10,-8)){
            continue;
            }
              // calculation of the absolute deviation for priority kval
            zvalue = zvalue + valb[kval][row]*rhs[row];
            continue;
           }

        if ((ktest==0)||(net>0)){
           fprintf(fp2,"\tP%d\t\t\t%7.2f\n",net,zvalue);
        }
      else{ fprintf(fp2,"\tArtificial:\t\t%7.2f",zvalue);
      if(zvalue != 0.0){
          fprintf(fp2,"\n\n ******* YOUR PROBLEM MODEL IS INFEASIBLE ******* \n");
          fprintf(fp2,"\nArtificial variable MUST be zero.\n\
  Adjust goal constraints:\n\ Add or drop some goals then re-run the model.\n");
         fprintf(fp2,"Value of artificial variable = extent of a constraint violation\n");
          }
        }
      continue;
      }

      fprintf(fp2,"\n\n\tProblem solved in  %d iterations\n\n",iter);

  end:  ;
       ended = clock();
     fprintf(fp2,"\tProcess time is  %f sec\n",difftime(ended, start)/CLK_TCK);
   return 0;
  }
```

Appendix C
Goal Program Input File Description

The format of the goal programming input data is sequentially described by the following components:

1. Title for the production schedule, e.g. date and shift number. Maximum of 80 characters are allowed all typed in one line.
2. Number of constraints: Enter the total number of constraints defining the system which includes both the system and goal constraints.
3. Enter the number of decision variables
4. Enter the number of priorities in the model, the minimum is number is one.
5. Enter the total number of constraints that have both their deviations in the objective function.
6. For all the constraints, enter the sign between the usage of resource (the left-hand side) and the resource (the right hand side) of the constraint. The allowed characters describing these signs are 'e' or 'E' for equality, 'g' or 'G' for greater than, 'l' or 'L' for less than, and 'b' or 'B' for both which represent those constraints with both deviations appearing in the objective function. These sign characters are entered in one line with a spacing between consecutive constraints. The format of this data card for a six constraint problem is of the form: b L e B G l or b l e b g l. Mixing capital and small letters is permitted.
7. For each constraint, the permissible direction of deviation is entered as '+' or '-' for positive and negative respectively. The sign is then followed by the constraint number, priority and weight in a format such as

   + 5 2 1.5

   where + = positive deviation is acceptable
   5 = constraint number 5
   2 = priority 2 for this constraint and
   1.5 = weight of 1.5 for this constraint
8. The technological coefficients of each non-zero decision variable in each constraint is entered by describing the constraint number followed by the column number of the decision variable and then the coefficient value. The format is of the form:

   6 21 244

   where 6 = constraint number 6
   21 = decision variable (equivalent to column number) 21 and
   244 = technological coefficient of variable 21 in constraint 6

   When all constraints' technological coefficients are entered, mark the end by a null decision variable containing zeros.

9. The last data card is the list of the resources. For each constraint enter its right-hand side value in the correct order i.e. if the constraint is number 10 then the right-hand side for this constraint falls in position ten. A typical format of this card is as follows:

   200 0.0 43 1099 0.045 ....

   These values must be zero or positive only. If a negative value exist, multiply the respective constraint by minus one to change the sign of the right-hand side. This requirement is necessary since the basis of goal program is to minimize or make the positive deviations from the target resources equal to zero

A typical data file used for the case example described in Chapter 5 is shown below. It has to be noted that the braced text in this data file is only meant for highlighting the start and end of the various data cards and must be deleted before the data file can be used in the computer goal program.

Table C1 Typical example of goal program input datafile

```
Production Shift Schedule   (Title card number 1)
24 9 5 2   (Cards numbers 2 to 5 consecutively)
g 1 b b 1 1 1 1 1 1 1 1 1 1 1 e 1 1 1 1 1 1 1 1 (Card # 6)
+ 1 5 1 (Card # 7 for permissible goal sign)
- 2 4 1
- 3 2 1
+ 3 2 1
- 4 3 1
+ 4 3 1
- 5 1 1
- 6 1 1
- 7 1 1
- 8 1 1
- 9 1 1
- 10 1 1
- 11 1 1
- 12 1 1
- 13 1 1
- 14 1 1
- 15 1 1
- 16 1 1
- 17 1 1
- 18 1 1
- 19 1 1
- 20 1 1
- 21 1 1
- 22 1 1
- 23 1 1
- 24 1
1 1 1(Card #8 for variable location and coefficient)
1 2 1
1 3 1
1 4 1
1 5 1
1 6 1
1 7 1
1 8 1
1 9 1
2 1 4
2 2 2
2 3 3
2 4 1.5
2 5 1.5
2 6 5.0
2 7 1.86
2 8 2.6
2 9 2.0
3 1 0.17
3 2 0.17
3 3 -0.13
3 4 -0.13
3 5 -1.63
3 6 -1.63
3 7 0.27
3 8 -1.73
4 1 -1.13
4 2 -1.13
4 3 0.47
```

[File continues to next column]

[File continuing in this column]

```
4 4 0.47
4 5 0.37
4 6 0.37
4 7 0.57
4 8 0.42
5 1 1
5 2 1
5 3 1
5 4 1
5 5 1
5 6 1
6 7 1
6 8 1
6 9 1
7 1 1
7 3 1
7 5 1
8 2 1
8 4 1
8 6 1
9 7 1
9 8 1
10 9 1
11 1 1
11 2 1
12 3 1
12 4 1
13 5 1
13 6 1
14 7 1
15 8 1
16 9 1
17 1 0.003
17 2 0.003
17 3 0.0023
17 4 0.0023
17 5 0.0026
17 6 0.0026
18 7 0.0021
18 8 0.00295
18 9 0.00226
19 1 1
20 2 1
21 3 1
22 4 1
23 5 1
24 6 1
0 0 0
2250 6200 0.0 0.0 1550 710 1500 1000 1000
1000 800 800 500 600 400 100 6 4 500 500 500
500 500 500 (End of resource card number 9)
```

Appendix D

Underground Active Dispatch and Control Model Listing

```
//------------------- UNDERGROUND ACTIVE DISPATCH MODEL ---------------
#include <dos.h>
#include <fstream.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <conio.h>
#include <time.h>
#include <ctype.h>
#include <graphics.h>
#define  ASIZE          5000
#define SPAN             5          // moving average interval
#define MID            20 * 60
#define MAX              5          //levels
#define STP             15          //stopes
#define BIN             10          //bins
#define MAC             20          //machines
#define LOWL            0.5
#define ALLOW            3
#define CLIP_ON          0          //graphical parameters
#define THICKNESS1         3
#define THICKNESS2         2.5
#define USER_PATTERN         4
#define XOFFSET          120
#define YOFFSET          130
#define XMAX              500
#define YMAX              380
#define BLOCKAGE         (0.5/(shift_end/smp_tim))
#define BBLOCKAGE        (0.08/(shift_end/smp_tim))
#define DELAY            60            //time lost due interference X-over

// schedule input variable declarations
int  level_stopes[MAX],level_bins[MAX];  //rule,#stopes,#bins on each level
int  bf_stcap[MAX][STP],buf_bin_cap[MAX][BIN];//buf capacities & # of mach/level
int  equ_assign[MAX][STP], bin_assign[MAX][BIN]; //assigned machines at each source or destn

int sumd;                               /* # of mach */
int lev[MAC];                             /* machine status & level location */
int sequnt[MAX][STP][MAC], bsequnt[MAX][BIN][MAC]; /*seq at source/dest */

// equipment characteristics (work type distributions)
float gamma_1[MAC],alpha_1[MAC],beta_1[MAC];  //loading parameters
float alpha_2[MAC], beta_2[MAC], gamma_2[MAC]; //travelling full parameters
float alpha_3[MAC], beta_3[MAC],gamma_3[MAC]; //dumping parameters
float gamma_4[MAC],alpha_4[MAC],beta_4[MAC]; //travelling empty parameters
float rate[MAC], cap[MAC];                    //rating of mach & parameters

// weights for grade-tonnage schedules
float d2[MAX];              //operation thresholds on targets
float man[25], minn=999;  //parameters returned by fn feed(), minimum machine capacity
float rank[MAX][STP] ={0};

//breakdown/blockage arrays
int sblock[MAX][STP], bblock[MAX][BIN]; //flag of blockages of source & destn resp
int KATE[MAX];                      //flags of job completion on levels

// haulage network data
 float dist[MAX][STP][BIN], VEL_1[MAX][STP][BIN], VEL_2[MAX][STP][BIN];
 float disc[MAX][STP], vect[MAX][STP];  //inter-level dist and speed resp.
 float metr[MAX][STP][BIN],velo[MAX][STP][BIN]; //inter-source on level dist & speed
 float rmpds[MAX], rmpv[MAX];
 int  route[MAX][STP][BIN];
 int  in_road[MAX][STP][BIN]; //allowed machines per road section

// statistics variables
float stope_grade[MAX][STP]; /* grade of source j on level i  */
float tot_stope_cap[MAX][STP];
float tot_bin_cap[MAX][BIN]; //scheduled capacities
float sch_ton[MAX][STP];
float fix_space[MAX][BIN]; //scheduled qty for source ij.

// queue arrays
char que[MAX][STP][MAC],  bque[MAX][BIN][MAC]; //order of assigned machines in queue
char mach_type[MAC], equp[30];      //machine model, mach input filename, driver names
char mat_flag[MAX][STP], bmat_flg[MAX][BIN]; //flag of source & destination materials resp.
char arrar[50];

/* ------------- graphical variables ----------------------------------*/
float plan_ton[16], tall_ton[16], tall_grd[16];
int bstx[MAX][BIN], bsty[MAX][BIN];   /* LHD's bin start co-ord */
int sstx[MAX][STP], ssty[MAX][STP];   /* LHD's stope start co-ord */
void far *buffer[MAC];

float far *EAT = (float *)malloc(sizeof(EAT));
float far *TEA = (float *)malloc(sizeof(TEA));
float far *bdown = (float *)malloc(sizeof(bdown));
float far *SERVE = (float *)malloc(sizeof(SERVE));
char far *ptr = (char *)malloc(sizeof(ptr));

int checkbuf(int space,int k,int n,int m, int z, int s, float &SERVE);

/* -------------------------------------------------------------------*/

float randomm()
{ /* random number generator */
  float ramber=0.0;
```

```
    randomize();
    for (int i = 0; i < 100; i++)
    ramber = (float) random(ASIZE);
    ramber = ramber/ASIZE;
    return ramber;
}

/* ------------------------------------------------------------------*/

 float weibull( float a, float b, float g)  // describe load/travel/dump distribution
{
   float temp=0.0,increment=0.0;

   temp = randomm();
   if (temp==0){ temp = 0.001;}
    temp = -log(temp);
     if(temp < 0) { temp = 0;}
      if (a!=0){
           temp = pow(temp, (1/a));
            increment = g + (b * temp);
           }
      else {
           printf("Error. Weibull -ve nos.\n"); //Non-negative alpha value
           getch();
           exit(EXIT_FAILURE);            //currently overwritten possible to 0
          }
  return increment;          //incremental factors added to deterministic qties
}
/* -----------------------------------------------------------------*/

float load_cap(int s, float under_load, float over_load)
{
 /*------- TRIANGULAR DISTRIBUTION OF FILL FACTORS ON LOADING -------- */
 float load=0.0, randx=0.0, peak = 1.0, lest=0, maxt=0, relpk = 0;

 randx = randomm();  //random generation of a load

 lest = peak - under_load;
 maxt = peak + over_load;
if((lest<0)||(maxt >2)) abort();
 relpk = (peak - lest)/(maxt-lest);

if(randx <= relpk){
     load = peak - lest;
     load = load * randx;
     load = load * (maxt - lest);
     load = sqrt(load);
     load = lest + load;
     }
else if(randx >relpk){
      load = 1.0 - randx;
```

```
        load = load * (maxt - peak);
        load = load * (maxt - lest);
        load = sqrt(load);
        load = maxt - load;
      }
        load = cap[s] * load;
 return load;   //weight put on machine in appropriate units
}
/* ---------------------------------------------------------------------*/

float runtime(int k, int p, int q, int z,float a, float b, float g, int s)
{      //calculates travel times between points
    float tr_tim=0.0, my=0.0;

  if (z==0){                    //travelling empty: Z=Flag of bin to source
         my = weibull(a,b,g);  //a is non_zero
        if(VEL_2[k][p][q] <=0){
              printf("VEL_2[][][] is zero in runtime()\n");
              exit(EXIT_FAILURE);
            }
          else tr_tim = dist[k][p][q]/(VEL_2[k][p][q] * rate[s]) + my;
         }
  else if (z==1){                        //Z=1 => source to bin travel full
            my = weibull(a,b,g);          //travelling full
         if(VEL_1[k][p][q]<=0){
            printf("VEL_1[][][] is zero in runtime()\n");
            exit(EXIT_FAILURE);
         }
        else tr_tim = dist[k][p][q]/(VEL_1[k][p][q]*rate[s]) + my;  //rate[c] non-zero
      }
   return tr_tim;  //return a time length to complete a journey
}
/* ---------------------------------------------------------------------*/

int factorial(int n)
{  //factorial function
    int result = 1;
    if (n > 0) {
    do {
        result *= n;
        --n;
        } while (n > 1);
      }
    else if (n < 0) {
        printf("Factorial Argument is negative \n");
        exit(EXIT_FAILURE);
        }
  return result;   //return an integer for weibull mean fn calc
}
/* ---------------------------------------------------------------------*/
```

```
float mean_load(int s,int z)    //s is machine number, z=> location (source/dest)
{  //calculates load or dump times of queuing machines
 int zee = 0, kk = 0;
 float ave_load = 0, randx=0, theta = 0, GG=0,AA=0,BB=0;

 if (z==0){   /* use parameters for loading */
       GG=gamma_1[s];AA=alpha_1[s]; BB=beta_1[s];
       }
  else if (z==1){ /* use parameters for dumping */
        GG=gamma_3[s];AA=alpha_3[s];BB=beta_3[s];
    }

 if(BB != 0){
        theta = (1 + 1/BB) - 1;
        zee = (int) theta;       // casting theta to int zee
        kk = factorial(zee);  //calling factorial function in Weibull fn mean
         randx = randomm();
          ave_load = GG + AA * kk;
      if(randx <=0.5){
          ave_load = ave_load - (ave_load - (LOWL*ave_load))*(randx/0.5);
         }
       else if(randx >0.5){
           ave_load = ave_load + ((1+(1-LOWL))*ave_load -ave_load)*(1 - randx)/0.5;
        }
       }
  else {
       printf("Beta[%d] value is undefined\n",s);  //Non-zero beta value
      }
 return ave_load;    //returns time to execute a job
}
/* ---------------------------------------------------------------*/

void arang(int k, int i, int zum, int z, char ch)
{ //arranges the queue if machine removed: Queue by Type
 int n=0;

  switch(z){
    case 1: for (n=0; n < zum; n++){        //random access of departing machine
               if(que[k][i][n] == ch )  que[k][i][n] =NULL;
           }
           for (n=0; n < zum; n++){
               if(que[k][i][n] == NULL){
                que[k][i][n] = que[k][i][n+1];
                que[k][i][n+1] = NULL;
                }
             }
           break;
    case 0: for (n=0; n < zum; n++){        //random access from queue of machine
               if(bque[k][i][n] == ch) bque[k][i][n] =NULL;
           }
           for (n=0; n < zum; n++){
                if (bque[k][i][n] == NULL){
                bque[k][i][n] = bque[k][i][n+1];
                bque[k][i][n+1] = NULL;
                }
              }
            break;
  }
 return;
}
//=================================================

void range(int k, int i, int zum, int z, int s)
{   //arranges the queue if machine removed: Queue by Machine No.
 int n=0;

   switch(z){
     case 1: for (n=0; n<zum; n++){          //delete.head() of stope queues
                if (sequnt[k][i][n] == s) sequnt[k][i][n] = -1;
             }
             for (n=0; n< zum; n++){
                 if (sequnt[k][i][n] == -1){
                    sequnt[k][i][n] = sequnt[k][i][n+1];
                    sequnt[k][i][n+1] = -1;
                 }
             }
            break;
     case 0: for (n=0; n<zum; n++){          //delete.head() of bin queues
                if (bsequnt[k][i][n] == s) bsequnt[k][i][n] = -1;
             }
            for (n=0; n< zum; n++){
                if (bsequnt[k][i][n] == -1){
                bsequnt[k][i][n] = bsequnt[k][i][n+1];
                bsequnt[k][i][n+1] = -1;
                }
            }
            break;
  }
 return;
}
/* ------------------------------------------------------------------*/

float ques(int k, int i, int z)
{     //gives earliest service for expected new client in queue
      //k = level, i = source or dest on level k, z = 0 or 1
  int n=0;       float summ=0.0;

  switch (z){
      case 0: for (n=0; n<equ_assign[k][i];n++){  //stope char array of machines
                 summ += mean_load(sequnt[k][i][n],z); //find average service time of Q
                }
            break;
      case 1: for (n=0; n<bin_assign[k][i];n++){    // array of machines @ bin
                  summ += mean_load(bsequnt[k][i][n],z); //find service time of Q
                  }
              break;
   }
 return summ;   //time of servicing the clients in queue i.e XX_assign[][]
}
/* -------------------------------------------------------------------*/

int adjunt(int k,int clock,float &TEA, int s, int shift_end, int levels)
{ /* determine where to send client {NEXT} and the time to get there {TIMEX} */
 int next=0; float timex=0;

 if ((KATE[k-1]!=99)&&((k-1)!=-1)&&(KATE[k+1]!=99)&&(k+1<levels)){
    if (clock <(int)(LOWL * shift_end)){         //criteria for routing to next level
       next=(d2[k-1]>d2[k+1]) ? k+1 : k-1;        //production & timing basis
       timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]); //call incremental delay for travel
       if((rmpv[next]!=0)&&(next==k-1))
        timex = timex + rmpds[next]/(rmpv[next]*rate[s]*0.9);  //add to deterministic time
        if((rmpv[next]!=0)&&(next==k+1))
        timex = timex + rmpds[next]/(rmpv[next]*rate[s]*1.1);
       }
   else{ next=((rmpds[k-1]/rmpv[k-1])>(rmpds[k+1]/rmpv[k+1])) ? k+1 :k-1;
         timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]); //call incremental delay for travel
           if((rmpv[next]!=0)&&(next==k-1))
              timex = timex + rmpds[next]/(rmpv[next]*rate[s]*0.9);
           if((rmpv[next]!=0)&&(next==k+1))
              timex = timex + rmpds[next]/(rmpv[next]*rate[s]*1.1);
         }
     }
 else if ((KATE[k-1]!=99)&&((k-1)!=-1)){ //shortest travel time as close
           next = (k-1);                   //to end of shift
           timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]); //call incremental delay for travel
           if(rmpv[next]!=0) timex = timex + rmpds[k]/(rmpv[k]*rate[s]*0.9);        //add to
deterministic time
            }
   else if((KATE[k+1]!=99)&&((k+1)<levels)){
         next = k+1;
         timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]);  //call incremental delay for travel
         if(rmpv[next]!=0)
         timex = timex + rmpds[next]/(rmpv[next]*rate[s]*1.1);   //add to deterministic time
      }
   else{ timex = 0;
         next = 155;     //work done on either levels
       }
 TEA = timex;
return next;
}
/* ------------------------------------------------------------------- */
int validate(int next, int &waste_okay)
{ int i = 0, ore_okay = 0, zip = 0;
```

```
 for (i = 0; i < level_bins[next]; i++){
   if((bblock[next][i]==1)&&(bmat_flg[next][i] == 'o'))
       ore_okay += 1;
   else if((bblock[next][i]==1)&&(bmat_flg[next][i] == 'w'))
       zip += 1;
  }
  waste_okay = zip;
  return ore_okay;
}
/* ---------------------------------------------------------------------*/

int to_assign(int next,int machines)
{ int i=0, okay=0, ore_okay=0, wte_okay=0, *waste_okay=0;
  float sum = 0;

  ore_okay = validate(next, *waste_okay);  //count bins unblocked of each type
   wte_okay = *waste_okay;
    for(i=0; i<level_stopes[next]; i++){
      if((sblock[next][i]==1)&&(mat_flag[next][i]=='o')&&(ore_okay>=1))
           sum += tot_stope_cap[next][i];
      else if((sblock[next][i]==1)&&(mat_flag[next][i]=='w')&&(wte_okay>=1))
           sum += tot_stope_cap[next][i];
           }
    if((sum/((0.01+machines)*minn))>ALLOW) okay = 1;   //assign extra lhds
    else okay = 0;                                      //no extra lhds
 return okay;
}
/* ---------------------------------------------------------------------*/

  int dump_state(int k, char flag)
  { /* prevent loading if no dump space available */
   int bin=0, count=0;
    for (bin = 0; bin < level_bins[k]; bin++){
      if((flag == bmat_flg[k][bin])&&(tot_bin_cap[k][bin]-\
          bin_assign[k][bin]*minn > minn)&&(bblock[k][bin]!=-1)){
           count += 1;                     /* space available */
         }
      }
    return count;
 }
/* ---------------------------------------------------------------------*/

int true(int k, int z)
{   /* assesses if other machines still small enough to do job left */
int i=0, j=0, kiss=0, kount=0;
switch(z){
     case 0: for (i=0; i<sumd; i++){
              if(lev[i]==k){
               for (j=0; j<level_stopes[k]; j++){
                 if (tot_stope_cap[k][j] < minn){  //stope depleted for all machs
                    kiss = 99;                  //flag for all work done
```

```
                }
                else kount +=1;          //flag of undepleted stopes wrt some machs
              }
            }
          }     //end_for sumd
          if(kount !=0) kiss = 66;
          else kiss = 99;
          break;
     case 1: for (i=0; i<sumd; i++){
            if(lev[i]==k){
            for (j=0; j<level_bins[k]; j++){
              if (tot_bin_cap[k][j] < minn){  //bins full for all machs cap.
                kiss = 99;                      //flag all work done
                }
              else kount += 1;        //selecting smaller machine for remaining job
            }                       //end_for
          }
          }                         //end_for_sumd
          if(kount !=0) kiss = 66;
          else kiss = 99;
         break;
   }
 return kiss;        //assess availability of work/space at sites on section
}
/*----------------------------------------------------------------------*/

int break_down(float MTBF, float MRT, float time,int status, float &bdown)
{
 int down=0;
 float randx=0, temp=0, total=0;

