



National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

A Structural Study and Algorithms in Vertex Coloring

Eric Masson

McGill University, Montreal



January 1996

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment
of the requirements for the degree of Doctor of Philosophy.

© Eric Masson, 1996



National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-12433-9

Canada

Abstract

The problem of vertex coloring holds an important place in engineering as it models situations in which a number of shared resources must be minimized and distributed across the sub-components of a system. The objective is to ensure a valid and cost effective implementation of the overall system. The problem surfaces in a great number of applications, many of which fall within the area of digital systems design. However, despite its wide range of applications, vertex coloring remains one of the most complex optimization problems known and to this day no efficient method has been shown to provide optimal answers in all instances of the general case.

This dissertation explores characteristics of optimality of vertex colorings, bounds on the chromatic number and coloring heuristics. The first few characteristics are based around the fundamental and residual nodes of an optimal coloring. An examination of the subset of fundamental nodes will reveal necessary subgraph properties for an optimal coloring and its graph; one of which is based on Kempe chains. In turn, this leads to a bound relating the chromatic number and the number of odd cycles in a graph.

Subsequently, a continuous variable formulation of the vertex coloring problem is presented along with an analysis of its solution space. The characterization of the space illustrates the problem's complexity and the nature of its local minima relates to the Gallai-Roy theorem. The results given will have algorithmic significance since their proofs are constructive.

The WWI pair of vertex coloring heuristics is then disclosed. The algorithms are based on successive compressions of pairs of non-adjacent nodes each reducing the problem in-

stance by one node until a complete graph is obtained. The criteria for selecting pairs of nodes concentrate on the affinity and conflict values calculated from structural properties of the graphs. The heuristics are justified by upper bounds on the chromatic number and approximation arguments. It is demonstrated that some compressions preserve the chromatic number or the maximal clique size of a graph, thus resulting into some identifiably optimal selections by the algorithms. Ultimately, this leads to the characterization of a class of perfect graphs. Finally, a set of benchmarks for the WWI algorithms on random graphs and k -colorable random graphs is given and favorable comparisons are offered with existing algorithms.

Résumé

La coloration des graphes joue un rôle important en ingénierie car elle représente les situations où une quantité minimale de ressources similaires doit être allouée aux sous-composantes d'un système. L'objectif est d'assurer un faible coût total au système. Une multitude de problèmes est directement reliée à la coloration, dont plusieurs se situent dans la conception des systèmes digitaux. Cependant, malgré la grande variété d'applications pratiques et l'attention particulière que le problème a reçu au cours des ans, la coloration des graphes demeure un des problèmes les plus complexes de la recherche opérationnelle et pour lequel il n'existe toujours pas de méthode de résolution optimale qui soit efficace.

Cette thèse étudie la coloration par le biais des caractéristiques d'une coloration optimale, des bornes sur l'index chromatique d'un graphe et d'heuristiques. Les premières caractéristiques examinées se rattachent aux sommets fondamentaux et résiduels d'une coloration. L'analyse du sous-ensemble des sommets fondamentaux identifie plusieurs propriétés structurelles nécessaires à une coloration optimale, dont l'une est basée sur les chaînes de Kempe. Cette dernière observation mène à une borne reliant l'index chromatique et le nombre de circuits impairs d'un graphe.

Subséquentement, le problème de coloration des graphes est formulé et analysé sur un espace continu. La caractérisation de l'espace des solutions illustre la complexité du problème avec l'appui d'un résultat qui établit un lien entre les minimums locaux et le théorème de Gallai et Roy. Les résultats présentés ont des implications algorithmiques étant donné la nature constructive de leurs preuves.

Pour terminer, une paire d'heuristiques de coloration est dévoilée. Les algorithmes

procèdent par la compression successive de sommets non-adjacents jusqu'à ce qu'un graphe complet soit obtenu. Les principes de compressions sont fondés sur les propriétés structurelles d'affinité et de conflit d'un graphe. Des arguments d'approximation ainsi que des bornes sur l'index chromatique servent à justifier les décisions heuristiques. En outre, il est démontré que certaines compressions préservent l'index chromatique ou la clique maximale d'un graphe tout en permettant une réduction optimale du problème. Ceci mène à la caractérisation d'un ordre de graphes parfaits. Finalement, une batterie de tests sur des graphes aléatoires sert de comparaison favorable entre les algorithmes présentés et des algorithmes provenant d'autres sources.

Acknowledgements

First, I wish to acknowledge my co-advisors Prof. Vinod K. Agarwal and Prof. Pramod C. P. Bhatt. I am grateful for their guidance, encouragements and insights during the course of my research. They have helped me develop a reasoning process which will carry over to my future endeavors. To Professor Bhatt, thank you for making me understand that a correct result is not complete until it becomes as clear and simple as it possibly can. *Dhanyavād.*

To my parents, Gilles and Monique, which have given me their unrelenting support and advice throughout my existence. To them I owe all.

To Betina Hold, *danke schön* for your friendship, the insights, the projects, your encouragements, your drive to organize events and your proofreading. To Anthony Botzas for the friendship, the design projects, and the involvement in graduate student affairs. *Merci à Pauline Masson pour son aide en temps critique.*

My special thanks to Jacek Slaboszewicz, Charles Arsenault and Alain Magloire for their friendship and the most reliable and complete computing environment I have ever used.

Throughout my years at the MACS laboratory I have met several friends with common interests and with whom I shared ideas, good times and mind soothing soccer games. My regards to Abdolreza Nabavi-Lishi, Abid Zaidi, Ajai Jain, Al Lu, Andy Bishop, Avrum Warshawsky, Benoît Veillette, Boonchuay Supmonchai, Chinsong Sul, Choon Haw Leong, Dimitri Lambidonis, Erik Altman, Fidel Muradali, Francis Larochelle, Irene Song, Jean-Charles Maillet, Jean-François Côté, John Abcarius, Kasia Radecka, Loai Louis, Manoj Verghese,

Marco Baratta, Mark Kassab, Michael Kim, Michael Moscovitch, Michael Toner, Morie Malowany, Mourad El-Gamal, Nadime Zacharia, Nagesh Tamarapalli, Nilanjan Mukherjee, Oryal Tanir, Palash Desai, Peter Sinn, Philip Crawley, Pierre Parent, Pierre Racz, Sanjay Gupta, Shashank Nemawarkar, Sherif Sherif, Stéphane Gagnon, Thomas Obenaus, Victor Tyan, Victor Zia, Xavier Haurie.

Finalement, merci à tous mes copains de longue date qui ont su comprendre lorsque je n'ai pu être avec eux: Günther Baßler, Carl Joli, Stéphane Tremblay, Sylvain Tremblay, Yves Du Sault.

I wish to extend my gratitude to the *Free Software Foundation* and its GNU project for providing software tools on which this research was developed. The dedication of its participants has shown that quality need not fetch a price in this day and age.

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), *Les Fonds pour la Formation de Chercheurs et l'aide à la Recherche du Québec* (FCAR), and MicroNet, the Canadian Federal Network for centers of excellence dealing with Microelectronics.

Claim of originality

The core of the original work is located in chapters 3 , 4 , 5 and appendices A , B , C. Chapter 2's contribution is to provide a classification of compression algorithms on the properties of order, adaptability and localization.

- Chapter 3 introduces the concept of fundamental nodes on which the main contributions are theorem 3.2, corollary 3.4 and the algorithm of appendix B. Theorem 3.2 is a necessary characteristic of optimal vertex colorings identifying structural properties of graphs and their colorings. Corollary 3.4 is a cubic bound relating the chromatic number and the number of odd cycles in a graph. This bound has further consequences in the heuristic designs of chapter 5. The proofs of the characteristics of optimality in chapter 3 are constructive and can be used as refinements for sub-optimal colorings. Appendix B provides such an algorithm for a refinement on fundamental nodes in $O(|V|^2)$.
- The theoretical content of chapter 4 is original with exception to theorem 4.3 and corollary 4.2 which can be demonstrated as consequences of the known Gallai-Roy theorem (theorem 2.3). The chapter begins with a transformation of the vertex coloring problem into a continuous variable mathematical formulation (theorems 4.1 and 4.2). Observations are then made on the nature of the solution space (corollary 4.1, propositions 4.1 and 4.2, and corollary 4.3). The introduction of colored stratifications led to a new and constructive proof to the necessary characteristic of optimality of theorem 4.3. In turn this proof led to the refinement algorithm of appendix C. Finally

theorem 4.4 fully characterizes the local minima of the continuous variable representation and it relates them to the Gallai-Roy theorem.

- Chapter 5 concerns two new vertex coloring heuristics: WWI on affinities and WWI on conflicts. The implementation described in appendix A provides a mechanism which permits the algorithms to operate one order of magnitude faster (in the number of vertices) than other algorithms based on similar metrics. The use of the chromatic number preserving compressions of theorem 5.2 and the approximation of a second order criterion via propositions 5.3 and 5.4 is also novel. The theoretical consequences of the algorithms are new results as well: theorems 5.3 and 5.4, and the algorithmically identifiable perfect graphs of section 5.6.

Contents

Abstract	i
Résumé	iii
Acknowledgements	v
Claim of originality	vii
1 Introduction	1
1.1 Focusing on Digital System Synthesis	2
1.1.1 Modeling Complexity	4
1.1.2 Computational Complexity	5
1.2 Vertex Coloring	7
1.2.1 Channel Routing Example	7
1.2.2 Register Allocation	10
1.2.3 Other problems related to vertex coloring	13
1.3 A Brief Dissertation outline	13
2 Defining Vertex Coloring	15
2.1 Definitions	15

2.2	Algorithmic Complexity	20
2.3	A Bound on the Chromatic Number and Characteristics of Optimality . . .	23
2.4	Studying the solution space	25
2.5	More on Algorithms	26
2.5.1	Greedy Algorithm	26
2.5.2	Coloring through compressions	26
2.5.3	Tseng's Algorithm	29
2.5.4	A Classification of Compressions: order, adaptability and localization	31
3	Some Characteristics of the Chromatic Number	34
3.1	A First Characteristic	34
3.2	Going further with fundamental nodes	38
3.3	Some possibilities	46
3.4	A brief recapitulation	47
4	A Structural Study of the Solution Space	48
4.1	Mathematical programming models	48
4.2	A Slight Modification	54
4.3	Refinement method	60
4.4	Relating Characteristics of Optimality	62
4.5	Conclusion	73
5	Algorithms	75
5.1	Further Observations	75
5.2	WWI Algorithms	82

5.2.1	WWI on Affinities	83
5.2.2	WWI on Conflicts	84
5.2.3	Discussion on Algorithms	86
5.3	An Example of WWI on Affinities at Work	86
5.4	Implications of WWI on affinities	87
5.4.1	A Heuristic	92
5.5	Benchmarks	93
5.5.1	Random Graphs	93
5.5.2	k -Colorable Random Graphs	96
5.5.3	The first order approximation to the second order	101
5.6	Perfect class	105
5.7	Improving the WWI Algorithms	113
5.8	Conclusion	113
6	Conclusion	115
6.1	Possible directions	116
	Bibliography	118
A	WWI Implementation	124
A.1	Initialization	127
A.2	Finding the best compression	129
A.3	Adjusting EC	130
A.4	Adjusting A, C, K, χ	133
A.5	Complexity of the implementation	136

B Fundamental Node Refinement	138
C Colored Path Refinement	141
C.1 Greedy Algorithm	143

List of Figures

1.1	Y-chart illustrating the levels of abstraction in digital design automation . .	3
1.2	A PCB board to route	8
1.3	Set of wires to channel route	9
1.4	Conflict graph for wire segments	10
1.5	A channel routing	11
1.6	Register allocation example	12
2.1	A graph and a coloring	16
2.2	Another graph and its coloring	17
2.3	A compression of vertices v_1 and v_3 in figure 2.1	19
2.4	Coloring through compressions	29
3.1	An example for theorem 3.1	36
3.2	A cycle of chromatic number 3	38
4.1	A simple 3 vertex graph	58
4.2	The effect of ϵ	59
4.3	The local minimum might not be achieved	62
4.4	An example color stratification	63

4.5	Refining a coloring	68
4.6	A sub-optimal coloring	69
5.1	Triangles and compressions	83
5.2	WWI on affinities example	88
5.2	WWI on affinities example (<i>continued</i>)	89
5.3	A partial graph with dependencies between $v_p, v_q, v_\alpha, v_\beta$	90
5.4	Two possibilities with v_γ	91
5.5	Diamond Graph	94
5.6	Benchmark results for random graphs and WWI on affinities	97
5.6	Benchmark results for random graphs and WWI on affinities (<i>continued</i>)	98
5.7	Benchmark results for random graphs and WWI on conflicts	99
5.7	Benchmark results for random graphs and WWI on conflicts (<i>continued</i>)	100
5.8	Benchmark results for $n = 128, 256, 512, 1024$ and $p = 0.5$	102
5.9	Benchmark results for $n = 128$ and $p = 0.15, 0.3, 0.70, 0.85$	103
5.10	Benchmark results for $n = 256$ and $p = 0.15, 0.3, 0.70, 0.85$	104
5.11	2nd order approximation on 128 nodes and $p = 0.5$	105
5.12	A sequence of conflict free compressions	112
A.1	Data structure examples	126
A.2	Cases 2,3,4 and 5.	134
A.3	Case 6.	135

Chapter 1

Introduction

From a fundamental standpoint the goal of the engineering process is to seek solutions to practical problems and subsequently apply them. Quite often there is more than one solution offering itself to a given problem and the process also entails finding the best possible one given a determined set of criteria. In that respect the engineering process is directly comparable to a mathematical optimization process for which the goal is to determine the best possible solution given objectives and constraints. Due to the wide spectrum of engineering problems, the complexity of the optimization models that underlie engineering varies greatly. Some problems have simple and elegant solutions whereas others elude us entirely due to their complexity. Furthermore, the very nature of the complexity is itself a variant from problem to problem.

Particular problems are difficult because their criteria and objectives are nebulous and intangible. For such problems precise or quantitative mathematical formulations are difficult if not impossible. Some examples of such engineering problems are software quality metrics problems [Glas92] or the slew of problems that fuzzy logic or neural networks have attempted to address [Wels94]. These cases correspond to the optimization problems for which it is hard to generate a model of abstraction or to prioritize the multiple objectives and criteria imposed. And hence they are problems of *high modeling complexity*.

But there are also a great number of highly complex engineering problems for which

there are clear, precise and even concise mathematical formulations. Their complexity arises because there are no known methods to compute their optimal solutions efficiently within a limited set of resources. These are the problems of *high computational complexity*. Some of the problems within this category deal with infinite solution spaces from which it is sometimes difficult to generate a solution let alone determine whether a solution encountered is indeed an optimal one. For example the many questions of systems theory which lead to non-linear formulations on continuous variable representations [PaWi88].

But one does not need to go to such extremes to find problems for which no computationally efficient methods are known. For example, several problems of combinatorial nature have finite solution spaces on which optimal solutions can be found within finite time and resources [GaJo79, Gibb85]. However they still fall within the computationally complex category because of the sheer magnitude of their solution spaces and the inefficient methods we have to prune them. It is the prohibitive amount of resources (often the time resource) demanded by current tools which renders impractical the goal of a guaranteed optimal solution. The core of this work will concentrate on such a problem which has several practical occurrences in engineering, namely the vertex coloring of graphs. This specific problem is of importance since an adequate solution method, even if sub-optimal, has an impact on a very broad class of practical problems [GaJo79].

It is important to point out that the categorization of complexity into modeling complexity and computational complexity used herein is not a formal classification but rather a descriptive tool to highlight the two principal sources of difficulty found in practical optimization. Nor are the two categories mutually exclusive as some tenacious problems confront us with both the modeling and computational difficulties.

1.1 Focusing on Digital System Synthesis

In order to bring the points of the previous section to a further depth, the scope is now reduced to the particular optimization issues found in the synthesis of digital systems. Over the last few decades computer automation has become a necessity for the design of digital

systems. This is a direct result of the explosion in size of digital systems and the shortened time cycles allotted for their production [Sher93]. Automation tools have seen their level of abstraction steadily augment to meet the increasing demands of designers. The Y-diagram of figure 1.1 first introduced by Gajski and Kuhn [GaKu83] is a succinct depiction of the levels of abstraction which have appeared over the years. The extent of an abstraction is established from the distance of its corresponding ring to the center of the chart. At the lower levels of abstractions, such as the circuit or logic levels, one must go through the tedium and detail of transistors, differential equations, boolean equations, logic gates and polygons of materials to produce a circuit. At a higher level such as the algorithmic one it ideally suffices to produce an algorithmic description of a design to automatically churn out a circuit corresponding to it. And therefore, the use of higher levels of abstraction has the effect of shortening and simplifying the design cycle as one does not have to dedicate as much time to details as opposed to concepts [MiLD92].

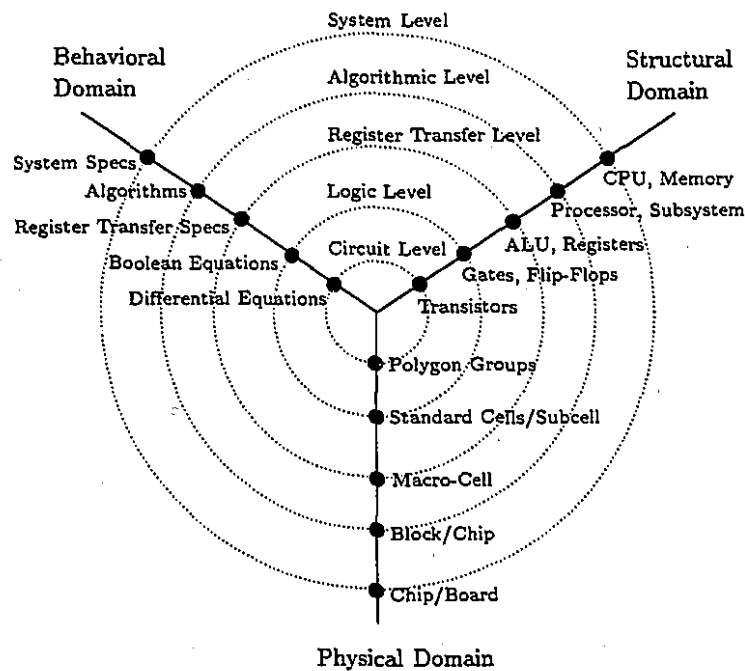


Figure 1.1: Y-chart illustrating the levels of abstraction in digital design automation

However the introduction of several layers of abstraction has generated several problems of high modeling complexity within the field of digital synthesis, especially at the higher levels of abstractions. As well, the discrete structure of digital systems leads to many integer optimization and combinatorial problems of great computational complexity (although at the lowest levels there are many problems of a continuous variable since many models are based on classical physics). As a consequence, one of the main factors impeding the progress of digital design automation is the need for advancement in optimization techniques. In fact, if one foregoes the satisfaction of constraints and optimization criteria associated with the general problems of digital synthesis then relatively simple mechanisms exist to derive a physical circuit from a system's description in a high level language such as VHDL [Perr91, Merm92, GDWL92]. Much of the difficulty arises in the fulfillment of constraints and optimization methods.

1.1.1 Modeling Complexity

In high level synthesis, the issues surrounding modeling complexity are numerous. The optimization process deals with intangible constraints such as reusability, modularity, upgradability of designs [GDWL92, MiLD92] and thus it becomes tentative to model such objectives on automation tools based on computational platforms. Other objectives such as design testability and verifiability are also hard to quantify unless restricted to specific sets of faults and assumptions [MiLD92, GhDN92].

Furthermore, there is also the *multi-objective* nature of the problems encountered. Optimal values are simultaneously desired for multiple dimensions such as monetary cost, silicon area and timing [KuDe92, GDWL92]. This poses difficult modeling problems as there is no complete mathematical methodology to deal with multiple objectives. One of the widely used methods to treat multi-objectives is akin to lexicographical optimization techniques [Zlob78] in which the objectives are prioritized and a sequence of optimizations is conducted on each objective. Such is the case with the sequencing approach to partitioning, allocation and scheduling found in many high-level synthesis tools [KuDe92, GDWL92, THKR83]. Another approach employed transforms the multiple objectives into a single

objective through a weighted cost function on the original objectives. Many examples of this can be found in the clustering algorithms of high-level synthesis [MiLD92, LaTh89]. As well there exists other multicriteria techniques such as Pareto optimization [Pa1896] which have yet to be explored in digital synthesis. Unfortunately the point of contention with these techniques is that they may very well bypass significant portions of the solution regions. The selection of a satisfactory model becomes a laborious task as different methods generate solutions often differing in optimality. Although prevalent in high level synthesis, multi-criteria problems occur at all levels of abstraction of digital synthesis and present the same difficulties. At the lower level it is the tradeoff between area and timing which is most evident.

There are other elements of modeling complexity as well. For example the design representations made available to the designer restrict the possible forms of expressions of a design and impose further constraints on the solution space [GDWL92, KuDe92]. Similar drawbacks exist with the internal representations selected for synthesis computations [GDWL92, MiLD92]. Overall, these combined factors contribute to render digital design automation rich in model complexity issues.

1.1.2 Computational Complexity

As well, digital synthesis is certainly not exempt of computationally complex problems. As previously stated, the discrete structure of digital systems produces several integer and mixed integer optimization problems; many of which fall within the classes of NP complete or NP hard problems [GaJo79]. To name a few, the general formulations of wire routing, logic minimization, scheduling, allocation, finite state minimization, partitioning, binding [AsDN92, MiLD92, Sher93, GDWL92] all contain NP hard or NP complete problems and they span across all levels of abstraction in synthesis. In fact most optimization problems of synthesis [Sher93] are NP hard or NP complete and quite a few directly translate to integer linear programs [NeWo88] or well known graph theoretical problems such as maximum clique, vertex coloring, steiner tree problem, and hamiltonian circuit [GaJo79, Gibb85]. As with all NP type problems, this implies that there are no known algorithms which provably

solve them optimally within polynomial time of the instance sizes. And nor do we know if such algorithms do exist. However, algorithms of exponential time complexity can be used to generate optimal answers. Unfortunately, they become much too slow as input instances grow in size and thus alternative methods must be found for the larger cases that occur in practice.