 MRT = 1.0/MRT;                  /* mean service rate */
 total = - time/MTBF;
 total = exp(total);           /* cumulative probability of a failure */
 if (status != 1){             /* checks earlier breakdown in shift of same LHD*/
    randx = randomm();
    delay(20);                /* prevent sampling same random # due to fast CPU */
 if(randx > total){ down = 1;   /* machine broken */
   randx = randomm();
   if (randx > MRT) {  bdown = 0;
            down = 0;
           }
    else { temp = randx/MRT;
           temp = log(temp);
           temp = -(temp/MRT);
           bdown = temp * 3600;
         }
     }
   }
else {
    bdown = 0;
    down = 0;        //flag machine OK
   }
 return down;     //machine gone on breakdown or not
}
/* -----------------------------------------------------------------------*/

int route1(int z, int s, int k, int p,char zita, float &EAT)
{   //Maximize contained metal relative to priorities
  float  ss=0, keep=0, slack=0, prio=0, test=0, loaded=9999999;
  int i=0, j=0, muza=0, kount=0, koot=0, kiss=-1, space=0;

    switch(z){
      case 0:for(i=0; i<level_stopes[k]; i++){        //polling each server
            if ((sblock[k][i]==1)&&(tot_stope_cap[k][i]-\
              equ_assign[k][i]*minn >= cap[s])&&(dump_state(k,mat_flag[k][i])>0)){
              space = bf_stcap[k][i] - equ_assign[k][i];
              muza = checkbuf(space,k,i,p,z,s, *SERVE); //check if space at site ki

            if ((muza ==1)&&(route[k][i][p]>in_road[k][i][p])){                    //if true
continue

             if(stope_grade[k][i]!=0) keep = (1.0+stope_grade[k][i]) * tot_stope_cap[k][i];
             else keep = tot_stope_cap[k][i];
               test = (tot_stope_cap[k][i]* rank[k][i])/sch_ton[k][i];
               prio = keep * test;
              if  (prio > slack){
                  slack = prio;
                  kiss = i;
                 }
               }
            else  {                         //destn mach space inavailable
                   koot += 1;                  // counter of sources
                if(kiss== -1) kiss = 999;
                if (koot ==level_stopes[k]){
                   kiss = 999;             //flag for over capacity
                  }
                  continue;
                }
              }                           //end_if cap
        else{                             //mach>>large or destn blocked
            kount +=1;                    //counter of sites
            if (kount == level_stopes[k]){
               kiss=true(k,z);            //check if small machine can do job
              }
           }
         }                 //end_for test

      if ((kiss != 99)&&(kiss!=66)&&(kiss!=999)){   //find travel time = ss
         ss = runtime(k, kiss,p,z,alpha_4[s],beta_4[s],gamma_4[s],s);
         ss = ss + DELAY * in_road[k][kiss][p];
         EAT=ss;
```

```
          }
        else {ss = 0;           //else can't travel: over_cap / jobs finished
            EAT = ss;           //or too large for job left.
          }
       break;

   case 1: for (j=0; j<level_bins[k]; j++){  // fixed stope, var bins: row X col
        if((bblock[k][j]==1)&&(bmat_flg[k][j]==zita)&&\
          (tot_bin_cap[k][j]-bin_assign[k][j]*minn > cap[s])){
            space = buf_bin_cap[k][j] - bin_assign[k][j];
              muza = checkbuf(space,k,p,j,s,z,*SERVE); //check if space at site kj
          if ((muza == 1)&&(route[k][p][j]>in_road[k][p][j])){
                 prio = *SERVE;              //service time of total queue
                if(prio < loaded ){
                   loaded = prio;         //selecting earliest service site
                   kiss = j;             //flag of best site so far
                  }
                }
               else {koot += 1;    //mach space inavailable
                  if(kiss== -1) kiss = 999;
                  if (koot==level_bins[k]){
                     kiss = 999;          //flag over_capacity
                  }
                }
           }                     //end_if cap
        else{                    //mach>>large or destn blocked
            kount +=1;
            if (kount == level_bins[k]){
              kiss=true(k,z);  //return 66 or 99: analyze job status
            }
          }
        }                      //end_for test

   if ((kiss !=99)&&(kiss !=66)&&(kiss !=999)){ //travel time to best site
       ss = runtime(k,p,kiss,z,alpha_2[s],beta_2[s],gamma_2[s],s);
       ss = ss + DELAY * in_road[k][p][kiss];
       }
    else {ss = 0;             //no travel here
        }
        EAT = ss;
        break;
       }                    //end_switch
 return kiss;            //new destination assignment and time of travel EAT
}
/* ----------------------------------------------------------------------*/

int route2(int z, int s, int k, int p,char zita, float &EAT)
{         //Min-SLACK considers both travel time & destination service rates
   int i=0, kiss=-1,muza=0, kount=0, koot=0, space=0;
   float wwe=0, nn=0, mm=0,test=0, prio=0, slack=0, loaded=9999999;
```

```
switch (z){
  case 0: for(i=0; i<level_stopes[k]; i++){
          if ((sblock[k][i]==1)&&(tot_stope_cap[k][i]-\
             equ_assign[k][i]*minn > cap[s])&&(dump_state(k,mat_flag[k][i])>0)){
              space = bf_stcap[k][i] - equ_assign[k][i];
              muza = checkbuf(space,k,i,p,z,s,*SERVE);  //check queue space at ki

             if ((muza == 1)&&(route[k][i][p] > in_road[k][i][p])){
                nn = *SERVE;           //find total service time at site
                mm = runtime(k,i,p,z,alpha_4[s],beta_4[s],gamma_4[s],s);
                mm = mm + DELAY * in_road[k][i][p];
                slack = fabs(nn - mm);
                test = sch_ton[k][i]/(tot_stope_cap[k][i]* rank[k][i]);
                prio = slack *  test;

              if (prio < loaded){
                 loaded = prio;      //goto minimum slack destination
                 wwe  = mm;
                 kiss = i;           //store that best site
                 }
                }
            else {         //no machine space at destn
                koot +=1;
             if(kiss== -1) kiss = 999;
             if (koot == level_stopes[k]){
                kiss = 999;  //flag over-capicity
                wwe = 0;
               }
             }
           }              //end-if cap
     else{                 //mach>>large or dest blocked
          kount +=1;
         if (kount == level_stopes[k]){
            kiss=true(k,z);  //job status check; return 66 or 99
            wwe = 0;
            }
          }
        }             //end_for test
        EAT = wwe;
        break;

  case 1: for( int j = 0; j<level_bins[k]; j++){
     if((bblock[k][j]==1)&&(bmat_flg[k][j]==zita)&&\
        (tot_bin_cap[k][j]-bin_assign[k][j]*minn > cap[s])){
             space = buf_bin_cap[k][j] - bin_assign[k][j];
             muza = checkbuf(space,k,p,j,z,s,*SERVE);  //check queue space at kj

         if ((muza ==1)&&(route[k][p][j] > in_road[k][p][j])){
              nn = *SERVE;                //total service time at site
              mm = runtime(k,p,j,z,alpha_2[s],beta_2[s],gamma_2[s],s);
              mm = mm + DELAY * in_road[k][p][j];
              slack = fabs(nn - mm);

             if (slack < loaded){         //determine min slack for destn
                  loaded = slack;
                  wwe  = mm;           //store travel time
                  kiss = j;          //store site number
                  }
                 }
             else { koot +=1;  //returns a temp. space incapacity
              if(kiss== -1) kiss = 999;
              if (koot == level_bins[k]){
                kiss = 999;            //flag over_capacity
                wwe = 0;               //no travel
               }
              }
             }                //end_if cap
       else{                       //bins full or mach cap>>large
           kount +=1;
           if (kount == level_bins[k]){
             wwe = 0;
             kiss=true(k,z);      //assess job status & return 66 or 99
             }
           }
         }                    //end_for of cap
      EAT = wwe;
     break;
    }                      //end_switch
  return kiss;              //new destination assignment and time of travel EAT
}
/* ---------------------------------------------------------------------*/

int route3(int z, int s, int k, int p, char zita,float &EAT)
{  // EEST only considers earliest service.
   // Travel times not considered explicitely
  float mm = 0, nn = 0, loaded = 9999999, prio=0, temp=0;
  int i=0, j=0, muza=0, kount=0, koot=0, kiss=-1, space=0, count=0;

   switch(z){  //ok 30/5/95
      case 0: for(i=0; i<level_stopes[k]; i++){ //selecting approp loading
            if((sblock[k][i]==1)&&((tot_stope_cap[k][i]-\
               equ_assign[k][i]*minn) > cap[s])&&(dump_state(k,mat_flag[k][i])>0)){
               space = bf_stcap[k][i] - equ_assign[k][i];
               muza = checkbuf(space,k,i,p,z,s,*SERVE); //check queue space at ki

           if ((muza ==1)&&(route[k][i][p] > in_road[k][i][p])){
                  nn = *SERVE;              //service time of total queue
             prio = nn * sch_ton[k][i]/(tot_stope_cap[k][i]* rank[k][i]);

                if ( prio <=loaded ){
                  count += 1;
                  if((i!=0)&&(count > 1)){
                    temp = randomm();
                    if(temp > 0.5){          //random selection of server
                      loaded = prio;
                      kiss = i;
                      }
                    else continue;
                  }
                else {
                    loaded = prio;             //select minimum service time
                    kiss = i;              //keep best site
                 }
                }
              }
           else{                          //no machine space
               koot +=1;                       //sites counter
               if(kiss == -1) kiss = 999;
               if (koot == level_stopes[k]){
                  kiss=999;                  //over-capacity
                  }
                }
              }                           //end_if cap
           else{                          //when cap is not enough or blocked
               kount +=1;
               if (kount == level_stopes[k]){
                  kiss=true(k,z); //assess job status & return 66 or 99
                  }
                }
              }                          //end_for

      if((kiss != 99)&&(kiss != 66)&&(kiss != 999)){   //travel time to site
         mm = runtime(k,kiss,p,z,alpha_4[s],beta_4[s],gamma_4[s],s);
         mm = mm + DELAY * in_road[k][kiss][p];
        }
       else mm =0;                         //no travel otherwise
      EAT = mm;
      break;

case 1: for(j=0; j<level_bins[k]; j++){ //selection of approp destination
        if((bblock[k][j]==1)&&(bmat_flg[k][j]==zita)&&\
          (tot_bin_cap[k][j]-bin_assign[k][j]*minn > cap[s])){
              space = buf_bin_cap[k][j] - bin_assign[k][j];
              muza = checkbuf(space,k,p,j,z,s,*SERVE);  //check queue space at kj

          if ((muza ==1)&&(route[k][p][j] > in_road[k][p][j])){
           nn = *SERVE;                   //service time of total queue
          if (nn<=loaded ){
           count += 1;
          if((j!=0)&&(count > 1)){
              temp = randomm();
            if(temp > 0.5){          //random selection of server
              loaded = nn;
              kiss = j;
              }
            else continue;
            }
           else {                        // nn < loaded & count == 1
             loaded = nn;                 //select minimum service time
             kiss = j;                    //store best destn site
             }
             }
            }
        else {                          //temporal over-capacity
              koot += 1;                    //counter of sites polled
              if (kiss == -1) kiss = 999;
           if (koot == level_bins[k]){
              kiss=999;                  //over capacity at all sites
              }
             }
            }                            //endif for destn
        else {                          //bins full or mach_cap >>large
             kount += 1;
           if (kount == level_bins[k]){
              kiss=true(k,z);   //assess job status & return 66 or 99
            }
           }
          }                  //end_for
      if((kiss !=99)&&(kiss !=66)&&(kiss!=999)){ //for destn find the travel time
        mm = runtime(k,p,kiss,z,alpha_2[s],beta_2[s],gamma_2[s],s);
        mm = mm + DELAY * in_road[k][p][kiss];
      -}
    else mm = 0;      //no travel otherwise
      EAT = mm;
     break;
  }                   //end switch
 return kiss;           //new destination assignment and time of travel EAT
}
/* -----------------------------------------------------------------*/

int route4(int z, int s, int k, int p, char zita, float &EAT)
{   //STT consider travel times & no destination service rates
   float mm=0, prio=0, loaded=0, test=9999999;
   int   i=0, j=0, muza=0, kiss=-1, koot=0, kount=0, space=0;

  switch (z){
       case 0:
            for(i=0; i<level_stopes[k]; i++){
            if ((sblock[k][i]==1)&&(tot_stope_cap[k][i]-\
              equ_assign[k][i]*minn > cap[s])&&(dump_state(k,mat_flag[k][i])>0)){
             space = bf_stcap[k][i] - equ_assign[k][i];
              muza = checkbuf(space,k,i,p,z,s,*SERVE);   //check if space at site ki

               if ((muza ==1)&&(route[k][i][p] > in_road[k][i][p])){
```

```
          mm = runtime(k,i,p,z,alpha_4[s],beta_4[s],gamma_4[s],s);
          mm = mm + DELAY * in_road[k][i][p];
           prio = mm  * (1.0/rank[k][i]) * (sch_ton[k][i]/tot_stope_cap[k][i]);
           if (prio < test){
               test = prio;
               loaded = mm;              //select min travel time
               kiss = i;              //store that site
              }
            }
        else {                        //mach space inavailable
              koot +=1;                  //counter of sites read
              if(kiss== -1) kiss = 999;
              if (koot == level_stopes[k]){
                 loaded = 0;
                 kiss = 999;            //flag of overcapacity at sites
              }
            }
          }                  //end_if cap
      else {                  //all stopes depleted or mach too large
              kount +=1;
              if(kount == level_stopes[k]){
                 loaded = 0;
                 kiss = true(k,z);  //assess job status & return 66 or 99
              }
            }
        }                    //end_for test
    EAT = loaded;
    break;

case 1:
      for(j=0; j<level_bins[k]; j++){
          if((bblock[k][j]==1)&&(bmat_flg[k][j]==zita)&&\
            (tot_bin_cap[k][j]-bin_assign[k][j]*minn > cap[s])){
        space = buf_bin_cap[k][j] - bin_assign[k][i];
          muza = checkbuf(space,k,p,j,z,s,*SERVE);  //check if space at site kj
          if ((muza ==1)&&(route[k][p][j] > in_road[k][p][j])){
              mm = runtime( k,p,j,z,alpha_2[s],beta_2[s],gamma_2[s],s );
              mm = mm + DELAY * in_road[k][p][j];
              prio = mm;
              if (prio < test){
                  test = prio;
                  loaded = mm;         //min travel time found
                  kiss = j;           //best site stored
                }
              }
           else {
                  koot +=1;          //temp machine over-capacity
                if(kiss == -1) kiss = 999;
                if (koot == level_bins[k]){
                    loaded = 0;
                    kiss =999;         //sites over-capacity
                    }
                  }
                }                     //end_if cap
            else {                  //all bins are full or machine too large
                    kount +=1;
                if (kount == level_bins[k]){
                    kiss = true(k,z);   //assess job status & return 66 or 99
                    loaded = 0;
                  }
                }
              }                     //end_for destn test
         EAT =loaded;
        break;
      }            //end_switch
return kiss;          //new destination assignment and time of travel EAT
}
/* ------------------------------------------------------------------*/

int route5(int z, int s, int k, int p, char zita, float &EAT)
{ //Tonnage-grade-Min_Slack utilities: service & travel times assessed
  int i=0, j=0, kiss=-1,space=0, muza=0, koot=0, kount=0;
  float wwe=0, prod=0, keep=0, loaded=9999999;
  float mm=0, nn=0, slack=0, goal=0, d_fact=0;

   switch (z){
     case 0: for(i=0; i<level_stopes[k]; i++){
            if ((sblock[k][i]==1)&&(tot_stope_cap[k][i]-\
              equ_assign[k][i]*minn > cap[s])&&(dump_state(k,mat_flag[k][i])>0)){
              space = bf_stcap[k][i] - equ_assign[k][i];
              muza = checkbuf(space,k,i,p,z,s,*SERVE); //check if space at site ki
              if((muza ==1)&&(route[k][i][p] > in_road[k][i][p])){
                nn = *SERVE;                  //service time found
                mm = runtime(k,i,p,z,alpha_4[s],beta_4[s],gamma_4[s],s);  //travel time
                mm = mm + DELAY * in_road[k][i][p];
                slack = fabs(nn - mm);
                slack = slack + 1;
          if(stope_grade[k][i]!=0) keep = (1.0+stope_grade[k][i]) * tot_stope_cap[k][i];
          else keep = tot_stope_cap[k][i];

          if(dist[k][i][p]<=50) d_fact = 2.0;
          else if((dist[k][i][p]>50)&&(dist[k][i][p]<=100)) d_fact = 1.5;
          else if((dist[k][i][p]>100)&&(dist[k][i][p]<=150)) d_fact = 1.0;
          else if(dist[k][i][p]>150) d_fact = 0.5;
          prod = keep * d_fact * (1.0/slack) * (tot_stope_cap[k][i]*rank[k][i]/sch_ton[k][i]);
//product utility of variables
                if (prod > goal){
                   goal = prod;           //select max utility value
                   wwe  = mm;               //keep travel time to site
                   kiss = i;             //keep this best site
                  }
                }
          else {                          //mach-space inavailable
               koot +=1;                    //site counter
            if(kiss== -1) kiss = 999;
            if (koot == level_stopes[k]){
               kiss = 999;                //sites over-capacitated
               wwe = 0;
               }
             }
           }                          //end_if space test
     else{                            //mach>>large or destn blocked
          kount +=1;                    //site counter of blocked, depleted or 66
          if (kount == level_stopes[k]){
             kiss=true(k,z);          //assess job status & return 66 or 99
            }
          }
        }                             //end_for destn test
   EAT = wwe;
   break;

case 1: for (j=0; j<level_bins[k]; j++){   //dispatch based on min slack time
     if((bblock[k][j]==1)&&(bmat_flg[k][j]==zita)&&\
       (tot_bin_cap[k][j]-bin_assign[k][j]*minn > cap[s])){
         space = buf_bin_cap[k][j] - bin_assign[k][j];
         muza = checkbuf(space,k,p,j,z,s,*SERVE);     //check if space at site kj
         if((muza ==1)&&(route[k][p][j] > in_road[k][p][j])){
          nn = *SERVE;                       //  total service time of total queue at destination
          mm = runtime(k,p,j,z,alpha_2[s],beta_2[s],gamma_2[s],s);
          mm = mm + DELAY * in_road[k][p][j];
          slack = fabs(nn - mm);
          prod = slack;
        if(prod<=loaded){
           loaded = prod;
            kiss = j;
            wwe = mm;
            }
           }
        else {                    //mach space inavailable
             koot += 1;
             if(kiss== -1) kiss = 999;
             if (koot == level_bins[k]){
             kiss = 999;
             wwe = 0;
           }
          }
         }                    //end_if cap
     else{                    //mach>>large or destn blocked
         kount +=1;
         if (kount == level_bins[k]){
           kiss=true(k,z);       //assess job status & return 66 or 99
           }
          }
```

```
            }                       //end_for destn test
        EAT = wwe;
        break;
        }               //end-switch
   return kiss;        //new destination assignment and time of travel EAT
}
/* -----------------------------------------------------------------*/

int route6(int z, int s, int k, int p, char zita, float &EAT)
{  //Maximum remaining work: expediting using the critical ratio
   //Maximize the production from stopes with largest work remaining
  int i=0, j=0, kiss=-1, kount=0, koot=0, muza=0, space=0, count=0;
  float mm=0, nn=0, ss=0, goal=0, wwe=0, loaded=999, temp=0;
  float keep = 0, prio = 0, test=0;
   switch (z){
     case 0: for(i=0; i<level_stopes[k]; i++){
             if ((sblock[k][i]==1)&&(tot_stope_cap[k][i]-\
                 equ_assign[k][i]*minn > cap[s])&&(dump_state(k,mat_flag[k][i])>0)){
               space = bf_stcap[k][i] - equ_assign[k][i];
               muza = checkbuf(space,k,i,p,z,s,*SERVE);      //check if space at site ki
               if((muza ==1)&&(route[k][i][p] > in_road[k][i][p])){      //find travel time
                  mm = runtime(k,i,p,z,alpha_4[s],beta_4[s],gamma_4[s],s);
                  mm = mm + DELAY * in_road[k][i][p];
                  keep = tot_stope_cap[k][i]/sch_ton[k][i] * rank[k][i];
                  test = (tot_stope_cap[k][i]-equ_assign[k][i]*minn)/sch_ton[k][i];
                  prio = test * keep;
              if (prio >= goal){
                  goal = prio;                    //select site
                   kiss = i;                     //with largest goal and keep it
                    wwe = mm;                       //store its travel time
                   }
                }
             else {                  //mach space inavailable
                  koot += 1;      //site counter
                  if(kiss == -1) kiss = 999;
                  if (koot == level_stopes[k]){
                      kiss = 999;   //over-capacity
                   }
                }
              }                //end_if cap
        else{                  //mach >>large or blocked destn
             kount +=1;
             if (kount == level_stopes[k]){
                kiss=true(k,z);   //assess job status & return 66 or 99
                wwe = 0;
                }
              }
            }                //end_for test
        EAT = wwe;
        break;
   case 1: for (j=0; j<level_bins[k]; j++){  // select bin on shortest service time
        if((bblock[k][j]==1)&&(bmat_flg[k][j]==zita)&&\
          (tot_bin_cap[k][j]-bin_assign[k][j]*minn > cap[s])){
           space = buf_bin_cap[k][j] - bin_assign[k][j];
            muza = checkbuf(space,k,p,j,z,s,*SERVE);    //check if space at site kj
           if((muza ==1)&&(route[k][p][j] > in_road[k][p][j])){
             nn = *SERVE;          //service time of total queue
              prio = nn;
           if ( prio <=loaded ){
                count += 1;
             if((j!=0)&&(count > 1)){
                     temp = randomm();
                   if(temp > 0.5){          //random selection of server
                     loaded = prio;
                     kiss = j;
                    }
                   else continue;
                  }
             else {                          // count == 1 and j == 0;
                loaded = prio;                //select minimum service time
                kiss = j;                    //keep best site
                }
              }
            }          //end of if muza
        else {                  //mach cap inavailable
              koot += 1;        //counter of sites assessed
            if(kiss== -1) kiss = 999;
            if(koot == level_bins[k]){
              kiss = 999;         //over_capacity
              wwe = 0;
              }
            }                   //end else muza
          }                     //end_if cap
    else{                      //blocked/not rt material/no cap
        kount +=1;              //counter of sites assessed
        if (kount == level_bins[k]){
            kiss=true(k,z);     //assess job status & return 66 or 99
           }
        }
      }                   //end_for test

  if ((kiss !=99)&&(kiss!=66)&&(kiss!=999)){  //travel time generation
        ss = runtime(k,p,kiss,z,alpha_2[s],beta_2[s],gamma_2[s],s);
        ss = ss + DELAY * in_road[k][p][kiss];
        }
    else ss = 0;      //no travel
         EAT = ss;
         break;
       }           //end_switch
  return kiss;         //new destination assignment and time of travel EAT
}
```

```
/* ------------------------------------------------------------------*/

int checkbuf( int space,int k,int n,int m, int z, int s, float &SERVE )
{   /* checks if assignment to a point has no space problem  */
  int  buf_status = 0, xex = 0;
  float mm = 0.0, sume = 0.0, AA = 0.0, BB = 0.0, GG = 0.0;

if(z==0){ xex = n;
        AA = alpha_4[s]; BB = beta_4[s]; GG = gamma_4[s];
       }
  else{xex = m;
      AA = alpha_2[s]; BB = beta_2[s]; GG = gamma_2[s];
      }
if (space >=0){
     buf_status = 1;        // 1==> space is available
     SERVE = 0;
     }
   else if(space<0){
     switch (space){
        case -1:  sume = ques(k,xex,z);  //find total queue service time
                  mm = runtime(k,n,m,z,AA,BB,GG,s); //tr_time
                   if (sume < mm ){   //space at site OK if true
                       buf_status=1;
                       }
                   else {
                        buf_status=0;     //0==> no space at this site
                        }
                       SERVE = sume;
                  break;
       case -2: sume = ques(k,xex,z);      //find total queue service time
                  mm = runtime(k,n,m,z,AA,BB,GG,s);
                   if (sume < mm ){
                       buf_status=1;     //OK to send client
                       }
                   else {
                         buf_status=0;  //no space for client
                         }
                        SERVE = sume;
                    break;
    default:     buf_status=999;         //over-capacitated system
                 break;
             }                            //end_witch
      }                                   //end_else_if
  return buf_status;        //returns 0, 1, or 999
}
/* --------------------------------------------------------------------*/

int we_next(int k, int s, float &EAT)
{  //function for selecting server on next level
  float nn=0, loaded=999, timex=0;
  int i=0, kiss=-1, z=0, kount=0;       //z=0 always for stope selection as server
  for(i=0; i<level_stopes[k]; i++){ //selecting approp loading
   if((sblock[k][i]==1)&&(tot_stope_cap[k][i]-equ_assign[k][i]*minn > cap[s])){
     if((disc[k][i]!=0)&&(vect[k][i]!=0)){
        timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]); //delay
        timex = disc[k][i]/vect[k][i] + timex;          //total time travel'ng
         nn = ques(k,i,z);                   //service time of total queue
          if ( fabs(timex-nn) <loaded ){  //min slack to destination criteria
              loaded = fabs(timex - nn);
              kiss = i;                 //keep new destn site
              EAT = timex/rate[s];                //keep travel time
             }
            }
           }
   else{ kount += 1;                   //count # of sites depleted or blocked
       }
     }
  if(kount == level_stopes[k]){
     kiss = -1;
     EAT = 0;
    }
 return kiss;      //destination on next work area
}
/* ----------------------------------------------------------------------*/
 int stoop(int k, int pos, int s, float &EAT)
{//function for inter source travel times when blocked at POS
 int i=0, kiss=-1, kount = 0;  float timex = 0, loaded = 999;

 for(i=0; i<level_stopes[k];i++){
  if((sblock[k][i]==1)&&(tot_stope_cap[k][i]-equ_assign[k][i]*minn> cap[s])){
   if(( i < pos)&&( velo[k][i][pos] != 0)){               //lower #sites to current position
      timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]);
      timex = timex + (metr[k][i][pos]/velo[k][i][pos]); //find total time
      timex = timex/rate[s];
      }
     else if((i > pos)&&(velo[k][pos][i] != 0)){         //upper #sites to current position
      timex = weibull(alpha_4[s],beta_4[s],gamma_4[s]);
      timex = timex + (metr[k][pos][i]/velo[k][pos][i]); //travel times
      timex = timex/rate[s];
      }

  if ( timex < loaded){                        //select shortest travel time
       loaded = timex;
       kiss = i;                         //keep destn site
       EAT = timex;
      }
     }
   else  kount += 1;                         //track blocked or empty sites
 }                                      //end for loop

 if( kount == level_stopes[k] ){ kiss = 66; EAT = 0;}
 return kiss;      //new destin assignment and time of travel EAT
```

```
}
/*-----------------------------------------------------------------------*/

int feed(char type)
{         //reads in appropriate machine characteristics
   FILE *fp;
   char  mt[256], one_char, msg[80];

   if(( fp = fopen(equp,"r"))==NULL){
     sprintf(msg,"Failed to open equipment file: \"%s\"",equp);
     outtextxy(200,getmaxy()/2-40,msg);
     getch();
    exit(1);
   }

   while (fgets(mt,256,fp)!=NULL){
       one_char = tolower(mt[0]);
      if(one_char==type){
        fscanf(fp, "%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f\n", &man[0], &man[1],\
         &man[2], &man[3], &man[4], &man[5], &man[6], &man[7],\
         &man[8], &man[9], &man[10], &man[11], &man[12], &man[13], &man[14]);
        fclose ( fp );
       return ( 1 );
      }
     }
    fclose( fp );
   return ( 0 );
}
/* ---------------------------------------------------------------------*/

float simg( float t)
{              //simulates variability of face grade
  float vary = 0.0, x = 0.0;
  int num=0;

   x = randomm();
   if (x < 0.5 ) { num = 0; }
    else { num = 1; }

   switch(num){
       case 0: vary = sin(0.5*t)+(sin(0.5*t)*randomm());
              break;
       case 1: vary = cos((0.5*t) + 90)+(cos((0.5*t)+90)*randomm());
              break;
    }
   return vary;           //+/- deviation from expected target
}
/* ----------------------------------------------------------------------*/
int comp(int k)      //assess bin & stopes for work availability per level
{int i=0, back=0, bin_ore=0, bin_waste=0, stop_ore=0, stop_waste = 0;
 int bore = 0, bwste = 0, sore = 0, swste = 0;
```

```
for(i=0; i<level_bins[k];i++){      //check filled areas
      if((bblock[k][i] != -2)&&(bmat_flg[k][i] == 'o')) bin_ore += 1;//maybe available!
      if((bblock[k][i] != -2)&&(bmat_flg[k][i] == 'w')) bin_waste += 1;
      if((bblock[k][i] == 1)&&(bmat_flg[k][i] == 'o')) bore += 1;  //trully available!
      if((bblock[k][i] == 1)&&(bmat_flg[k][i] == 'w')) bwste += 1;
      }

for(i=0; i<level_stopes[k];i++){   //check depleted areas
    if((sblock[k][i] != -2)&&(mat_flag[k][i] == 'o')) stop_ore += 1; //maybe available
    if((sblock[k][i] != -2)&&(mat_flag[k][i] == 'w')) stop_waste += 1;
    if((sblock[k][i] == 1)&&(mat_flag[k][i] == 'o')) sore += 1;  //trully available
    if((sblock[k][i] == 1)&&(mat_flag[k][i] == 'w')) swste += 1;
   }
if(((sore > 0)&&(bore >0))||((swste >0)&&(bwste > 0)))
     back = 2;                    //feasible work available
else if(((bin_ore > 0)&&(stop_ore > 0))||((bin_waste > 0)&&( stop_waste > 0)))
     back = 0;                    //probably infeasible work available (blockages)
  else back = 1;                  //work complete here

return back;
}
/* ----------------------------------------------------------------*/
int complete(int levels)
{  //determines if all work is finished on all levels: bins full &/|| stopes empty
 int  i=0, kount=0, rest=0, yeild = 0;

 for (i = 0; i < levels; i++){
     yeild =  comp(i);
     if (yeild == 1)           /* work complete on level i          */
        kount += 1;            /* sum number of completed sections */
     }

   if (kount == levels) rest = 1;   /* all levels are complete      */
   else rest = 0;                   /* some sections still working */
  return rest;
}
/* ----------------------------------------------------------------------*/
int policies(int choice, int z, int s, int k, int i, char flag, float &EAT)
 { //lists the dispatch rules
    int zip=0;
    switch(choice){  // using the particular rule==choice select...
      case 1:     // maximization of destination utility:WEIGHTED CRITERIA
            zip= route1(z,s,k,i,flag,EAT);
            break;
      case 2: //minimization of slack time between resources: machines & servers
            zip= route2(z,s,k,i,flag,EAT);
            break;
      case 3: //optimize the earliest service times at servers
            zip= route3(z,s,k,i,flag,EAT);
            break;
      case 4: //work on basis of shortest travel times to servers
```

```
                zip= route4(z,s,k,i,flag,EAT);
                break;
        case 5: //work on basis of max utility & min slack of client-server
                zip= route5(z,s,k,i,flag,EAT);
                break;
        case 6: //work on basis of max work remaining: critical ratio
                zip= route6(z,s,k,i,flag,EAT);
                break;
        default: printf("Improper dispatch rule\n");
                getch();
                exit(EXIT_FAILURE);
                break;
        }
   return zip;
 }
/* -------------------------------------------------------------------*/
char *rules(int choice)
{ // outputs the rule in use at that time
 switch(choice){
        case 1: strcpy(arrar, "Work_Quality_Utility");
               break;
        case 2: strcpy(arrar, "Minimun_System_Slack");
               break;
        case 3: strcpy(arrar, "Earliest_Expected_Service_Time");
               break;
        case 4: strcpy(arrar, "Shortest_Travel_Times");
               break;
        case 5: strcpy(arrar, "Max_Work_Quality_Min_Slack_Rule");
               break;
        case 6: strcpy(arrar, "Maximun_Critical_Ratio");
               break;
        }
 return arrar;
}
/* -------------------------------------------------------------------*/
/*                  GRAPHICAL FUNCTIONS                               */
/* -------------------------------------------------------------------*/

 void far stopefill(int k, int i, int z)
 {   int col = 460, y=0; char msg[10];
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    setfillstyle(SOLID_FILL,WHITE);
    y = (int) (tot_stope_cap[k][i]*60)/sch_ton[k][i];    /* background of stope materials */
        col = col + i*30;
      if(k==1) bar(col,410,col+10,350);
      else bar(col,200,col+10,140);

  if(z==-1) setfillstyle(SLASH_FILL,LIGHTGRAY);
      else setfillstyle(SOLID_FILL,GREEN);

        col = 460;                              /* re-initialize */
    y = (int) (tot_stope_cap[k][i]*60/sch_ton[k][i]);   /* progressive material mucked */
        col = col + i*30;
       if (k==1){
        bar(col,410,col+10,350+(60-y));
        sprintf(msg,"S%d",i+1);
        outtextxy(col+2,420,msg);
        }
      else{
         bar(col,200, col+10,140+(60-y));
         sprintf(msg,"S%d",i+1);
         outtextxy(col+2,220,msg);
      }
   }
/* --------------------------------------------------------------------*/

 void far binfill(int k, int i, int z)
 { int col = 460, y=0; char msg[10];
    setlinestyle(SOLID_LINE,0,THICKNESS1);
    settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
    setfillstyle(SOLID_FILL,WHITE);
         col = col + i*30;
      if(k==0) bar(col,100,col+10,40);          /* bin outlines */
        else  bar(col,320,col+10,260);
  if(z==-1) setfillstyle(SLASH_FILL,LIGHTGRAY);
      else setfillstyle(SOLID_FILL,YELLOW);
         col = 460;
    y = (int) (tot_bin_cap[k][i]*60/fix_space[k][i]);   /* progressive material dumped */
         col = col + i*30;
      -if(k==0){
         bar(col,100,col+10,100-(60-y));
         sprintf(msg,"B%d",i+1);
         outtextxy(col+2,120,msg);
         }
      else{bar(col,320,col+10,320-(60-y));
         sprintf(msg,"B%d",i+1);
         outtextxy(col+2,335,msg);
       }
 }
 /* --------------------------------------------------------------------*/

  void animate(int yy)
 {int i, j;
  char msg[10];
  unsigned size[MAC];
    setviewport(100,100,130,130,0);
    size[yy] = imagesize(110,110,125,125);       /* making the image icon */
    buffer[yy] = malloc(size[yy]);               /* allocating space for image */
    rectangle(110,110, 125,125);
    for(i=111; i<125; i++){
      for(j=111; j<125; j++){
        getpixel(i,j);
        putpixel(i,j,RED);
        }
       }
      sprintf(msg,"%d",yy+1);
     outtextxy(117,117, msg);
    getimage(110,110,125,125,buffer[yy]);      /* saving image into memory */
  cleardevice();
  }
/* -------------------------------------------------------------------*/

 int far  motion(int k, int star, int endr,float TRFULL,float TREMP,\
           float ttrip,int flag,int carry,int *x_axis,int *y_axis)

 { int x=700, y=700, endx, endy;      //animation of vehicle movements

 if((endr!=-1)&&((carry==3)||(carry==4))){
     x = bstx[k][endr];      /* stay at same position e.g. temp blockage */
     y = bsty[k][endr];
     }
 else if((endr!=-1)&&((carry==6)||(carry==1))){
       x = sstx[k][endr];
       y = ssty[k][endr];
     }
 else if((endr==-1)&&((carry==4)||(carry==1))){
     x = bstx[k][star];      /* stay at same position e.g. temp blockage */
     y = bsty[k][star];
     }
 else if((endr==-1)&&(carry==6)){
       x = sstx[k][star];
       y = ssty[k][star];
     }
 else if((TRFULL!=0)&&(k==0)){        /* travelling empty       */
    switch (star){              /* local position switch  */
     case 0:                     /* currently at  1st stope */
       switch(endr){
         case 0: endx = 150; endy = 200;
                x = sstx[k][star]  + (int)((1-(TRFULL/ttrip))* \
                    abs(sstx[k][star] - endx));
                y = ssty[k][star] + (int)((1 - (TRFULL/ttrip)) * abs(ssty[k][star] - endy));
                break;
          case 1:if(TRFULL>=(0.5*ttrip)){
                endx = 150; endy = 200;
                x = sstx[k][star] + (int)(((ttrip-TRFULL)/(.5*ttrip)) * abs(sstx[k][star] - endx));
                y = ssty[k][star] + (int)(((ttrip-TRFULL)/(.5*ttrip)) * abs(ssty[k][star] - endy));
               }
               else { endx = 375; endy = 200;
                x = 150  + (int)(((.5*ttrip-TRFULL)/(.5*ttrip))*abs(150 - endx));
                y = 200;
               }
               break;
            }
      break;
case 1:
    switch(endr){          /* from stope 2 to bin 1 on level 1 */
       case 0: endx = 150; endy = 200;
               x = sstx[k][star] + (int)((1-(TRFULL/ttrip))* \
                   abs(sstx[k][star] - endx));
               y = ssty[k][star];
             break;
       case 1: if(TRFULL>=(0.5*ttrip)){ /* to bin 2 */
               endx = 150; endy = 200;
               x = sstx[k][star] + (int)(((ttrip-TRFULL)/(.5*ttrip)) * abs(sstx[k][star] - endx));
               y = ssty[k][star] - (int)(((ttrip-TRFULL)/(.5*ttrip)) * abs(ssty[k][star] - endy));
              }
              else { endx = 375; endy = 200;
               x = 150  + (int)(((.5*ttrip-TRFULL)/(0.5*ttrip))*abs(150 - endx));
               y = endy;
              }
           break;
          }
       break;
 case 2:                        /* at stope 3 on level 1 */
       switch(endr){
          case 0: endx = 150; endy = 200;        /* to bin 1 */
               x = sstx[k][star] - (int)((1-(TRFULL/ttrip))* \
                   abs(sstx[k][star] - endx));
               y = endy;
               break;
           case 1: endx = 375; endy = 200;        /* to bin 2 */
               x = sstx[k][star] + (int)((1-(TRFULL/ttrip))* \
                   abs(sstx[k][star] - endx));
               y = endy;
              break;
            }
          break;
    case 3:                  /* at stope 4 & to travel to bin 1 and 2 resp */
      switch(endr){
       case 0:if(TRFULL>=(0.75*ttrip)){
            x = sstx[k][star] - (int)(((ttrip-TRFULL)/(ttrip*0.25))*abs(450-375));
            y = ssty[k][star] + (int)(((ttrip-TRFULL)/(ttrip*0.25))*abs(ssty[k][star]-200));
            }
       else{ endx = 150; endy = 200;
            x = 375 - (int)((1 - (TRFULL/(0.75*ttrip)))*abs(375-endx));
            y = endy;
             }
             break;
       case 1: endx = 375; endy = 200;        /* to bin 2 from stope 4 */
                x = sstx[k][star] - (int)((1-(TRFULL/ttrip)) * abs(sstx[k][star] - endx));
                y = ssty[k][star] + (int)((1-(TRFULL/ttrip)) * abs(ssty[k][star] - endy));
              break;
             }
           break;
      case 4:              /* at stope 5 & travelling to bins 1 and 2 resp */
        switch(endr){
          case 0: x = sstx[k][star] - (int)((1-(TRFULL/ttrip))* abs(450-150));
                  y = ssty[k][star];
                  break;
          case 1: endx = 375; endy = 200;   /* to bin 2 from stope 5 */
                  x = sstx[k][star] - (int)((1-(TRFULL/ttrip)) * abs(sstx[k][star] - endx));
                  y = endy;
                break;
              }
         break;
        }
      }
/* ----------------- next level routes ------------------------ */

 else if((TRFULL!=0)&&(k==1)){ /* travelling full  LEVEL2  */
   switch (star){              /* local position switch */
    case 0:                     /* currently at  1st stope */
      switch(endr){             //GOTO DUMP 1 FROM STOPE 1
         case 0: endx = 300; endy = 350;
                 x = sstx[k][star]  + (int)((1-(TRFULL/ttrip))* \
                     abs(sstx[k][star] - endx));
                 y = ssty[k][star] + (int)((1 - (TRFULL/ttrip)) * abs(ssty[k][star] - endy));
                 break;
          }
          break;
     case 1:
           switch(endr){           //GOTO DUMP 1 FROM STOPE 2
            case 0:  endx = 300; endy =350;
                 x = sstx[k][star] + (int)((1-(TRFULL/ttrip))* abs(sstx[k][star] - endx));
                 y = ssty[k][star] - (int)((1 -(TRFULL/ttrip)) * abs(ssty[k][star] - endy));
                 break;
          }
          break;
     case 2:
            switch(endr){          //GOTO DUMP 1 FROM STOPE 3
              case 0: endx = 300; endy = 350;           /* goto next draw point */
                x = sstx[k][star] - (int)((1-(TRFULL/ttrip))* abs(sstx[k][star] - endx));
                y = ssty[k][star] + (int)((1-(TRFULL/ttrip)) * abs(ssty[k][star] - endy));
                break;
          }
         break;
        }
      }
/* ---------------------------------------------------------*/
else if((TREMP!=0)&&(k==0)&&(flag!=1)){      /* bin --> stope on level 1 */
 switch(star){
  case 0:                              /* bin 1 */
    switch(endr){
         case 0: endx = 80; endy = 100;      /* to stope 1 */
             x = bstx[k][star] - (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = bsty[k][star] - (int)((1-(TREMP/ttrip))*abs(bsty[k][star]-endy));
            break;
        case 1: endx = 80; endy = 200;      /* to stope 2 */
            x = bstx[k][star] - (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = bsty[k][star] + (int)((1-(TREMP/ttrip))*abs(bsty[k][star]-endy));
            break;
        case 2: endx = 250; endy = 200;      /* to stope 3 */
            x = bstx[k][star] + (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = endy;
            break;
        case 3: if(TREMP>=(0.25*ttrip)){ /* to stope 4 */
            endx = 375; endy = 200;
            x = bstx[k][star] + (int)(((ttrip-TREMP)/(0.75*ttrip))*abs(bstx[k][star]-endx));
            y = endy;
            }
            else{ endx = 450; endy = 80;
             x = 375 + (int)((1-(TREMP/(0.25*ttrip))) * abs(375 - endx));
             y = 200 - (int)((1 -(TREMP/(.25*ttrip))) * abs(200 - endy));
            }
            break;
        case 4: endx = 450; endy = 200;      /* to stope 5 */
            x = bstx[k][star] + (int)((1 - (TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = endy;
           break;
       }
      break;

  case 1:                               /*  bin 2 */
    switch(endr){
        case 0: if(TREMP>(0.25*ttrip)){ /* to stope 1 */
              endx = 150; endy = 200;
              x = bstx[k][star] - (int)(((ttrip-TREMP)/(.75*ttrip))*abs(bstx[k][star]-endx));
              y = endy;
              }
              else{ endx = 80; endy = 100;
              x = 150 - (int)((1-(TREMP/(0.25*ttrip)))*abs(150 - endx));
              y = 200 - (int)((1-(TREMP/(0.25*ttrip)))*abs(200 - endy));
              }
              break;

        case 1:endx = 80; endy = 200;       /* to stope 2 */
            x = bstx[k][star] - (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = endy;
            break;

        case 2: endx = 250; endy = 200;      /* to stope 3 */
            x = bstx[k][star] - (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = endy;
            break;

        case 3: endx = 450; endy = 80;       /* to stope 4 */
```

```
            x = bstx[k][star] + (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = bsty[k][star] - (int)((1-(TREMP/ttrip))*abs(bsty[k][star]-endy));
            break;

        case 4: endx = 450; endy = 200;      /* to stope 5 */
            x = bstx[k][star] + (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
            y = bsty[k][star];
            break;
          }
        break;
      }
     }
/* ----------------------- level 1: STOPE TO STOPE ----------------------- */
 else if((TREMP!=0)&&(k==0)&&(flag==1)){    /* stope --> stope on level 1 */
 switch(star){
   case 0:                               /* stope 1 */
    switch(endr){
     case 1: if((int)TREMP>(int)(0.5*ttrip)){  /* stope 1 to stope 2 */
            endx = 150; endy = 200;
            x = sstx[k][star] + (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(sstx[k][star]-endx));
            y = ssty[k][star] + (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(ssty[k][star]-endy));
            }
          else{ endx = 80; endy = 200;
           x = 150 - (int)((TREMP/(.5*ttrip))*(150 - 80));
           y = endy;
           }
           break;

    case 2: if((int)TREMP>(int)(0.5*ttrip)){ /* stope 1 to stope 3 */
            endx = 150; endy = 200;
            x = sstx[k][star] + (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(sstx[k][star]-endx));
            y = ssty[k][star] + (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(ssty[k][star]-endy));
            }
          else{ endx = 250; endy = 200;
           x = 150 + (int)((1-(TREMP/(.5*ttrip)))*(250 - 150));
           y = endy;
           }
           break;

     case 3: if((int)TREMP>(int)(0.75*ttrip)){ /* stope 1 to stope 4 */
            endx = 150; endy = 200;
            x = sstx[k][star] + (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(sstx[k][star]-endx));
            y = ssty[k][star] + (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(ssty[k][star]-endy));
            }
          else if(((int)TREMP>(int)(.25*ttrip))&&((int)TREMP<(int)(0.75*ttrip))){
           endx = 375; endy = 200;
           x = 150 + (int)((1-(TREMP-(0.25*ttrip))/(.5*ttrip))*(375 - 150));
           y = endy;
           }
           else { endx = 450; endy = 80;
           x = 375 + (int)((1-(TREMP/(0.25*ttrip)))*abs(450-375));
```