Several algorithmic techniques are used to address the synthesis problems for which no optimal polynomial time algorithms are known. One approach consists in finding polynomial time approximation algorithms which guarantee solutions within bounds of the optimal solutions [Sher93]. Another is to use polynomial time heuristics shown to be effective through benchmarking and theoretical justifications but which are without guarantees of optimality. The heuristic approach is by far the most common one used in synthesis.

There also exist special case algorithms which concentrate on subspaces of the problems for which optimal polynomial time solution generators are known. For instance, there are many types of perfect graphs which have polynomial time solutions to problems otherwise NP complete in the most general case [Golu80]. Although there are some synthesis problems which are always guaranteed to generate instances within these special cases [HaSt71, Gav72, KuPa87], these problems are the rare exceptions. In most situations where specialized algorithms are used, the original problems cannot be directly associated with special cases. Instead the constructs are artificially modified and constraints added so that the problem instances generated meet the requirements of an optimally solvable special case. For example, most of the Olympus synthesis system [KuDe92] operates under that principle. However, due to the additional constraints introduced, the optimal answers to the modified problems may very well be suboptimal answers to the original problems. Therefore the method compares to heuristics as it provides no guarantee of optimality. In fact, many heuristics are designed by extending the methods which are provably correct for special cases [Sher93].

Advances in integer linear programming (ILP) methods [NeWo88] have made it practical to utilize integer programming tools to solve limited size synthesis problems optimally [GeEl90]. Although these new methods remain exponential in the worst case, they

do operate sufficiently fast on small but significant cases. Finally, probabilistic algorithms such as simulated annealing [PFTV88] and simulated evolution [Sher93] have been used on a number of complex synthesis problems. They do provide quality answers for problems such as partitioning or scheduling but their excessive run time often renders them impractical [GDWL92].

1.2 Vertex Coloring

Given the importance of optimization in the field of synthesis it is imperative that the topic occupies a significant part of the synthesis research areas. However, undertaking work which tackles all of the aforementioned issues at once would be overwhelming. Instead much of the synthesis research concentrates on specific optimization problems for which the objectives are less broad but more tangible. Such is the case with this dissertation which concentrates its effort on the combinatorial problem of vertex coloring. The problem is of importance since it has several occurrences in the digital systems synthesis area and it belongs to the NP-complete class of problems. It is one of the most persistent problems even amongst the NP-complete problems as it falls within a group for which there are no likely satisfactory approximation algorithms [LuYa94]. Yet sub-optimal solutions translate directly into digital systems of larger area, poorer performance, and higher cost. Therefore, there are imminent needs for finding better heuristics and further understanding of the problem. Two specific instances of vertex coloring are now used to exemplify synthesis applications in which the problem is found.

1.2.1 Channel Routing Example

One of the first synthesis applications involving vertex coloring was channel routing [HaSt71]. As an abstract example consider the printed circuit board (PCB) of figure 1.2. The board consists of 3 rows of 4 integrated circuits which must be interconnected. Vertical wires are made to run on one side of the board and the horizontal wires on the opposite. Junctions between horizontal and vertical wires are achieved by contacts (vias) drilled into the board.

Between each row of circuits there are *horizontal spaces* in which five wire *channels* can run in parallel. The same applies for *vertical spaces* between circuit columns. Each channel can be made to run several disconnected wire segment as long as they occupy disjoint regions of the channel.

At first, an algorithm attempting to minimize connection distances is used to decide the general path of connections by determining which horizontal and vertical spaces the wires will run through. For example the eight interconnection paths shown on figure 1.3 are desired for a single PCB.

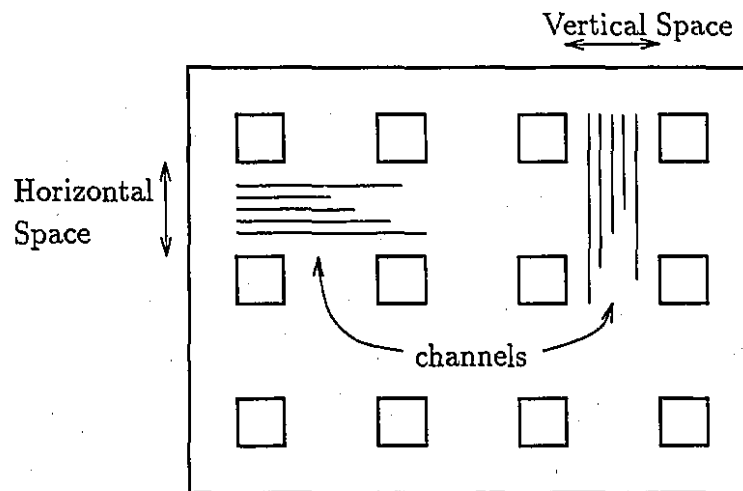
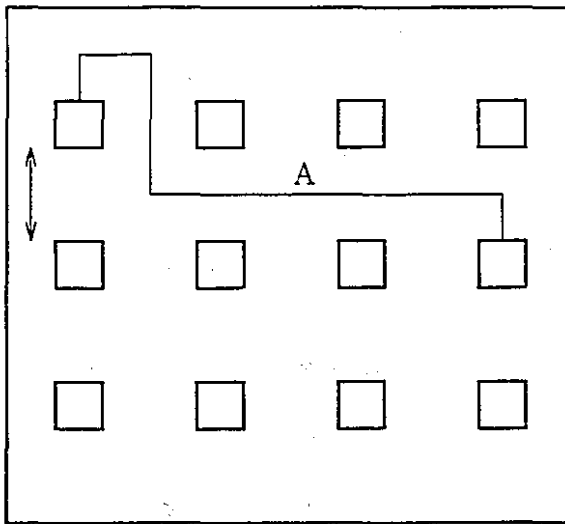
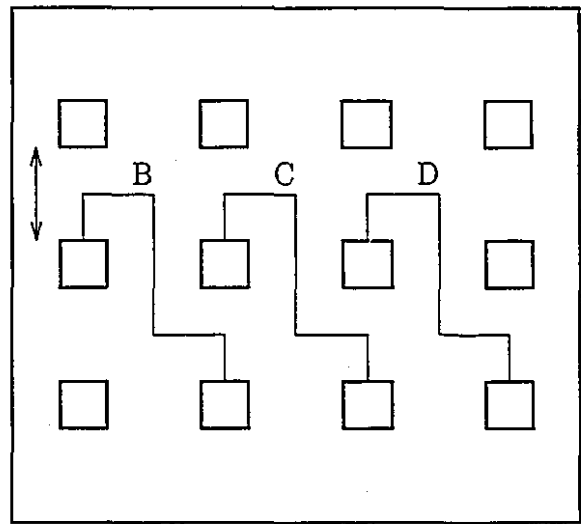


Figure 1.2: A PCB board to route

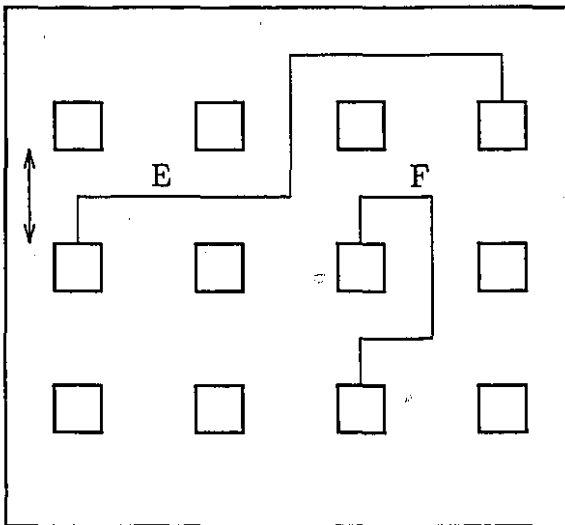
Once the general paths are determined, channel assignment follows. For each horizontal and vertical space on the board the wire segments running through it are extracted. The goal of channel routing is to minimize the number of channels needed to run all the wire segments falling within a given space. Returning to figure 1.3 consider the eight wire segments labeled A through H located on the horizontal space between the first and second row of circuits. The wire segments E and H can run on the same channel as they occupy different sectors of the space. However A and E cannot share a common channel as they both run through a common part of the horizontal space. A table indicating whether or



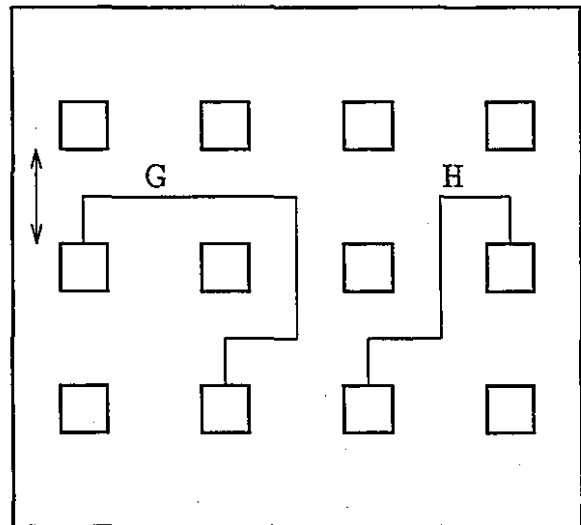
(a)



(b)



(c)



(d)

Figure 1.3: Set of wires to channel route

not each pair of wire segments can share a channel is easily produced. From this table a *conflict graph* is generated. Each node of the graph represents a given wire segment and each edge denotes that two segments cannot share a common wire channel. The conflict graph for the horizontal space between first row and second row of circuits in figure 1.3 is given in figure 1.4. The ensuing task consists of labeling the nodes of the graph (wire

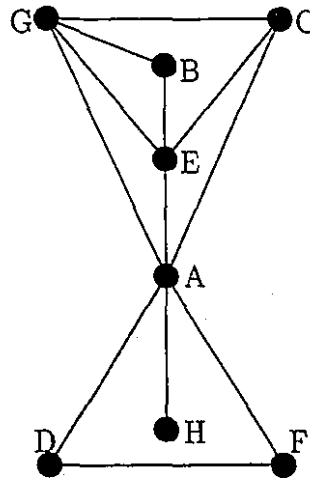


Figure 1.4: Conflict graph for wire segments

segments) with the fewest possible labels (channel numbers) so that no two adjacent nodes carry the same identifier (no short circuits). In this particular case the minimal number of labels is four and a possible solution is to label A,B with 1; E,F,H with 2; G,D with 3; and finally C with 4. The resulting channel routing is shown on figure 1.5. The exercise of labeling the nodes of the graph with the fewest possible labels is equivalent to that of vertex coloring.

1.2.2 Register Allocation

Register allocation is the task of minimizing the number of register elements needed to implement a given set of computations. The immediate gain in finding optimal solutions is to use the least possible silicon area in the case of digital synthesis [MiLD92], or im-

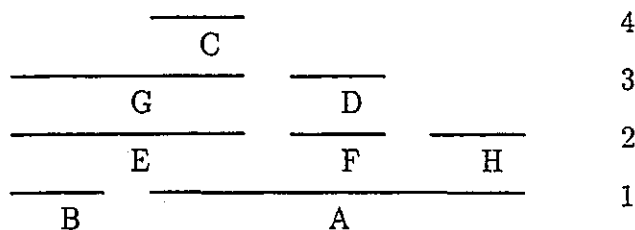


Figure 1.5: A channel routing

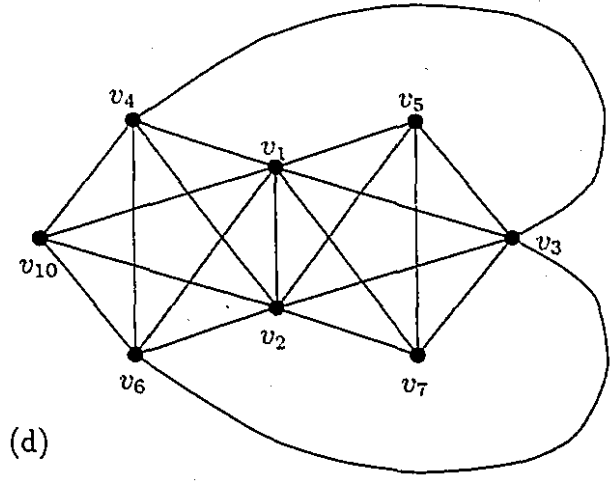
proved efficiency in the case of compiled software [CACCS81, AhSU86] as more variables are stored into CPU registers as opposed to slower storage. Consider the sequence of instructions shown on figure 1.6 (a) with $v_1, v_2, v_4, v_6, v_{10}$ as input variables and v_1, v_2 as output variables [GDWL92]. Assume the dataflow graph for the hardware schedule is shown in figure 1.6 (b). The dashed lines delimit clock periods within which several operations may run in parallel if they are not interdependent on their values. At each period the registers are read for input and then written to once the results are computed. Each operation outcome is associated with the variable below its operation circle. The line segments of figure 1.6 (c) show the useful lifetimes of the variables; the times at which variables convey desired information. Two variables which are not simultaneously live during any time segment may share the same register. Such is the case for v_6 and v_8 for example. However two variables which are both live at any given moment must be stored into distinct registers. For example v_1 and v_6 are in conflict. Extending this idea, a conflict graph is built by letting nodes represent variables and edges pairs of variables which overlap in time. In the example of figure 1.6 the conflict graph is shown in (d). The minimization task involves labeling the nodes with the fewest labels possible such that no two adjacent nodes get the same identifier. Nodes sharing a label are then made to share a register. For the graph of figure 1.6 (d) at least five labels are required and one possible labeling is v_1, v_8 with 1; v_2, v_3, v_{11} with 2; v_4, v_5 with 3; v_6, v_7 with 4; and v_9, v_{10} with 5. Once again this minimization exercise is equivalent to vertex coloring.

input $v_1, v_2, v_4, v_6, v_{10}$

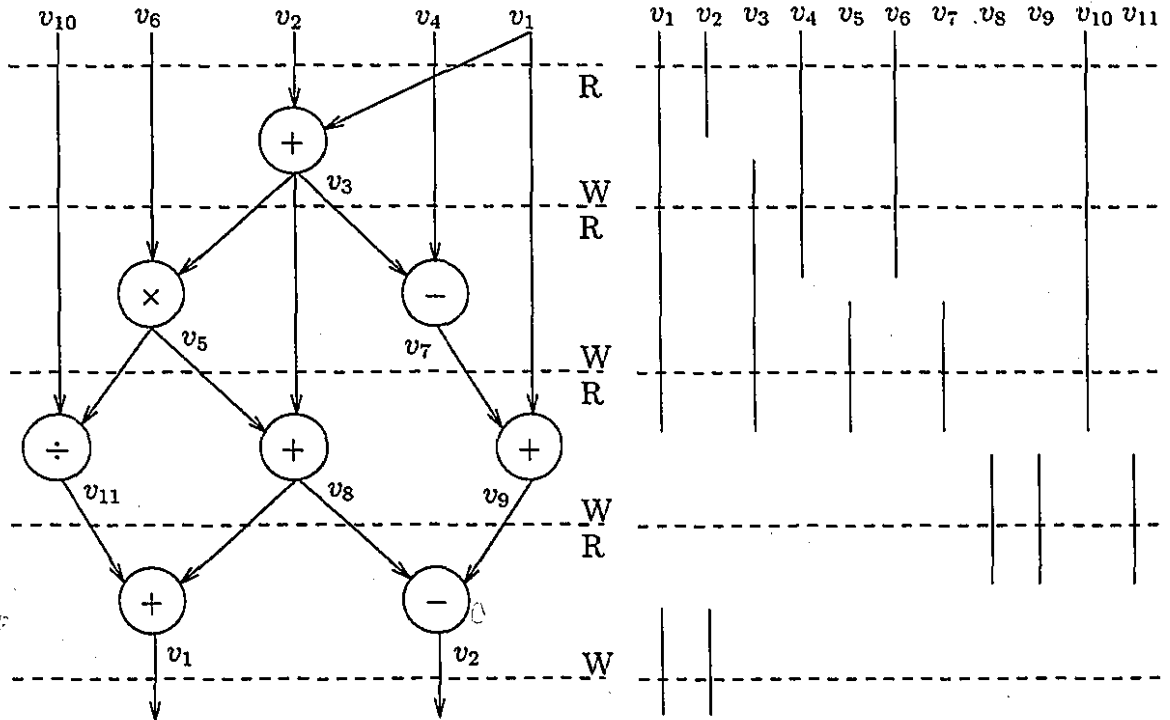
$v_3 = v_1 + v_2$
 $v_5 = v_3 \times v_6$
 $v_7 = v_3 - v_4$
 $v_8 = v_3 + v_5$
 $v_9 = v_1 + v_7$
 $v_{11} = v_{10} \div v_5$
 $v_1 = v_8 + v_{11}$
 $v_2 = v_8 - v_9$

return v_1, v_2

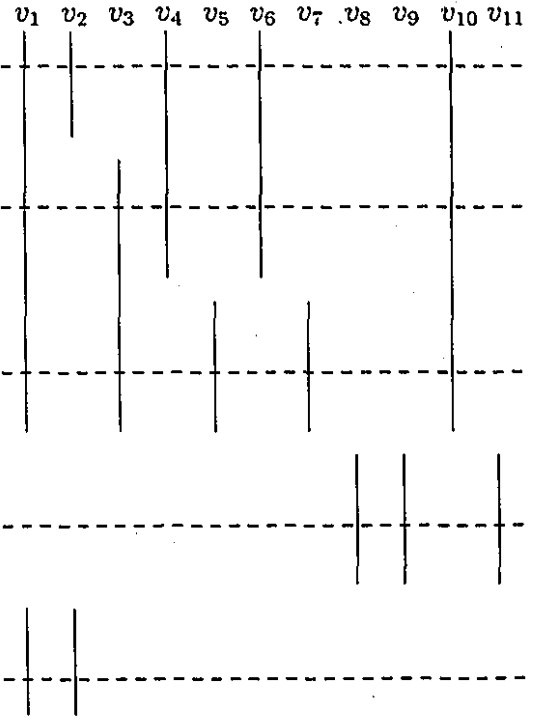
(a)



(d)



(b)



(c)

Figure 1.6: Register allocation example

1.2.3 Other problems related to vertex coloring

In addition to the examples given in the previous sections vertex coloring has other occurrences in digital synthesis. Functional units such as adders, multipliers, or any other combinatorial function can be allocated by reformulating the problem to vertex coloring [GDWL92]. As well interconnection unit allocation (buses and multiplexers) can be reduced to vertex coloring [TsSi86]. Scheduling problems involving mutually exclusive events also reduce to vertex coloring. All of the aforementioned problems exemplify the great importance of the vertex coloring in the digital synthesis framework.

But vertex coloring does not limit itself to synthesis. It models optimization and operations research problems in which the resources utilized by a system must be brought to a minimal number. Colors, each representing a resource, are distributed across the parts of the system so that the system's functional objective is achieved. Different parts may share a resource to reach their respective subgoal; however there are constraints guided by physical impossibility that prohibit some parts from sharing a common resource. Finding the minimal number of resources which satisfies both the constraints and the parts composing the system consists of the vertex coloring problem. The nature of the systems modeled by vertex coloring varies greatly: from map coloring [Berg73] to class scheduling [Gibb85] with our attention particularly set on the occurrences in digital systems design. And so does the meaning of the colors vary greatly: from time units to materials.

1.3 A Brief Dissertation outline

Now that the practical importance of the vertex coloring problem has been stressed, the remaining chapters of this dissertation will concentrate on the specifics of vertex coloring. At first a theoretical study of the problem will be presented followed by practical heuristics resting on the theory. Chapter 2 will introduce basic definitions and some work related to the theoretical and practical elements of subsequent chapters. Chapter 3 delves into necessary characteristics of optimal colorings. Chapter 4 will provide a structural analysis of the solution space and highlight its complexity. Finally, chapter 5 and the appendices are

dedicated to vertex coloring heuristics based upon the theoretical work of previous chapters.

Chapter 2

Defining Vertex Coloring

The problem of vertex coloring is not a recent one and it has been extensively studied over the last sesquicentenary [BiLW76]. The collection of published works on the subject represents an immense library and it would be futile to attempt a complete overview within this dissertation. Instead the overview presented within this chapter will concentrate on results directly relevant to the core of the subsequent work. For compendia of results on vertex coloring the reader is referred to [Lova79, Berg73, GrWa77, Tome85, Golu80].

2.1 Definitions

This section presents definitions and examples which have direct bearing on the work presented thereafter. For the remainder, the graph theoretical notation and terminology followed can be found in [Gibb85, Bigg85]. Simple, finite, undirected graphs are dealt with throughout. There will be other definitions introduced in upcoming chapters but they will pertain specifically to the sections which contain them.

Definition 2.1 (Vertex Coloring Problem) *Given an undirected graph $G = (V, E)$ with vertices*

$$V = \{v_1, v_2, \dots, v_n\}$$

and the edges defined on pairs of adjacent vertices

$$E \subseteq \{e_k = \{v_i, v_j\} \mid v_i, v_j \in V\}$$

a vertex coloring C is a mapping from the vertices to a subset S of natural numbers

$$C : V \mapsto S \subset \mathbb{N}$$

which must satisfy the following condition:

$$\{v_i, v_j\} \in E \Rightarrow C(v_i) \neq C(v_j)$$

Hence if two vertices are adjacent then they cannot be assigned the same integer.

As a pictorial aid, the integers in S are often directly associated with distinct colors of the visible spectrum. This is where the problem derives its name from. Figures 2.1 and 2.2 show graphs and their respective colorings; the vertices being represented by nodes, the edges by line segments and the colors being associated with nodes.

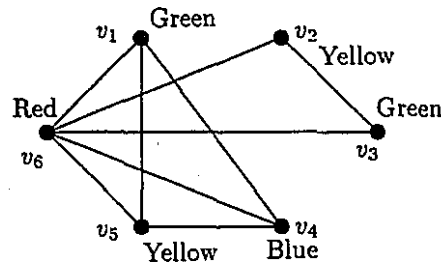


Figure 2.1: A graph and a coloring

Definition 2.2 (Optimal Vertex Coloring) For a graph $G = (V, E)$ as given in definition 2.1, an optimal vertex coloring C is one which minimizes the cardinality of S :

$$\min |S|$$

such that

$$C : V \mapsto S \subset \mathbb{N}$$

\wedge

$$C(v_i) \neq C(v_j) \quad \forall \{v_i, v_j\} \in E$$

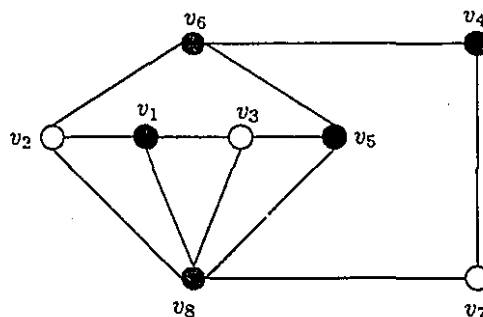


Figure 2.2: Another graph and its coloring

Note that there may be several vertex colorings which are optimal for a given graph. However there is only one possible optimal value to the minimum $|S|$ (the minimum number of colors).