```
            y = 200 - (int)((1-(TREMP/(0.25*ttrip)))*abs(200-80));
            }
            break;

          case 4: if((int)TREMP>(int)(0.75*ttrip)){ /* stope 1 to stope 5 */
             endx = 150; endy = 200;
             x = sstx[k][star] + (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(sstx[k][star]-endx));
             y = ssty[k][star] + (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(ssty[k][star]-endy));
             }
           else{ endx = 450; endy = 200;
            x = 150 + (int)((1-(TREMP/(.75*ttrip)))*abs(150 - 450));
            y = endy;
            }
            break;
       }
       break;
  case 1:                        /* stope 2 */
      switch(endr){
       case 0:if((int)TREMP>(int)(.5*ttrip)){    /* stope 2 to stope 1 */
            endx = 150; endy = 200;
            x = sstx[k][star] + (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(150-80));
            y = endy;
            }
            else{ endx = 80; endy = 100;
            x = 150 - (int)((1-(TREMP/(0.5*ttrip)))*abs(150 - endx));
            y = 200 - (int)((1-(TREMP/(0.5*ttrip)))*abs(200 - endy));
            }
            break;
       case 2: endx = 250; endy = 200;    /* stope 2 to stope 3 */
          x = sstx[k][star] + (int)((1-(TREMP/ttrip))*abs(sstx[k][star]-endx));
          y = endy;
          break;
       case 3: if((int)TREMP>(int)(0.25*ttrip)){ /* stope 2 to stope 4 */
          endx = 375; endy = 200;
          x = sstx[k][star] + (int)(((ttrip-TREMP)/(.75*ttrip))*abs(sstx[k][star]-endx));
          y = endy;
          }
          else { endx = 450; endy = 80;
           x = 375 + (int)((1-(TREMP/(0.25*ttrip)))*abs(450-375));
           y = 200 - (int)((1-(TREMP/(0.25*ttrip)))*abs(200-80));
          }
          break;
       case 4: endx = 450; endy = 200;         /* stope 2 to stope 5 */
          x = sstx[k][star] + (int)((1-(TREMP/ttrip))*abs(sstx[k][star]-endx));
          y = endy;
          break;
      }
       break;

 case 3:                        /* stope 4 */
    switch(endr){
        case 4: endx = 450; endy = 200;          /* stope 4 to stope 5 */
            x = sstx[k][star];
            y = ssty[k][star] + (int)((1-(TREMP/ttrip))*abs(ssty[k][star]-endy));
            break;
         case 2: if((int)TREMP>(int)(0.5*ttrip)){ /* stope 4 to stope 3 */
            x = sstx[k][star] - (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(450-375));
            y = ssty[k][star] - (int)(((ttrip-TREMP)/(0.5*ttrip))*abs(200-80));
            }
            else {
            x = 375 -(int)((1-(TREMP/(0.5*ttrip)))*abs(375-250));
            y = 200;
            }
            break;

         case 1: if((int)TREMP>(int)(0.75*ttrip)){        /* stope 4 to stope 2 */
            x = sstx[k][star] - (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(450-375));
            y = ssty[k][star] - (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(200-80));
            }
            else{
            x = 375 -(int)((1-(TREMP/(0.75*ttrip)))*abs(375-80));
            y = 200;
            }
            break;

         case 0: if((int)TREMP>(int)(0.75*ttrip)){ /* stope 4 to stope 1 */
            x = sstx[k][star] - (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(450-375));
            y = ssty[k][star] - (int)(((ttrip-TREMP)/(0.25*ttrip))*abs(200-80));
            }
          - else if(((int)TREMP>(int)(0.25*ttrip))&&((int)TREMP<(int)(.75*ttrip))){
              x = 375 - (int)((1-(TREMP-(0.25*ttrip))/(.5*ttrip))*(375- 150));
              y = 200;
              }
           else {
              x = 150 - (int)((1-(TREMP/(0.25*ttrip)))*abs(150-80));
              y = 200 - (int)((1-(TREMP/(0.25*ttrip)))*abs(200-100));
              }
              break;
         }
        break;
    case 4:                                   /* stope 5 */
        switch(endr){
         case 0: if((int)TREMP>(int)(0.25*ttrip)){ /* stope 5 to stope 1 */
             endx = 150; endy = 200;
             x = sstx[k][star] - (int)(((ttrip-TREMP)/(0.75*ttrip))*abs(450-150));
             y = endy;
             }
             else {
             x = 150 - (int)((1-(TREMP/(0.25*ttrip)))*abs(450-150));
             y = 200 - (int)((1-(TREMP/(0.25*ttrip)))*abs(200-100));
             }
             break;
    case 1: endx = 80; endy = 200;          /* stope 5 to stope 2 */
       x = sstx[k][star] - (int)((1-(TREMP/ttrip))*abs(sstx[k][star]-endx));
       y = endy;
       break;
     case 2: endx = 250; endy = 200;        /* stope 5 to stope 3 */
       x = sstx[k][star] - (int)((1-(TREMP/ttrip))*abs(sstx[k][star]-endx));
       y = endy;
       break;
     case 3: endx = 450; endy = 80;              /* stope 5 to stope 4 */
       x = sstx[k][star];
       y = ssty[k][star] - (int)((1-(TREMP/ttrip))*abs(ssty[k][star]-endy));
       break;
       }
   break;
  }
 }
/* ==================== LEVEL 2 ====================== */
else if((TREMP!=0)&&(k==1)&&(flag!=1)){      /* bin --> stope travel */
    switch(star){
     case 0:                            /* at bin 1 on level 2 to travel to stopes! */
      switch(endr){                      /* to stope 1 */
       case 0: endx = 80; endy = 300;
           x = bstx[k][star] - (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
           y = bsty[k][star] - (int)((1-(TREMP/ttrip))*abs(bsty[k][star]-endy));
           break;
       case 1: endx = 100; endy = 400;     /* to stope 2 */
           x = bstx[k][star] - (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
           y = bsty[k][star] + (int)((1-(TREMP/ttrip))*abs(bsty[k][star]-endy));
           break;
        case 2: endx = 400; endy = 350;    /* to stope 3 */
           x = bstx[k][star] + (int)((1-(TREMP/ttrip))*abs(bstx[k][star]-endx));
           y = bsty[k][star];
           break;
          }
      break;
    }
   }
/* -------------------------------------------------------------------*/
//on level 2 but current location blocked or finished
else if((TREMP!=0)&&(k==1)&&(flag==1)){      /* stope --> stope travel */
  switch(star){
    case 0:                             /* currently at 1st stope */
     switch(endr){
      case 1:if (TREMP>=(0.5*ttrip)){   /* to stope 2 */
            endx = 300; endy =350;
            x = sstx[k][star] + (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(sstx[k][star] - endx));
            y = ssty[k][star] + (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(ssty[k][star] - endy));
            }
            else { endx = 100; endy = 400;
            x = 300 - (int)(((.5*ttrip-TREMP)/(.5*ttrip)) * abs(300 - endx));
            y = 350 + (int)((1 - (TREMP/(.5*ttrip))) * abs(350 - endy));
        }
        break;
  case 2:if (TREMP>=(0.5*ttrip)){ //stope 1 to stope 3
        endx =300; endy =350;
        x = sstx[k][star] + (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(sstx[k][star] - endx));
        y = ssty[k][star] + (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(ssty[k][star] - endy));
        }
       else{ endx = 400; endy =350;
         x = 300  + (int)(((.5*ttrip-TREMP)/(0.5*ttrip))*abs(300 - endx));
         y = 350;
         }
        break;
       }
      break;        /* end of stope 1 on to other stopes on same level 2*/
case 1:                    /* currently at 2nd stope */
 switch(endr){
  case 0:if (TREMP>=(0.5*ttrip)){ // stope 2 to stope 1
        endx = 300; endy =350;
        x = sstx[k][star] + (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(sstx[k][star] - endx));
        y = ssty[k][star] - (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(ssty[k][star] - endy));
        }
        else { endx = 80; endy = 300;
        x = 300  - (int)((1-(TREMP/(.5*ttrip))) * abs(300 - endx));
        y = 350  - (int)((1-(TREMP/(.5*ttrip))) * abs(350 - endy));
        }
        break;
  case 2:if (TREMP>=(0.5*ttrip)){ //stope 2 to stope 3
        endx =300; endy =350;
        x = sstx[k][star]  + (int)((ttrip-TREMP) * abs(sstx[k][star] - endx)/(.5*ttrip));
        y = ssty[k][star]  - (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(ssty[k][star] - endy));
        }
       else{ endx = 400; endy= 350;
         x = 300  + (int)(((.5*ttrip-TREMP)/(0.5*ttrip))*abs(300 - endx));
         y = 350;
         }
        break;
       }
      break;        /* end of stope 2 on to other stopes on same level 2 */

case 2:                    /* currently at 3rd stope */
 switch(endr){
  case 0:if (TREMP>=(0.5*ttrip)){ // stope 3 to stope 1
        endx = 300; endy =350;
        x = sstx[k][star] - (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(sstx[k][star] - endx));
        y = 350;
        }
        else { endx = 80; endy = 300;
        x = 300  - (int)(((.5*ttrip-TREMP)/(.5*ttrip)) * abs(300 - endx));
        y = 350  - (int)(((.5*ttrip-TREMP)/(.5*ttrip)) * abs(350 - endy));
        }
        break;
        case 1:if (TREMP>=(0.5*ttrip)){ //stope 3 to stope 2
              endx =300; endy =350;
              x = sstx[k][star] - (int)(((ttrip-TREMP)/(.5*ttrip)) * abs(sstx[k][star] - endx));
              y = 350;
              }
             else{ endx = 100; endy= 400;
               x = 300  - (int)(((.5*ttrip-TREMP)/(.5*ttrip))* abs(300 - endx));
               y = 350  + (int)(((.5*ttrip-TREMP)/(.5*ttrip)) * abs(350 - endy));
               }
              break;
             }
            break;        /* end of stope 3 on to other stopes on same level 2 */
        }                /* end if before switch(star) */
    }
   *x_axis = x;
  *y_axis = y;
 return (1);
}
/* ----------------------------------------------------------------------*/

 void network1(int k, int X1,int Y1, int X2, int Y2)
 { //draws the haulage routes, one at a time on the screen
 char msg[15];
 setfillstyle(CLOSE_DOT_FILL,LIGHTGRAY);
 bar(75,210, 455,290);
 setlinestyle(SOLID_LINE,USER_PATTERN,THICKNESS2);
   line(X1, Y1, X2, Y2);
   line(350,350,280,200);

  sprintf(msg,"IDLE LHD: ");  /* labels for machines on breakdown and idle */
  outtextxy(100,40,msg);
  sprintf(msg,"B/D LHD: ");
  outtextxy(100,70,msg);

  if(k==0){
     sprintf(msg, "LEVEL %d",k+1);
     outtextxy(getmaxx()/2-40, 160, msg);
     }
   else { sprintf(msg, "LEVEL %d",k+1);
         outtextxy(getmaxx()/2-40, 430, msg);
    }
 }
/*---------------------------------------------------------------------*/

void network2(int k)
 { //draws in the bin and stope symbols on the network on screen
 int i;
 setlinestyle(SOLID_LINE,USER_PATTERN,THICKNESS2);

    setfillstyle(SOLID_FILL,GREEN);
 for (i = 0; i < level_stopes[k]; i++)
```

```
     circle(sstx[k][i],ssty[k][i],7);

     setfillstyle(SOLID_FILL,YELLOW);
 for (i = 0; i < level_bins[k]; i++)
    ellipse(bstx[k][i], bsty[k][i],0,360,7,4);
  }
/* ---------------------------------------------------------------------*/

 void plot_axes( float delta2, float eta2,int shift_end,int smp_tim, int tally)
{ /* creates a progressive variation graph of grade deviation from target */
  int  i=0, j=0, k=0, max=0, range=0;
  char msg[20];

  for (i=0; i< tally; i++){                /* max to use in scaling the y-axis */
     if((int)(fabs(tall_grd[i])) >= max)  /* conditional MUST include for multi-*/
        max =(int)(tall_grd[i]);          /* variable plots: still to be done */
    }
  cleardevice();
  setlinestyle(SOLID_LINE,USER_PATTERN,THICKNESS1);
  rectangle(2,2,getmaxx()-2,getmaxy()-2);
  setfillstyle(SOLID_FILL, 4);
  bar(30, 30, 600, 440);

  settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(getmaxx()/2, 60, "SHIFT GRADE DEVIATIONS");
  line(135, 70, 510,70);

  /* draw x axis and y-axis resp */
  line (XOFFSET, (YMAX-YOFFSET), XMAX + 80, YMAX-YOFFSET); //x-axis
  line (XOFFSET, YMAX-10, XOFFSET, YMAX-YOFFSET-130);      //y-axis

  /* draw markings on x-axis indicating key times */
  settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
  /* labels to x-axis and sign of deviation */
  outtextxy(getmaxx()/2, 380, "SHIFT TIME (mins) -->");

  range = (int) (shift_end/smp_tim);
  for (i = 0; i < range+1; i++){
     line((XOFFSET+(int)((i*smp_tim)/60)),(YMAX-YOFFSET),XOFFSET+(int)((i*smp_tim)/60),
YMAX-YOFFSET+3);
     sprintf(msg, "%3d", (int)(i*smp_tim)/60);
     outtextxy(XOFFSET+(int)(i*smp_tim)/60, (YMAX-YOFFSET+5 + textwidth(msg)), msg);
    }

  /* draw upper and lower deviation limits resp  */
  setlinestyle(DOTTED_LINE, USER_PATTERN, THICKNESS2);

  line(XOFFSET, (YMAX-YOFFSET-(int)(delta2*100)/30), XMAX+80,
YMAX-YOFFSET-(int)(delta2*100)/30);
  line(XOFFSET, (YMAX-YOFFSET+(int)(delta2*100)/30), XMAX+80,
YMAX-YOFFSET+(int)(delta2*100)/30);
  line(XOFFSET, (YMAX-YOFFSET-(int)((delta2+eta2)*100)/30), XMAX+80,
YMAX-YOFFSET-(int)((delta2+eta2)*100)/30);
  line(XOFFSET, (YMAX-YOFFSET+(int)((delta2+eta2)*100)/30), XMAX+80,
YMAX-YOFFSET+(int)((delta2+eta2)*100)/30);
  outtextxy(XMAX,YMAX-(YOFFSET + (int)((delta2*100+5)/30)),"delta");
  outtextxy(XMAX,YMAX-(YOFFSET + (int)(((delta2+eta2)*100+5)/30)),"delta + eta");
  outtextxy(XMAX,YMAX-(YOFFSET - (int)((delta2*100-5)/30))," -delta");
  outtextxy(XMAX,YMAX-(YOFFSET - (int)(((delta2+eta2)*100-5)/30)),"- delta - eta");
  outtextxy(XMAX,YMAX-YOFFSET-5,"plan grade");

  /*  label the y axis increments  */
  for  (i = 0; i <= 3; i++){
                                        /* negative deviations */
      line( XOFFSET, YMAX-(YOFFSET + (int)(i*100)/3), XOFFSET-4, YMAX-(YOFFSET
+(int)(i*100)/3));
      sprintf(msg,"%3d",i*10);
      outtextxy(XOFFSET - textwidth(msg), YMAX-(YOFFSET + (int)(i*100)/3),msg);
                                        /* positive deviations */
      line( XOFFSET, YMAX-(YOFFSET - (int)(i*100)/3), XOFFSET-4, YMAX-(YOFFSET
-(int)(i*100)/3));
      sprintf(msg,"%3d",-i*10);
      outtextxy(XOFFSET - textwidth(msg), YMAX-(YOFFSET -(int)(i*100)/3),msg);
    }
    moveto(XOFFSET,YMAX-YOFFSET);              /*  moves cursor position to zero position  */
    for (i = 0; i < tally; i++){              /*  draws a variation diagram */
        j = (int)((tall_grd[i]/30) * 100);   /* proportion the values */
        j = YMAX - (YOFFSET + j);
        k = XOFFSET + (int)((i+1)*smp_tim/60);
        setlinestyle(CENTER_LINE,USER_PATTERN,THICKNESS2);
       lineto(k,j);                        /*  connecting consecative points by a line */
       }
   /*  labeling the y-axis by name   */
    settextstyle(DEFAULT_FONT, VERT_DIR, 1);
    outtextxy(75, getmaxy()/2, "deviation (%)");
    settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);           /*resetting to horiz print */
 }
 /* -----------------------------------------------------------------*/

   void plot_axed(float max, int shift_end, int smp_tim,int tally )
{ /* cumulative production graph of tonnages */
  int i, j=0, k=0, range=0;
  char msg[20];
  setlinestyle(SOLID_LINE,USER_PATTERN,THICKNESS1);
  rectangle(2,2,getmaxx()-2,getmaxy()-2);
  setfillstyle(SOLID_FILL, 4);
  bar(30, 30, 600, 440);
  setlinestyle(SOLID_LINE,USER_PATTERN,THICKNESS1);


  settextstyle(DEFAULT_FONT, HORIZ_DIR, 2);
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(getmaxx()/2, 60, "CUMULATIVE SHIFT PRODUCTION");
  line(100, 70, 532,70);

  /* draw x axis and y-axis resp */
  line (XOFFSET, (YMAX-YOFFSET+100), XMAX + 80, YMAX-YOFFSET+100); //x-axis
  line (XOFFSET, 350, XOFFSET, 120);                               //y-axis

  /* draw markings on x-axis indicating key times */
  settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
  range = (int) shift_end/smp_tim;
  for (i = 0; i < range+1; i++){
line((XOFFSET+(int)((i*smp_tim)/60)),(YMAX-YOFFSET+100),XOFFSET+(int)((i*smp_tim)/60),
YMAX-YOFFSET+103);
     sprintf(msg, "%3d", (int)(i*smp_tim)/60);
     outtextxy(XOFFSET+(int)(i*smp_tim)/60, (YMAX-YOFFSET+100 + textwidth(msg)), msg);
//100==103
    }

  /* labels to x-axis and sign of deviation */
   outtextxy(getmaxx()/2, 390, "SHIFT TIME (mins) ->");

  /* label the y axis increments   */
  for (i = 0; i <= 20; i=i+2){
      line( XOFFSET, 350 - (int)((i*200)/20), XOFFSET-4, 350-(int)((i*200)/20));
      sprintf(msg,"%3d",(int)(i*100)/20);
      outtextxy(XOFFSET - textwidth(msg), 350 -(int)((i*200)/20),msg);
      }
        moveto(120,350);
     for (i=0; i< tally; i++){   /*  draws a cumulative diagram */
         k = XOFFSET + (int)(((i+1)*smp_tim)/60);
         j = (int)(tall_ton[i]*200/max);
          setlinestyle(DOTTED_LINE, USER_PATTERN,THICKNESS2);
          lineto(k,350-j);            /*  connecting consecative points by a line */
         }
        moveto(120,350);
     for (i=0; i< tally; i++){   /*  draws a cumulative diagram */
        j = (int)(plan_ton[i]*200/max);
        k = XOFFSET + (int)(((i+1)*smp_tim)/60);
         setlinestyle(SOLID_LINE, USER_PATTERN,THICKNESS2);
         lineto(k,350-j);
        }
    /*  labeling the y-axis by name   */
   settextstyle(DEFAULT_FONT, VERT_DIR, 1);
   outtextxy(75, getmaxy()/2, "cum production (%)");
   settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);        /*resetting to horiz print */
   setlinestyle(SOLID_LINE,0,THICKNESS1);
  getch();
 }
/* ---------------------------------------------------------------*/
```

```
 void maint()
{  /*  creates the menu of priority rules  */
   settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
   setviewport(400,40,600,250,1);
   setlinestyle(SOLID_LINE, USER_PATTERN, THICKNESS1);
   rectangle(100,30, 190,180);
   line(130,30, 130,180);
   line(160,30, 160,180);
   line(100,80,190,80);
   line(100,130,190,130);
   outtextxy(45,55,"GRADE");
   outtextxy(45,105,"TONNES");
   outtextxy(45,155,"UTILZE");
   outtextxy(110,55,"1");
   outtextxy(110,105,"2");
   outtextxy(110,155,"3");
   outtextxy(140,55,"4");
   outtextxy(140,105,"5");
   outtextxy(140,155,"6");
   outtextxy(170,55,"7");
   outtextxy(170,105,"8");
   outtextxy(170,155,"9");
   outtextxy(100,10,"P1");
   outtextxy(130,10,"P2");
   outtextxy(160,10,"P3");
   }
/*===============================================================*/
/*                    MAIN PROGRAM                              */
/* ==============================================================*/

main()
{
 int d = 0, w = 0, level_rule[MAX]={0};
 int k = 0, i = 0, j = 0, s = 0, choice=0;
 int move =0, mum = 0, tick = 0, tock = 0;

//shift input variables (set-up times etc)
 int b_lunch=0, e_lunch=0, shift_end=0, overs=0, pedza=0;
 int SAP_TIME=0, icrem=0,  smp_tim=0, clock=0, ending=0;
 int *x_axis, *y_axis;
 int spt_tim=0, dpt_tim=0;

// flags of position/activity
 int z=0, AVE=0, DSAVE=0, undone[MAC]  ={0};
 int next=0, stope[MAX][STP]={0}, bin[MAX][BIN]={0}, turns=0;
 int tally=0, levels=0;
 int endr[MAC]={0}, star[MAC]={0};            /*end & start of LHD travel indicator*/
 int sos[MAC]={0};                         /* material source  */
 int cht[MAC]={0};                         /* material destination */
 int X1[MAX][15]={0}, Y1[MAX][15]={0};       /* starting tunnel co-ordinates */
 int X2[MAX][15]={0}, Y2[MAX][15]={0};       /* ending tunnel co-ordinates */
```

```
// counts variable names
int p=0, cnt3[MAX][BIN]={0}, cnt2[MAX][STP]={0};
int orig[MAX][BIN][30] = {0}, oldstate[MAC] ={0}, ba = 0, rule =0;
int oldpos[MAC] ={0}, red[MAC] = {0}, no_lines[MAX]={0}, nodes[MAX]={0};
int mach[MAC] ={0}, level_mach[MAC]={0}, status[MAC]={0};

// variability criteria
float delta2 = 0, delta3 = 0;
float eta2 = 0, eta3 = 0, theta = 0;
float delta1=0, eta1 =0;
int priority = 0;

// operating variable of equipment
  float ld_tim[MAC]={0}, dp_tim[MAC]={0},bd_time[MAC]={0}, load[MAC]={0};
  float wt_load[MAC]={0}, wt_dump[MAC]={0}, tfull[MAC]={0}, tempty[MAC]={0}; //times of
activities
  float under_load=0, over_load=0;                    //load distr. parameters
  float sbk_tm[MAX][STP]={0}, bbk_tm[MAX][BIN]={0}; //blockage times of source & destn resp
  float TREMP[MAC]={0}, TRFULL[MAC]={0},ttrip[MAC] ={0};
  float opcost[MAC]={0}, totcost=0.0, lhd_ton[MAC] = {0};    /* operating cost calcs */
  float st_tim[MAC]={0}, start_up=0, overtime[MAC]={0};

// production variable statistics:times/grades/tonnages/utilitization/etc
  float Gave[MAX]={0}, Lton[MAX][STP]={0}, ScLton[MAX]={0}, LAgrd[MAX]={0};
  float LTgrd[MAX]={0}, g2[MAX]={0}, gdz=0, temp2 =0;
  float available =0, utilize = 0;
  float Agrandtn=0, Pgrandtn=0, Agrandgd=0, Pgrandgd=0, Wton[MAX]={0}, TWton=0,
goom=0;
  float ScWton[MAX]={0}, wton[MAX][STP]={0};
  float ALton[MAX]={0}, gomo[MAX]={0}, travD[MAX]={0}, travDE[MAX]={0};
  float Block[MAX]={0},Slock[MAX]={0},b_block[MAX][BIN]={0},s_block[MAX][STP]={0};

  float Tbd_dn=0, bd_dn[MAC]={0}, TBlock=0, TSlock=0;
  float WTlost=0, travF[MAX]={0}, travE[MAX]={0};
  float prodeff=0, gradeff=0, Accm[MAX]={0}, oreton=0;
  float too[MAX][STP][30]={0}, tot[MAX][STP][30]={0},
wt_tal[MAX][STP]={0},ave_gd[MAX][STP]={0};
  float d_ton[MAX][STP][30]={0}, au_tal[MAX][STP][30]={0}, g_move[MAX][STP]={0};
  float mgrade[MAX][STP]={0}, dd=0, d1[MAX][STP]={0}, g1[MAX][STP]={0}, find=0;
  float dump_ton[MAX][BIN]={0}, sumt=0, empt=0, Agr=0, Pgr=0;
  float rungd[MAX][BIN][30]={0}, target_ton[MAX][STP]={0};
  float acc_tim[MAC]={0}, MTBF=0, MRT=0;  //mean time between failures and mean service time

// character variables
 char type[64], finame[30], layout[30], results[30];
 char name[MAC][15], gogo[30], chabva[30], sign, aa;
 char msg[80], mine = ' ';
 time_t first, second;

 FILE *fpt, *fp2, *fp3, *fp4, *fp5, *fp6;
```

```
/* -----------------------------------------------------------*
 *          BEGIN INITIALISATION OF INPUT DATA...             *
 * -----------------------------------------------------------*/
  x_axis = (int *)malloc(sizeof(x_axis));
  y_axis = (int *)malloc(sizeof(y_axis));

  for (i = 0; i < MAC; i++){    /* initializing arrays of destinations */
    sos[i] = -1;
    cht[i] = -1;
    star[i] = -1;
    endr[i] = -1;
    red[i] = -1;
    oldpos[i] = -1;
    oldstate[i] = -1;
  }

for (i=0; i< MAX; i++){
 for (j = 0; j < STP; j++){
  for ( k=0; k< MAC; k++){
   sequnt[i][j][k] = -1;
   }
  }
 }

for (i = 0; i< MAX; i++){
 for (j = 0; j < BIN; j++){
  for ( k= 0; k< MAC; k++){
    bsequnt[i][j][k] = -1;
   }
  }
 }

for (k = 0; k< MAX; k++){
 for (i = 0; i < STP; i++){
  for ( j= 0; j< BIN; j++){
    route[k][i][j] = 0;
    in_road[k][i][j] = 0;
   }
  }
 }

int graphdriver=DETECT, graphmode, gerror;      /* auto-detection */

detectgraph(&graphdriver, &graphmode);
if( graphdriver < 0){
   printf("No graphics hardware available.\n");
   exit(1);
   }

initgraph(&graphdriver, &graphmode, "c:\\borlandc\\bgi");
```

```
 gerror = graphresult();

 if(gerror < 0){
   printf("initgraph error: %s.\n", grapherrormsg(gerror));
   exit(1);
   }

   setviewport(0,0,getmaxx()-1, getmaxy()-1,CLIP_ON);
   setbkcolor(BLUE);
   setcolor(YELLOW);
   setlinestyle(SOLID_LINE,USER_PATTERN,THICKNESS1);
   rectangle(2,2,getmaxx()-3, getmaxy()-3);
   setfillstyle(SOLID_FILL, 4);
   bar(30, 30, 600, 440);
   settextstyle( TRIPLEX_FONT,HORIZ_DIR,2);
   settextjustify(CENTER_TEXT,CENTER_TEXT);
   outtextxy(getmaxx()/2-5,getmaxy()/2-90,"UNDERGROUND MINING DYNAMIC");
   outtextxy(getmaxx()/2-5,getmaxy()/2-40,"UNDERGROUND ACTIVE DISPATCH MODEL ");
   outtextxy(getmaxx()/2-25, getmaxy()/2+10,"McGill University, Montreal");
   outtextxy(getmaxx()/2-50, getmaxy()/2+60,"C.M. Tsomondo");
   getch();
   cleardevice();
   rectangle(2,2,getmaxx()-3, getmaxy()-3);
   setfillstyle(SOLID_FILL, 4);
   bar(30, 30, 600, 440);
   settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
   settextjustify(LEFT_TEXT,TOP_TEXT);
   outtextxy(200, getmaxy()/2-100, "Enter the schedule input file");
   gotoxy(25,12); gets(finame);

   if((fp4=fopen(finame, "r"))==NULL){
      outtextxy(200,getmaxy()/2-75,"Failed to open  file \n");
      exit(EXIT_FAILURE);
     }
 outtextxy(200,getmaxy()/2-50, "Enter equipment file ");
 gotoxy(25,15); gets(equp);

//initializing the schedule input: goal parameters
fscanf(fp4,"%d\n",&levels);
fscanf(fp4,"%f %f\n",&under_load,&over_load);

//reading the scheduled work areas: sources & dumps, equipment # & utility values
for(k=0; k<levels; k++){
fscanf(fp4,"%d %d %d\n",&level_stopes[k],&level_bins[k],&level_mach[k]);
      nodes[k]=level_stopes[k];
      }

//reading the material sources, type and qualities
  for(k=0; k<levels; k++){
    for( i=0; i<level_stopes[k]; i++){ //read in stope tonnages and grades
      fscanf(fp4,"%f %f %d %d %d
%c\n",&tot_stope_cap[k][i],&stope_grade[k][i],&priority,&bf_stcap[k][i],&sblock[k][i],&mat_flag[k][i]
);
        sch_ton[k][i]=tot_stope_cap[k][i]; //storing the qties scheduled for site
       mat_flag[k][i] = tolower(mat_flag[k][i]);
      if(priority == 1) rank[k][i] = 3;
      else if(priority == 2) rank[k][i] = 2;
      else if(priority == 3) rank[k][i] = 1;
      else {printf("Allowed stope priorities are only 1,2 & 3.");
           getch();
           exit(EXIT_FAILURE);
           }
      if(stope_grade[k][i] != 0.0){
           Gave[k] += (stope_grade[k][i] * sch_ton[k][i]);
           ScLton[k] += sch_ton[k][i];
          }
       else ScWton[k] += sch_ton[k][i];
     }
       Pgrandtn += ScLton[k];     //total scheduled ore tonnage
       oreton = Pgrandtn;
      Pgrandgd += Gave[k];
     TWton += ScWton[k];          //total scheduled waste tonnage
    gomo[k] = Gave[k]/ScLton[k]; //planned level weighted mean grade
   }
  goom = Pgrandgd/Pgrandtn;

// reading dump-point characteristics
  for(k=0; k<levels; k++){
   for(j=0; j<level_bins[k]; j++){ //read in bin tonnage capacity
      fscanf(fp4,"%f %d %d
%c\n",&tot_bin_cap[k][j],&buf_bin_cap[k][j],&bblock[k][j],&bmat_flg[k][j]);
      fix_space[k][j] = tot_bin_cap[k][j];     //stores initial space avail. for graph
      bmat_flg[k][j] = tolower(bmat_flg[k][j]);
    }
  }
  fscanf(fp4,"%f\n",&start_up); //start-up time for the shift

 //initializing machine positions, type, drivers and machine characteristics
 for (k=0; k<levels; k++){     //read machine #, type, features & location
  for (s=sumd; s<(sumd+level_mach[k]); s++){
   lev[s] = k;                 //initializing work section of machine
    fscanf(fp4,"%s %c %d %c\n",&name[s][0], &mach_type[s], &w, &sign); //***name=>driver
name PROBLEM!!!
   if( sign == '+'){
        sos[s] = w;            //machine at material source at start of shift
       }
    else if(sign == '-'){
         cht[s] = w;           //machine at dump-point at start of shift
         }
    else {
        printf("Improper equipment location (check sign in input filename)\n");
        getch();
        exit(EXIT_FAILURE);
        }
     mach_type[s] = tolower(mach_type[s]);
    type[0] = mach_type[s];          //mach_type is prefered if a STRING not char
    feed(type[0]);
     alpha_1[s] = man[0];            //loading features (a_1,b_,g_1)
     beta_1[s] = man[1];
     gamma_1[s] = man[2];
     alpha_2[s] = man[3];            //hauling features (a_2,b_2,g_2)
     beta_2[s] = man[4];
     gamma_2[s] = man[5];
     alpha_3[s] = man[6];            //dumping features (a_3, b_3,g_3)
     beta_3[s] = man[7];
     gamma_3[s] = man[8];
     alpha_4[s] = man[9];            //running empty (a_4, b_4, g_4)
     beta_4[s] = man[10];
     gamma_4[s] = man[11];
     cap[s] = man[12];               //machine capacity/b_d/speeds
     rate[s] = man[13];              //machine efficiency rating
     opcost[s] = man[14];            //operating cost of machine per hour
   if(cap[s]<minn) minn = cap[s];      //finding the smallest machine

if(sign=='+'){
  for (d=0; d<level_stopes[k]; d++){
   if (sos[s]==d){
        que[k][d][equ_assign[k][d]] = mach_type[s]; //assign type to queue slot
        sequnt[k][d][equ_assign[k][d]] = s;
        equ_assign[k][d] += 1;
        mach[s] = 6;                 //flag for ready for loading
        star[s] = sos[s];            //start location of travel
       }
      }
     }
else{
  for (d=0; d<level_bins[k]; d++){
   if (cht[s]==d){                   //equipment at draw areas
        bque[k][d][bin_assign[k][d]] = mach_type[s]; //assign a type to a queue slot
        bsequnt[k][d][bin_assign[k][d]] = s;
        bin_assign[k][d] += 1;
       mach[s] = 4;                  //flag for ready to travel empty
       star[s] = cht[s];             //start location of travel
      }
     }
    }
    if(randomm() > 0.5)
    st_tim[s] = start_up + randomm() * start_up;  //start_up times
    else st_tim[s] = start_up - randomm() * start_up;
    delay(40);
   }
  sumd +=level_mach[k];
 }
```

```
fscanf(fp4,"%d %d %d %d %d\n",&b_lunch,&e_lunch,&shift_end,&icrem,&smp_tim); //shift times
fscanf(fp4,"%f %f %f\n",&delta1,&delta2,&delta3);      //set control parameters
fscanf(fp4,"%f %f %f %f\n",&eta1, &eta2, &eta3,&theta);
fscanf(fp4,"%f %f\n",&MTBF, &MRT);      // mean time between failure and mean repair time

for (i = 0; i < sumd; i++)
fscanf(fp4,"%f",&acc_tim[i]);          //hrs machine running since last b/down
fclose(fp4);

//opening the mine network filename
outtextxy(200, getmaxy()/2,"Enter mine layout file ");
 gotoxy(25,18); gets(layout);

if((fpt=fopen(layout,"r"))==NULL){
   sprintf(msg,"Failed to open the haulage \"%s\" data file\n",layout);
   outtextxy(200,getmaxy()/2+50,msg);
   getch();
   exit(EXIT_FAILURE);
  }

// reading in the mine layout: distances and permitted speeds
 for(k=0; k<levels; k++){
     fscanf(fpt,"%f %f\n",&rmpds[k], &rmpv[k]);      //inter-level distance
   for (i=0; i<level_stopes[k]; i++){
    for (j=0; j<level_bins[k]; j++){                 //level travelling
     fscanf(fpt,"%f %f %f %d",&dist[k][i][j],&VEL_1[k][i][j],&VEL_2[k][i][j],&route[k][i][j]);
     }
   }
  }

  for (k=0; k<levels; k++){      //distances from exit of ramp to load points
    for(i=0;i<level_stopes[k]; i++){
     fscanf(fpt,"%f %f", &disc[k][i],&vect[k][i]);
     }
   }

  for (k=0; k<levels; k++){  //distances between load points used in case of bkage
    for(i=0;i<level_stopes[k]; i++){
     for(j=0; j<nodes[k]; j++){
      fscanf(fpt,"%f %f",&metr[k][i][j], &velo[k][i][j]);
     }
    }
   }
 fclose(fpt);
/*--------------------------------------------------------------*/
 outtextxy(200,getmaxy()/2+50,"Is a mine plan available? Y or N? ");
 gotoxy(25,21); scanf("%c",&mine);

 if(tolower(mine)!='n'){
   outtextxy(200, getmaxy()/2+100, "Enter the mine plan file ");
   gotoxy(25,24); gets(chabva);
```

```
if(gets(chabva)==NULL) outtextxy(200,getmaxy()/2+150,"Can't open file ");
if((fp6 = fopen(chabva,"r"))== NULL){
    outtextxy(200,getmaxy()/2+150,"Fail to open file");
    getch();
    exit(0);
   }

 for (k = 0;  k<levels; k++){
 fscanf(fp6, "%d\n",&no_lines[k]);              /* read number of tunnels present */

 for (i = 0; i< no_lines[k]; i++)
    fscanf(fp6,"%d %d %d %d\n",&X1[k][i],&Y1[k][i],&X2[k][i],&Y2[k][i]);

 for (i = 0; i < level_bins[k]; i++)            /* reading co-ordinates of bins */
     fscanf(fp6,"%d %d", &bstx[k][i], &bsty[k][i]);

 for (i = 0; i < level_stopes[k]; i++)          /* reading co-ordinates of stopes */
    fscanf(fp6, "%d %d", &sstx[k][i], &ssty[k][i]);
 }                              /* end of drawing a level network data */
 fclose(fp6);                    /* close input file */
 }                              /* end of case with a programmed file of layout */
/* --------- selection of dispatch rule to apply ----------------------- */
run: outtextxy(200, getmaxy()/2+170,"Press any key...");
   getch();
   cleardevice();
   setcolor(YELLOW);
   rectangle(2,2,getmaxx()-3, getmaxy()-3);
   setfillstyle(SOLID_FILL, 4);
   bar(30, 30, 600, 440);
  outtextxy(200,80,"CHOICE DISPATCH POLICY:");
  outtextxy(200,100,"1 = Work_Quality_Utility");
  outtextxy(200,120,"2 = Minimun_System_Slack");
  outtextxy(200,140,"3 = Earliest_Expected_Service_Time");
  outtextxy(200,160,"4 = Shortest_Travel_Times");
  outtextxy(200,180,"5 = Max_Work_Quality_Min_Slack_Rule");
  outtextxy(200,200,"6 = Maximun_Critical_Ratio");

  gotoxy(25,15); scanf("%d",&choice);
      switch(choice){
            case 1: for(i=0; i<levels;i++){
                  level_rule[i] = choice;
                  }
                  break;
            case 2: for(i=0; i<levels;i++){
                  level_rule[i] = choice;
                  }
                  break;
            case 3: for(i=0; i<levels;i++){
                  level_rule[i] = choice;
                  }
                  break;
            case 4: for(i=0; i<levels;i++){
                    level_rule[i] = choice;
                    }
                    break;
            case 5: for(i=0; i<levels;i++){
                    level_rule[i] = choice;
                    }
                    break;
            case 6: for(i=0; i<levels; i++){
                    level_rule[i] = choice;
                    }
                    break;
            default: clrscr();
                     outtextxy(200,200,"Please enter the proper rules:1,2,3,4,5 or 6");
                    goto run;
        }
  outtextxy(200,240,"Enter production stats output filename");
  gotoxy(25,18); gets(results);
 if( gets(results) == NULL) printf("Cannot read the output filename\n");

  if((fp3=fopen(results,"w"))==NULL){    //opening output filename
    gotoxy(25,21);sprintf("Failed to create output file:\" %s\"",results);
    outtextxy(200,280,msg);
    getch();
    exit(EXIT_FAILURE);
   }
  outtextxy(200,290, "Enter the event record filename");
  gotoxy(25,22); gets(gogo);

  if((fp5=fopen(gogo,"w"))==NULL){          //openning output filename
   gotoxy(25,24); sprintf(msg,"Failed to create output file:\" %s \"",gogo);
   outtextxy(200,310,msg); getch();
   exit(EXIT_FAILURE);
   }
    cleardevice();
    setcolor(YELLOW);
    rectangle(2,2,getmaxx()-3, getmaxy()-3);
    setfillstyle(SOLID_FILL, 4);
    bar(30, 30, 600, 440);

outtextxy(140,60,"Priority Rules:");
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(160,80,"123 = Q_T_U=");
outtextxy(160,90,"126 = Q_T_>_U");
outtextxy(160,100,"234 = T_U_>_Q");
outtextxy(160,110,"135 = Q_U_>_T");
outtextxy(160,120,"156/165 = Q_>_T_U");
outtextxy(160,130,"246/264 = T_>_Q_U");
outtextxy(160,140,"345/354 = U_>_Q_T");
outtextxy(160,150,"159 = Q_>_T_>_U");
outtextxy(160,160,"168 = Q_>_U_>_T");
outtextxy(160,170,"249 = T_>_Q_>U");
outtextxy(160,180,"267 = T_>_U_>_Q");
outtextxy(160,190,"357 = U_>_T_>_Q");
outtextxy(160,200,"348 = U_>_Q_>T");

settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
outtextxy(160,265,"Enter the priority rule (3 figure integer)");
maint();    // set up a menu box for priority selections
setviewport(0,0,getmaxx()-1, getmaxy()-1,CLIP_ON);
gotoxy(20,19);scanf("%d",&rule);
outtextxy(160,310,"Press any key to continue");
      getch();
   SAP_TIME = smp_tim;
  first = time(NULL);
 /* ---------------------------------------------------------------*/
   for (i=0; i< sumd; i++){    /* create the machine icons and put in memory */
      animate(i);
      }
/* ------------ units conversion to percentage ------------------------ */
   delta1 = delta1*100; delta2 = delta2*100; delta3 = delta3*100;
   eta1 = eta1*100; eta2 = eta2*100; eta3 = eta3*100;

/* -------------------------------------------------------------------*
 *                     START SIMULATION                               *
 * -------------------------------------------------------------------*/
  for (clock=0; clock<(shift_end); clock+=icrem){

  setviewport(0,0,getmaxx()-XOFFSET,getmaxy()-YOFFSET,CLIP_ON);
  setbkcolor(BLUE);
  setcolor(YELLOW);

  setlinestyle(SOLID_LINE,0,THICKNESS1);
  rectangle(2,2,getmaxx()-2,getmaxy()-2);       /* settings of screen definition */
  setfillstyle(SOLID_FILL, 4);
  bar(30, 30, 600, 440);                        /* brown inner screen          */

  settextstyle(TRIPLEX_FONT, HORIZ_DIR, 2);
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(getmaxx()/2, 20, "SIMULATION IN PROGRESS");
  settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);    /* settings for viewports */

char mss[30];
if(tolower(mine)!='n'){                     /* mine layout plan */
for (k=0; k<levels; k++){
 for (j=0; j<no_lines[k];j++){
    network1(k,X1[k][j],Y1[k][j],X2[k][j],Y2[k][j]);
    }
    network2(k);
  }
 }
 sprintf(mss,"TIME NOW: %5.1f mins",(float)clock/60);
 outtextxy(getmaxx()/2, getmaxy()-20, mss);

  if (SAP_TIME <=0){
/* -----------------------------------------------------------------------*
 *              EVALUATION OF PLANNED vs ACTUAL PRODUCTION                      *
 * -----------------------------------------------------------------------*/

fin:  Agrandtn = 0; Pgrandtn = 0; TBlock = 0; TSlock = 0;
      Agrandgd = 0; Pgrandgd = 0; Agr = 0; Pgr = 0;

for (k=0;k<levels;k++){    // outputs statistics
 Gave[k]=0; ScLton[k]=0; ALton[k]=0; Accm[k]=0;
 Block[k]=0; Slock[k] =0; ScWton[k] = 0; Wton[k] = 0;

  for (i=0; i<level_stopes[k]; i++){
    if((sblock[k][i]==1)&&(tot_stope_cap[k][i]>=minn)){ //test for stope blockages
         dd = randomm();
       if (dd < BLOCKAGE){              //probability of 5%/#of sample times
         delay(50);
         sblock[k][i] = -1;
         dd = randomm();
         fprintf(fp5,"Stope #%d on Level %d blocked at time %d\n", i+1,k+1, clock);
         if(dd < 0.5){
           sbk_tm[k][i] = MID - (MID * randomm()); //blockage duration
           s_block[k][i]+=sbk_tm[k][i];
           }
         else {sbk_tm[k][i] = MID + (MID * randomm());
             s_block[k][i]+=sbk_tm[k][i];
             }
          }
       }
      Slock[k] += s_block[k][i];

 if(sch_ton[k][i] > 0){                    //calculation of deviations &
   Wton[k] += wton[k][i];                    //summing waste in last window of work
   ALton[k] += Lton[k][i];                   //updating production stats on ore
     Gave[k] += (ave_gd[k][i] * Lton[k][i]); //weighted grades being used here
      target_ton[k][i] = (sch_ton[k][i] * clock)/shift_end;
      if(stope_grade[k][i]==0){
         ScWton[k] += target_ton[k][i];
         d1[k][i] = (wton[k][i] - target_ton[k][i])/target_ton[k][i]; //ton dev'n
         }
       else {
           ScLton[k] += target_ton[k][i];
           d1[k][i] = (Lton[k][i] - target_ton[k][i])/target_ton[k][i]; //ton dev'n
           g1[k][i]= ((ave_gd[k][i])-(stope_grade[k][i]))/stope_grade[k][i]; //grd dev'n
           Accm[k] += (target_ton[k][i] * stope_grade[k][i]);
          }
   if(cnt2[k][i] != 0){
   if (cnt2[k][i] <=SPAN){                  //moving average grade at each stope
```