Definition 2.3 (Chromatic Number) For a graph $G = (V, E)$ as given in definitions 2.1 and 2.2, the optimal value of $|S|$ is called the **chromatic number** $\chi(G)$.

The graph of figure 2.1 is optimally colored since vertices v_1, v_4, v_5, v_6 are fully interconnected and form a *complete* subgraph; therefore they each must have a different color. The chromatic number is 4 for this particular graph.

Definition 2.4 (Degree) The *degree* $d(v)$ of a vertex v is number of vertices which are adjacent to it. The **maximum degree** $\Delta(G)$ of a graph $G = (V, E)$ is the maximum of all vertex degrees in the graph: $\Delta(G) = \max_{v_i \in V} d(v_i)$.

Definition 2.5 (Edge Complement E^c) Given a graph $G = (V, E)$ as in definition 2.1, the set E^c is called the **edge complement** and is defined as follows:

$$E^c = \{(v_i, v_j) \in V \times V \mid \{v_i, v_j\} \notin E\}$$

Note that E^c consists of *ordered* pairs whereas E does not. For example both (v_1, v_3) and $(v_3, v_1) \in E^c$ for the graph of figure 2.1. This property will simplify upcoming definitions, especially that of *conflict*.

Definition 2.6 (Affinity) *The affinity α of a graph $G = (V, E)$ is a mapping from the edge complement to the natural numbers:*

$$\alpha : E^c \mapsto \mathbb{N}$$

such that:

$$\begin{aligned} \alpha(v_i, v_j) = \\ |\{v_k \in V \mid \{v_i, v_k\} \in E \wedge \{v_j, v_k\} \in E\}| \end{aligned}$$

In other words, the affinity between two vertices v_i and v_j is the number of vertices which are adjacent to both. Note the symmetry: $\alpha(v_i, v_j) = \alpha(v_j, v_i)$. In figure 2.1, $\alpha(v_1, v_3) = \alpha(v_3, v_1) = 1$.

Definition 2.7 (Conflict) *The conflict γ of a graph $G = (V, E)$ is a mapping from the edge complement to the natural numbers:*

$$\gamma : E^c \mapsto \mathbb{N}$$

such that:

$$\begin{aligned} \gamma(v_i, v_j) = \\ |\{v_k \in V \mid \{v_i, v_k\} \in E \wedge \{v_j, v_k\} \notin E\}| \end{aligned}$$

So, the conflict between two vertices v_i and v_j is the number of vertices which are adjacent to v_i but not to v_j . In figure 2.1 $\gamma(v_1, v_3) = 2$ and $\gamma(v_3, v_1) = 1$.

Definition 2.8 (Compression Transformation) *Suppose an undirected graph $G = (V, E)$ which is not complete. Then consider a graph $G' = (V', E')$ obtained by removing any two non adjacent vertices v_i and v_j in G and replacing them with a single vertex $v_{i,j}$ connected to all the adjacencies of v_i and v_j . Graph G' has one less vertex and it is said to be obtained from a **compression** of vertices v_i and v_j on G to a vertex $v_{i,j}$ in G' . In the new graph G' we have:*

$$V' = (V \setminus \{v_i, v_j\}) \cup \{v_{i,j}\}$$

and

$$E' = \{ \{v_c, v_{i,j}\} \mid \exists \{v_c, v_i\} \in E \vee \{v_c, v_j\} \in E \} \cup \\ (E \setminus \{ \{v_a, v_b\} \in E \mid (v_a = v_i) \vee (v_b = v_j) \})$$

The compression is denoted as

$$\succ (G : v_i, v_j; G' : v_{i,j})$$

We notice that in the literature a compression is also known as a *contraction-connection* [Berg73]. It is different from an *elementary contraction* or a *contraction* [Berg73] [Gibb85]. Figure 2.3 shows a compression of vertices v_1 and v_3 of the graph in figure 2.1. Compressions are instrumental in the design of some vertex coloring algorithms.

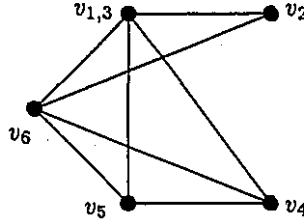


Figure 2.3: A compression of vertices v_1 and v_3 in figure 2.1

Definition 2.9 (Clique) A *clique* of a graph $G = (V, E)$ is a subset of the vertices $W = \{w_1, w_2, \dots, w_k\} \subseteq V$ such that $\forall w_i, w_j \in W \exists \{w_i, w_j\} \in E$. If W has k elements then it is said to be a k -clique.

Definition 2.10 (Clique number) The *clique number* of a graph $G = (V, E)$ is the maximum size of all cliques in G . Let \mathcal{W} be the set of all cliques in G and let $\omega(G)$ be the symbolic representation for the maximal clique number. Then:

$$\omega(G) = \max_{w_i \in \mathcal{W}} |W_i|$$

Definition 2.11 (Perfect Graph) A graph is *perfect* if and only if the chromatic number of the graph $\chi(G)$ is equal to its clique number $\omega(G)$: $\chi(G) = \omega(G)$.

Definition 2.12 (Path) A *path* is a sequence of vertices (v_1, v_2, \dots, v_p) such that consecutive vertices are adjacent and all vertices are distinct. If p is even then it is an *even path*, otherwise it is an *odd path*.

Definition 2.13 (Cycle) A *cycle* is a sequence of vertices $(v_1, v_2, \dots, v_d, v_{d+1})$ such that consecutive vertices are adjacent and all vertices are distinct except for $v_1 = v_{d+1}$. If d is even then it is coined an *even cycle*, otherwise it is an *odd cycle*. d is the length of the cycle and if $d = 3$ then we also refer to the cycle as a *triangle*. A cycle is sometimes called a *circuit*.

Figure 2.1 depicts a perfect graph since the chromatic number is 4 and $\{v_1, v_4, v_5, v_6\}$ form a clique of size 4. On the same graph, $(v_1, v_6, v_5, v_4, v_1)$ forms an even cycle of length 4.

Definition 2.14 (Orientation) An *orientation* of an undirected graph $G = (V, E)$ is the transformation of each of its edges into an ordered pair called a *directed edge*. Under an orientation each $\{v_a, v_b\} \in E$ becomes one of (v_a, v_b) or (v_b, v_a) and thus there are $2^{|E|}$ possible orientations to a graph. Given an orientation, a *directed path* is a path in which pairs of successive nodes are directed edges of the orientation.

Definition 2.14 will be of use when discussing the forthcoming Gallai-Roy theorem and related results.

2.2 Algorithmic Complexity

As demonstrated in [GaJo79] the vertex coloring problem belongs to the *NP-complete* class where no polynomial time solution has been found which will solve it under all instances. And nor do we know if one exists. Furthermore, vertex coloring appears to be a problem which cannot be approximated [Gibb85]. As opposed to some other NP-complete problems,

such as edge coloring [Vizi64, Gibb85], there is no known polynomial time approximation algorithm that guarantees solutions to vertex coloring within a reasonable bound of the optimal answer. And unless $P=NP$ then no such approximation exists. The first result in this direction was shown in [GaJo79]. It was demonstrated that if there is a polynomial time approximation to vertex coloring which is certain to return a number of colors within a factor of two of the optimal answer (for any graph in general) then there exists an optimal polynomial time algorithm for vertex coloring and $P=NP$. Short of proving $P=NP$, this result is quite negative in critical applications such as hardware synthesis. But, as will be discussed shortly, the forecast gets even grimmer.

Over the last two decades there has been a concerted effort to improve the performance of approximation algorithms for vertex coloring. Much of the research has focused on improving the performance ratio metric:

Definition 2.15 (Performance Ratio of Vertex Coloring Algorithms) *Let ϕ be a vertex coloring algorithm performing on a class of graphs C . For any graph $G \in C$, define $\phi(G)$ to be the number of colors returned by algorithm ϕ for graph G and let $\chi(G)$ be the chromatic number of G . A bound B is a performance ratio of the vertex coloring algorithm ϕ over the class C if it is provable that*

$$\frac{\phi(G)}{\chi(G)} \leq B \quad \forall G \in C$$

B is not necessarily a constant and it is often the case that a bound is known to exist within the order of a function $f(n)$ where n is the number of vertices. In such cases the performance ratio is expressed as $O(f(n))$.

After demonstrating the poor performance ratio of many graph coloring heuristics on the general problem, Johnson [John74] proposed an algorithm with a performance ratio in the order of $O(\frac{n}{\log n})$. A decade later Wigderson [Wigd83] produced an algorithm of performance ratio $O(n^{\frac{1}{k-1}})$ for the class of graphs with chromatic number smaller or equal to k (k -colorable graphs). To date the best known performance ratio is of the order $O(\frac{n(\log \log n)^2}{(\log n)^3})$ [Hall93] for general graphs. The specialized class of 3-colorable graphs has also been examined as it represents a subclass for which the problem remains NP

complete. A recent result for three colorable graphs has yielded an algorithm of performance ratio $O(n^{\frac{2}{3}} \log^{\frac{2}{3}} n)$ [Blum94]. It was in turn bested by the performance ratio of $\min\{O(\Delta^{\frac{1}{3}} \log^{\frac{2}{3}} \Delta), O(n^{\frac{1}{4}} \log n)\}$ [KaMS95] where Δ is the maximum degree. Although these results have theoretical significance they have little practical value since the ratio becomes rapidly large with increasing graph size and thus provides little in terms of practical guarantees.

By demonstrating the following theorem Lund and Yannakakis [LuYa93, LuYa94] have dismissed all aspirations of finding a constant or well behaved performance ratio:

Theorem 2.1 *There is an $\epsilon > 0$ such that vertex coloring cannot be approximated in polynomial time with performance ratio n^ϵ unless $P=NP$.*

Proof The proof can be found in [LuYa94]. □

Consequently, unless $P=NP$, there is little hope of finding approximation algorithms which have significantly better performance bounds than those previously discussed. Furthermore, unless some specific classes of problems are shown to satisfy equivalence properties, the exponent ϵ of theorem 2.1 can be shown to be strictly greater than $\frac{1}{10}$ [BeSu93] at the least. The dire outcome of theorem 2.1 is that, unless $P=NP$, the performance ratio of approximation algorithms is certain to diverge with increasing graph size and further work to improve the best known ratio can only achieve a better rate of change to the divergence. The main criticism of theorem 2.1 is that it does not apply to the specific cases in which the input instances are guaranteed to be of the k -colorable class for small k [BeSu93].

Due to the hardness of the problem, algorithms promising better bounds on specialized classes of graphs have been investigated. Polynomial time algorithms which operate on some classes of perfect graphs have been proposed to color their respective subclass optimally [Golu80, Sher93]. As well, the important class of planar graphs can be colored in linear time within a performance ratio of $\frac{5}{3}$ [ChNS80]. There are also probabilistic techniques which are known to be efficient on some simple distributions of random graphs [Wilf84]. Unfortunately much of the practical cases are not covered by these restricted algorithms and broader solutions must be found. Finally, there have been attempts

to find an efficient exponential time algorithm to optimally color a graph $G(V, E)$ of reasonably small size but the best result to date has yielded a time complexity in the order of $O(|E||V|(1 + \sqrt[3]{3})^{|V|})$ [Law176].

In later chapters, a pair of practical vertex coloring heuristics will be presented. They are called the WWI¹ algorithms. Instead of aiming at improving the overall performance ratio, the algorithms will concentrate on identifying substeps which are provably optimal during the coloration of a particular graph. Most often optimal substeps cannot be found and the algorithms fall back onto heuristics based on theoretical bounds of the chromatic number and checks on characteristics that optimal colorings must obey. Through their mechanism of operation the WWI algorithms focus on the particular instances under computation rather than providing guarantees for classes of graphs. Nevertheless it will be shown that they can be used to identify some perfect graphs which they color optimally.

2.3 A Bound on the Chromatic Number and Characteristics of Optimality

Most bounds on the chromatic number are only valid for graphs obeying a strict set of properties. However there are a few, yet weaker, bounds which are applicable to all graphs in general and thus can be used in all cases of heuristic decision making. The following bound is probably the most inked bound on the chromatic number due to its generality and the simplicity of its proof:

Theorem 2.2 *Given a graph $G = (V, E)$ with maximum degree $\Delta(G)$, the chromatic number $\chi(G)$ is bounded above by the following condition:*

$$\chi(G) \leq \Delta(G) + 1$$

Proof See [Gibb85] for a proof. □

¹The name was chosen to reflect the resemblance between the algorithms' mechanisms and the alliances of the Great War in which countries with common interests and few divergent ones sided with each other out of convenience.

As an example for the graph of figure 2.2 the usage of theorem 2.2 yields $\chi \leq 5 + 1 = 6$. Theorem 2.2 will play a role in the WWI algorithms but it will be of little significance. A quadratic bound on the number of edges and a cubic bound on the number of odd cycles shall play the key parts.

Short of enumerative and exponential methods, there are no known *sufficient* conditions which universally determine whether or not a coloring is optimal. However there does exist a set of *necessary* conditions which optimal colorings must satisfy and for which coloring solutions can be verified efficiently. One such characteristic of optimality is the Gallai-Roy theorem and it will have direct bearing on the work of chapter 4:

Theorem 2.3 [Gallai-Roy] *Consider a graph $G(V, E)$ with chromatic number $\chi(G)$. For each orientation of its edges there exists a directed path with $\chi(G)$ vertices. Furthermore there exists an orientation for which there is no directed path with more than $\chi(G)$ vertices.*

Proof See [Berg73] for a proof. □

For example, since the chromatic number of figure 2.1 is 4 it is certain that there will be a directed path of length 4 on any orientation of its edges. Theorem 2.3 will be revisited in a form which deals with colorings as opposed to directed paths. As well it will be related to a property of the local minima in the vertex coloring solution space.

Another well known result on structural properties classifies the entire class of two colorable graphs:

Theorem 2.4 *A graph $G(V, E)$ can be colored with two colors if and only if it contains no odd cycles.*

Proof See [CLiu68] for a proof. □

Beyond two colors there are no known structural characteristics such as that of theorem 2.4 which clearly delimit the chromatic number. As previously stated the optimal coloring of 3-colorable graphs remains an NP-complete problem and thus the prospect of finding a structural property which fully characterizes 3-colorable graphs is not promising.

Furthermore it has even been shown that the four coloring of 3-colorable graphs is also NP-complete [KhLS92]. As for 2-colorable graphs they can be optimally colored within polynomial time and in fact one of the two WWI algorithms will serendipitously achieve that goal. Chapter 3 will further highlight the structural importance of odd cycles with a result relating the chromatic number and odd cycles.

2.4 Studying the solution space

The sheer difficulty of coloring graphs warrants an examination of the solution space involved. A detailed study of the solution space is the object of chapter 4. It will be conducted by transforming the graph coloring problem into a non-linear mathematical program over a continuous space.

It will be apparent that graph coloring is closely related to the problem of mapping a set of variables subject to a partial ordering [AhHU82] onto a smallest possible set of values. However in a partial ordering the inequalities relating pairs of variables are of type $<, >$; thus predetermining an order between two variables. In vertex coloring the inequalities are of type $x_i \neq x_j$, therefore leaving it undetermined whether $x_i > x_j$ or $x_j < x_i$ when the problem is specified. It is this single non-deterministic difference which renders graph coloring much more difficult. Leaving the order of the inequalities undetermined fundamentally changes the solution space of graph coloring with respect to the partial ordering problem due to a combinatorial explosion of the possible solution regions in hyperspace. The partial order problem onto a smallest set of values is confined to a single solution region and it can be solved in polynomial time (in the number of variables) by using a tool such as linear programming [Khac79] or even simpler.

The solution space study will reveal additional necessary conditions which optimal colorings must obey and it proposes a simple and rapid algorithm which will improve any coloring which does not meet them. The algorithm may be used as a post-processing refinement to heuristics or it can be used to color graphs directly.

2.5 More on Algorithms

2.5.1 Greedy Algorithm

The most widespread technique to color graphs is the greedy algorithm [Bigg85]. Although it yields a poor performance it is easy to implement and has an efficient time complexity of $O(|V|^2)$. In addition the algorithm belongs in the proof of several theorems. It proceeds by sequentially traversing a set of vertices $\{v_1, v_2, \dots, v_n\}$ in order and coloring each vertex to the first color unused by its adjacent predecessors. The following is a description of the algorithm in which S represents the set of colors adjacent to a vertex and C the coloring itself:

```
Get A graph  $G = (V, E)$ 
// Color the graph
 $C(v_1) \leftarrow 1$ 
for  $i = 2$  to  $n$  do
     $S \leftarrow \emptyset$ 
    for  $j = 1$  to  $i - 1$  do
        if  $\{v_i, v_j\} \in E$  then
             $S \leftarrow S \cup \{C(v_j)\}$ 
     $k \leftarrow 1$ 
    while  $k \in S$  do
         $k \leftarrow k + 1$ 
     $C(v_i) \leftarrow k$ 
Return  $C$ 
```

In chapter 4 the greedy algorithm will be used to establish a property of the solution space and in appendix C it is modified to provide an alternate solution for the verification of a necessary characteristic of optimal colorings.

2.5.2 Coloring through compressions

The vertex coloring algorithms that are of interest within this dissertation are all based on a mechanism which follows from these simple observations on compressions. The observations are evident but their proofs have been included for sake of completeness.

Observation 2.1 Let $G' = (V', E')$ be a graph obtained from a compression on an incomplete graph $G(V, E)$. Let v_i and v_j be the vertices of V compressed into $v_{i,j}$ in V' . Then the following statements hold:

(A) Let Q_a be a vertex coloring of G' . The color assignment C_a defined on G such that

$$C_a(v_k) = Q_a(v_k) \quad \forall v_k \in V \setminus \{v_i, v_j\}$$

$$\text{and } C_a(v_i) = C_a(v_j) = Q_a(v_{i,j})$$

is a vertex coloring of G .

(B) Let C_b be a vertex coloring of G and let κ be a color which is not used in C_b . The color assignment Q_b defined on G' such that

$$Q_b(v_k) = C_b(v_k) \quad \forall v_k \in V' \setminus \{v_{i,j}\}$$

$$\text{and } Q_b(v_{i,j}) = \kappa$$

is a vertex coloring of G' .

(C) Let C_c be a vertex coloring of G with $C_c(v_i) = C_c(v_j)$. The color assignment Q_c defined on G' such that

$$Q_c(v_k) = C_c(v_k) \quad \forall v_k \in V' \setminus \{v_{i,j}\}$$

$$\text{and } Q_c(v_{i,j}) = C_c(v_i) = C_c(v_j)$$

is a vertex coloring of G' .

Proof

(A): It will be shown that $C_a(v_x) \neq C_a(v_y) \forall \{v_x, v_y\} \in E$. Consider an arbitrary edge $\{v_x, v_y\} \in E$. $\{v_x, v_y\} \neq \{v_i, v_j\}$ since two nodes must be non-adjacent to be compressed. If $v_x \in \{v_i, v_j\}$ then $\{v_y, v_{i,j}\} \in E'$ by the definition of compression. And since Q_a is a coloring, it follows that $Q_a(v_y) \neq Q_a(v_{i,j})$. From the assignment C_a it follows that $C_a(v_y) = Q_a(v_y)$ and $C_a(v_x) = Q_a(v_{i,j})$, thus $C_a(v_x) \neq C_a(v_y)$. Hence $C_a(v_x) \neq C_a(v_y)$ if $v_x \in \{v_i, v_j\}$. Similarly $C_a(v_x) \neq C_a(v_y)$ if $v_y \in \{v_i, v_j\}$. Finally if both v_x and v_y differ from v_i and v_j then $\{v_x, v_y\} \in E' \Rightarrow Q_a(v_x) \neq Q_a(v_y)$. It follows that $C_a(v_x) \neq C_a(v_y)$ and

$C_a(v_y) = Q_a(v_y)$, thus $C_a(v_x) \neq C_a(v_y)$. Therefore in all possible cases it has been shown that $C_a(v_x) \neq C_a(v_y)$ and C_a is thus a valid coloring.

(B): Consider an arbitrary edge $\{v_x, v_y\} \in E'$. If $v_x = v_{i,j}$ then $Q_b(v_x) = \kappa$ and $Q_b(v_y) = C_b(v_y)$. Since κ is not a color used in C_b it follows that $Q_b(v_x) \neq Q_b(v_y)$. Similarly, $Q_b(v_x) \neq Q_b(v_y)$ if $v_y = v_{i,j}$. If both v_x and v_y are different from $v_{i,j}$ then $\{v_x, v_y\} \in E$ and $C_b(v_x) \neq C_b(v_y)$. Since $Q_b(v_x) = C_b(v_x)$ and $Q_b(v_y) = C_b(v_y)$ then $Q_b(v_x) \neq Q_b(v_y)$. This covers all possibilities and thus $Q_b(v_x) \neq Q_b(v_y) \forall \{v_x, v_y\} \in E'$. This implies that Q_b is a coloring.

(C): Let $\{v_x, v_y\} \in E'$. If both v_x and v_y are different from $v_{i,j}$ then $\{v_x, v_y\} \in E$. Therefore $C_c(v_x) \neq C_c(v_y)$. This implies $Q_c(v_x) \neq Q_c(v_y)$. If $v_x = v_{i,j}$ then either $\{v_i, v_y\} \in E$ or $\{v_j, v_y\} \in E$. Since $C_c(v_i) = C_c(v_j)$ then $C_c(v_y) \neq C_c(v_i)$ and $C_c(v_y) \neq C_c(v_j)$. But $Q_c(v_x) = Q_c(v_{i,j}) = C_c(v_i) = C_c(v_j)$. Hence $Q_c(v_x) \neq Q_c(v_y)$. Similarly, $Q_c(v_x) \neq Q_c(v_y)$ if $v_y = v_{i,j}$. Therefore if $\{v_x, v_y\} \in E'$ then $Q_c(v_x) \neq Q_c(v_y)$ and Q_c is a coloring. \square

Via observation 2.1 (A) an algorithm can reduce the vertex coloring problem of a graph G by making an appropriate selection of two non-adjacent vertices and compressing them into a graph G' . In turn the graph G' is compressed and this procedure is repeated until a complete graph results. The action of compressing two non-adjacent nodes results in assigning them the same color in the final coloring. The number of nodes remaining on the complete graph determines the number of colors required by the algorithm. This intuitive technique for coloring graphs has long been used, see [Berg73] for example.

Figure 2.4 shows an example of the use of observation 2.1 (A) on a graph borrowed from [Bigg85]. By repeatedly using observation 2.1 (A) one can construct a coloring for the graph in (a) through successive compressions leading to (d). A possible coloring result is $C(v_1) = \text{red}$, $C(v_2) = C(v_5) = \text{green}$ and $C(v_3) = C(v_4) = C(v_6) = \text{blue}$. A practical way to achieve this is by keeping track of the compressions through the indices of the vertices. Vertices with a common color in the original graph are grouped under the index of a vertex in the final graph. For this particular example the coloring returned is an optimal one but that is not always the case.

What distinguishes compression heuristics (and in fact most vertex coloring algorithms) is the criteria used to select which pair of nodes to compress at each step. A simple scheme shall soon be presented to classify compression algorithms under general terms. But prior to that the description of a well known compression algorithm is given as a detailed example.

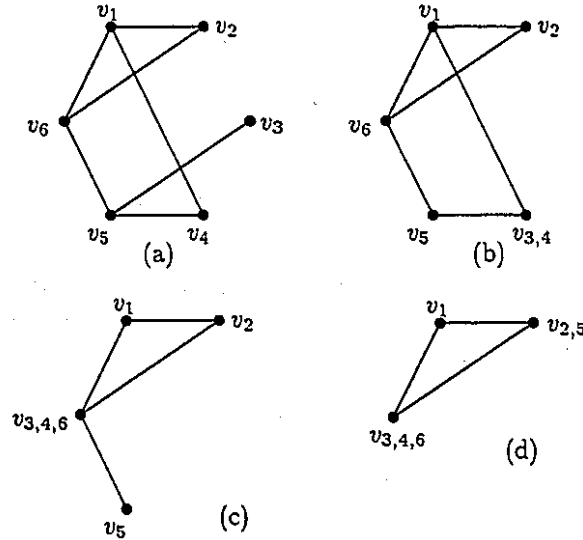


Figure 2.4: Coloring through compressions

2.5.3 Tseng's Algorithm

Tseng's algorithm [TsSi86] is a clique partitioning heuristic and it is not a vertex coloring algorithm per se. However clique partitioning and vertex coloring are two equivalent problems related through the *complement* of a graph and thus it is possible to directly translate Tseng's algorithm into a vertex coloring algorithm with little effort. For uniformity and clarity the version of Tseng's algorithm presented herein is an adapted version for vertex coloring which was obtained from the clique partitioning counterpart described in [GDWL92, MiLD92]. As with the WWI algorithms, Tseng's algorithm is based on affinity and conflict calculations (although [TsSi86] uses the values and terminology of *common neighbors* and *total neighbors* instead). A stepwise description of Tseng's heuristic follows:

1. Calculate all the affinities and conflicts for all pairs of non-adjacent vertices in the graph.
2. This step involves finding a pair of non-adjacent vertices as candidates for compression. If this is the first compression step or if a compression is no longer possible with the vertex resulting from the previous compression step then find the pair of non-adjacent vertices v_i and v_j with the minimum sum of affinities and conflicts: $\alpha(v_i, v_j) + \gamma(v_i, v_j) + \gamma(v_j, v_i)$. Otherwise if v_c was the resulting vertex from the last compression then find the vertex v_k which is non-adjacent to v_c and which yields the minimal sum: $\alpha(v_c, v_k) + \gamma(v_k, v_c) + \gamma(v_c, v_k)$. In the plausible case that several vertex pairs offer themselves as best compression candidates then select the pair v_m, v_n with the maximal affinity $\alpha(v_m, v_n)$ amongst them.
3. Compress the pair of vertices found in step 2.
4. Recalculate all the affinities and conflicts in the newly compressed graph.
5. Unless the compressed graph is complete, return to step 2.

In Tseng's algorithm there is no need to distinguish on the direction of conflict as $\gamma(v_i, v_j)$ and $\gamma(v_j, v_i)$ are always lumped into a sum. But in the WWI algorithms the separate contributions are crucial and thus Tseng's algorithm has been expressed in such terms for ease of comparison between the algorithms.

Tseng's heuristic has widespread use in digital hardware CAD systems. Compared to other known graph coloring heuristics it is of a slow time complexity of $O(|V|^2|E^c|)$ because affinity and conflict values must be recalculated after each compression step. Despite its good performance, the large complexity of the algorithm renders it less appropriate for applications which are less critical of the quality of results as opposed to their promptness (such as register allocation in standard software compilers).

As a speed up, Springer and Thomas [SpTh94] have recently introduced the EGAD heuristic based on the model of Tseng's algorithm. EGAD operates in $O(|V||E^c|)$. However it targets graphs with structures close to that of comparability graphs. In [SpTh94] it is

demonstrated that EGAD yields a performance similar to Tseng's heuristic for the problem of bus allocation.

The WWI algorithms presented in chapter 5 have a worst case performance bound of $O(|V|^3)$ and operate in a time directly proportional to $O(|V||E^c|)$. As with Tseng's heuristic, the WWI algorithms also operate on a re-evaluation of the affinities and conflicts at each compression step. However a significant speed improvement over the existing version of Tseng's algorithm is possible due to a more efficient method for recalculating the affinities and conflicts at each step. The technique is disclosed in appendix A and it can be applied to improve Tseng's algorithm to $O(|V|^3)$ as well. Finally, benchmarks on random graphs will be used to compare the performance of the WWI algorithms over Tseng's algorithm.

2.5.4 A Classification of Compressions: order, adaptability and localization

Vertex coloring heuristics are akin to approximating functions through Taylor series. Higher order Taylor series approximations signify better answers at the cost of a greater computational complexity. Similarly a notion of order can be brought to vertex coloring. Heuristics which base their compression decisions on large subgraph structures surrounding the compression pair can yield better results but they do so at the expense of a greater time spent exploring for the preferable choice. Algorithms which look at the immediate adjacencies of a pair of compression candidates are coined **first order** compressions. Algorithms which go beyond and examine the adjacencies of adjacencies of compression pairs are **second order** algorithms. And so on; the **order** being the maximal depth of the adjacencies being examined. By this definition Tseng's algorithm is clearly a first order algorithm since it depends upon affinity and conflict values. So are the WWI algorithms since they are also based on the same metrics. Chapter 5 will briefly discuss a second order algorithm of superior performance but the resulting computational complexity renders it slow and impractical. Instead, one of two WWI algorithms will be a first order approximation of this second order algorithm.

For general graphs, affinities and conflicts fully determine the nature of the first order

neighborhood of a potential compression pair and thus many first order algorithms can be described with these metrics. Finding the best criteria based on these values remains the main design issue of first order algorithms. As for designing algorithms of higher order the tradeoff for time complexity becomes rapidly costly and the useful range of the algorithms becomes limited to the smaller input cases.

The comparison with Taylor series can be taken further. Heuristics designed to work well for graphs resembling those of a particular class are comparable to adjusting the point about which a Taylor series is taken. Points nearby the Taylor series' origin are better approximated than those distant. Similarly the graphs of a structure unrelated to those of a specialized algorithm are less likely to be properly colored.

The next means which can be used to distinguish compression algorithms is their adaptability. All the algorithms presented in this thesis are **adaptive** because after each compression step the affinities and conflicts are readjusted to reflect the compressed graph. As such adaptive algorithms take profit of the past history of decisions made by the algorithm but they do so at the expense of time complexity. Algorithms which do not readjust these values but rely on the initial values of the problem throughout the compression steps are **non-adaptive**. In appendix A it is demonstrated that a non-adaptive version of the WWI algorithms or Tseng's algorithm can be implemented in the time complexity of matrix multiplication.

Finally there is a need to distinguish between a localized compression scheme versus a globalized one. A **localized** algorithm performs all of its compressions about a single vertex until it is no longer possible to compress on that vertex. Then and only then does it proceed to a new vertex. The sequence of compressions from figure 2.4 is localized since the compressions always occur on the last compressed vertex until it is adjacent to all nodes of the graph. Had v_2 and v_5 been compressed in (b) instead of $v_{3,4}$ and v_6 then the localization rules would have been violated as there were still compressions possible with $v_{3,4}$. From the second step of Tseng's algorithm it is clear that it is a localized algorithm. Algorithms which do not limit themselves to compressions on the last compressed vertex are **globalized** algorithms.

Localized algorithms do limit the selection of compressions amongst those possible with the last compressed vertex but the restriction of choices does not necessarily have a negative impact on the performance of the algorithms. For example, Tseng's algorithm performs much worse if globalized. In the case of the WWI algorithms, one will operate globally and the other locally. As it will be demonstrated localized algorithms tend to have a better behavior on graphs with a high edge density.

Chapter 3

Some Characteristics of the Chromatic Number

This chapter discusses necessary characteristics of an optimal coloring based on the concept of fundamental nodes (defined later). For colorings which do not meet the optimality criteria, methods are presented to reduce the number of colors and an algorithm is introduced. This will lead to a theorem on bicolored paths within a graph and an upper bound on the chromatic number with respect to the number of odd cycles. These results also have relevance in the design of the algorithms in chapter 5.

3.1 A First Characteristic

The following theorem is closely related to the theory of the γ -critical subgraphs [Berg73] and could be derived as a result of that theory. The characteristic the theorem introduces forms the basis of the results presented thereafter:

Theorem 3.1 *Let $G = (V, E)$ be a graph and let C be an optimal coloring which uses the set of colors S . For each $s \in S$ there exists a node $v_i \in V$ which has color $C(v_i) = s$ and which is adjacent to at least one node of each other color of $S \setminus \{s\}$ under coloring C .*

Proof Let C be an optimal coloring with the set of colors S and suppose $s \in S$ is a color for which there is no vertex $v_i \in V$ with $C(v_i) = s$ which is adjacent to at least one node of all other colors different from s in C . Then it is possible to construct a coloring Q having one less color than C :

First assign a color in Q to all the nodes colored different from s in C .

$$Q(v_j) = C(v_j) \quad \forall v_j \in V \ni C(v_j) \neq s$$

Such a partial assignment on Q is valid as C itself is a valid coloring. At this point, nodes having color s in C have not been assigned a color in Q .

Recall that there is no vertex v_i , with $C(v_i) = s$, connected to a vertex of all other colors in $S \setminus \{s\}$. This implies that for each node v_j with $C(v_j) = s$ there is at least one color $\sigma \in S \setminus \{s\}$ for which all nodes v_k with $C(v_k) = \sigma$ have the property that $\{v_j, v_k\} \notin E$. For each such v_j one can assign it the color of a respective σ ($Q(v_j) = \sigma$) since no edge can exist between v_j and all other nodes with color σ or s in C , and only nodes of color σ or s in C can be assigned a color σ in Q . This completes the coloring of all nodes in Q . Therefore Q is a vertex coloring of G which uses colors $S \setminus \{s\}$ as its set of colors. This contradicts that C is an optimal coloring. *Reductio ad absurdum.* \square

The proof of theorem 3.1 has practical use since it is constructive. After a heuristic returns a coloring, a check of theorem 3.1 can be done and if the coloring fails the criterion then a better coloring can be rapidly extracted with an algorithmic test. Such an algorithm is presented in appendix B and it operates in the order of $O(|V|^2)$ for time complexity. The algorithm can also be used to color graphs directly as well. For the benchmarks and heuristics of chapter 5 this simple check on optimality improves the coloring of many graphs of a reasonable size (more than 25 nodes).

For example, as shown in [Bigg85] if one colors the graph of figure 3.1 by using the greedy coloring algorithm one finds the coloring C with: $C(v_1) = C(v_3) = 0$, $C(v_2) = C(v_4) = 1$, $C(v_5) = 2$ and $C(v_6) = 3$. By theorem 3.1 this cannot be an optimal coloring since there is no vertex of color 0 which is connected to a node of all other colors. v_1 is not adjacent to any nodes of color 2 and v_3 is not adjacent to any nodes of color 1. Therefore one can

obtain a coloring with one less color by letting v_1 have color 2 and v_3 have color 1. That is $Q(v_2) = Q(v_3) = Q(v_4) = 1$, $Q(v_1) = Q(v_5) = 2$ and $Q(v_6) = 3$.

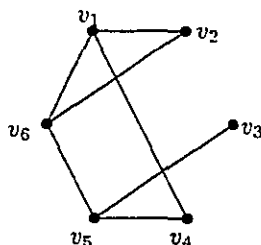


Figure 3.1: An example for theorem 3.1

One can now provide a simple proof to the following known bound on the chromatic number of a graph:

Corollary 3.1 *Let $G(V, E)$ be a graph with chromatic number $\chi(G)$ and let c be a natural number. If $\chi(G) \geq c$ then the number of vertices in G with degree $\geq c - 1$ is $\geq c$.*

Proof Let C be an optimal coloring of a graph with chromatic number $\chi(G)$. C must use $\chi(G)$ colors and c be an integer smaller or equal to $\chi(G)$. By theorem 3.1 for each color used in C there is a vertex v with an adjacency to nodes of all other colors in C . There are $\chi(G)$ such nodes (one for each color) and they must have degree of at least $\chi(G) - 1$ so that a connection is possible to nodes of all other colors. Since $c \leq \chi(G)$ it follows that the graph has at least c nodes of degree $c - 1$ or more. Hence the corollary is proven. \square

Corollary 3.1 is not a new result. The contrapositive was shown earlier, see [Berg73] for a proof. The result has been included because it is a simple consequence of theorem 3.1 and the proof relies on elementary mathematics. In figure 3.1 the graph fails the condition of corollary 3.1 for $\chi(G) \geq 4$ since it does not have at least 4 vertices of degree 3 or more. However it passes the test for $\chi(G) = 3$. Hence the chromatic number of that particular graph is at most 3. In chapter 5, corollary 3.1 will be used in the proof of another bound on the chromatic number.

Interestingly, it also is possible to derive an even stronger check for optimality from theorem 3.1. It is based on distinguishing the nodes which are adjacent to nodes of all other colors but their own:

Definition 3.1 Let C be a coloring of a graph $G = (V, E)$. And let S be the set of colors used by C . $\forall s_i \in S$ define the following sets:

$$F_C(s_i) = \{ v \mid \forall s \in S \setminus \{s_i\}, \exists w \ni: \{v, w\} \in E \wedge C(w) = s \}$$

$$R_C(s_i) = \{ v \mid C(v) = s_i \wedge v \notin F_C(s_i) \} \quad (3.1)$$

$F_C(s_i)$ is called the set of *fundamental nodes* of color s_i and $R_C(s_i)$ is the set of *residual nodes* of color s_i .

The residual nodes of any color can always be assigned different colors since each has at least one color to which it is not adjacent. And only fundamental nodes will be left in the color from which the residuals were assigned. Taken a bit further this observation yields:

Corollary 3.2 Let C be a coloring of a graph $G = (V, E)$ using the set of colors S . Consider cases for which there are two colors s_i and s_j in S such that no edge exists between a fundamental node of $F_C(s_i)$ and $F_C(s_j)$. If there is a recoloring of all nodes in $R_C(s_i)$ such that no node in $R_C(s_j)$ becomes fundamental then C is not an optimal coloring.

Proof Recolor all nodes in $R_C(s_i)$ which are adjacent to color s_j in a way that no node of $R_C(s_j)$ becomes fundamental. Then recolor all nodes in $R_C(s_j)$ which are adjacent to a node of color s_i . This leaves no edge between nodes of color s_i and s_j and therefore all nodes of color s_i and s_j can be regrouped under one color. Hence resulting in a coloring which uses one less color than C and thus contradicting that C is optimal. \square

The proof is again constructive and an algorithm can be devised to perform a check. As an example, the graph of figure 2.2 colored with $C(v_1) = 3, C(v_2) = C(v_4) = C(v_5) = 2, C(v_3) = C(v_6) = C(v_7) = 1$ and $C(v_8) = 0$. The figure shows that only 3 colors are required and therefore C is not an optimal coloring. But C still obeys the characteristic of optimality of theorem 3.1 since $F_C(3) = \{v_1\}$, $F_C(2) = \{v_2\}$, $F_C(1) = \{v_3\}$ and $F_C(0) =$

$\{v_8\}$. However note that there is no edge between v_2 and v_3 and thus no edge exists between the elements of $F_C(2)$ and $F_C(1)$. Furthermore it is possible to recolor the nodes of $R_C(2) = \{v_4, v_5\}$ which are adjacent to color 1 without making any nodes of $R_C(1) = \{v_6, v_7\}$ fundamental: let $C'(v_4) = C'(v_5) = 3$. Now recolor the nodes of $R_C(1) = \{v_6, v_7\}$ which are adjacent to color 2: let $C'(v_6) = 0$. The result is that the remaining nodes of color 2 ($\{v_2\}$) and color 1 ($\{v_3, v_7\}$) are mutually non-adjacent. Hence a common color 2 may be chosen for $\{v_2, v_3, v_7\}$. This yields a new coloring $C'(v_1) = C'(v_4) = C'(v_5) = 3$, $C'(v_2) = C'(v_3) = C'(v_7) = 2$, $C'(v_6) = C'(v_8) = 0$ which requires only three colors and the outcome is shown on figure 2.2. An interesting challenge is to find a coloring for a graph which meets the conditions of corollary 3.2 but which is not optimal.

3.2 Going further with fundamental nodes

Another question of interest pertains to the subgraph structures relating the fundamental nodes in a coloring. It is well known that not all graphs are *perfect* i.e. they might not contain a complete subgraph with as many nodes as the chromatic number. Similarly it is not a necessary condition that there is an edge between the fundamental nodes of any two distinct colors in an optimal coloring. The cycle graph of figure 3.2, with the fundamental nodes in box, illustrates this. Using the technique of Kempe chains [Kel879, Gibb85, BiLW76] we can show that for an optimal coloring C there must lie a particularly colored path between the fundamental nodes of each pair of colors. We use the term *bicoloring* as in [Berg73] instead of Kempe chain.

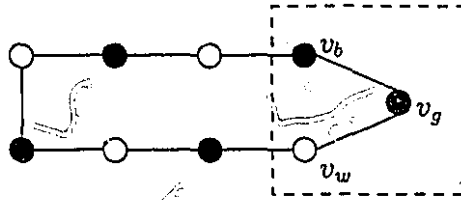


Figure 3.2: A cycle of chromatic number 3

Definition 3.2 (Bicolored Path) Let C be a coloring using the set of colors S for a graph $G = (V, E)$. A **bicolored path** over two colors $s_a, s_b \in S$ is a path such that the colors of successive nodes in the path alternate between s_a and s_b . The **length** of a bicolored path is the number of nodes in the path.

Theorem 3.2 Let C be an optimal coloring using the set of colors S for a graph $G = (V, E)$. For any two colors $s_a, s_b \in S$ there must exist a bicolored path from a fundamental node of $F_C(s_a)$ to one in $F_C(s_b)$.

Proof Suppose there are two colors $s_i, s_j \in S$ such that no bicolored path exists between a fundamental node of $F_C(s_i)$ and one in $F_C(s_j)$. With this stipulation, consider a fundamental node $f \in F_C(s_i)$ and the subgraph $G_f = (V_f, E_f)$ consisting of all nodes and edges in G that are on a bicolored path from f over colors s_i and s_j in coloring C .

Clearly, every node of V_f will either be of color s_i or s_j . It is possible to modify coloring C to C' by recoloring all nodes of color s_i in V_f to s_j and all nodes of color s_j in V_f to s_i while all the other nodes of $V \setminus V_f$ preserve their color from C . This is because any node of $V \setminus V_f$ which is adjacent to a node in V_f must have a color different from s_i or s_j . Hence interchanging colors in V_f cannot cause any coloring conflicts and C' is a valid optimal coloring of G since it uses the same number of colors as C .

V_f cannot have any fundamental nodes of $F_C(s_j)$, otherwise there would be a bicolored path from $f \in F_C(s_i)$ to a fundamental node of $F_C(s_j)$ and that would contradict the initial assumption. Therefore all nodes of color s_j in V_f are in $R_C(s_j)$. And since each node of color s_j in G_f is adjacent to at least one node of color s_i then it must be that each is non-adjacent to a color different from s_i under coloring C . Therefore when nodes of $R_C(s_j)$ in V_f are recolored to s_i in C' they each will remain non-adjacent to the same color to which they were not adjacent in C since no modifications are brought to colors other than s_i and s_j in the transformation from C to C' . Hence nodes of color s_j in V_f colored with C all become nodes of $R_{C'}(s_i)$. In a similar manner, all nodes of $R_C(s_i)$ which are also in V_f become nodes of $R_{C'}(s_j)$.

And the nodes of $R_C(s_i)$ which are in $V \setminus V_f$ must remain residual in C' . Otherwise; suppose that there is a node $r \in R_C(s_i)$ such that $r \in V \setminus V_f$ and $r \in F_{C'}(s_i)$. Since $r \in R_C(s_i)$ there is at least one color s to which r is not adjacent in C . If $s \neq s_j$ then r will not be adjacent to s in C' since the set of nodes of color s in C' is the same as in C and thus r could not be fundamental in C' . So s_j must be the only color to which r is not adjacent in C . However only a node r' of color s_i in C can change color to s_j in C' for r to gain a neighbor of color s_j so that it becomes fundamental. But this implies that two adjacent nodes r, r' both have color s_i under coloring C and that would contradict that C is a valid coloring. Hence r cannot exist and **all nodes of $R_C(s_i)$ also in $V \setminus V_f$ are residual in $R_{C'}(s_i)$.** Similarly **all nodes of $R_C(s_j)$ also in $V \setminus V_f$ are residual nodes in $R_{C'}(s_j)$.**

Node f and all other nodes of $F_C(s_i)$ also in V_f become fundamental nodes in $F_{C'}(s_j)$. This is because they are recolored to s_j as all other nodes of color s_i in V_f . And their neighbors different from color s_j in C are retained in C' as they preserve their color from C to C' . And the adjacent nodes of color s_j in C are recolored to s_i in C' since they must also be in V_f . Therefore nodes of $F_C(s_i)$ also in V_f are adjacent to all colors but s_j in C' and they become fundamental nodes of $F_{C'}(s_j)$.