```
      j = cnt2[k][i] - 1;
     for (w = 0; w < (j+1); w++){
       tot[k][i][j] += au_tal[k][i][w];
       too[k][i][j] += d_ton[k][i][w];
       }
       if (too[k][i][j]!=0) g_move[k][i] = tot[k][i][j]/too[k][i][j];
       fprintf(fp3,"Moving average grade of stope %d^%d is %4.3f\n",k+1,i+1,g_move[k][i]);
     }
 else {
     for (int w = 0; w<(j+1); w++){
         tot[k][i][j] += au_tal[k][i][w];
         too[k][i][j] += d_ton[k][i][w];
         }
        if(too[k][i][j] - too[k][i][j-SPAN] != 0)
         g_move[k][i] = (tot[k][i][j] - tot[k][i][j - SPAN])/\
          (too[k][i][j] - too[k][i][j - SPAN]);
       fprintf(fp3,"Moving average grade of stope %d^%d is %4.3f\n",k+1,i+1,g_move[k][i]);
     }
    }
   }
  }    //end of level_stopes

 for (i=0; i<level_bins[k]; i++){ //test for bin blockages
    if((bblock[k][i]==1)&&(tot_bin_cap[k][i]>=minn)){
        dd = randomm();
          if (dd < BBLOCKAGE){          //probability of 8%/# of sample windows
              bblock[k][i] = -1;
             dd = randomm();
             delay(50);
            fprintf(fp5,"Bin #%d on Level %d blocked at time %d\n", i+1,k+1, clock);
           if(dd > 0.5){
               bbk_tm[k][i] = LOWL*(MID - (MID * randomm())); //blockage duration
               b_block[k][i] += bbk_tm[k][i];
             }
            else {bbk_tm[k][i] = LOWL*(MID + (MID * randomm()));
                 b_block[k][i] += bbk_tm[k][i];
                }
             }
          }
        Block[k] += b_block[k][i];
      }
/*--------------- calculation of individual level totals -------------------*/

 if(ScLton[k]>0){   //level target grade
    LTgrd[k] =  (Accm[k]/ScLton[k]);
    }

 if(ALton[k]>0){   //level actual grades
     LAgrd[k] = Gave[k]/ALton[k];  //ratio control: cf est. mean grade
   }
 if((ScLton[k]>0)||(ScWton[k]>0)){   // ton deviation from target
    d2[k] = ((ALton[k] + Wton[k])-(ScLton[k]+ScWton[k]))/(ScLton[k]+ScWton[k]); //level ton
dev'nt
    }

 if(LTgrd[k]>0){   //level grades deviation from target
   g2[k] = (LAgrd[k]-LTgrd[k])/LTgrd[k]; //level grade dev'n
   }

//------------- accumulating the grand totals production updates -----------
    Agrandtn += ALton[k] + Wton[k];
    Pgrandtn += ScLton[k] + ScWton[k];
    Agrandgd += LAgrd[k] * ALton[k];
    Pgrandgd += LTgrd[k] * ScLton[k];
    Agr += ALton[k];
    Pgr += ScLton[k];
    TBlock += Block[k];        //total accum. bin blockage time to date
    TSlock += Slock[k];        //total accum. bin blockage time to date
}                              //end of levels accumulation

   WTlost = 0; Tbd_dn = 0;
for (s=0; s<sumd; s++){           //summation of idle times
  if(break_down(MTBF, MRT, acc_tim[s],status[s], *bdown)==1){
    if(mach[s]==4) bin[lev[s]][cht[s]] = 0;    //clear servers if mach breaks down
    if(mach[s]==1) stope[lev[s]][sos[s]] = 0;  // " "
      mach[s] = 7;                          //flag for machine on break down
      status[s] = 1;
      acc_tim[s] = 0.0;
      bd_time[s] = *bdown;                  //returned duration of being down
      bd_dn[s] += bd_time[s];
      fprintf(fp5,"Mach #%d has broken down at time %d\n", s+1,clock);
      }
   else acc_tim[s] +=(float)smp_tim/3600;  //accumulate run time without b/d
    WTlost += (wt_load[s] + wt_dump[s]);   //time-idle at source and destn
  Tbd_dn += bd_dn[s];                      //total accumulated break down times
 }

if (clock < b_lunch){                     //equip utilization
       utilize = (float)sumd*clock;
       utilize = utilize - Tbd_dn - WTlost;
       utilize = utilize/((float)sumd*clock);
       utilize = utilize * 100.0;
      }
  else if (clock > e_lunch){
        utilize = (float)(e_lunch - b_lunch);
        utilize = (float)sumd * (clock - utilize) - Tbd_dn - WTlost;
        utilize = utilize/((float)sumd*(clock - e_lunch + b_lunch));
        utilize = utilize * 100.0;
       }

 if(clock < b_lunch){
    available = (float)sumd * clock - Tbd_dn;
    available = available/((float)sumd*clock);
    available = available * 100.0;
    }
 else if(clock > e_lunch){
     available = (float)(e_lunch - b_lunch );
     available = (float)sumd * (clock - available) - Tbd_dn;
     available = available/((float)sumd * (clock - e_lunch + b_lunch));
     available = available * 100.0;
     }
/*----------- criteria measurements to date -----------------------------*/
     prodeff = 100*(Agrandtn - Pgrandtn)/Pgrandtn; //deviations from targets
     gradeff = 100*((Agrandgd/Agr) - (Pgrandgd/Pgr))/(Pgrandgd/Pgr); //    """""
     gdz = (Agrandgd/Agr);

    tall_ton[tally] = Agrandtn;
    plan_ton[tally] = Pgrandtn;
    tall_grd[tally] = gradeff;
    tally += 1;

/* -------------------------------------------------------------------*
 *              OUTPUT OF INTERVAL STATISTICS                       *
 * -------------------------------------------------------------------*/
 if(((clock+ending)>=shift_end)||(pedza==1000)){
     fprintf(fp3,"\n\n-------------------> SUMMARY PRODUCTION REPORTS
<-------------------\n\n");
     }
 else {
     fprintf(fp3,"\n\n----------------> PRODUCTION STATISTICS TO DATE <----------------\n\n");
     }

 fprintf(fp3,"\t\t\tTIME IS NOW: %d\n\n",(clock+ending));
 ptr = rules(choice);
 fprintf(fp3,"\tThe dispatch rule in use is: \" %s \"\n\n",ptr);
 fprintf(fp3,"\tProduction deviation = %5.2f \%\n",prodeff);
 fprintf(fp3,"\n\tPlan_grd = %5.2f\tActual grade = %5.2f\n",goom,gdz);
 fprintf(fp3,"\tGrade deviation = %5.2f \%\n",gradeff);
 fprintf(fp3,"\tAvailability = %5.2f \%",available);
 fprintf(fp3,"\tUtilization  = %5.2f \%\n\n",utilize);

 fprintf(fp3,"\tProduction Reconciliation by Levels or Work Districts\n");

fprintf(fp3,"\t==========================================
=============\n\n");
 fprintf(fp3,"\t\tLevel    Pln_Ton        Act_Ton         Devn \n");
 fprintf(fp3,"---------------------------------------------------------------\n");

 for (k=0; k<levels; k++){
  fprintf(fp3,"Tons:Ore\t%d\t%5.1f\t\t%5.1f\t\t%5.2f\n",k+1,ScLton[k],ALton[k],d2[k]);
  fprintf(fp3,"Grades:\t\t%d\t%5.2f\t\t%5.2f\t\t%5.2f\n",k+1,LTgrd[k],LAgrd[k],g2[k]);
  if(ScWton[k]!=0)
```

```
fprintf(fp3,"Tons:Waste\t%d\t%5.1f\t\t%5.1f\t\t%5.2f\n",k+1,ScWton[k],Wton[k],(Wton[k]-ScWton[k
])/ScWton[k]);
 }
 fprintf(fp3,"------------------------------------------------------------------\n");
 fprintf(fp3,"\n\t\tMaterial Sources Production Reconcilation\n");
 fprintf(fp3,"\t\t==========================================
===\n");
 fprintf(fp3,"\nMt_type  Lev/Sp  Tar_Ton    Act_Ton    Dev_Ton    Sp_Ton_left\n");
 fprintf(fp3,"--------------------------------------------------------------\n");

 for (k=0; k<levels; k++){
 for (i=0;i<level_stopes[k]; i++){
 if((mat_flag[k][i] =='o')||(mat_flag[k][i] =='O'))
   fprintf(fp3,"O:  \t%d^%d         %4.1f        %4.1f        %5.3f
%5.1f\n",k+1,i+1,target_ton[k][i],Lton[k][i],d1[k][i],tot_stope_cap[k][i]);
 else
    fprintf(fp3,"W:\t%d^%d         %4.1f        %4.1f        %5.3f
%5.1f\n",k+1,i+1,target_ton[k][i],wton[k][i],d1[k][i],tot_stope_cap[k][i]);
  }
 }
fprintf(fp3,"--------------------------------------------------------------\n");

fprintf(fp3,"\n\tOre Sites Grade Variations From Targets\n");
fprintf(fp3,"\t=========================================
\n");
fprintf(fp3,"\nLev/Sp  Tar_Grade     Act_Grade     Grade_Dev\n");
fprintf(fp3,"----------------------------------------------\n");

 for (k=0; k<levels; k++){
 for (i=0;i<level_stopes[k]; i++){
  if((mat_flag[k][i] == 'o')||(mat_flag[k][i] =='O'))
     fprintf(fp3,"%d^%d\t%5.2f\t\t%5.2f\t\t%5.3f\n",k+1,
i+1,stope_grade[k][i],ave_gd[k][i],g1[k][i]);
  }
 }
fprintf(fp3,"----------------------------------------------\n");
fprintf(fp3,"\nBin no.  Tons Dumped     Bin Space Left \n");
fprintf(fp3,"-----------------------------------------\n");
 for (k=0; k<levels; k++){
  for (i=0; i<level_bins[k]; i++){
     fprintf(fp3,"%d^%d\t%5.1f\t\t%5.1f\n",k+1,i+1,dump_ton[k][i],tot_bin_cap[k][i]);
   }
  }
fprintf(fp3,"-----------------------------------------\n");

fprintf(fp3,"\n\tProduct Mix in the Bins\n");
fprintf(fp3,"\t==============================\n\n");
fprintf(fp3,"Bin no.      Sources of Dumped Material \n");
fprintf(fp3,"----------------------------------------------\n");
for( k=0; k<levels; k++){
 for(j=0; j<level_bins[k]; j++){
     fprintf(fp3,"%d^%d\t",k+1,j+1);
       for (p=0; p<cnt3[k][j];p++){
           fprintf(fp3,"<%3d>", orig[k][j][p]+1);
         }
         fprintf(fp3,"\n%d^%d\t",k+1,j+1);
        for (p=0; p<cnt3[k][j];p++){
             fprintf(fp3,"%3.2f=",rungd[k][j][p]);
      }
     fprintf(fp3,"\n");
    }
  fprintf(fp3,"\n");
}
fprintf(fp3,"--------------------------------------------\n\n");

fprintf(fp3,"Equipment Utilization and Production Costs\n");
fprintf(fp3,"==========================================
=\n\n");
 fprintf(fp3,"Mach   \t\ttrv_FUL\ttrv_EMP\twt_times Cost_$/ton\n");
 fprintf(fp3,"-----------------------------------------------\n");
  totcost = 0.0;
  for (s=0; s<sumd;s++){
       fprintf(fp3,"%s\t\t%5.2f\t%5.2f\t%5.2f\t%5.3f\n",name[s],tfull[s],tempty[s],\
       (wt_load[s]+wt_dump[s]),(opcost[s]*clock/3600)/lhd_ton[s]);
    if(clock >e_lunch)
       totcost +=((opcost[s]/3600)*(clock-(e_lunch-b_lunch)))/lhd_ton[s];
    else  totcost += ((opcost[s]/3600)*clock)/lhd_ton[s];
       }
fprintf(fp3,"Average unit operating cost:=\t\t$%5.3f/ton\n", totcost/sumd);
fprintf(fp3,"-----------------------------------------------\n");
empt=0; sumt=0;
fprintf(fp3,"\nLevel     Tot_Dist_Full   Tot_Dist_Emp      Tot_Wait_Time\n");
fprintf(fp3,"-------------------------------------------------------\n");
 for( k=0; k<levels; k++){
   fprintf(fp3,"%d\t\t%4.1f\t\t%4.1f\n",k+1,travD[k],travDE[k]);
       empt += travDE[k];
       sumt += travD[k];
    }
fprintf(fp3,"-------------------------------------------------------\n");
fprintf(fp3,"Totals:\t\t%4.1f\t\t%4.1f\t\t%4.2f\n",sumt,empt,WTlost);
fprintf(fp3,"-------------------------------------------------------\n\n");
fprintf(fp3,"\n\tOperation Status: Equipment and Production Centres\n");
fprintf(fp3,"\t=========================================
==========\n");
fprintf(fp3,"\nMach  \t\tStatus\t\tTime broken\tTonnage\n");
fprintf(fp3,"----------------------------------------------------------\n");
   for(i=0; i<sumd; i++){
     fprintf(fp3,"%s\t\t%d\t\t%5.2f\t\t%5.2f\n",name[i],mach[i],bd_dn[i],lhd_ton[i]);
     }
fprintf(fp3,"----------------------------------------------------------\n");
fprintf(fp3,"\nStope  \t\tStatus\t\tTime Blocked\n");
fprintf(fp3,"-------------------------------------------\n");
      for (k=0; k<levels; k++){
       for(i=0; i<level_stopes[k];i++){
          fprintf(fp3,"%d^%d\t\t%d\t\t%5.2f\n",k+1,i+1,sblock[k][i],s_block[k][i]);
        }
      }
fprintf(fp3,"-------------------------------------------\n");

fprintf(fp3,"\nBin  \t\tStatus\t\tTime Blocked\n");
fprintf(fp3,"-------------------------------------------\n");
  for (k=0; k<levels; k++){
   for(i=0; i<level_bins[k];i++){
     fprintf(fp3,"%d^%d\t\t%d\t\t%5.2f\n",k+1,i+1,bblock[k][i],b_block[k][i]);
    }
  }
fprintf(fp3,"-------------------------------------------\n");
fprintf(fp3,"\nTotal time stopes blocked  = %5.2f\n",TSlock);
fprintf(fp3,"Total time bins blocked    =  %5.2f\n",TBlock);
fprintf(fp3,"Total time machines b_down =  %5.2f\n",Tbd_dn);
fprintf(fp3,"Total time stopes not available =  %d\n",spt_tim);
fprintf(fp3,"Total time bins not available =  %d\n",dpt_tim);

fprintf(fp3,"\n=========================================
=================================\n");
if(((clock+ending)>=shift_end)||(pedza==1000)){
    for (s=0; s<sumd; s++){
        overs += overtime[s];
        }

fprintf(fp3,"\nProduction ended at  %3d units\n",clock+ending-icrem);
if(clock>=shift_end) fprintf(fp3,"Total overtime done: %4d units\n",overs);
 fprintf(fp3,"\n\n-----------------> END OF SHIFT PRODUCTION SIMULATION
<-----------------\n\n");
  goto wedza;
  }
//End of printing interval statistics
/*-----------------------------------------------------------------------*
 *                  CONTROL PROCESS LOGIC                              *
 * ----------------------------------- -----------------------------------*/
if(clock < shift_end) goto inza;
sum: cleardevice();
     rectangle(2,2,getmaxx()-3, getmaxy()-3);
     setfillstyle(SOLID_FILL, 4);
     bar(30, 30, 600, 440);

   maint();     //again show the selection menu
   setviewport(0,0,getmaxx()-XOFFSET,getmaxy()-YOFFSET,CLIP_ON);
   settextstyle(TRIPLEX_FONT, HORIZ_DIR, 1);
   sprintf(msg,">>>> PROCESS SUMMARY NOW <<<<");
   outtextxy(200,35,msg);
   sprintf(msg,"TIME IS %4.1f mins",(float)clock/60);
   outtextxy(200,60,msg);
   settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
   settextjustify(LEFT_TEXT, TOP_TEXT);
if(clock<b_lunch){ sprintf(msg,"Time left before lunch is %4.1f mins",(float)(b_lunch-clock)/60);
   outtextxy(40,90,msg);
   }
else if(clock>e_lunch){ sprintf(msg,"Time left to end of shift is %4.1f
mins",(float)(shift_end-clock)/60);
   outtextxy(40,90,msg);
   }
if ((rule==123)||(rule==132)||(rule==213)||(rule==231)||(rule==312)||(rule==321)){
//P1=P2=P3
   if((prodeff>= -delta1)&&(utilize >=100-delta3)&&(fabs (gradeff) <= delta2)){
      sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
      outtextxy(40,120,msg);
      sprintf(msg,"\"%s\"",ptr);
       outtextxy(40,130,msg);
        sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
         outtextxy(40,140,msg);
         sprintf(msg,"To make changes: Enter Y/y");
          outtextxy(40,150,msg);
          aa = getche();
       if(tolower(aa)=='y')  goto treat;
        }
   else {
        sprintf(msg,"Out-of-control under dispatch rule:");
         outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
         outtextxy(40,130,msg);
         sprintf(msg,"and priority rule %d",rule);
         outtextxy(40,140,msg);
      goto test;
    }
   }
else if((rule==126)||(rule==216)){ //P1=P2>P3
   if((prodeff >= -delta1)&&(fabs(gradeff) <= delta2)){
      sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
       outtextxy(40,120,msg);
       sprintf(msg,"\"%s\"",ptr);
       outtextxy(40,130,msg);
       sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
        outtextxy(40,140,msg);
        sprintf(msg,"To make changes: Enter Y/y");
         outtextxy(40,150,msg);
          aa = getche();
       if(tolower(aa)=='y')  goto treat;
       }
   else {
        sprintf(msg,"Out-of-control under dispatch rule:");
         outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"and priority rule %d",rule);
        outtextxy(40,140,msg);
       goto test;
      }
    }
  else if((rule==234)||(rule==324)){ //P2=P3>P1
    if((prodeff >=-delta1)&&(utilize >=100-delta3)){
      sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
       outtextxy(40,120,msg);
       sprintf(msg,"\"%s\"",ptr);
       outtextxy(40,130,msg);
       sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
       outtextxy(40,140,msg);
       sprintf(msg,"To make changes: Enter Y/y");
       outtextxy(40,150,msg);
         aa = getche();
       if(tolower(aa)=='y')  goto treat;
       }
    else {
        sprintf(msg,"Out-of-control under dispatch rule:");
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"and priority rule %d",rule);
        outtextxy(40,140,msg);
       goto test;
       }
     }
  else if((rule==135)||(rule==315)){  //P1=P3>P2
    if((utilize >=100-delta3)&&(fabs(gradeff) <= delta2)){
       sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
       outtextxy(40,120,msg);
       sprintf(msg,"\"%s\"",ptr);
       outtextxy(40,130,msg);
       sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
       outtextxy(40,140,msg);
       sprintf(msg,"To make changes: Enter Y/y");
       outtextxy(40,150,msg);
         aa = getche();
       if(tolower(aa)=='y')  goto treat;
       }
    else {
       sprintf(msg,"Out-of-control under dispatch rule:");
       outtextxy(40,120,msg);
       sprintf(msg,"\"%s\"",ptr);
       outtextxy(40,130,msg);
       sprintf(msg,"and priority rule %d",rule);
       outtextxy(40,140,msg);
       goto test;
```

```
        }
      }
else if((rule==156)||(rule==165)){ //P1>P2=P3
 if((prodeff >= -(delta1+eta1))&&(utilize >=100-(delta3+eta3))&&(fabs(gradeff)<= delta2)){
      sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
      outtextxy(40,120,msg);
      sprintf(msg,"\"%s\"",ptr);
      outtextxy(40,130,msg);
      sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
      outtextxy(40,140,msg);
      sprintf(msg,"To make changes: Enter Y/y");
      outtextxy(40,150,msg);
        aa = getche();
      if(tolower(aa)=='y')  goto treat;
   }
   else {
       sprintf(msg,"Out-of-control under dispatch rule:");
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"and priority rule %d",rule);
        outtextxy(40,140,msg);
     goto test;
   }
 }
else if((rule==246)||(rule==264)){ //P2>P1=P3
 if((fabs(gradeff)<=(delta2+eta2))&&(prodeff >= -delta1)&&(utilize/100 >=100-(delta3+eta3))){
     sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
      outtextxy(40,120,msg);
      sprintf(msg,"\"%s\"",ptr);
      outtextxy(40,130,msg);
      sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
       outtextxy(40,140,msg);
       sprintf(msg,"To make changes: Enter Y/y");
        outtextxy(40,150,msg);
         aa = getche();
      if(tolower(aa)=='y')  goto treat;
      }
   else {
       sprintf(msg,"Out-of-control under dispatch rule:");
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"and priority rule %d",rule);
        outtextxy(40,140,msg);
       goto test;
    }
  }
else if((rule==345)||(rule==354)){ //P3>P1=P2
 if((utilize >=100-delta3)&&(prodeff>=-(delta1+eta1))&&(fabs(gradeff)<=(delta2+eta2))){
     sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE: ");
```