And nodes of $F_C(s_i)$ also in $V \setminus V_f$ are nodes of $F_{C'}(s_i)$. If $v \in V \setminus V_f$ and v is a node of $F_C(s_i)$ then none of its adjacencies of color s_j in C is a node of V_f . All adjacencies of v which have color s_j must also be in $V \setminus V_f$ and they will therefore preserve their color in C' . And all other nodes adjacent to v but colored differently from s_j in C will also keep their coloring intact in C' since the transformation from C to C' only affects nodes of color s_i and s_j . Therefore v remains adjacent to all colors but s_i in C' and it is a fundamental node of $F_{C'}(s_i)$. Similarly all fundamental nodes of $F_C(s_j)$ also in $V \setminus V_f$ are in $F_{C'}(s_j)$. This implies that **all nodes of $F_C(s_j)$ are in $F_{C'}(s_j)$** since there are no nodes of $F_C(s_j)$ in V_f . So far we have shown that:

$$v \in V \setminus V_f \wedge v \in R_C(s_j) \Rightarrow v \in R_{C'}(s_j) \quad 1)$$

$$v \in V_f \wedge v \in R_C(s_i) \Rightarrow v \in R_{C'}(s_j) \quad 2)$$

$$v \in V \setminus V_f \wedge v \in R_C(s_i) \Rightarrow v \in R_{C'}(s_i) \quad 3)$$

$$v \in V_f \wedge C(v) = s_j \Rightarrow v \in R_{C'}(s_i) \quad 4)$$

$$v \in V_f \wedge v \in F_C(s_i) \Rightarrow v \in F_{C'}(s_j) \quad 5)$$

$$v \in F_C(s_j) \Rightarrow v \in F_{C'}(s_j) \quad 6)$$

$$v \in V \setminus V_f \wedge v \in F_C(s_i) \Rightarrow v \in F_{C'}(s_i) \quad 7)$$

$$v = f \Rightarrow v \in F_{C'}(s_j) \quad 8)$$

$$v \in F_C(s_j) \Rightarrow v \in V \setminus V_f \quad 9)$$

Therefore covering all possibilities for nodes of color s_i and s_j in C' . Since only implication (7) produces vertices in $F_{C'}(s_i)$ it must be that any element of $F_{C'}(s_i)$ is also an element of $F_C(s_i)$ i.e. $F_{C'}(s_i) \subseteq F_C(s_i)$. Furthermore, by implication (8), there is one node $v = f \in F_C(s_i)$ such that $v = f \notin F_{C'}(s_i)$. And, by implication (5), there are perhaps more such nodes. Hence there are strictly fewer fundamental nodes of color s_i in C' and $F_{C'}(s_i) \subset F_C(s_i)$.

Now we will show that there cannot be a bicolored path between nodes of $F_{C'}(s_i)$ and $F_{C'}(s_j)$. First notice that if a bicolored path over colors s_i and s_j exists under coloring C' then the same bicolored path also exists under C since the transformation from C to C' only recolors some nodes of color s_i to s_j and vice versa.

Now suppose there is a bicolored path in C' between $u \in F_{C'}(s_i)$ and $w \in F_{C'}(s_j)$. From the implications above we note that:

$$(a) \quad u \in F_C(s_i) \quad (\text{see 7 above})$$

$$(b) \quad w \in V \setminus V_f \wedge w \in F_C(s_j) \quad (\text{see 6 and 9})$$

or

$$w \in V_f \wedge w \in F_C(s_i) \quad (\text{see 5})$$

We treat both cases of w separately and show contradiction.

If $w \in V \setminus V_f \wedge w \in F_C(s_j)$ then there is a bicolored path in C' between a node $u \in F_{C'}(s_i)$ and a node $w \in F_{C'}(s_j)$. Since a bicolored path over s_i and s_j in C' is also a

bicolored path over s_i and s_j in C then there is a bicolored path between $u \in F_C(s_i)$ and a node $w \in F_C(s_j)$ under coloring C . This contradicts the initial assumption that such a path does not exist and thus this case cannot occur.

If $w \in V_f \wedge w \in F_C(s_i)$. Since there is a bicolored path in C' between u and w over s_i and s_j then there is at least one bicolored path p_1 from w to u over s_i and s_j in C . But since $w \in V_f$ there is also at least one bicolored path p_2 from f to w in C . p_2 prolonged by p_1 forms a bicolored path over s_i and s_j from f to u under coloring C and therefore $u \in V_f$. This contradicts that $u \in V \setminus V_f$ and thus this case cannot occur either.

Therefore there cannot be a bicolored path from a node of $F_{C'}(s_i)$ to a node of $F_{C'}(s_j)$ in C' . Hence from C another optimal coloring C' was generated which has at least one less fundamental node of color s_i and which also has no bicolored path between an element of $F_{C'}(s_i)$ and one in $F_{C'}(s_j)$. This was achieved by selecting an arbitrary $f \in F_C(s_i)$ and performing a recoloring about it. Emulating the procedure by selecting a node $f' \in F_{C'}(s_i)$ and performing another recoloring will yield a coloring C'' which has at least one less fundamental node of color s_i than C' and which also has no bicolored path between nodes of $F_{C''}(s_i)$ and $F_{C''}(s_j)$. Repeatedly applying the same procedure will eventually lead to a coloring C^* which has no fundamental node of color s_i . By theorem 3.1, C^* cannot be an optimal coloring and it can easily be transformed in a coloring \tilde{C} which uses one less color by recoloring the residual nodes of color s_i in C^* . Since C^* does not use more colors than C , it follows that C cannot be an optimal coloring and a contradiction arises. Hence there must always be at least one bicolored path between the fundamental nodes of any two colors of an optimal coloring. \square

Clearly the optimal coloring of figure 3.2 obeys theorem 3.2 as there is a bicolored path between all fundamental nodes, notably one of length 8 between v_b and v_w . Unlike the case illustrated in the figure, when there is more than one fundamental vertex of each color it is not always the case that there is a subset of fundamental nodes, one from each color, such that all vertices in the subset are mutually interconnected by a bicolored path. Since the proof of theorem 3.2 is constructive, it is straightforward to implement an algorithm which improves a coloring which does not meet the characteristic by emulating the proof. And

although the characteristic is in a sense weaker than that of corollary 3.2, an algorithm to check for the characteristic of theorem 3.2 is easier to implement efficiently.

No work based on Kempe chains should go without an attempt to simplify Appel and Haken's lengthy enumerative proof of the four colorability of maps [ApHa77]. However we were unable to prove or disprove the existence of a subgraph of a graph which has the same chromatic number but has an optimal coloring with only one fundamental node of each color:

Corollary 3.3 *A planar graph G which has an optimal vertex coloring requiring only one fundamental vertex of each color has chromatic number $\chi(G) \leq 4$.*

Proof Suppose such a planar graph requires 5 or more colors. Then consider an optimal coloring C which requires only one fundamental node of each color. And especially consider an arbitrary set of 5 of these fundamental vertices, say $\{f_1, f_2, f_3, f_4, f_5\}$. By theorem 3.2 there must be a bicolored path between all possible pairs of $\{f_1, f_2, f_3, f_4, f_5\}$.

For the same reason that it is impossible to draw a complete graph of 5 nodes without crossing two edges joining 4 distinct vertices it is also impossible to draw the bicolored paths between all pairs of $\{f_1, f_2, f_3, f_4, f_5\}$ without crossing two of the paths joining four distinct elements of $\{f_1, f_2, f_3, f_4, f_5\}$. Let one of the intersecting paths be from w_1 to w_2 and the other be from w_3 to w_4 such that $\{w_1, w_2, w_3, w_4\} \subseteq \{f_1, f_2, f_3, f_4, f_5\}$.

The bicolored path from w_1 to w_2 only has nodes of color $C(w_1)$ and $C(w_2)$ whereas the path from w_3 to w_4 only has nodes of color $C(w_3)$ and $C(w_4)$. Therefore the crossing of the paths cannot occur at a node and edges must be crossed. Thus contradicting that the graph is planar. \square

A necessary and sufficient condition for a graph to be two colorable is that it holds no cycles (or circuits) of odd length (theorem 2.4 [CLiu68, Bigg85]). For that reason the value of the chromatic number of a graph can be interpreted as a measure of the odd cycles and their interdependence. The following consequence to theorem 3.2 places a lower bound on the number of odd cycles that a graph of chromatic number $\chi(G)$ must have.

Corollary 3.4 (1) Let $G = (V, E)$ be a graph with chromatic number χ . Then there is a connected component H of G which has at least

$$\frac{\chi(\chi - 1)(\chi - 2)}{6}$$

odd cycles.

(2) Let G be a graph with ψ odd cycles. If $\psi \geq 1$ then the chromatic number χ of G obeys the following upper bound:

$$\chi \leq \left\lceil 1 + \sqrt[3]{3\psi + \sqrt{(3\psi)^2 - \frac{1}{27}}} + \sqrt[3]{3\psi - \sqrt{(3\psi)^2 - \frac{1}{27}}} \right\rceil$$

Otherwise, for $\psi = 0$ then $\chi \leq 2$.

Proof (1) First consider the cases for which $\chi \geq 3$. Since G has chromatic number χ then there must exist a connected subgraph H of G with chromatic number χ .

One by one, remove selected vertices of H and their incident edges such that the chromatic number of the resulting graph remains at χ . Repeat the procedure until it is no longer possible to remove a vertex without reducing the chromatic number to $\chi - 1$. The outcome will be a graph H_1 from which we can remove an arbitrary vertex v_1 to obtain a $\chi - 1$ colorable graph H'_1 . Optimally color H'_1 with C'_1 and transform C'_1 into an optimal coloring C_1 of H_1 by adding a new color for v_1 . Since v_1 is the only vertex of its color in C_1 it follows that all fundamental nodes of the $\chi - 1$ remaining colors in C_1 must be adjacent to v_1 . By theorem 3.2 it follows that there are at least

$$\frac{(\chi - 1)(\chi - 2)}{2}$$

distinct bicolored paths between the fundamental nodes of these $\chi - 1$ colors. Since these particular bicolored paths begin and terminate at fundamental nodes of different color it follows that they must have even length. And since the beginning and terminating nodes of these paths are both adjacent to v_1 it follows that there must be:

$$\frac{(\chi - 1)(\chi - 2)}{2} = \frac{(\chi - (0) - 1)(\chi - (0) - 2)}{2}$$

odd cycles including v_1 in H_1 .

Now consider H'_1 again and continue removing vertices until it is no longer possible to do so without reducing the chromatic number to $\chi - 2$. Call the resulting graph H_2 and consider the $(\chi - 2)$ colorable graph H'_2 in which an arbitrary vertex v_2 is removed from H_2 . Color H'_2 with an optimal coloring C'_2 which can be transformed into an optimal coloring C_2 of H_2 by adding a new color for v_2 . With an argument similar to that used in H_1 there must be at least:

$$\frac{(\chi - 2)(\chi - 3)}{2} = \frac{(\chi - (1) - 1)(\chi - (1) - 2)}{2}$$

distinct odd cycles which include vertex v_2 in H_2 . Furthermore since v_1 is not in H'_1 and thus H_2 , it follows that these new cycles are all different from those about v_1 in H_1 .

This procedure can be repeated until a 3 colorable graph $H_{\chi-3}$ for which a vertex $v_{\chi-3}$ must have at least:

$$\frac{2 \cdot 1}{2} = \frac{(\chi - (\chi - 3) - 1)(\chi - (\chi - 3) - 2)}{2}$$

distinct odd cycles all different from those in the previous steps. We stop at three colors since the subsequent two colorable graph has no bicolored paths when a sufficient number of vertices are removed to render it 1 colorable. Summing up the contributions we get that there are at least:

$$\begin{aligned} & \frac{(\chi - (0) - 1)(\chi - (0) - 2)}{2} + \frac{(\chi - (1) - 1)(\chi - (1) - 2)}{2} \\ & + \dots + \frac{(\chi - (\chi - 3) - 1)(\chi - (\chi - 3) - 2)}{2} \\ & = \frac{1}{2} \sum_{i=0}^{\chi-3} (\chi - (i+1))(\chi - (i+2)) = \frac{\chi(\chi-1)(\chi-2)}{6} \end{aligned} \quad (3.2)$$

odd cycles in H for $\chi \geq 3$. The summation was reduced using the following properties of integers: $\sum_{i=1}^n i = \frac{n(n+1)}{2}$ and $\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$.

The formula is also valid for $\chi = 1$ or $\chi = 2$ since it vanishes at those points. Hence a graph G of chromatic number χ must have a connected component with at least

$$\frac{\chi(\chi-1)(\chi-2)}{6}$$

odd cycles.

(2) Let ψ be the largest number of odd cycles in a connected component. The formula derived in (1) indicates that:

$$\frac{\chi(\chi-1)(\chi-2)}{6} \leq \psi$$

which transforms into the following polynomial inequality:

$$\chi^3 - 3\chi^2 + 2\chi - 6\psi \leq 0$$

For $\psi = 0$ we must have $\chi \leq 2$ since the largest root of the polynomial is at $\chi^* = 2$ and the polynomial remains of strictly positive value for all $\chi > 2$.

By using Cardano's method [Ca1545] one can determine that the above cubic has only one real root for all $\psi > \frac{1}{9\sqrt{3}}$ and it is located at

$$\chi^* = 1 + \sqrt[3]{3\psi + \sqrt{(3\psi)^2 - \frac{1}{27}}} + \sqrt[3]{3\psi - \sqrt{(3\psi)^2 - \frac{1}{27}}}$$

Therefore it follows that for $\psi \geq 1$

$$\chi \leq \left\lfloor 1 + \sqrt[3]{3\psi + \sqrt{(3\psi)^2 - \frac{1}{27}}} + \sqrt[3]{3\psi - \sqrt{(3\psi)^2 - \frac{1}{27}}} \right\rfloor$$

The integer floor is taken since χ must be an integer. □

For example the graph G of figure 3.1 has only two odd cycles: (v_1, v_2, v_6, v_1) and $(v_2, v_6, v_5, v_4, v_1, v_2)$. From corollary 3.4 we compute that $\chi(G) \leq \lfloor 3.4348 \rfloor = 3$. It is not always that the bound is so tight. Moreover, although the statement of corollary 3.4 does not directly state it, it is clear from the proof that the cycles must also obey a particular arrangement in the graph. Finally, corollary 3.4 will play a useful role in the design of the algorithms of chapter 5.

3.3 Some possibilities

Theorems 3.1, 3.2 and corollary 3.2 suggest heuristic techniques to improve colorings. For example, consider the subgraph made up the fundamental nodes. If a better coloring is to be found then that particular subgraph must be colored with fewer colors. An improvement heuristic could concentrate on re-coloring that subgraph and then proceed with the

rest of the graph in the case that a better coloring can be found for the subgraph. This can be viewed as an approximation to finding a γ -critical subgraph. Other techniques can concentrate on recoloring residual nodes so that the conditions of theorems 3.1, 3.2 or corollary 3.2 become violated. In fact such recoloring methods could provide ascent directions for techniques such as simulated annealing [PFTV88] or neural networks [TaLe91].

3.4 A brief recapitulation

This chapter presented necessary characteristics of an optimal coloring along with some upper bounds on the chromatic number. In all cases constructive proofs of the characteristics were given so that they have practical use as refinement algorithms to coloring heuristics. A simple and efficient coloring refinement algorithm was detailed in appendix B to validate this claim.

Chapter 4

A Structural Study of the Solution Space

This chapter demonstrates that although the solution space of vertex coloring problems can be quite vast, there exists a representation for which the space is very regular and well behaved. Thus dispelling beliefs that this problem of class NP is solved on irregular and rough solution spaces ([KuDe92]). The mathematical programming terminology used throughout the chapter is as in [Naza87]. The results that will be presented are not the first attempt to bring discrete NP complete problems to continuous variable representations, for some other examples see [PaRo87]. Nor is it the first attempt to bring graph coloring to a mathematical programming model. In [TCHu69] graph coloring is transformed into an integer linear programming model.

4.1 Mathematical programming models

The following theorem introduces a continuous variable mathematical programming model for vertex coloring:

Theorem 4.1 *Suppose the graph $G = (V, E)$ with $|V| = n$ must be vertex colored and assume without loss of generality that the vertices are indexed from 1 to n : $V = \{v_1, v_2, \dots, v_n\}$.*

To each $v_i \in V$ associate a real variable $x_i \in \mathbb{R}$. Let $x^* = [x_1^*, x_2^*, \dots, x_n^*]$ be an optimal solution to the following real valued mathematical program:

$$\begin{aligned} & \min x_1 \\ & \text{such that} \\ & |x_i - x_j| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ & 0 \leq x_j \leq x_1 \quad \forall v_j \in V \setminus \{v_1\} \end{aligned}$$

Then the coloring C^* defined as

$$[C^*(v_1), C^*(v_2), \dots, C^*(v_n)] = [\lfloor x_1^* \rfloor, \lfloor x_2^* \rfloor, \dots, \lfloor x_n^* \rfloor]$$

is an optimal coloring of G .

Proof We shall demonstrate the validity of this formulation by a sequence of transformations over definition 2.2. In this definition $C(v_i)$ and $C(v_j)$ are integers and the condition that $C(v_i) \neq C(v_j)$ can be replaced by the condition that $|C(v_i) - C(v_j)| \geq 1$. Also since $C(v_i) \in \mathbb{N} \quad \forall v_i \in V$ the constraint $C(v_i) \geq 0$ can be added and the problem can be restated as:

$$\begin{aligned} & \min |S| \\ & \text{such that} \\ & C : V \mapsto S \subset \mathbb{N} \\ & |C(v_i) - C(v_j)| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ & 0 \leq C(v_j) \quad \forall v_j \in V \end{aligned}$$

Without loss of generality one can impose that the integers in S must be consecutive starting with 0. This is because a coloring not obeying this rule can be trivially transformed into a coloring which does. In such a case minimizing $|S|$ is equivalent to minimizing the maximum integer (color) attributed to any node and the problem can be expressed as:

$$\begin{aligned} & \min(\max_{v_i \in V} C(v_i)) \\ & \text{such that} \\ & C : V \mapsto S \subset \mathbb{N} \\ & |C(v_i) - C(v_j)| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ & 0 \leq C(v_j) \quad \forall v_j \in V \end{aligned}$$

Since the colors of a graph can always be permuted amongst groups of nodes having the same color there is no loss in predetermining one vertex to have the highest possible color. Let this vertex be v_1 . This implies that $\min(\max_{v_i \in V} C(v_i)) = C(v_1)$ and that $C(v_j) \leq C(v_1) \forall v_j \in V \setminus \{v_1\}$. Also $C : V \mapsto S \subset \mathbb{N}$ can be changed to $C(v_j) \in \mathbb{N} \forall v_j \in V$. Hence a solution to

$$\begin{aligned} & \min C(v_1) \\ & \text{such that} \\ & |C(v_i) - C(v_j)| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ & 0 \leq C(v_j) \leq C(v_1) \quad \forall v_j \in V \setminus \{v_1\} \\ & C(v_i) \in \mathbb{N} \quad \forall v_i \in V \end{aligned} \tag{4.1}$$

will provide an optimal coloring for the graph. The formulation in program 4.1 is almost that of the theorem statement with the exception to the added constraint that variables must be integers. The remainder of the proof concentrates on removing this constraint.

Consider the less tightly constrained program in which the variables can be real valued:

$$\begin{aligned} & \min x_1 \\ & \text{such that} \\ & |x_i - x_j| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ & 0 \leq x_j \leq x_1 \quad \forall v_j \in V \setminus \{v_1\} \end{aligned} \tag{4.2}$$

and suppose $\mathbf{x}^* = [x_1^*, x_2^*, \dots, x_n^*]$ is an optimal solution. Now consider the point $\mathbf{y}^* \in \mathbb{N}^n$ defined as

$$\mathbf{y}^* = [y_1^*, y_2^*, \dots, y_n^*] = [\lfloor x_1^* \rfloor, \lfloor x_2^* \rfloor, \dots, \lfloor x_n^* \rfloor]$$

First notice that $z_i - z_j \geq 1 \Rightarrow \lfloor z_i \rfloor - \lfloor z_j \rfloor \geq 1 \quad \forall z_i, z_j \in \mathbb{R}$. This result is easily extended to $|z_i - z_j| \geq 1 \Rightarrow |\lfloor z_i \rfloor - \lfloor z_j \rfloor| \geq 1 \quad \forall z_i, z_j \in \mathbb{R}$ by observing the symmetry in the property. From the last observation a conclusion is that point \mathbf{y}^* must obey all the constraints with absolute values in program 4.2 since \mathbf{x}^* is itself a feasible point.

Then also notice that $0 \leq z_j \leq z_1 \Rightarrow 0 \leq \lfloor z_j \rfloor \leq \lfloor z_1 \rfloor \quad \forall z_j, z_1 \in \mathbb{R}$. Through this observation and the relationship of \mathbf{y}^* with the feasible point \mathbf{x}^* it follows that \mathbf{y}^* obeys all the remaining constraints of program 4.2. Hence \mathbf{y}^* is a feasible point of program 4.2. Since \mathbf{y}^* consists of integers it is also a feasible solution of 4.1.

Since x^* is an optimal solution to the minimization problem in 4.2 it follows that the optimal value of the objective function is x_1^* . But at feasible point y^* the value of the objective function is $y_1^* = \lfloor x_1^* \rfloor \leq x_1^*$. Therefore y^* must also be an optimal solution of 4.2.

In addition to the integer constraints, program 4.1 has all the constraints that program 4.2 has. Therefore it must be that the set of feasible points of 4.1 is a subset of those in 4.2. Hence the objective function cannot yield a better value in program 4.1 than in 4.2 since any optimal point of 4.1 would also be in 4.2. Since y^* is an optimal solution of 4.2 and it is also a feasible solution of 4.1 it follows that y^* must be an optimal solution to 4.1.

Therefore the coloring C^* obtained through

$$[C^*(v_1), C^*(v_2), \dots, C^*(v_n)] = [\lfloor x_1^* \rfloor, \lfloor x_2^* \rfloor, \dots, \lfloor x_n^* \rfloor]$$

is an optimal coloring of graph G and the graph coloring can be resolved by solving 4.2 and then transforming the solution as demonstrated here. \square

As $x = [n-1, n-2, \dots, 0]$ is always a feasible solution to the program in theorem 4.1 then there always exists a non-empty feasible space to the program. Note that the choice of v_1 as a vertex colored with the highest possible value was arbitrary. $|V| - 1$ other programs could have been derived with each other vertex having the highest possible value. And in fact it is the union of all the $|V|$ solution spaces which covers all possible vertex colorings for a graph. The following addresses the problem of the k -colorability of a graph:

Theorem 4.2 *Suppose the graph $G = (V, E)$ with $|V| = n$ must be vertex colored and assume without loss of generality that the vertices are indexed from 1 to n : $V = \{v_1, v_2, \dots, v_n\}$. And let k be an integer such that $1 \leq k \leq n$. Graph $G = (V, E)$ can be colored with k or fewer colors if and only if there exists a point*

$$\begin{aligned} x &= [x_1, x_2, \dots, x_n] \in \mathbb{R}^n \ni: \\ |x_i - x_j| &\geq 1 \quad \forall \{v_i, v_j\} \in E \\ 0 &\leq x_i \leq k-1 \quad \forall v_i \in V \end{aligned}$$

Proof (\Rightarrow) If the graph can be colored in less than or equal to k colors it will be demonstrated that the point exists in real space. Let C be an optimal coloring of the graph G

and let S be the set of colors it uses. Suppose $|S| = h$. Since the chromatic number of G is k or smaller it must be that $h \leq k$. Let d_0, d_1, \dots, d_{h-1} be distinct elements of S . Then construct the color assignment D as follows:

$$\begin{aligned} D(v_i) &= 0 & \forall v_i \ni: C(v_i) = d_0 \\ D(v_i) &= 1 & \forall v_i \ni: C(v_i) = d_1 \\ &\vdots \\ D(v_i) &= h-1 & \forall v_i \ni: C(v_i) = d_{h-1} \end{aligned}$$

D assigns the same integer only to those nodes which have a common color in C . Therefore it follows that:

$$D(v_i) \neq D(v_j) \quad \forall \{v_i, v_j\} \in E$$

And since D maps to integers,

$$|D(v_i) - D(v_j)| \geq 1 \quad \forall \{v_i, v_j\} \in E \quad (4.3)$$

Furthermore from the nature of the assignment D ,

$$0 \leq D(v_i) \leq h-1 \quad \forall v_i \in V$$

Since $h \leq k$ it follows that

$$0 \leq D(v_i) \leq k-1 \quad \forall v_i \in V \quad (4.4)$$

Now consider the point $x \in \mathbb{R}^n$ such that

$$x = [x_1, x_2, \dots, x_n] = [D(v_1), D(v_2), \dots, D(v_n)]$$

Because properties 4.3 and 4.4 hold, x obeys all the criteria of the theorem statement.

Hence the existence of a point has been verified.

(\Leftarrow) Suppose $\exists x = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n$ such that

$$\begin{aligned} |x_i - x_j| &\geq 1 \quad \forall v_i, v_j \in E \\ 0 &\leq x_i \leq k-1 \quad \forall v_i \in V \end{aligned}$$

Now consider the color assignment C such that

$$[C(v_1), C(v_2), \dots, C(v_n)] = [\lfloor x_1 \rfloor, \lfloor x_2 \rfloor, \dots, \lfloor x_n \rfloor]$$

Since

$$\begin{aligned} \forall \{v_i, v_j\} \in E, |x_i - x_j| \geq 1 &\Rightarrow ||x_i| - |x_j|| \geq 1 \\ &\Rightarrow |C(v_i) - C(v_j)| \geq 1 \Rightarrow C(v_i) \neq C(v_j) \end{aligned}$$

This implies that C' is a coloring for G . Also note that:

$$\begin{aligned} \forall v_i \in V, 0 \leq x_i \leq k-1 &\Rightarrow 0 \leq |x_i| \leq k-1 \\ &\Rightarrow 0 \leq C(v_i) \leq k-1 \end{aligned}$$

Because there are only k integers from 0 to $k-1$ it must be that C requires k or fewer colors. And therefore graph G can be colored with k or fewer colors. \square

A simple consequence of the previous two theorems characterizes the solution space structure in the following way:

Corollary 4.1 *The programs of theorem 4.1 and 4.2 have feasible solution spaces which are the union of a finite number of disjoint convex regions in hyperspace.*

Proof If the absolute values are dropped from their respective constraints then the programs of 4.1 and 4.2 become linear programs for which the feasible region is convex.

First notice that the constraints with absolute values are of type $|y_i - y_j| \geq 1$ and that this implies that $y_i - y_j \geq 1$ or $y_j - y_i \geq 1$. Note that the "or" is exclusive since only one of $y_i - y_j \geq 1$ or $y_j - y_i \geq 1$ may hold true; representing that there could be two subprograms for defining the feasible regions. The subprograms are constructed by selecting a constraint of type $|y_i - y_j| \geq 1$ in the original program and replacing it by $y_i - y_j \geq 1$ in one subprogram and $y_j - y_i \geq 1$ in the other with all the other constraints preserved. Clearly these two subprograms have disjoint feasible region. But the union of their feasible regions is equal to that of the original problem since any point feasible in the original problem is also feasible in one of the subproblems and any point which is not in the feasible region of the original problem is not feasible in both the subprograms.

If there are other constraints with absolute values left one can further divide each of the two subprograms by selecting another absolute value constraint and applying the same principle as before. This will yield 4 subprograms which have disjoint spaces but for which

the union of their feasible spaces is equal to that of the original feasible space. One can repeat this procedure until the constraints with absolute values are all exhausted leaving only linear constraints finally. Since there are $|E|$ constraints with absolute values in the original program this will result in $2^{|E|}$ programs with disjoint feasible regions. And since these remaining programs are all linear programs it follows that each of them has a convex feasible region. Therefore the feasible spaces of the programs in theorems 4.1 and 4.2 can be expressed as the union of a finite number of convex and disjoint regions in hyperspace. And the corollary has been demonstrated. \square

This last proof justifies the earlier claim that the graph coloring problem can be transformed into one for which the solution space is vast but well behaved and regular. Furthermore if differentiability is a concern then a constraint of type $|x_i - x_j| \geq 1$ can be replaced by the equivalent constraint that $(x_i - x_j)^2 \geq 1$ without affecting any of the other properties. Also it is a consequence that there cannot be any saddle points since all the regions are convex.

From the proof of corollary 4.1, a specific convex solution region is fully determined by the sign of the expressions within the absolute values of the constraints¹. If one is to actually carry out the construction of the $2^{|E|}$ programs as described in the proof of corollary 4.1, it can be noted that for many graphs several of the resulting linear programs will have null feasible spaces. To calculate a bound on the number of non-empty convex regions the reader is referred to a discussion on chromatic polynomials in [Gibb85].

4.2 A Slight Modification

A slight modification to the program of theorem 4.2 ensures that the program will have feasible regions with interior points. It relies upon the following two observations.

Proposition 4.1 *Let $G = (V, E)$ be a graph with $|V| = n$ and $V = \{v_1, v_2, \dots, v_n\}$. Also let k be an integer such that $1 \leq k \leq n$ and ϵ be a real number such that $0 \leq \epsilon < 1$. Then*

¹A graphical interpretation of a convex region is a directed graph with the same nodes and edges in which the edges assume the direction from the lowest colored node to the highest colored node.

the following holds:

$$\begin{aligned} \exists \mathbf{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n \ni: \\ |x_i - x_j| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ 0 \leq x_i \leq k-1 \quad \forall v_i \in V \end{aligned} \tag{4.5}$$

if and only if

$$\begin{aligned} \exists \mathbf{y} = [y_1, y_2, \dots, y_n] \in \mathbb{R}^n \ni: \\ |y_i - y_j| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ 0 \leq y_i \leq k-1+\epsilon \quad \forall v_i \in V \end{aligned} \tag{4.6}$$

Proof (\Rightarrow) Since $\epsilon \geq 0$, any feasible point of program 4.5 is also a feasible solution of program 4.6 as program 4.6 is the same as program 4.5 with the exception to some constraints being slacker. So it only remains to be shown that if there exists a feasible point for program 4.6 then there is also a feasible point in program 4.5.

(\Leftarrow) Suppose $\mathbf{y} = [y_1, y_2, \dots, y_n]$ is a feasible point to program 4.6 and then consider the point:

$$\mathbf{x} = [x_1, x_2, \dots, x_n] = [\lfloor y_1 \rfloor, \lfloor y_2 \rfloor, \dots, \lfloor y_n \rfloor]$$

With the same argument as in the proofs of theorems 4.1, 4.2 the point \mathbf{x} satisfies all the constraints with absolute value expressions. And since

$$\begin{aligned} \forall v_i \in V, \quad 0 \leq y_i \leq k-1+\epsilon \Rightarrow 0 \leq \lfloor y_i \rfloor \leq \lfloor k-1+\epsilon \rfloor \\ \Rightarrow 0 \leq x_i \leq k-1 \end{aligned}$$

Because $k-1$ is a natural number and $0 \leq \epsilon < 1$ and x_i is the integer part of y_i it follows that: $0 \leq x_i \leq k-1 \quad \forall v_i \in V$. Hence \mathbf{x} is a feasible point of program 4.5 and the proposition is proven. \square

Therefore, in the light of theorem 4.2 and proposition 4.1, if one wishes to verify if a graph $G = (V, E)$ can be colored with k or fewer colors then one can use program 4.6 instead of program 4.5. The advantage of program 4.6 is that if ϵ is strictly greater than 0 then the feasible regions of program 4.6 have $|V|$ dimensions containing interior points.

Proposition 4.2 Let $G = (V, E)$ be a graph with $|V| = n$ and $V = \{v_1, v_2, \dots, v_n\}$. Also let k be an integer such that $1 \leq k \leq n$ and ϵ be a real number such that $0 < \epsilon < 1$. If the following program has feasible regions

$$\begin{aligned} \exists \mathbf{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n \ni: \\ |x_i - x_j| \geq 1 \quad \forall \{v_i, v_j\} \in E \\ 0 \leq x_i \leq k - 1 + \epsilon \quad \forall v_i \in V \end{aligned} \tag{4.7}$$

then each of the feasible regions are n dimensional and they have interior points.

Proof Suppose there exist feasible solutions. Let $\mathbf{x}^* = [x_1^*, x_2^*, \dots, x_n^*]$ be an arbitrary feasible solution to program 4.7. Then $\mathbf{y}^0 = [y_1^0, y_2^0, \dots, y_n^0] = [\lfloor x_1^* \rfloor, \lfloor x_2^* \rfloor, \dots, \lfloor x_n^* \rfloor]$ is also a feasible point. Note that feasible points \mathbf{y}^0 and \mathbf{x}^* are within the same convex feasible region (same argument as in corollary 4.1) since all the absolute value expressions have arguments of the same sign for points \mathbf{x}^* and \mathbf{y}^0 . We now proceed to construct n other feasible points $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^n$ also within the same convex region. The individual coordinates to each of the points are calculated as follows:

$$\forall i, j \quad 1 \leq i \leq n, 1 \leq j \leq n$$

$$y_j^i = \begin{cases} y_j^0 & \text{if } y_j^0 \leq y_i^0 \text{ with } j \neq i \\ y_j^0 + \epsilon & \text{if } i = j \\ y_j^0 + \epsilon & \text{if } y_j^0 > y_i^0 \end{cases}$$

It is a matter of bookkeeping to verify that $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^n$ are also feasible points and that they are all distinct since $\epsilon > 0$. Also, they all lie within the same feasible region as \mathbf{x}^* and \mathbf{y}^0 since they do not change the signs of the arguments within the absolute value expressions.

Furthermore the point assignment ensures that the vectors $\bar{\mathbf{w}}_1 = \mathbf{y}^1 - \mathbf{y}^0, \bar{\mathbf{w}}_2 = \mathbf{y}^2 - \mathbf{y}^0, \dots, \bar{\mathbf{w}}_n = \mathbf{y}^n - \mathbf{y}^0$ are linearly independent. Briefly, this is shown through a Gaussian elimination on the matrix W whose rows are $\bar{\mathbf{w}}_1, \dots, \bar{\mathbf{w}}_n$. The elimination first targets a node v_j of the highest color at \mathbf{y}^0 and uses the j th row of W (i.e. $\bar{\mathbf{w}}_j$) to eliminate the j th coordinate on the other rows. The remaining coordinates are eliminated in the order of

the coordinates of the highest value (color) down to those of the lowest value at y^0 . If the k th coordinate is being eliminated then the k th row should be used to annihilate the k th coordinate on the other rows. This results in a matrix whose rows form a standard basis for \mathbb{R}^n and therefore $\bar{w}_1, \dots, \bar{w}_n$ are linearly independent. In the assignment of the y_j^i 's it is important that the coordinates incremented by ϵ must have the y_j^i th entry strictly greater than the y_i^0 th entry unless $i = j$; otherwise linear independence could no longer be guaranteed.

Therefore there are n linearly independent feasible directions from point y^0 and the n dimensional simplex formed by points y^0, y^1, \dots, y^n is contained in the convex feasible region under study since any convex combination of these points is also feasible. Hence the feasible region which contains $x^*, y^0, y^1, \dots, y^n$ is n dimensional and it therefore has interior points. Since x^* was an arbitrary feasible point the last statement holds for all feasible regions of the program. \square

The graph of figure 4.1 (a) is used to illustrate the discussion in proposition 4.2. Figure 4.1 (b) holds the mathematical program to verify if this graph can be colored with two colors and recall that ϵ is a real number such that $0 < \epsilon < 1$. Clearly the point $y^0 = [1, 0, 0]$ is a feasible point to this problem. Using the same construction as in proposition 4.2, three other feasible points are generated: $y^1 = [1 + \epsilon, 0, 0]$, $y^2 = [1 + \epsilon, \epsilon, 0]$, and $y^3 = [1 + \epsilon, 0, \epsilon]$. These points yield the following linearly independent feasible directions from point y^0 : $\bar{w}_1 = [\epsilon, 0, 0]$, $\bar{w}_2 = [\epsilon, \epsilon, 0]$, $\bar{w}_3 = [\epsilon, 0, \epsilon]$. Figure 4.1 (c) shows the 3-D simplex formed by points y^0, y^1, y^2, y^3 . Figure 4.2 gives an example of the necessity to have $\epsilon > 0$ to ensure that the feasible regions are n -dimensional. Figure 4.2 (a) is the example graph and (b) gives the associated program to verify that it can be colored with two or fewer colors. Figure 4.2 (c) clearly shows that if $\epsilon = 0$ then the feasible region only consists of two points. When $\epsilon > 0$ as in figure 4.2 (d) then the feasible regions (highlighted) become n -dimensional. Proposition 4.2 adds even further to the well behavedness of the feasible space both from the point of view of program stability and numerical stability. It should be apparent at this stage that choosing ϵ as close as possible to 1 will yield the highest possible volumes for the feasible regions. Having the guarantee of a feasible n -dimensional volume gives the feasible point event a finite, non-zero probability within the n -dimensional

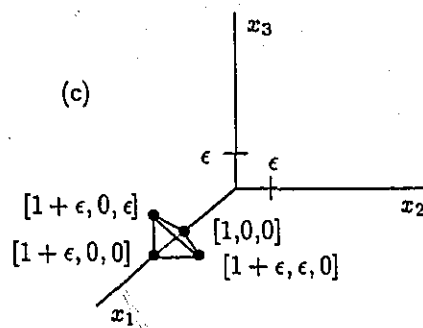
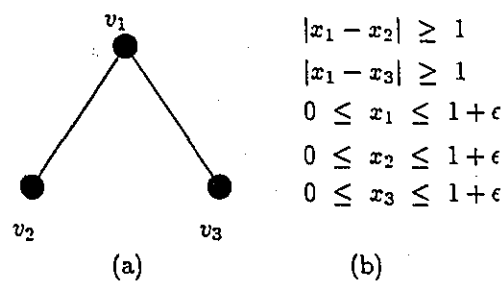


Figure 4.1: A simple 3 vertex graph

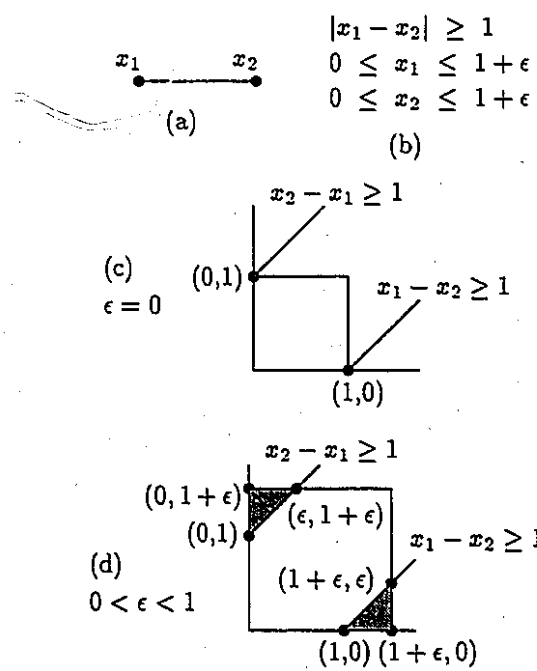


Figure 4.2: The effect of ϵ

hypercube of side $k - 1 + \epsilon$. This can be of importance if one wishes to apply search or pseudo-random techniques on the programs. Also the presence of interior points can be of importance if non-linear programming techniques are attempted on the programs. Finally note that there is a similar modification which can be brought to the programs of theorem 4.1.

4.3 Refinement method

In addition to being a tool to solve for colorings directly, the mathematical programs can be used to refine colorings produced by some heuristics. An interesting question pertains as to whether or not the colorings returned by some heuristics are guaranteed to be local minima of the convex feasible region within which their solutions lie. If not then a post processing check based on linear programming can be performed to ensure that any coloring returned is at least a local minimum. For example, the WWI heuristics of chapter 5 can return feasible colorings which are not necessarily the local minimum of their convex feasible region. As an illustration recall the graph of figure 2.2 for which the program to obtain the best coloring would be:

$$\begin{aligned} & \min x_1 \\ & \text{such that} \\ & |x_1 - x_2| \geq 1, \quad |x_1 - x_3| \geq 1, \quad |x_1 - x_8| \geq 1 \\ & |x_2 - x_6| \geq 1, \quad |x_2 - x_8| \geq 1, \quad |x_3 - x_5| \geq 1 \\ & |x_3 - x_8| \geq 1, \quad |x_4 - x_6| \geq 1, \quad |x_4 - x_7| \geq 1 \\ & |x_5 - x_6| \geq 1, \quad |x_5 - x_8| \geq 1, \quad |x_7 - x_8| \geq 1 \\ & 0 \leq x_i \leq x_1 \quad i = 2, 3, \dots, 8 \end{aligned}$$

As it will be shown, one of the WWI heuristics returns a non-optimal coloring for this particular graph:

$$\begin{aligned} C(v_1) &= 3 \\ C(v_2) &= C(v_4) = C(v_5) = 2 \\ C(v_3) &= C(v_6) = C(v_7) = 1 \\ C(v_8) &= 0 \end{aligned}$$

With respect to the mathematical program the coloring C is a feasible point in space. But is C the best point within the convex region within which it lies ? First note that the coloring C predetermines the signs within all of the absolute values of the program and thus fully specifies its convex region. The answer to the question is obtained by dropping all the absolute values and adjusting the subtraction operations so that they reflect that of the coloring:

$$\begin{aligned} & \min x_1 \\ & \text{such that} \\ & x_1 - x_2 \geq 1, \quad x_1 - x_3 \geq 1, \quad x_1 - x_8 \geq 1, \quad x_2 - x_6 \geq 1 \\ & x_2 - x_8 \geq 1, \quad x_5 - x_3 \geq 1, \quad x_3 - x_8 \geq 1, \quad x_4 - x_6 \geq 1 \\ & x_4 - x_7 \geq 1, \quad x_5 - x_6 \geq 1, \quad x_5 - x_8 \geq 1, \quad x_7 - x_8 \geq 1 \\ & 0 \leq x_i \leq x_1 \quad i = 2, 3, \dots, 8 \end{aligned}$$

When solving the above program the optimal objective function value was found to be 2 and a point which yielded it was $x = [2, 1, 1, 2, 2, 0, 1, 0]$; thereby revealing that the graph can be colored with 3 instead of 4 colors. This has the implication that the WWI heuristics may not even stop at a point which is a local minimum of the programming space. However if the graph colorings obtained by WWI are always post-processed by a linear program then it is certain that at least a local minimum will have been achieved. This situation is depicted in figure 4.3 (a) which shows that the feasible point obtained by WWI can possibly be different from the local minimum of the convex feasible region in which it is contained. However the diagram in (b) shows that after solving the associated linear program it implies the resulting point will indeed be the minimum of the region. The next section will present a specialized algorithm to solve for these programs without having to resort to generalized linear programming methods.

Unfortunately if the minimum point to the convex region is not a global minimum to the coloring problem then the only way to achieve a better coloring is to enter another feasible convex region. Since the regions are disjoint this means that the search for a better solution requires that an infeasible region of real space must be traversed before another feasible region is entered. This, in turn, implies that some graph coloring constraints must be violated in the hope that eventually a better coloring can be found. This is a difficult

problem and instead we have settled for a heuristic (WWI) which tends to find a feasible solution within a "good" feasible region and then refine them so that a local minimum is ensured. The importance of a "good" feasible region is crucial since it follows from upcoming corollary 4.3 that the ratio of a local minimum value of colors over the global minimum number of colors required can be proportional to the number of vertices and thus can become arbitrarily large as the graph size increases.

An advantage of the mathematical programming approach is that the problems which lead to a graph coloring formulation are often associated with secondary constraints which are not directly related to the problem. The linear programming methodology is very suitable for the introduction of other constraints. One should first solve the problem while ignoring these constraints if the initial heuristic used is hard to adapt to take into account the secondary constraints. Then once the linear programming stage is introduced the constraints should be added. Furthermore the additional costs incurred while introducing the secondary constraints will become apparent.