```
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
        outtextxy(40,140,msg);
        sprintf(msg,"To make changes: Enter Y/y");
        outtextxy(40,150,msg);
          aa = getche();
        if(tolower(aa)=='y')  goto treat;
         }
     else {
         sprintf(msg,"Out-of-control under dispatch rule: ");
          outtextxy(40,120,msg);
          sprintf(msg,"\"%s\"",ptr);
          outtextxy(40,130,msg);
          sprintf(msg,"and priority rule %d",rule);
          outtextxy(40,140,msg);
       goto test;
      }
     }
  else if((rule==159)||(rule==168)){ //P1>P2>P3
   if((prodeff >= -(delta1+eta1))&&(fabs(gradeff)<=delta2)){
        sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE: ");
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
        outtextxy(40,140,msg);
        sprintf(msg,"To make changes: Enter Y/y");
        outtextxy(40,150,msg);
          aa = getche();
        if(tolower(aa)=='y')  goto treat;
       }
     else {
         sprintf(msg,"Out-of-control under dispatch rule: ");
          outtextxy(40,120,msg);
          sprintf(msg,"\"%s\"",ptr);
          outtextxy(40,130,msg);
          sprintf(msg,"and priority rule %d",rule);
          outtextxy(40,140,msg);
         goto test;
       }
      }
  else if((rule==249)||(rule==267)){ //P2>P1>P3
    if((fabs(gradeff) <= (delta2+eta2))&&(prodeff>-delta1)){
       sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE: ");
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
         outtextxy(40,140,msg);
        sprintf(msg,"To make changes: Enter Y/y");
         outtextxy(40,150,msg);
          aa = getche();
       if(tolower(aa)=='y')  goto treat;
       }
   else {
         sprintf(msg,"Out-of-control under dispatch rule:");
         outtextxy(40,120,msg);
         sprintf(msg,"\"%s\"",ptr);
         outtextxy(40,130,msg);
         sprintf(msg,"and priority rule %d",rule);
         outtextxy(40,140,msg);
       goto test;
     }
   }
else if((rule==348)||(rule==357)){ //P3>P1>P2 / P3>P2>P1
  if((utilize >100 - delta3)&&(prodeff>= -(delta1+eta1))){
      sprintf(msg,"PROCESS IN CONTROL UNDER DISPATCH RULE:");
       outtextxy(40,120,msg);
       sprintf(msg,"\"%s\"",ptr);
       outtextxy(40,130,msg);
       sprintf(msg,"AND PRIOIRTY RULE: %d",rule);
        outtextxy(40,140,msg);
        sprintf(msg,"To make changes: Enter Y/y");
         outtextxy(40,150,msg);
          aa = getche();
       if(tolower(aa)=='y')  goto treat;
     }
   else {
        sprintf(msg,"Out-of-control under dispatch rule:");
        outtextxy(40,120,msg);
        sprintf(msg,"\"%s\"",ptr);
        outtextxy(40,130,msg);
        sprintf(msg,"and priority rule %d",rule);
        outtextxy(40,140,msg);
  /*  listing of the possible outcomes of each variable  */
test: if(gradeff < delta2){
         sprintf(msg,"Grade below target by Dev'tion of %4.1f %",gradeff);
         outtextxy(40,170,msg);}
      if(gradeff > delta2){
        sprintf(msg,"Grade above target by Dev'tion of +%4.1f %",gradeff);
        outtextxy(40,170,msg);}
      if((gradeff < 0)&&(gradeff > -delta2)){
         sprintf(msg,"Grade within T_range with Dev'tion of %4.1f %",gradeff);
         outtextxy(40,190,msg);}
      if((gradeff > 0)&&(gradeff < delta2)){
         sprintf(msg,"Grade within T_range with Dev'tion of +%4.1f %",gradeff);
         outtextxy(40,190,msg);}

      if(prodeff < -delta1){
         sprintf(msg,"Production is below target by %4.1f %",prodeff);
        outtextxy(40,210,msg);}
      if(prodeff > delta1){
        sprintf(msg,"Production is above target by %4.1f %",prodeff);
        outtextxy(40,210,msg);}
      if((prodeff < 0)&&(prodeff > -delta1)){
        sprintf(msg,"Production within T_range by Dev'tion of %3.1f %",prodeff);
        outtextxy(40,220,msg);}
      if((prodeff > 0)&&(prodeff < delta1)){
        sprintf(msg,"Production within T_range by Dev'tion of %3.1f %",prodeff);
        outtextxy(40,220,msg);}

      if(utilize <100 - delta3){
        sprintf(msg,"Utilization below target by %4.1f %",100 - utilize);
        outtextxy(40,240,msg);}
      if(utilize > 100-delta3){
        sprintf(msg,"Utilization within target range by Dev'tion of %4.1f %",100-utilize);
        outtextxy(40,240,msg);}

treat:  sprintf(msg,"Do you want to change the DISPATCH rule? Enter Y/N ");
        outtextxy(40,260,msg);
        gotoxy(25,10); aa = getche();
       if(tolower(aa)=='y'){
         sprintf(msg,"Enter the new rule (An integral value: 1-6!)");
         outtextxy(40,280,msg);
         gotoxy(25,12); scanf("%d",&choice);
         }
        sprintf(msg,"Do you want to change the PRIORITY rule? Enter Y/N");
        outtextxy(40,300,msg);
        gotoxy(25,14); aa = getche();
       if(tolower(aa)=='y'){
         sprintf(msg,"Enter the new rule (As an integral value!)");
          outtextxy(40,330,msg);
          gotoxy(25,16); scanf("%d",&rule);
          }
       else {sprintf(msg,"Do you want a GOAL PROGRAMMING re-schedule? Enter Y/N");
            outtextxy(40,350,msg);
            gotoxy(25,18); aa = getche();
            if(tolower(aa)=='y'){
                closegraph();
                goto wedza;              //terminate program if 'wedza'
         }
        }
       }
      }
     SAP_TIME = smp_tim;    //re-initialize next sample time
    sprintf(msg,"Press any key to continue..");
    outtextxy(getmaxx()/2,410,msg);
   getch();
graph: cleardevice();
       setbkcolor(BLUE);
       setviewport(0,0,getmaxx()-XOFFSET,getmaxy()-YOFFSET,CLIP_ON);
```

```
        plot_axes(delta2,eta2,shift_end,smp_tim,tally);
        sprintf(msg,"Press any key to continue..");
        outtextxy(getmaxx()/2,410,msg);
        getch();
        plot_axed((oreton+TWton),shift_end,smp_tim,tally);
        sprintf(msg,"Enter 1,2,3 to CONT., see GRADE, see SUMMARY..");
        outtextxy(getmaxx()/2,410,msg);
        gotoxy(25,25); aa = getche();

       if(aa=='2'){ goto graph;}
        else if(aa=='3'){goto sum;}
       else ;
}
 /* ----------------------------------------------------------------------*
 *          SIMULATION PROCESS AND DATA ACCUMULATION                      *
 * ----------------------------------------------------------------------*/
inza:
for (k=0; k<levels; k++){
 for (i=0; i<level_stopes[k];i++){ //assesses blockage time remaining
       stopefill(k,i,sblock[k][i]);               /* stope outlines */
  if (sblock[k][i] == -1){          //stope is blocked if true
       sbk_tm[k][i] -=icrem;        //decrememt blockage time
        if (sbk_tm[k][i] <= 0.0){
            sbk_tm[k][i] = 0.0;
            sblock[k][i] = 1;       //re-set to available
            fprintf(fp5,"Stope #%d on Level %d unblocked at time %d\n", i+1,k+1, clock);
            }
        }
     }
  for (i=0; i<level_bins[k]; i++){     //assesses bin block time remaining
    binfill(k,i,bblock[k][i]);                 /* bin outlines */
  if (bblock[k][i] == -1){              //bin is blocked if true
       bbk_tm[k][i] -=icrem;            //decrementing blockage time
     if (bbk_tm[k][i] <= 0.0){
          bbk_tm[k][i] = 0.0;
          bblock[k][i] = 1;         //re-set to available
          fprintf(fp5,"Bin #%d on Level %d unblocked at time %d\n", i+1,k+1, clock);
          }
      }
   }
}

k=0;                    /* re-initialize k as its affected by above loops */
move =0; mum = 0;        /* initializing location poster of bd and idle mach */
tick = 0; tock = 0;
for (s=0; s<sumd; s++){
   if(tolower(mine)!='n'){
     motion(lev[s],star[s],endr[s],TRFULL[s],TREMP[s],ttrip[s],red[s], mach[s],x_axis,y_axis);
     d = *x_axis;
     w = *y_axis;
     if((mach[s]==2)||(mach[s]==3)||(mach[s]==4)){
        putimage(d, w,buffer[s],3);
        putimage(d - 1, w,buffer[s],4);
       }
     else if ((mach[s]==5)||(mach[s]==6)||(mach[s]==1)){
        putimage(d, w,buffer[s],1);
        putimage(d - 1, w,buffer[s],3);
     }
     else if (mach[s]==8){ move += 1;
           putimage(120+(move*20),40,buffer[s],2);
           }
     else if (mach[s]==7){ mum += 1;
           putimage(120+(mum*20),60,buffer[s],0);
          }
     delay(50);
   }
  else {if((mach[s]==2)||(mach[s]==3)||(mach[s]==4)){tick += 1;
        putimage(120+(tick*20),100,buffer[s],3);
        }
     else if ((mach[s]==5)||(mach[s]==6)||(mach[s]==1)){tock += 1;
        putimage(120+(tock*20), 140,buffer[s],1);
     }
     else if (mach[s]==8){ move += 1;
           putimage(120+(move*20),40,buffer[s],2);
           }
     else if (mach[s]==7){ mum += 1;
           putimage(120+(mum*20),60,buffer[s],0);
          }
     delay(50);
   }

   if (st_tim[s] >= clock){       //work proceeds only after start-up delays finished
      wt_load[s] +=icrem;
      continue;
     }

   if(complete(levels)==1){   //work completed usually before shift_end
    pedza = 1000;
    goto home;
    }

  if (clock >= shift_end){
      goto home;            //end of shift: ensure machine complete current jobs
      }
/* ------------------------------------------------------------------*/
  while((clock >= b_lunch)&&(clock < e_lunch)){ //lunch break, increment time only
    if (bd_time[s] > 0){          //maintenace continues work through lunch break
       bd_time[s] -= icrem;
       if(bd_time[s]<0){
         bd_time[s] = 0;   //<--- coming out of service/breakdown
         fprintf(fp5,"Mach #%d on Level %d been repaired at time %d\n", s+1,k+1, clock);
         if(ld_tim[s] !=0){mach[s]=1;}   //re-assign to last machine activity
```

```
            else if(TRFULL[s] !=0){mach[s]=2;}
            else if(load[s] !=0){mach[s]=3;}
            else if(dp_tim[s] !=0){mach[s]=4;}
            else if(TREMP[s] !=0){mach[s]=5;}
            else if((ld_tim[s]==0)&&(TRFULL[s]==0)&&\
                  (load[s]==0)&&(dp_tim[s]==0)&&(TREMP[s]==0)) {mach[s]=6;}
            else mach[s] = 8;      //machine had stopped work when went on b/d
          }
        }
        goto mogo;
      }

   if (bd_time[s] > 0){          //checking for breakdown of equipment
         bd_time[s] -= icrem;
         if(bd_time[s]<0){
            bd_time[s] = 0;   //<--- coming out of service/breakdown
            fprintf(fp5,"Mach #%d on Level %d been repaired at time %d\n", s+1,k+1, clock);
            if(ld_tim[s] !=0){mach[s]=1;}   //re-assign to last machine activity
            else if(TRFULL[s] !=0){mach[s]=2;}
            else if(load[s] !=0){mach[s]=3;}
            else if(dp_tim[s] !=0){mach[s]=4;}
            else if(TREMP[s] !=0){mach[s]=5;}
            else if((ld_tim[s]==0)&&(TRFULL[s]==0)&&\
                   (load[s]==0)&&(dp_tim[s]==0)&&(TREMP[s]==0)) {mach[s]=6;}
            else mach[s] = 8;      //machine had stopped work when went on b/d
          }
        else continue;         //still on b/d
      }
  /* ------------------------------------------------------------------*/
   if(sos[s]!=-1){
      AVE = sos[s];    //location of machine (stopes)
      k=lev[s];
      }
   else if(cht[s]!=-1){
           DSAVE = cht[s];
           k=lev[s];    //location of machine: dump point
         }
  /* ------------------------------------------------------------------*/
   if ((mach[s]==6)&&(sblock[k][AVE]== -1)){  /* waiting for service & server down*/
        oldpos[s] = sos[s];
        sos[s] = stoop(k,sos[s],s,*EAT);      /* determine new server */
        turns = *EAT;
        temp2 = sbk_tm[k][oldpos[s]] - turns; /*if time diff small remain here */
     if(sos[s]==66){                      /* finished on this level */
         sos[s] = oldpos[s];
        goto xex;
        }
    else if (temp2 > turns){            /* need new server as waiting too large */
         TREMP[s] = *EAT;                   //simulated travel time on selected route
         ttrip[s] = *EAT;                //duration time of trip for graph
         tempty[s] += *EAT;                 //stats
        travE[k] += *EAT;                //stats
        if(sos[s] > oldpos[s])
        travDE[k] += metr[k][oldpos[s]][sos[s]];
        else travDE[k] += metr[k][sos[s]][oldpos[s]];
        j = equ_assign[k][sos[s]];
        que[k][sos[s]][j] = mach_type[s];
        sequnt[k][sos[s]][j] = s;
        z = 1;
        arang(k,oldpos[s],equ_assign[k][oldpos[s]],z,mach_type[s]); //arange current TYPE queue
        range(k,oldpos[s],equ_assign[k][oldpos[s]],z,s);  //re-arrange current machine NO. queue
        equ_assign[k][oldpos[s]] -= 1;               //decrement old queue length
        equ_assign[k][sos[s]] +=1;  //incrementing total machine queue at stope
        cht[s] = oldpos[s];              /* record origin */
        endr[s] = sos[s];                /* graphics usage: sos as dest'n */
        star[s] = cht[s];
        fprintf(fp5,"Travelling to other server. Current blocked %d^%d\n",\
              k+1,oldpos[s]+1);
        }
     else { sos[s] = oldpos[s];              //stay at same position
          wt_load[s] += icrem;
          continue;
        }
      }
   /* ------------- loading  and weighing and grade scanning    --------------*/

   if((mach[s]==6)&&(stope[k][AVE]==0)&&(sblock[k][AVE]==1)&&\
      (tot_stope_cap[k][AVE] >= minn)&&(dump_state(k,mat_flag[k][AVE]) > 0)){
      load[s] = load_cap(s,under_load,over_load);  //generate a load size
      cnt2[k][AVE]  += 1;                //loads from each stope
      j = cnt2[k][AVE] - 1;
      d_ton[k][AVE][j] = load[s];        //array of loads
      lhd_ton[s] += load[s];            //accumulating machine production
     if (mat_flag[k][AVE]!='w'){         //waste flagging
         Lton[k][AVE] += load[s];
         tot_stope_cap[k][AVE] -= load[s];
       if(tot_stope_cap[k][AVE] < minn){  //scheduled work finished at stope
         sblock[k][AVE] = -2;              //flag of depleted stope
        }
        find = simg(j+1);                  //generate a grade variability based on tons drawn
        mgrade[k][AVE] = stope_grade[k][AVE] + find;  //add to stope average:i.e. load grade
         au_tal[k][AVE][j] = load[s] * mgrade[k][AVE]; //array of contained metal
          wt_tal[k][AVE] += load[s] * mgrade[k][AVE];   //sum of contained metal in loads
           ave_gd[k][AVE] = wt_tal[k][AVE]/Lton[k][AVE];  //weighted average grade
         }      // end of ore grade sensing
    else {       //waste material
         wton[k][AVE] += load[s];
         tot_stope_cap[k][AVE] -= load[s];
       if(tot_stope_cap[k][AVE] < minn){  //scheduled work finished at WASTE stope
         sblock[k][AVE] = -2;  //stope flag depleted flag
       }
      }
```

```
/*   --------- generating the loading time (non-zero value) --------------- */
   ld_tim[s] = weibull(alpha_1[s],beta_1[s],gamma_1[s]); //generate loading time
       mach[s]=1;                          //machine being loaded
       stope[k][AVE]=1;                     //stope servicing: flagging
       fprintf(fp5,"Mach #%d start loading at stope %d^%d at time %d\n", s+1,k+1,AVE+1, clock);
       fprintf(fp5,"d_ton[%d%d%d] is %5.2f\n",k+1,AVE+1,cnt2[k][AVE],d_ton[k][AVE][j]);
       fprintf(fp5,"g_aveg[%d%d] is %5.2f\n",k+1,AVE+1,ave_gd[k][AVE]);
   continue;
  }
 /* -------------------------------------------------------------------*/
  else if ((mach[s]==6)&&(stope[k][AVE]==1)){ //machine waits for turn to load
          wt_load[s] += icrem;
          continue;
         }
/* ---------------------------------------------------------------------*/
  else if((mach[s]==1)&&(ld_tim[s]>0)){ //machine continue to load
          ld_tim[s] -= icrem;
        if(ld_tim[s] > 0)  continue;
        else {                          //finish loading in this increment
          stope[k][AVE] = 0;          //free server
          ld_tim[s] = 0;
          fprintf(fp5,"Mach #%d finish loading at stope %d^%d at time %d\n", s+1,k+1,AVE+1,
clock);
        }
    }
/* ----------------------------------------------------------------------*/
/*                          FULL TRAVEL                             */
/* ----------------------------------------------------------------------*/
 if((ld_tim[s]<=0)&&(mach[s]==1)){     //finish loading: request for destination now
  z = 1;
    cht[s] = policies(choice,z,s,k,AVE,mat_flag[k][AVE],*EAT);

  if(cht[s]==999){
     wt_dump[s] += icrem;
     dpt_tim += icrem;
     fprintf(fp5,"Temp. bin unavailability on Level %d at time %d\n",k+1, clock);
     cht[s] = -1;                   //can't travel therefore re-initialize
     continue;
    }
  else if ((cht[s] == 66)||(cht[s]==99)){
       if (cht[s] == 99)
           fprintf(fp5,"\tAll Bins on Level[%d] are full at time %d\n",k+1,clock);
       KATE[k] = cht[s];        /* flag current level (blocked/finished) */
xex:   next = adjunt(k,clock, *TEA, s, shift_end,levels);
     if(next == 155){                   /* no more work areas: done */
        mach[s] = 8;
        arang(k,AVE,equ_assign[k][sos[s]],z,mach_type[s]); //arange current TYPE queue
        range(k,AVE,equ_assign[k][sos[s]],z,s);  //re-arrange current machine NO. queue
        equ_assign[k][sos[s]] -= 1;               //decrementing total queue
        cht[s] = -1;
        fprintf(fp5, "Machine No. %d finished work at %d on level %d\n", s+1, clock,k+1);
```

```
      }                              /* adequate machines on NEXT */
    else if ((next!=155)&&(to_assign(next,level_mach[next])!=1)){
       oldstate[s] = mach[s];
       mach[s] = 8;
       arang(k,AVE,equ_assign[k][sos[s]],z,mach_type[s]); //arange current TYPE queue
       range(k,AVE,equ_assign[k][sos[s]],z,s);  //re-arrange current machine NO. queue
       equ_assign[k][sos[s]] -= 1;              //decrementing total queue
       cht[s] = -1;
       fprintf(fp5, "Machine No. %d finished work at %d on level %d\n", s+1, clock,k+1);
    }
    else if ((to_assign(next,level_mach[next])==1)&&(comp(next)==2)){
wiz:    mach[s] = 9;
       TREMP[s] = *TEA;
       lev[s] = next;
       travE[next] += TREMP[s];
       travDE[next] += rmpds[next];
       tempty[s] += TREMP[s];
       level_mach[k]  -= 1;         /* decrement on current section */
       level_mach[next] +=1;        /* increment on next section */
       arang(k,AVE,equ_assign[k][sos[s]],z,mach_type[s]); //arange current TYPE queue
       range(k,AVE,equ_assign[k][sos[s]],z,s);  //re-arrange current machine NO. queue
       equ_assign[k][sos[s]] -= 1;              //decrementing total queue
       sos[s] = -1;
       fprintf(fp5,"Ramp_travel time = %5.2f To Level %d of machine
%d\n",TREMP[s],next+1,s+1);
      }
    }                    /* end cht[s]== 66 or 99 if segment */
 else{                   /* MACHINE FIND DESTINATION ON SECTION */
  DSAVE = cht[s];        /* give dump destination & travel time there */
  TRFULL[s] = *EAT;      /* time to travel loaded to destn */
  ttrip[s] = *EAT;       /* fixed trip time for graph */
  endr[s] = cht[s];      /* record destination */
  star[s] = sos[s];      /* record origin */
  tfull[s] += TRFULL[s];        /* stats */
  travF[k] += TRFULL[s];        /* accumulate times & dist travelled */
  travD[k] += dist[k][AVE][DSAVE];
  j = bin_assign[k][DSAVE];
  bque[k][DSAVE][j] = mach_type[s]; /* machine type location in queue */
  bsequnt[k][DSAVE][j] = s;
  dump_ton[k][DSAVE] += load[s];  /* increment load-capacity simulated during loading*/
  tot_bin_cap[k][DSAVE] -= load[s];
  fprintf(fp5,"TRFULL[%d] = %5.2f from stope %d to bin %d^%d at time %d\n",\
       s+1,TRFULL[s],AVE+1,k+1,DSAVE+1,clock);

  if(tot_bin_cap[k][DSAVE] < minn){
     fprintf(fp5,"Bin[%d%d] full at time %d\n",k+1,DSAVE+1,clock);
     bblock[k][DSAVE] = -2;        /* flag for a filled bin */
  }
   cnt3[k][DSAVE] +=1;            /* counts of loads dumped in bin */
   p = cnt3[k][DSAVE] - 1;
   orig[k][DSAVE][p] = sos[s];     /*record origin of load of s_machine */
```