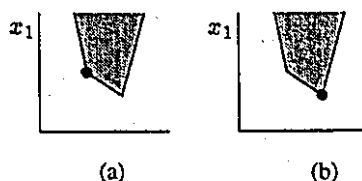


Figure 4.3: The local minimum might not be achieved

4.4 Relating Characteristics of Optimality

This section links up the local minima of the mathematical programs of theorem 4.1 and a known characteristic of optimality of graph colorings. Furthermore, efficient refinement algorithms will be provided to ensure that the characteristic is met. The results that we present require the following definition:

Definition 4.1 (Color Stratification) Suppose a graph $G = (V, E)$ and a coloring C which uses a set of m colors $K = \{k_1, k_2, \dots, k_m\}$. Let Π be an ordering of the colors in

$K, \Pi = (\pi_1, \pi_2, \dots, \pi_m)$ where each π_i corresponds to a distinct color in K . The *color stratification* of G with respect to coloring C and ordering Π is the graph

$$S = (V, \mathcal{E}) \quad \text{where}$$

$$\mathcal{E} = \{\{v_a, v_b\} \in E \mid \exists i \ni 1 \leq i \leq m-1 \text{ with}$$

$$C(v_a) = \pi_i \wedge C(v_b) = \pi_{i+1}\}$$

The color stratification is denoted as $\mathcal{L}(G, C, \Pi)$.

An important interpretation of the color stratification $\mathcal{L}(G, C, \Pi)$ is that of a stratified graph in which each stratum (row) possesses all the nodes assigned to a same color in C . The rows of nodes are stacked in the order of Π and the edges preserved in the stratified graph are only those that span across adjacent rows (successive colors in Π). Figure 4.4 is the color stratification $\mathcal{L}(G, C, (0, 1, 2, 3))$ for the graph in figure 2.2 colored with $C(v_1) = 3, C(v_2) = C(v_4) = C(v_5) = 2, C(v_3) = C(v_6) = C(v_7) = 1$ and $C(v_8) = 0$ obtained using WWI.

The motivation for the colored stratification is that it represents a two dimensional projection of the active constraints in the programs of theorems 4.1 and 4.2. The edges retained in the stratification represent the absolute value constraints which delimit the feasible point at which the coloring lies in the solution space.

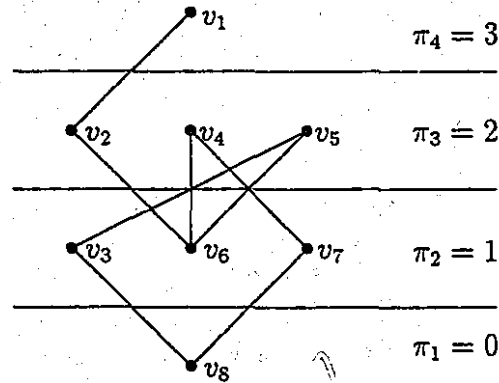


Figure 4.4: An example color stratification

Definition 4.2 (Colored Path) Given a graph $G = (V, E)$ and a coloring C , a *colored path* is a sequence of nodes in G such that successive nodes in the sequence are linked by an edge in E and each node in the sequence has a distinct color in C . The *length* of a colored path is the number of nodes in the sequence. The *stem* of a colored path is the first node in the sequence of nodes.

As an example in figure 2.2, $P = (v_4, v_6, v_2)$ is a colored path of length 3 which stems at v_4 .

Now suppose a graph $G = (V, E)$ and a coloring C which uses m colors. Let $S = (V, \mathcal{E})$ be a stratification of G to which coloring C is also applied (C is also a valid coloring of the stratified graph since $\mathcal{E} \subseteq E$). Clearly the only way that a colored path of length m can exist on the stratification is if there is a colored path which stems from the bottom row to the top row by traversing each row successively. This is because only edges between vertices of adjacent rows are preserved and backtracking is impossible since colors can only be used once in a colored path. On the stratified graph of figure 4.4, where $m = 4$, there are no colored paths of length 4 since there are no colored paths stemming at the bottom row and terminating at the top row. For that very reason it is possible to declare that the coloring associated with the stratification of figure 4.4 is not optimal. This last assertion is based on a theorem that will be presented shortly. The theorem is a characteristic of optimality which could be extracted as a consequence of theorem 2.3 and as such it does not represent a new finding. However the proof mechanism differs leading to the development of an algorithm and a relation between characteristics of optimality. For the sake of clarity we now point out that when we refer to a colored path on a stratification $\mathcal{L}(G, C, \Pi)$ it is always with respect to the coloring on which the stratification is based: C .

Theorem 4.3 Let $G = (V, E)$ be a graph and C be a coloring of G using m colors. If C is an optimal coloring of G then every stratification of G colored with C has a colored path of length m .

Proof The proof is by contradiction. Suppose there is a graph $G = (V, E)$ and optimal coloring C using m colors for which there is a stratification which has no colored path of length m . Then there is an ordering $\Pi = (\pi_1, \pi_2, \dots, \pi_m)$ to the colors in C for which the stratification $\mathcal{L}(G, C, \Pi)$ does not have a colored path of length m .

In the stratification $\mathcal{L}(G, C, \Pi)$ consider the longest colored paths which terminate at a node of color π_m . Since the stratification only preserves the edges of E between nodes of successive colors in Π , it must be that all longest colored paths leading to a node of color π_m must start at nodes of the same color, say π_x , traversing the colors in the order $\pi_x, \pi_{x+1}, \dots, \pi_{m-1}, \pi_m$. Otherwise would imply that two longest colored paths leading to color π_m could have different length, contradicting that both are of maximal length.

Because all longest colored paths ending at a node of color π_m all start from a node of color π_x , they each have length $m - x + 1$. Since there are no colored paths of length m in $\mathcal{L}(G, C, \Pi)$ it follows that $x \geq 2$. Therefore π_{x-1} is a color used in C . The remainder of the proof will make extensive use of the availability of π_{x-1} in C .

Now consider a vertex v_x of color π_x in C which is at the stem of a longest colored path $P_x = (v_x, v_{x+1}, \dots, v_{m-1}, v_m)$ terminating at a node v_m of color $C(v_m) = \pi_m$ in $\mathcal{L}(G, C, \Pi)$. It follows that v_x does not have an edge in E to any node of color π_{x-1} in C . Since the choice of v_x was arbitrary it follows that all nodes of color π_x which stem a longest colored path to a node of color π_m in $\mathcal{L}(G, C, \Pi)$ do not have any edges in E to nodes colored with π_{x-1} in C . This means that all the nodes stemming a longest colored path terminating at a node of color π_m in $\mathcal{L}(G, C, \Pi)$ could have been colored with π_{x-1} instead of π_x . It is precisely this alternate coloring C' of G which now retains our attention:

$$\forall v \in V, C'(v) = \begin{cases} \pi_{x-1} & \text{if } C(v) = \pi_x \text{ and } v \\ & \text{stems a longest colored} \\ & \text{path to a node of color} \\ & \pi_m \text{ in } \mathcal{L}(G, C, \Pi). \\ C(v) & \text{otherwise} \end{cases}$$

With coloring C' , there are three possible cases which can be distinguished. Each will contradict that C is an optimal coloring of G .

1. If the longest colored paths which terminate at a node of color π_m in $\mathcal{L}(G, C, \Pi)$ have length 1 (i.e. $\pi_x = \pi_m$) then all nodes of color π_x as each node on its own forms a path of length 1. Therefore all nodes of color π_x in C are transferred to π_{x-1} in C' . Hence, for the case $x = m$, it has been contradicted that C is an optimal coloring.

2. If the longest colored paths terminating at a node of color π_m in $\mathcal{L}(G, C, \Pi)$ have length > 1 and all nodes of color π_x stem a longest colored path to color π_m then all nodes of color π_x in C are colored with π_{x-1} in C' . As in case 1, C' requires one less color than C and therefore C cannot be an optimal coloring.

3. If the longest colored paths leading to a node of color π_m in $\mathcal{L}(G, C, \Pi)$ have length > 1 and not all nodes of color π_x stem a longest colored path to a node of color π_m then it follows that C' uses the same number of colors as C .

However the stratification $\mathcal{L}(G, C', \Pi)$ has longest colored paths terminating at color π_m which are one shorter than those in $\mathcal{L}(G, C, \Pi)$. This is because all nodes of color π_x stemming a longest colored path terminating at nodes of color π_m in $\mathcal{L}(G, C, \Pi)$ have been moved to color π_{x-1} in C' . By moving these nodes to color π_{x-1} they effectively have been disconnected from all colored paths leading to color π_m since none of these nodes can share an edge with nodes of color π_x in C' (otherwise coloring C could not have been valid since the nodes were all regrouped under color π_x within C). And none of the nodes remaining with color π_x in C' could possibly stem a colored path to a node of color π_m in $\mathcal{L}(G, C', \Pi)$. Therefore no colored path of length $m - x + 1$ terminating at a node of color π_m exists in C' . Hence it must be that the longest colored paths terminating at nodes of color π_m in $\mathcal{L}(G, C', \Pi)$ are shorter than those in $\mathcal{L}(G, C, \Pi)$. Since any such path in $\mathcal{L}(G, C, \Pi)$ with its stem removed from the beginning of the path is also present in $\mathcal{L}(G, C', \Pi)$, it follows that the longest colored paths terminating at a node of color π_m are exactly one shorter in $\mathcal{L}(G, C', \Pi)$.

This last observation is important since it guarantees that C' can be transformed into a coloring C'' in the same way that C was transformed into C' . Repeatedly applying similar transformations to generate new colorings will eventually produce a coloring C^* falling within case 1 or 2. That is because each time case 3 is encountered in the successive transformations we are guaranteed that the length of longest colored paths leading to a node of color π_m decreases by one (in the corresponding stratifications). Hence even if case 2 never occurs throughout the transformations it is guaranteed that case 1 will ultimately be reached when the length of the longest colored paths to color π_m will have shrunk to

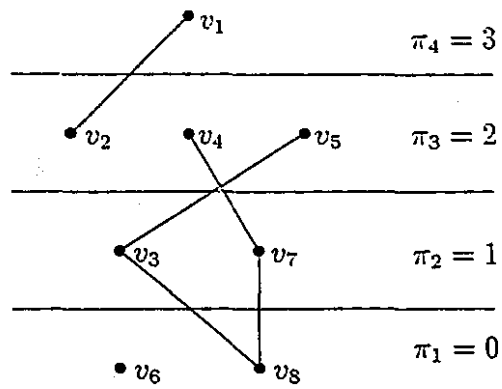
1. Therefore there will then be a coloring C^* of G which uses one less color than C , thus contradicting that C is an optimal coloring.

Hence in all the three possible cases it has been demonstrated that C is not an optimal coloring of G if there is a stratification of G and C which does not have a colored path of length m . \square

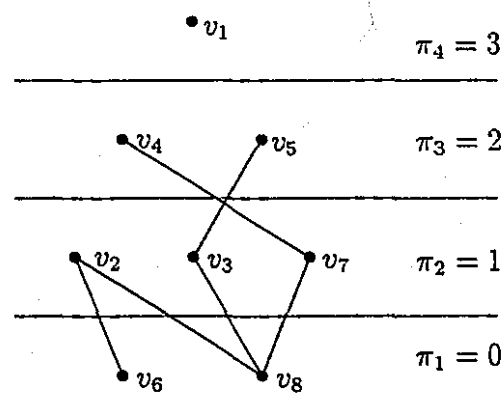
The proof of theorem 4.3 is once again constructive as it provides a simple mechanism to produce a better coloring from one which does not meet the characteristic of optimality. It is easy to devise an algorithm from the proof's methodology and figure 4.5 gives an example of the steps involved for the stratification of figure 4.4. In figure 4.4 there is only one longest colored path leading to a node of color 3 and that is $P = (v_6, v_2, v_1)$ stemming at v_6 of color 1. Bringing v_6 down to color 0 results in the stratification of figure 4.5 (a) in which the only longest colored path to a node of color 3 is $P = (v_2, v_1)$ of length 2. In turn, v_2 is brought down to color 1 and the resulting stratification is shown on figure 4.5 (b). In that particular stratification the longest colored path to a node of color 3 is $P = (v_1)$ of length 1. This signifies that the next transformation will produce a coloring which requires one less color since case 1 of the proof has now been met. When v_1 is lowered to color 2 the stratification of figure 4.5 (c) has three colors and C^* with $C^*(v_1) = C^*(v_4) = C^*(v_5) = 2$, $C^*(v_2) = C^*(v_3) = C^*(v_7) = 1$ and $C^*(v_6) = C^*(v_8) = 0$ is a better coloring for the graph of figure 2.2 than the one proposed in the stratification of figure 4.4. The following is an interesting corollary to theorem 4.3:

Corollary 4.2 *Suppose a graph $G = (V, E)$ and let C be a coloring of G using the set of colors K . If C is an optimal coloring then for each of the possible $|K|!$ orderings of the colors in K there is a colored path in G colored with C which traverses the colors in that precise order.*

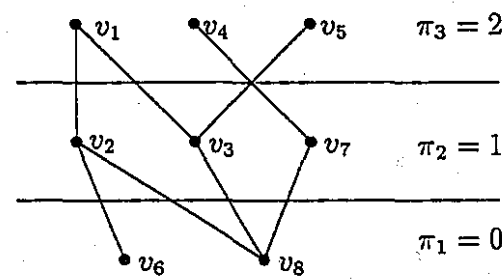
Proof Each of the $|K|!$ orderings of the colors has a corresponding colored stratification of G and C . If C is an optimal coloring then each of the stratifications has a colored path which traverses the colors of K in the same order as the stratification's ordering. Since the stratifications are formed with the same set vertices as G and a subset of the edges in G , it follows that any path in any of the stratifications is also in G . Therefore it must be that G



(a)



(b)



(c)

Figure 4.5: Refining a coloring

has a colored path which traverses the colors in all the possible orderings since there exists such paths in the stratifications. \square

For example on the graph of figure 2.2, which is optimally colored, the six colored paths $P_1 = (v_7, v_8, v_1)$, $P_2 = (v_7, v_4, v_6)$, $P_3 = (v_6, v_2, v_1)$, $P_4 = (v_8, v_5, v_3)$, $P_5 = (v_1, v_2, v_8)$, $P_6 = (v_1, v_8, v_3)$ all traverse the three colors in a different order. However the condition of corollary 4.2 is not a sufficient one. The coloring of figure 4.6 shows a colored path for all color permutations yet it is sub-optimal. As it will be shown in the upcoming theorem this unfortunately blurs further the distinction between a coloring which is only a local optimum and one which is a global optimum. This despite corollary 4.2's imposition of a possibly exponential number of constraints on a coloring. Appendix C presents a pair of

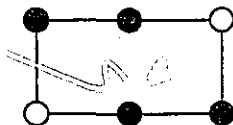


Figure 4.6: A sub-optimal coloring

simple algorithms to ensure a colored path, one of which is based on the greedy heuristic of section 2.5.1. The other algorithm is based on the proof of theorem 4.3 and it operates on the input of a graph, a coloring and an ordering to the colors. If the graph has a colored path which traverses the colors as in the ordering then the algorithm will return a coloring which requires the same number of colors as the one passed. Otherwise it will generate a coloring that requires fewer colors. The algorithm does not concern itself with finding nodes stemming longest colored paths as these are artifacts of the proof. Instead, it proceeds by indiscriminately recoloring all the nodes which it can. Both algorithms are of quadratic time complexity in the number of vertices.

The remainder of this section concerns itself with a theoretical result relating the programming approach and theorem 4.3.

Theorem 4.4 Consider the graph $G = (V, E)$ with $|V| = n$ and the vertices indexed from 1 to n : $V = \{v_1, v_2, \dots, v_n\}$. With each $v_i \in V$ associate a real variable $x_i \in \mathbb{R}$ and

define the following real valued program:

$$\begin{aligned}
 & \min x_1 \\
 & \text{such that} \\
 & |x_i - x_j| \geq 1 \quad \forall \{v_i, v_j\} \in E \\
 & 0 \leq x_j \leq x_1 \quad \forall v_j \in V \setminus \{v_1\}
 \end{aligned} \tag{4.8}$$

(1) If $\mathbf{x}^* = [x_1^*, x_2^*, \dots, x_n^*]$ is a local minimum of program 4.8 then the coloring C^* with

$$[C^*(v_1), C^*(v_2), \dots, C^*(v_n)] = [\lfloor x_1^* \rfloor, \lfloor x_2^* \rfloor, \dots, \lfloor x_n^* \rfloor]$$

has a colored path of length $C^*(v_1) + 1$ which traverses the colors in the order $0, 1, \dots, C^*(v_1)$.

(2) Let C' be a coloring of G arranged so that $C'(v_1)$ is the color with the highest integer value in the set of colors used by C' : $S = \{0, 1, \dots, C'(v_1)\}$. If C' has a colored path of length $C'(v_1) + 1$ which traverses the colors in the order $0, 1, \dots, C'(v_1)$ then $\mathbf{x}' = [C'(v_1), C'(v_2), \dots, C'(v_n)]$ is a local minimum of program 4.8.

Proof Recall from corollary 4.1 that program 4.8 has a feasible solution space which is divided into disjoint convex regions. Therefore the local minima of program 4.8 are simply the minimal points from each of the convex regions. Also recall that the characteristic distinguishing the convex regions was that a point \mathbf{x} in real space can only obey one of $x_i - x_j \geq 1$ or $x_j - x_i \geq 1$ when it is subject to the constraint $|x_i - x_j| \geq 1$.

(1) The proof is by contradiction. Suppose \mathbf{x}^* is a local minimum but C^* does not have a colored path traversing the colors in the order $0, 1, \dots, C^*(v_1)$. By corollary 4.2 C^* is not an optimal coloring and it is possible to transform C^* into a coloring \tilde{C} which requires fewer colors by using the algorithm of appendix C and the ordering $\Pi = (0, 1, \dots, C^*(v_1))$.

Node v_1 will not necessarily have the highest valued color in \tilde{C} . However, because v_1 is of the highest color in C^* , the algorithmic transformation from C^* to \tilde{C} ensures that v_1 cannot have any edges to nodes of color values above it in \tilde{C} . Hence it is possible to transform \tilde{C} into \hat{C} by recoloring v_1 to have the highest color in \tilde{C} :

$$\hat{C}(v) = \begin{cases} \tilde{C}(v) & \forall v \in V \setminus \{v_1\} \\ \max_{v_i \in V} (\tilde{C}(v_i)) & \text{if } v = v_1 \end{cases}$$

The algorithm proceeds by removing the highest ranked colors while it preserves the lower ranked ones. Therefore, since \hat{C} uses fewer colors than C^* it follows that $\hat{C}(v_1) < C^*(v_1)$. More precisely \hat{C} uses the colors $0, 1, \dots, \hat{C}(v_1)$.

Now consider the point:

$$\hat{x} = [\hat{C}(v_1), \hat{C}(v_2), \dots, \hat{C}(v_n)]$$

Because v_1 has the highest color in \hat{C} it is a consequence that \hat{x} obeys all the constraints of type $x_j \leq x_1$ in program 4.8. Furthermore \hat{x} obeys all the constraints with absolute values in 4.8, otherwise \hat{C} would not be a valid coloring of G . Finally, since \hat{C} uses the colors $0, 1, \dots, \hat{C}(v_1)$, all constraints of type $0 \leq x_j$ are also satisfied by \hat{x} . Hence \hat{x} obeys all the constraints and it is therefore a feasible point of program 4.8.

The manner in which the algorithm transforms coloring C^* into \tilde{C} is by repeatedly changing the color of a node to that of the previous color in the ordering when there are no conflicting edges. Operations of this nature never toggle the sign of the arguments within any of the absolute value expressions of program 4.8. The same observation applies to the trivial transformation from \tilde{C} to \hat{C} . Therefore it follows that \hat{x} is in the same convex region as x^* . Since $\hat{x}_1 < x_1^*$, it cannot be that x^* is a minimum of its convex region. Therefore a point x^* cannot be a local minimum of program 4.8 if its associated coloring C^* does not have a colored path of length $C^*(v_1) + 1$ which traverses the colors in the order $0, 1, \dots, C^*(v_1)$. Taken to its contrapositive this last conclusion justifies claim (1).

(2) Let $P = (v_{p_0}, v_{p_1}, \dots, v_{p_{C'(v_1)}})$ be a colored path which traverses the colors in the order $0, 1, \dots, C'(v_1)$ when C' colors G . Since sequential nodes in the path must be adjacent it follows that the following are constraints in program 4.8:

$$|x_{p_1} - x_{p_0}| \geq 1$$

$$|x_{p_2} - x_{p_1}| \geq 1$$

$$\vdots$$

$$|x_{p_{C'(v_1)-1}} - x_{p_{C'(v_1)}}| \geq 1$$

The point $x' = [C'(v_1), C'(v_2), \dots, C'(v_n)]$ must have $x'_{p_0} = C'(v_{p_0}) = 0, x'_{p_1} = C'(v_{p_1}) = 1, \dots, x'_{p_{C'(v_1)}} = C'(v_{p_{C'(v_1)}}) = C'(v_1)$; otherwise path P would not traverse the colors in the stated order. Therefore, in the convex region in which x' lies (x' is clearly a feasible point), all the feasible points have the following absolute value constraints oriented in these directions:

$$\begin{aligned} x_{p_1} - x_{p_0} &\geq 1 \\ x_{p_2} - x_{p_1} &\geq 1 \\ &\vdots \\ x_{p_{C'(v_1)-1}} - x_{p_{C'(v_1)}} &\geq 1 \end{aligned}$$

From which the following constraint implicitly holds:

$$x_{p_{C'(v_1)}} - x_{p_0} \geq C'(v_1)$$

Now since $x_{p_0} \geq 0$ is also a constraint, it follows that all points of the convex region of x' must satisfy:

$$x_{p_{C'(v_1)}} \geq C'(v_1)$$

Furthermore $x_1 \geq x_{p_{C'(v_1)}}$ is also a constraint. Therefore $x_1 \geq C'(v_1)$ for all points which lie in the same convex region as x' . This implies that x' with $x'_1 = C'(v_1)$ has the best possible value for the objective function of program 4.8 within its feasible region. Hence x' is a minimum point of its convex region and it is a local minimum of program 4.8. \square

Theorem 4.4 fully characterizes the local minima of the programs of theorem 4.1. Furthermore case (2) eliminates the need to perform general linear programming once a feasible region has been entered. It only suffices to pass a point of the convex region to the algorithm of appendix C and a (local) minimum will be returned. The algorithm of appendix C also can be used to test the existence of a path for several possible orders to the colors. For the case of graphs for which we have a coloring with few colors to start with then there are only a few orderings to check (in the case of four colors: $4! \div 2 = 12$).