```
if(mat_flag[k][AVE] == 'o')
   rungd[k][DSAVE][p] = mgrade[k][AVE];  /* record quality of that load */
else rungd[k][DSAVE][p] = stope_grade[k][AVE];

     arang(k,sos[s],equ_assign[k][sos[s]],z,mach_type[s]); /* arange current TYPE queue */
     range(k,sos[s],equ_assign[k][sos[s]],z,s);  /* re-arrange current machine NO. queue*/
     equ_assign[k][sos[s]] -= 1;                /* decrement old queue length  */
     bin_assign[k][DSAVE] += 1;        /* increment machines in queue @ server */
     in_road[k][sos[s]][DSAVE] += 1;  /* increase machines in road section */
     sos[s] = -1;          /* initialize origin to -1 becoz 0 is a valid location*/
     load[s] = 0.0;        /* re-initialize machine load */
     TRFULL[s] -= icrem;
     mach[s] = 2;          /* flag for travelling full */
     continue;
    }
 }
/*-----------------------------------------------------------------------*/
 else if((TRFULL[s]>0)&&(mach[s]==2)){  /* travelling loaded */
          TRFULL[s] -= icrem;          /* decrement by a time increment */
      if(TRFULL[s] >= 0) continue;
      else{
           TRFULL[s] = 0;
           ttrip[s] = 0;      /* end of trip to appear in graphics as arrival*/
           mach[s] = 3;
           in_road[k][star[s]][endr[s]] -= 1;  //remove machine from road section
           fprintf(fp5,"Mach #%d arrive at dump %d^%d at time %d\n", s+1,k+1,DSAVE+1, clock);
          }
        }
/* ------- MACHINE ARRIVE AT DUMP POINT AND CHECKS FOR CLEAR TO DUMP ------*/

 if((bin[k][DSAVE]==1)&&(mach[s]==3)){  //wait to dump
          wt_dump[s] += icrem;          //add lost time waiting
         continue;
      }
/*--------------- GENERATE DUMP TIME AT DESTINATION ---------------------- */

 else if((mach[s]==3)&&(bin[k][DSAVE]==0)&&(bblock[k][DSAVE]==1)){
         dp_tim[s] = weibull(alpha_3[s], beta_3[s], gamma_3[s]);   //call service time
         mach[s] = 4;            //flag for machine ready to travel
         bin[k][DSAVE]=1;        //start dumping
         fprintf(fp5,"Mach #%d start dumping at dump %d^%d at time %d\n", s+1,k+1,DSAVE+1,
clock);
        continue;
       }
/* ----------------------------------------------------------------------*/
 else if ((dp_tim[s] > 0)&&(mach[s]==4)){ //continue dumping
      dp_tim[s] -= icrem;          //reduce service time by increment
     if(dp_tim[s]>0)  continue;
     else {                  //finish dumping in this increment
          bin[k][DSAVE]=0;    //free bin and zero service time
          dp_tim[s]=0;
        fprintf(fp5,"Mach #%d finish dumping at dump %d^%d at time %d\n",
s+1,k+1,DSAVE+1, clock);
        }
      }
/* -------------------------------------------------------------------*/
/*                    EMPTY TRAVEL                                     */
/* -------------------------------------------------------------------*/
  if((dp_tim[s] <= 0)&&(mach[s]==4)){ //finish unloading
    z = 0;  /* positioning flag of bins */
     sos[s] = policies(choice,z,s,k,cht[s],bmat_flg[k][DSAVE],*EAT);
     if (sos[s] == 999){     //temp over capacity at servers:wait
        wt_load[s] += icrem;
        dp_tim[s] = 0;
          spt_tim += icrem;
        fprintf(fp5,"Temp. stope unavailability at time %d on level %d\n",clock,k+1);
        sos[s] = -1;   // maintain old position @ dump point
        continue;
     }
  if ((sos[s] == 99)||(sos[s] == 66)){   /* check other work sections */
     KATE[k] = sos[s];
     next =  adjunt(k, clock, *TEA, s, shift_end,levels);
     if(next == 155){                  /* no more work areas: done */
//      oldstate[s] = mach[s];
        mach[s] = 8;
        arang(k,cht[s],bin_assign[k][cht[s]],z,mach_type[s]); //arange current TYPE queue
        range(k,cht[s],bin_assign[k][cht[s]],z,s);  //re-arrange current machine NO. queue
        bin_assign[k][cht[s]] -= 1;                 //decrement old queue length
        sos[s] = -1;
        fprintf(fp5, "Machine No. %d finished work at %d on level %d\n", s+1, clock,k+1);
        continue;
     }                              /* adequate machines on NEXT */
     else if ((next!=155)&&(to_assign(next,level_mach[next])!=1)){
        oldstate[s] = mach[s];
        mach[s] = 8;
        arang(k,cht[s],bin_assign[k][cht[s]],z,mach_type[s]); //arange current TYPE queue
        range(k,cht[s],bin_assign[k][cht[s]],z,s);  //re-arrange current machine NO. queue
        bin_assign[k][cht[s]] -= 1;                 //decrement old queue length
        sos[s] = -1;
        fprintf(fp5, "Machine No. %d finished work at %d on level %d\n", s+1, clock,k+1);
        continue;
     }
     else if ((to_assign(next,level_mach[next])==1)&&(comp(next)==2)){
wez:    mach[s] = 9;
        TREMP[s] = *TEA;
        lev[s] = next;
        travE[next] += TREMP[s];
        travDE[next] += rmpds[next];
        tempty[s] += TREMP[s];
        level_mach[k]  -= 1;          /* decrement on current section */
        level_mach[next] +=1;          /* increment on next section */
        arang(k,DSAVE,bin_assign[k][cht[s]],z,mach_type[s]); /* arange current TYPE queue*/
        range(k,DSAVE,bin_assign[k][cht[s]],z,s);  /* re-arrange current machine NO. queue*/
        bin_assign[k][DSAVE] -= 1;                 /* decrementing total queue     */
        cht[s] = -1;
        fprintf(fp5,"Ramp_travel time = %5.2f To Level %d of machine
%d\n",TREMP[s],next+1,s+1);
        continue;
      }
    }                              /* end cht[s]== 66 or 99 if segment */
 else {                            /* found a destination on the same work section */
      TREMP[s] = *EAT;                  //simulated travel time on selected route
      ttrip[s] = *EAT;              //duration time of trip for graph
      endr[s] = sos[s];             //graphics usage
      star[s] = cht[s];
      tempty[s] += *EAT;              //stats
      travE[k] += *EAT;               //stats
      travDE[k] += dist[k][sos[s]][cht[s]];
      j = equ_assign[k][sos[s]];
      que[k][sos[s]][j] = mach_type[s];
      sequnt[k][sos[s]][j] = s;
      arang(k,cht[s],bin_assign[k][cht[s]],z,mach_type[s]); //arange current TYPE queue
      range(k,cht[s],bin_assign[k][cht[s]],z,s);  //re-arrange current machine NO. queue
      bin_assign[k][cht[s]] -= 1;                //decrement old queue length
      equ_assign[k][sos[s]] +=1;  //incrementing total machine queue at stope
      bin[k][cht[s]] = 0;         /* releasing the bin for new client   */
      in_road[k][sos[s]][cht[s]] +=1;  /* add machine to next route taken */
      cht[s] = -1;                /* initialize to -1 becoz 0 is a valid location */
      TREMP[s] -= icrem;
      mach[s] = 5;                /* ready to travel empty to server */
      fprintf(fp5,"TREMP[%d] = %5.2f from bin %d to stope %d^%d at time
%d\n",s+1,TREMP[s],DSAVE+1,k+1,sos[s]+1,clock);
     continue;
   }
 }
/* ---------------------------------------------------------------------*/
 else if((TREMP[s]>0)&&((mach[s]==5)||(mach[s]==9))){  //travelling empty
        TREMP[s] -= icrem;
    if(TREMP[s]>0)  continue;
    else {                         /* change status at resp. arrival */
        TREMP[s] = 0;
        ttrip[s] = 0;
     if(mach[s]==5){ mach[s] = 6;    /* MACHINE ON SECTION */
        fprintf(fp5,"Mach #%d start queuing at stope %d^%d at time %d\n",\
              s+1,k+1,sos[s]+1, clock);
        in_road[k][endr[s]][star[s]] -= 1;  /* machine out of road section */
        }
  else {                           /* MACHINE THAT RAMP ONTO SECTION */
     mach[s] = 5;                     /* ready to travel on next section */
     ba = we_next(lev[s],s,*EAT);   /* determine new destination */
   if (ba == -1){                   /* check another work section: current full */
      k = lev[s];
     goto wez;
    }
   sos[s] = ba;
   endr[s] = sos[s];                /* flags for graphics */
   star[s] = 6;                     /* flag for exit from ramps */
   TREMP[s] = *EAT;
   ttrip[s] = TREMP[s];             /* total trip length in scope whole duration */
   tempty[s] += TREMP[s];           /* stats */
   travE[lev[s]] += TREMP[s];
   j = equ_assign[lev[s]][ba];
   que[lev[s]][ba][j] = mach_type[s]; /* assign new LHD to queue */
   sequnt[lev[s]][ba][j] = s;       /* machine joins a new queue at tail */
   equ_assign[lev[s]][ba] += 1;     /* increment machines at destn server */
   fprintf(fp5,"Mach #%d ramp onto Level %d at time %d\n",s+1,k+1, clock);
   fprintf(fp5,"Ramp-to-stope travel = %5.2f on Level %d of MACH %d at time
%d\n",TREMP[s],lev[s]+1,s+1,clock);
   continue;
   }
  }
 }
/* ----------------------------------------------------------------------*/
 else if ((mach[s]==8)&&(comp(k)==2)&&((oldstate[s]==1)||(oldstate[s]==6))) /* IDLE
MACHINES */
        { mach[s] = 1;
          oldstate[s] = -1;
          fprintf(fp5,"Machine # %d going back to work at %d to stope %d^%d\n",\
                s+1, clock,k+1,i+1);
        }
 else if ((mach[s]==8)&&(comp(k)==2)&&(oldstate[s]==4))
        { mach[s] = 4;
          oldstate[s] = -1;
          fprintf(fp5,"Machine # %d going back to work at %d to stope %d^%d\n",\
                s+1, clock,k+1,i+1);
        }
 else if ((mach[s]==8)&&(oldstate[s]==4)){
          next = adjunt(k,clock,*TEA,s,shift_end,levels);
      if ((next != 155)&&(comp(next)==2)&&(to_assign(next,level_mach[next]) == 1))
          goto wez;
     else  wt_dump[s] += icrem;          /* accumulate idle time */
          continue;
 }
/* ----------------------------------------------------------------------*/
mogo:  ;
      }
     SAP_TIME -= icrem;
   }                    //simulation clock  loop ends here
/* ----------------------------------------------------------------------*/
home:  turns = 0;
       for(s=0;s<sumd; s++){
        if(mach[s]==1){
           turns +=3; undone[s] = 1;}
        else if(mach[s]==2) turns +=2;    //turns is flag for tracking machines
        else if(mach[s]==3) turns +=1;
       }
  do{  //control loop of jobs finished
     sprintf(msg,"TIME NOW: %5.1f mins",(float)clock/60);
     outtextxy(getmaxx()/2, getmaxy()-20, msg);
     for(s=0; s<sumd; s++){
      if(tolower(mine)!='n'){
      motion(lev[s],star[s],endr[s],TRFULL[s],TREMP[s],ttrip[s],red[s],mach[s],x_axis,y_axis);
      d = *x_axis;
      w = *y_axis;
      putimage(d, w,buffer[s],3);
      putimage(d - 1, w,buffer[s],4);
      delay(50);
      }
      if ((mach[s] == 1)&&(ld_tim[s]>0)){
         ld_tim[s] -= icrem;
          overtime[s] += icrem;
         }
      else if ((mach[s]==1)&&(ld_tim[s] <= 0)) {
              ld_tim[s] = 0;
              mach[s] = 2;
              z=1;
       cht[s] = policies(choice,z,s,lev[s],sos[s],mat_flag[lev[s]][sos[s]],*EAT);
       if ((cht[s] != 99)&&(cht[s]!=66)){
          TRFULL[s] = *EAT;
          ttrip[s] = *EAT;
          endr[s] = cht[s];
          star[s] = sos[s];
          in_road[lev[s]][sos[s]][cht[s]] += 1; /* machine enter a road section */
          turns -= 1;}
        else{ TRFULL[s] = 0;
             endr[s] = -1;
             star[s] = sos[s];
             mach[s] = 8;
             in_road[lev[s]][sos[s]][cht[s]] -= 1;  /* machine out of road section */
             turns -= 3;}
       }
       else if(mach[s]==2){
            TRFULL[s] -= icrem;
            overtime[s] += icrem;
            if(TRFULL[s]<=0){
               dp_tim[s] = weibull(alpha_3[s],beta_3[s],gamma_3[s]);
               mach[s] = 3;
               TRFULL[s] = 0;
               turns -=1;
               }
            if(overtime[s]>ending)
              ending = overtime[s];
             }
      else if(mach[s]==3){
             dp_tim[s] -= icrem;
            overtime[s] += icrem;
          if((dp_tim[s]<=0)&&(undone[s]==1)){
              dump_ton[lev[s]][cht[s]] += load[s];      //add load of machine
              tot_bin_cap[lev[s]][cht[s]] -= load[s];   //decrement cap left
              cnt3[lev[s]][cht[s]] +=1;                //increment loads at site
              p = cnt3[lev[s]][cht[s]] - 1;
              orig[lev[s]][cht[s]][p] = sos[s];        //record origin of load
            if (stope_grade[lev[s]][sos[s]] != 0)
               rungd[lev[s]][cht[s]][p] = mgrade[lev[s]][sos[s]]; //quality of load
            else rungd[lev[s]][cht[s]][p] = stope_grade[lev[s]][sos[s]];
               dp_tim[s] = 0;
               mach[s] = 4;                           //flag ready to travel
               turns -=1;                            //flag to ensure job finished
              }
           else if (( dp_tim[s] <= 0)&&(undone[s]==0)){
                 mach[s] = 4;
                turns -=1;
            }
          if(overtime[s]>ending)
            ending = overtime[s];
           }
     }              //end for loop
clock += icrem;
} while(!turns == 0);
         sprintf(msg,"Press any key ...");
         outtextxy(getmaxx()/2,410,msg);
         getch();
         goto fin;     //make a final production report

wedza: second = time(NULL);
        gotoxy(1,20); fprintf(fp3,"\tProcess time: %6.4f\n",difftime(second,first)/\
        CLOCKS_PER_SEC);
         free(x_axis); free(y_axis);
again:  plot_axes(delta2,eta2,shift_end,smp_tim,tally);
           sprintf(msg,"Press any key ..");
           outtextxy(getmaxx()/2, 410, msg);
         getch();
         plot_axed((oreton+TWton),shift_end,smp_tim,tally);
         sprintf(msg,"Enter 2,3 to CONT., see GRADE, see SUMMARY..");
         outtextxy(getmaxx()/2,410,msg);
         gotoxy(25,25); aa = getche();
         if(aa=='2'){ goto again;}
         else if(aa=='3'){goto sum;}
         else ;
         outtextxy(getmaxx()/2-50,getmaxy()-10,"SUCCESSFUL SIMULATION RUN");
         getch();
         free(EAT); free(TEA); free(SERVE); free(ptr); free(bdown);
        closegraph();
  return (0);
}
```

Appendix E
Underground Dispatch Operating Manual

The Program requires four input data files, namely:

1. Schedule input data file consists of the planned shift production targets, available equipment, its location within the mine and the draw/dump points quantities. This file also contains the shift length, times for scheduled breaks, and accumulated work hours for machines since their last service.

2. Machine file: This file holds the machine characteristics such as capacity, rating, cost per hour and three parameter Weibull distribution functions for loading, travelling full, dumping and travelling empty.

3. Mine tunnel network: consisting of the distances between allowed routes, the maximum upper limit velocities when travelling empty and loaded. Inter-level travel is also described in the same manner. In the event of a machine travelling to a next level, its travel in addition to the ramp travel includes the distance from the ramp exit to the destination on the new level. The distances to each possible/allowed destination are required.

4. Graphical mine layout file: this file is only required for the graphical representation of the mine layout. The data is specially created in graphical screen co-ordinates. This data file is static and is coded in the computer program. This requires recompiling of the computer code each time the mine layout changes i.e. when new tunnels are driven.

A detailed presentation of each data file follows. The sign > > is used to represent a new line in the data entry format.

1.0 Schedule Input File

The input is formatted and therefore its entry has to be as precise as is required.

a: Number of work sections or levels scheduled > >

b: Enter the largest probability of under-, and over-loading on the machines > >

c: Number of stopes, dump-points and machines per work section. > >

Repeat (c), as many times as the levels or sections scheduled.

d: Enter on the same line the scheduled stope tonnage, its expected grade, buffer space for machines queuing, priority of stope, status of stope (1 = ready, -1 = blocked) and the material type (characters O/o for ore, and W/w for waste) > >

Repeat for all stopes going systematically from level to level or section to section.

e: Enter on the same line the scheduled dump point capacity, its machine buffer space, status (1=ready, -1 = blocked) and material type allowed to dump there (W/w for waste dump and O/o for ore-dump). > >

Repeat for all dump points going sequentially from level to level (or section to section).

f: Enter the average start-up time for the shift in seconds. > >

g: Enter driver's name, machine type (A,B,or C; e.g. A=ST8, B=ST6A, etc), stope or dump number, and location sign i.e. + if at loading point and - if at dump point at start of shift. > >
Repeat for as there are machines scheduled to work sequentially entering from level to level.

h: Enter the start of lunch, end of lunch, shift end time, the simulating time increments (recommend small value to avoid missing events as they occur), and the time windows for reviewing the process. The times are all in units of seconds > >

k: Enter the inner control limits for grade, tonnage and utilization as decimal figures. > >

i: Enter the outer control limits of grade, tonnage and utilization as decimals then enter a time limit (in seconds) when no re-evaluation process would be required. > >

j: Enter the average time between breakdowns of the fleet and the average repair time. The time units of this card are hours. > >

l: Enter the accumulated time in hours for each machine since it last received a service. The order of entry of these times should be the same as that used in (g) to list the machines available. > >

Table E1 is an illustration of the schedule input file for a two level/section, with five stopes and two dump-points on level 1 and three stopes and one bin on level 2. One section is waste hence must have a waste dump-point. Five machines are available in all with three starting the shift on level 1 and the remainder on level 2.

Note: Under normal circumstances only items represented by **c, d, e, g and l** in the data file need to be changed to reflect each shift schedule or re-schedule. The number of stopes and draw points can also be kept constant with the only requirement that their capacities must be zero whenever they are not scheduled for production.

Table E1 A typical schedule input data file (Shift schedule used in Chapter 7)

```
2
0.1 0.15
5 2 3
3 1 2
400.0 2.0 2 1 1 o
100.0 2.0 1 2 1 W
210.0 3.0 1 2 1 o
200.0 2.8 1 1 1 o
300.0 3.0 1 1 1 o
500.0 2.5 1 2 1 o
250.0 3.0 2 2 1 o
300.0 2.8 1 1 1 o
1600.0 1 1 W
2000.0 2 1 o
1900.0 2 1 o
600.0
Inza a 0 +
Baba b 3 +
Amai a 2 +
Ogo b 2 +
Muza a 1 +
9000  12600 21600 10 1800
0.1 0.10 0.2
0.05 0.05 0.1 1200
14.5 1.5
0.0 2.0 1.0 0.0 3.0
```

2.0 Mine tunnel network file

This file represents the mine haulage network and has the speed limits that are imposed for safety reasons. Distances are in metres and velocities in metres per second.

a: Enter the inter-level ramp distance and allowed maximum speed. If only one level exists, enter zero for the ramp distance and any non-zero ramp speed. >>

b: Enter the stope to dump-point distance, maximum velocities loaded and empty respectively and maximum number of equipment on a road section on each route. Repeat as many times as the product of section dump-points and section stopes. If some dump-points to draw-points routes are infeasible, enter zero for the distance and non-zero values for the velocities. The program requires a fully assigned rectangular matrix of the three variables. This card has a free format that allows multiple entry of the variables in the same line without having to press >> for next line. >>

c: Go to (a) and repeat the process for as many times as there are levels scheduled.

d: Enter the distances and maximum allowed speeds from the ramp exit to all the possible destination on that level. If a route is infeasible, enter zero distance and some non-zero speed. This card has free format. >>

e: Repeat (d) as many times as there are levels scheduled.

f: Enter the distances and maximum allowed speeds between ALL draw-points (stopes) on each level. Infeasible routes have zero distance and a non-zero speed. This card is for possible re-dispatch to another draw-point in the event of a previously assigned one becoming blocked. Data is entered in free format allowing more than two variable values per line without pressing >> for new line.

g: Repeat (f) for as many times as there are scheduled levels. Note the total entries is sum of the number of stopes per level squared for each level.

Table E2 A typical mine layout data file (Inza Mine described in Chapter 7)

```
640.0 1.46
130.0 1.26 1.46 2 330.0 1.26 1.46 2
80.0 1.26 1.46 2 280.0 1.26 1.46 3
100.0 1.26 1.46 2 100.0 1.26 1.46 2
340.0 1.26 1.46 2 140.0 1.26 1.46 2
320.0 1.26 1.46 3 120.0 1.26 1.46 2
640.0 1.46
240.0 1.26 1.46 2 200.0 1.26 1.46 3 120.0 1.26 1.46 2
230.0 1.46 180.0 1.46 50.0 1.46 240.0 1.46 220.0 1.46
240.0 1.46 200.0 1.46 120.0 1.46
0   1.46 210 1.46 230 1.46 470 1.46 450 1.46
210 1.46 0   1.46 180 1.46 420 1.46 400 1.46
230 1.46 180 1.46 0   1.46 240 1.46 220 1.46
470 1.46 420 1.46 240 1.46 0   1.46 60  1.46
450 1.46 400 1.46 220 1.46 60  1.46 0   1.46
0 1.46 360 1.46 440 1.46
360 1.46 0 1.46 320 1.46
440 1.46 320 1.46 0 1.46
```

Note: The mine layout file is relatively static requiring occasional editing to include additional developments. By creating a number of these layout files for each level, then for adjacent levels, etc. it is possible to store the layouts in easily retrievable form to implement with any schedule that is generated. That is one simply calls the map file that corresponds to the scheduled levels and no time is lost in creating this file at implementation. This is particularly useful in instances of re-scheduling when no delays are permissible otherwise the benefits of these re-schedules is lost by becoming irrelevant when eventually implemented.

3.0 Machine characteristic file

The file contains the three term Weibull parameters for loading, travelling loaded, dumping and travelling empty. These parameters are then followed by the machine capacity, its rating (running efficiency) and the operating cost per hour.

a: Enter a machine type A, B, or C. These characters represent different makes, types or ages of fleet machine. > >
The types should be the same as those used to describe the available machines in the Schedule Input file by card number (g).

b: For the machine entered in (a), enter the 15 parameters in the SAME LINE in the following order:
- loading Weibull parameters: shape, scale and location factors (i.e. alpha1, beta1, gamma1)
- travelling loaded Weibull parameters: alpha2, beta2, gamma2
- dumping Weibull parameters: alpha3, beta3, gamma3
- travelling empty Weibull parameters: alpha4, beta4, gamma4
- capacity of machine (tons or cubic metres)
- rating (decimal)
- cost per hour. > >

c: Go to (a) and repeat as many times as there are machine types (a maximum of three types).

Note: All Weibull parameters are real numbers and note the shape and scale parameters are non-zero.

Table E3 A typical machine characteristic data file (data used in Chapter 7)

```
A
16.8 1.42 40.0 60.0 2.0 0.0 14.2 1.39 25.5 60.0 1.5 0.0 12.0 0.9 80.0
B
16.8 1.42 35.0 60.0 2.0 0.0 14.2 1.39 23.5 60.0 1.5 0.0 13.5 0.9 80.0
C
20.4 1.50 35.0 65.0 2.5 0.0 12.0 1.2 25.0 63.0 1.5 0.0 9.5 0.8 65.0
```

This file is static implying that it can be used for long time spans (several months) without modifications. However, if time studies and/or mechanical availability indicate a shift from the current parameters then it should be modified.

4.0 Mine graphical network file

This file consists of the physical description of the mine layout showing the tunnels and stopes (draw-points) and dumping points. The layout has to be transformed into screen co-ordinates and the particular layout has to be uniquely coded so as to enable the visual representation of the operations during the production dispatching. A typical graphical network that was coded is indicated in Table E4. The use of the Underground Dispatch Model can be done without the need of the graphical file. The graphical information that will be available to the user in that case is the cumulative draw and dump capacities as well as the progress reports at review times. This form requires no coding of a specific mine network.

Table E4 A typical graphical mine layout data file (Inza Mine layout)

```
7
80 100 150 200
80 200 150 200
150 200 250 200
250 200 375 200
375 200 450 80
375 200 450 200
450 80 450 200
150 200
375 200
80 100
80 200
250 200
450 80
450 200
3
```

```
80 300 300 350
100 400 300 350
300 350 400 350
300 350
80 300
100 400
400 350
```