An immediate consequence of theorem 4.4 concerns the disparity between local minimum values. It is shown in appendix C that the greedy algorithm of section 2.5.1 always yields

a coloring which obeys the conditions of theorem 4.4. Consequently the greedy algorithm always generates a coloring which is local minimum to its feasible region. Furthermore, it can easily be demonstrated that the greedy algorithm may yield colorings whose ratio of colors used versus the chromatic number of a graph can become arbitrarily large [BrBr87, Gibb85]. Hence,

Corollary 4.3 *For the programs of theorem 4.1, the ratio of local minimum values from the convex regions over the chromatic number can be proportional to the number of vertices and thus become arbitrarily large with increasing graph size. Furthermore convex regions of similar local minimum values are clustered together.*

This further portrays the difficulty of the solution space of graph coloring as divided regions in which the heights of the summits can vary greatly. The reasoning as to why convex regions of similar minimal values are clustered together follows from toggling the sign of a single absolute value argument defining a convex region. Such an operation can only produce a neighboring feasible region which requires one more color, one less or an equal number of colors at its local minimum.

Finally the characteristics of this section also suggest refinement heuristics for vertex coloring. Efforts can be spent on attempting to find a color permutation which has no longest colored path by stacking color strata with the intention to disrupt the colored paths as much as possible. Or target the residual nodes along the colored paths so as to break the paths when they are recolored.

4.5 Conclusion

A study of the solution space for a representation of the graph coloring problem was conducted. A characterization for the local minima of the space was found and the possible disparity of their values was shown. Although the individual solution regions were shown to be well behaved, the study also highlights the complexity introduced by the inequalities of type \neq and their divisive outcome on the solution space.

Finally, a characteristic of optimality was derived and related to the known Gallai-Roy theorem (see theorem 2.3). The constructive proof of the characteristic yielded the refinement algorithm of appendix C.

Chapter 5

Algorithms

This chapter presents the pair of WWI coloring algorithms, one of which is primarily based on the affinity metric while the other focuses on the conflict metric (defined in chapter 2). Both heuristics color graphs through compressions but they differ in their compression selection criteria. *WWI on affinities* uses the globalized compression approach of chapter 2 and is designed to target graphs of low edge densities. *WWI on conflicts* aims at graphs of higher edge density and uses a localized compression scheme.

New observations will be presented to provide a theoretical platform for the algorithms. After which the heuristics will be discussed in detail and examples will be provided. Implementation details are revealed in appendix A. Comparative benchmarks and theoretical consequences of the algorithms occupy the remainder of the chapter. The results will justify claims made about the algorithms. The final note being on classes of perfect graphs which the algorithms address.

5.1 Further Observations

The following results are instrumental in the design of the algorithms. Along with results from previous chapters, their purpose is to justify the compression criteria used by both WWI on affinities and WWI on conflicts. The terminology used is that of chapter 2.

Lemma 5.1 *A graph $G' = (V', E')$ obtained from a compression on $G = (V, E)$ has fewer or the same number of edges:*

$$|E'| \leq |E|$$

Proof Assume the compression removes vertices v_i and v_j in V and adds $v_{i,j}$ to V' . By the definition of compression, E' is constructed by removing all edges with an incidence to v_i or v_j in E and then adding one edge from $v_{i,j}$ to each adjacency of v_i or v_j in G . Since the number of nodes which are adjacent to $v_{i,j}$ can only be smaller than or equal to the number of edges incident to v_i or v_j , it follows that the number of edges added to $v_{i,j}$ can only be smaller or equal to the number of edges removed. Hence it must be that $|E'| \leq |E|$. \square

Lemma 5.2 *A graph $G' = (V', E')$ obtained from a compression on $G = (V, E)$ obeys the following chromatic number relationship:*

$$\chi(G') - 1 \leq \chi(G) \leq \chi(G')$$

Proof Observation 2.1 (A) provides a mechanism to transform any vertex coloring of G' into a vertex coloring of G without introducing any new colors. This of course includes all optimal colorings of G' and therefore, given any optimal coloring of G' there exists a coloring of G with the same number of colors. A direct consequence of this is that the chromatic number of G cannot be larger than that of G' :

$$\chi(G) \leq \chi(G')$$

Observation 2.1 (B) provides a method to transform any vertex coloring of G into a vertex coloring of G' while introducing at most one new color. Which means that for each optimal vertex coloring of G requiring $\chi(G)$ colors there exists a coloring of G' requiring $\chi(G) + 1$ colors. Therefore it must be that $\chi(G') \leq \chi(G) + 1$ which can be rewritten as:

$$\chi(G') - 1 \leq \chi(G)$$

This completes the proof as all inequalities have been verified. \square

Theorem 5.1 Given a graph $G = (V, E)$, the chromatic number $\chi(G)$ is bounded above by the following condition:

$$\chi(G) \leq \left\lfloor \frac{1 + \sqrt{1 + 8|E|}}{2} \right\rfloor$$

Proof If the graph G is complete then $\chi(G) = |V|$ and $|E| = \frac{|V| \cdot (|V| - 1)}{2}$ and the theorem statement holds since:

$$\begin{aligned} \chi(G) = |V| = \lfloor |V| \rfloor &= \left\lfloor \frac{1 + \sqrt{1 + 8 \cdot \frac{|V|(|V|-1)}{2}}}{2} \right\rfloor \\ &= \left\lfloor \frac{1 + \sqrt{1 + 8 \cdot |E|}}{2} \right\rfloor \end{aligned}$$

If the graph G is not complete then one can apply a compression transformation on G . And if the resulting graph is not complete then another compression can be applied. Clearly this process can be repeated until a complete graph G'' is reached. Suppose graph G'' has c vertices. Since it is complete then it has $\frac{c(c-1)}{2}$ edges. And by lemma 5.1 the number of edges in G'' must be smaller or equal to that of G since each compression step taken to arrive at G'' ensures the number of edges will never increase. Therefore the following inequality holds:

$$\begin{aligned} \frac{c(c-1)}{2} &\leq |E| \\ \Rightarrow c^2 - c - 2|E| &\leq 0 \end{aligned}$$

For the inequality to hold it must be that:

$$\frac{1 - \sqrt{1 + 8|E|}}{2} \leq c \leq \frac{1 + \sqrt{1 + 8|E|}}{2}$$

Since c is an integer, the statement can be strengthened to:

$$\left\lceil \frac{1 - \sqrt{1 + 8|E|}}{2} \right\rceil \leq c \leq \left\lfloor \frac{1 + \sqrt{1 + 8|E|}}{2} \right\rfloor$$

And since c is always positive the lower bound never has any practical use and it shall be dropped. Because G'' is a complete graph its chromatic number is $\chi(G'') = c$. Hence we have:

$$\chi(G'') \leq \left\lfloor \frac{1 + \sqrt{1 + 8|E|}}{2} \right\rfloor$$

By lemma 5.2 it must be that the chromatic number of G is smaller or equal to the chromatic number of G'' since G'' was obtained through successive compressions on G and the lemma ensures that the chromatic number will never decrease with compressions ($\chi(G) \leq \chi(G'')$). Therefore $\chi(G) \leq \chi(G'')$ which implies

$$\chi(G) \leq \left\lfloor \frac{1 + \sqrt{1 + 8|E|}}{2} \right\rfloor$$

Alternatively the result can be shown using corollary 3.1. However the proof is less illustrative of the upcoming compression algorithms. If $G = (V, E)$ has chromatic number $\chi(G)$ then by corollary 3.1 it follows that there are at least $\chi(G)$ nodes of degree $\chi(G) - 1$ or more in the graph. Hence the sum of the degrees for these particular nodes is at least $\chi(G)(\chi(G) - 1)$. By Euler's observation [Bigg85] the sum of the degrees for the entire graph is $2|E|$ and therefore it must be that:

$$\chi(G)(\chi(G) - 1) \leq 2|E| \Rightarrow \chi(G)^2 - \chi(G) - 2|E| \leq 0$$

By solving for the roots of the quadratic and remembering that $\chi(G)$ is positive and integral:

$$\chi(G) \leq \left\lfloor \frac{1 + \sqrt{1 + 8|E|}}{2} \right\rfloor$$

□

In the case of the graph of figure 2.2 the usage of theorem 5.1 tells us that $\chi \leq \left\lfloor \frac{1 + \sqrt{73}}{2} \right\rfloor = 4$. It is not always the case that the bound is this tight.

Theorem 5.2 Suppose two non-adjacent vertices v_i and v_j in a graph $G = (V, E)$ such that the conflict $\gamma(v_i, v_j) = 0$ or $\gamma(v_j, v_i) = 0$. Then there exists an optimal graph coloring C which assigns the same color to v_i and v_j : $C(v_i) = C(v_j)$.

Proof Suppose an optimal graph coloring C_{opt} of $G = (V, E)$. If $C_{opt}(v_i) = C_{opt}(v_j)$ then the theorem is verified. So the remainder of the proof concentrates on the case that $C_{opt}(v_i) \neq C_{opt}(v_j)$.

Now suppose that $\gamma(v_i, v_j) = 0$ and consider the following color assignment C to the vertices in V :

$$C(v_k) = C_{opt}(v_k) \quad \forall v_k \in V \setminus \{v_i\}$$

$$C(v_i) = C_{opt}(v_j)$$

In the assignment C it is clear that both v_i and v_j have the same color. Since all vertices other than v_i preserve their color from C_{opt} it must be that all pairs of adjacent vertices which exclude v_i must have different colors in assignment C . So one only needs to show that vertices adjacent to v_i have a different color to that of v_i to determine that C is a graph coloring.

In C_{opt} all adjacent vertices to v_j must be of a different color to v_j . This is also true in C since all these vertices carry over their color. Because $\gamma(v_i, v_j) = 0$ it is true that all adjacencies of v_i are also adjacencies of v_j . And since v_i is of the same color as v_j in C it must be that all adjacencies of v_i are of a different color than v_i . Therefore it has been verified that all adjacent vertices in G do not have a color match. Hence C is a graph coloring. And since it uses the same number of colors as C_{opt} this implies that it also is an optimal graph coloring. Therefore the theorem holds for $\gamma(v_i, v_j) = 0$. The argument for $\gamma(v_j, v_i) = 0$ is identical with the roles of v_i and v_j reversed. Hence the theorem holds. \square

By the very nature of the proof which constructs an optimal coloring with v_i and v_j to be of the same color from any optimal coloring with v_i and v_j differing in color, this implies that there are at least as many optimal colorings with v_i and v_j matching as there are some where v_i and v_j disagree. Hence the use of theorem 5.2 when coloring graphs will not make optimal solutions "harder" to find. In the graph of figure 2.3 $\gamma(v_2, v_5) = 0$. By theorem 5.2 there is an optimal coloring of this graph with v_2 and v_5 matching in color.

Corollary 5.1 *If there are two non adjacent vertices v_i, v_j in a graph $G = (V, E)$ such that the conflict $\gamma(v_i, v_j) = 0$ or $\gamma(v_j, v_i) = 0$ then the graph G' obtained from a compression of v_i, v_j on G has:*

$$\chi(G') = \chi(G)$$

Proof By theorem 5.2, graph G has an optimal coloring with v_i and v_j in the same color. By observation 2.1 (C) this optimal coloring can be transformed into a vertex coloring of G' without using any new colors to the $\chi(G)$ used in the optimal coloring. Therefore it

must be that $\chi(G') \leq \chi(G)$. But by lemma 5.2 $\chi(G) \leq \chi(G')$. Hence it can only be that:

$$\chi(G') = \chi(G)$$

□

Corollary 5.1 will have an important role in the algorithms presented in the next section. In conjunction with observation 2.1 (A) and compressions, it can be used to construct the coloring of a graph G from that of a compressed graph G' . In figure 2.4 (a) $\gamma(v_3, v_4) = 0$ and the vertices v_3 and v_4 are compressed for the graph in (b). In (b) $\gamma(v_{3,4}, v_6) = 0$ and the vertices $v_{3,4}$ and v_6 are compressed. In (c) $\gamma(v_5, v_2) = 0$ and there is a compression on v_2 and v_5 to yield the complete graph in (d) with chromatic number 3. Since the conditions of corollary 5.1 held at each compression step it must be that the chromatic number of the original graph is 3. It is interesting to note that as compressions progress, conflict between two arbitrary vertices may shrink to 0. For example in figure 2.4 (a) one can note that $\gamma(v_5, v_2) \neq 0$. Yet after a few compressions which transform the graph into (c) one notes that $\gamma(v_5, v_2) = 0$. It is this particular property of conflicts which renders corollary 5.1 useful in the algorithms.

Although they may appear intuitively obvious from the previous discussions, the following results will play a direct role in the design of the algorithms. That is why their presentation is given in detail.

Proposition 5.1 *If a graph $G = (V, E)$ undergoes a compression on two non-adjacent vertices v_i and v_j and the resulting graph is $G' = (V', E')$ with a new vertex $v_{i,j}$ then the degree of $v_{i,j}$ in G' is related to the following values in G :*

$$d(v_{i,j}) = \alpha(v_i, v_j) + \gamma(v_i, v_j) + \gamma(v_j, v_i)$$

Proof In G' , $v_{i,j}$ is connected to any vertex v_i or v_j was connected to in G . Since v_i and v_j are not adjacent in G and that they are the only two vertices to vanish in G' , we are guaranteed that all their adjacencies will be preserved in G' and that $v_{i,j}$ can be connected to them. The vertices for which v_i and v_j are adjacent to can be separated into three distinct groups:

1. Nodes adjacent to both v_i and v_j . $v_{i,j}$ will gain one adjacency in G' for each one of these nodes in G . By the definition of affinity there are $\alpha(v_i, v_j)$ such nodes.
2. Nodes adjacent to v_i but not v_j . $v_{i,j}$ will gain one adjacency in G' for each one of these nodes in G . By the definition of conflict there are $\gamma(v_i, v_j)$ such nodes.
3. Nodes adjacent to v_j but not v_i . $v_{i,j}$ will gain one adjacency in G' for each one of these nodes in G . By the definition of conflict there are $\gamma(v_j, v_i)$ such nodes.

Summing up the contributions to $v_{i,j}$ one gets $d(v_{i,j}) = \alpha(v_i, v_j) + \gamma(v_i, v_j) + \gamma(v_j, v_i)$. \square

Proposition 5.2 *If a graph $G = (V, E)$ undergoes a compression on two non-adjacent vertices v_i and v_j and the resulting graph is $G' = (V', E')$ then*

$$|E| - |E'| = \alpha(v_i, v_j)$$

Proof By the definition of compression, all the edges connected to v_i and v_j in G are lost and the edges connected to $v_{i,j}$ in G' are gained. So:

$$|E| - |E'| = d(v_i) + d(v_j) - d(v_{i,j})$$

Now $d(v_i)$ is the number of vertices adjacent to both v_i and v_j , $\alpha(v_i, v_j)$, in addition to the number of vertices adjacent to v_i but not v_j , $\gamma(v_i, v_j)$. Hence $d(v_i) = \alpha(v_i, v_j) + \gamma(v_i, v_j)$. Similarly, $d(v_j) = \alpha(v_i, v_j) + \gamma(v_j, v_i)$. And from proposition 5.1 $d(v_{i,j}) = \alpha(v_i, v_j) + \gamma(v_i, v_j) + \gamma(v_j, v_i)$. Therefore, by substitution,

$$\begin{aligned} |E| - |E'| &= +\alpha(v_i, v_j) + \gamma(v_i, v_j) \\ &\quad + \alpha(v_i, v_j) + \gamma(v_j, v_i) \\ &\quad - \alpha(v_i, v_j) - \gamma(v_i, v_j) - \gamma(v_j, v_i) \\ &= \alpha(v_i, v_j) \end{aligned} \tag{5.1}$$

\square

Proposition 5.3 *Let $G = (V, E)$ be a graph which is not complete. Consider the compression of two non-adjacent vertices v_i and v_j . Let*

$$\Gamma_{i,j} = \{\gamma \mid \{v_i, \gamma\} \in E \wedge \{v_j, \gamma\} \notin E\}$$

$$\Gamma_{j,i} = \{\gamma \mid \{v_j, \gamma\} \in E \wedge \{v_i, \gamma\} \notin E\}$$

The number of adjacencies between the conflicts of v_i and v_j :

$$|\{\{\gamma_a, \gamma_b\} \in E \mid \gamma_a \in \Gamma_{i,j} \wedge \gamma_b \in \Gamma_{j,i}\}|$$

represents the number of new triangles introduced by the compression. New triangles are triangles which were not present in the graph before the compression.

Proof New triangles must involve $v_{i,j}$ otherwise they would have existed in the graph prior to compression. Since all adjacencies of $v_{i,j}$ stem from adjacencies of v_i and v_j it follows that all triangles involving $v_{i,j}$ include two adjacent nodes which must belong in the affinities and conflicts of v_i and v_j . If both the nodes completing the triangle with $v_{i,j}$ are affinities of v_i and v_j then one triangle will be lost through the compression; see figure 5.1 (a). If the one of the nodes is an affinity and the other in $\Gamma_{i,j}$ or $\Gamma_{j,i}$ then the triangle is preserved through compression. Similarly if both nodes are in $\Gamma_{i,j}$ or both nodes are in $\Gamma_{j,i}$. However if one node is in $\Gamma_{i,j}$ and the other in $\Gamma_{j,i}$ then a new triangle is created through the compression. This is illustrated in figure 5.1 (b). \square

Proposition 5.4 *Let $G = (V, E)$ be a graph which is not complete. Consider the compression of two non-adjacent vertices v_i and v_j . And upper bound on the number of new triangles created by the compression is $\gamma(v_i, v_j) \cdot \gamma(v_j, v_i)$.*

Proof Clearly $|\Gamma_{i,j}| \cdot |\Gamma_{j,i}| = \gamma(v_i, v_j) \cdot \gamma(v_j, v_i)$ is the maximal number of edges between the elements of disjoint sets $\Gamma_{i,j}$ and $\Gamma_{j,i}$ of proposition 5.3. \square

5.2 WWI Algorithms

The pair of WWI algorithms is detailed in this section. As previously discussed, WWI based on affinities is a globalized compression algorithm which targets low edge density

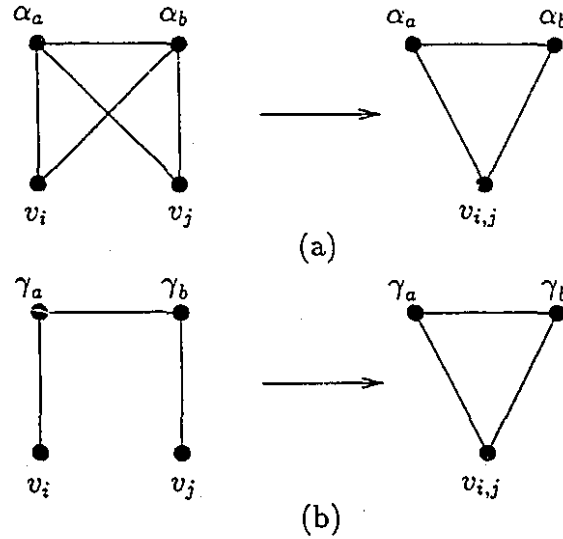


Figure 5.1: (a) A triangle is lost (b) A new triangle is created

graphs requiring few colors. By considering compression choices throughout the graph the algorithm's decisions are considerably better than if the search is limited to specific vertices. On the other hand, WWI on conflicts is a localized compression algorithm which aims at graphs of higher edge densities. At higher densities localized compressions are preferable since each compression runs a greater risk of increasing the chromatic number but a sequence of compressions about a single vertex can only increase the chromatic number by at most one. Therefore a localized compression mechanism provides further guarantees on graphs of higher density.

5.2.1 WWI on Affinities

WWI on affinities uses three criteria to determine which nodes are compressed at each stage of the algorithm. They are ranked in order of importance:

1. First the graph is examined for any pair of non-adjacent vertices v_i and v_j for which $\gamma(v_i, v_j) = 0$ or $\gamma(v_j, v_i) = 0$. If such a pair exists then v_i and v_j are compressed.

2. If no situation of 0 conflict exists then two nodes v_i and v_j with the largest possible affinity $\alpha(v_i, v_j)$ within the graph are selected to be compressed.
3. If several pairs of nodes v_i and v_j have maximal affinity then they are distinguished by the sum of their conflicts $\gamma(v_i, v_i) + \gamma(v_j, v_i)$. A pair with the largest affinity and then the smallest conflict sum is selected.

Criterion 1 is based on corollary 5.1 which guarantees that such a selection is an optimal choice by preserving the chromatic number. If zero conflicts are not present in the graph then the algorithm uses criterion 2. By proposition 5.2 the number of edges lost in the compressed graph is equal to the affinity of the compressed nodes in the graph prior to compression. Selecting a pair leading to the largest possible edge loss works in favor of the bound introduced in theorem 5.1. If several pairs of nodes have the highest affinity then one which has the smallest possible conflict sum is selected. In the light of proposition 5.1 this will create the compressed vertex with the smallest possible degree amongst the nodes of highest affinity and this will therefore work in favor of the bound of theorem 2.2. After each compression the affinities and conflicts are recalculated to reflect those of the compressed graph. Finally, criterion 2 can also be justified as a greedy step to keep the longest possible list of compression candidates for the next iteration.

Appendix A provides an implementation of WWI on affinities with an efficient method to recalculate conflicts and affinities. The resulting algorithm is of order $O(|V|^3)$ in the worst case.

5.2.2 WWI on Conflicts

The properties of affinity and conflict which drive the first order compression selection process only depend on the immediate neighbors of the compressed vertices. By concentrating on larger subgraph structures, such as those found in 2nd order criteria, a heuristic decision relies on more information and as such it could lead to better selections. However the analysis involved for second order selections is of greater complexity and the price is a worsened time performance for each compression of the algorithm.

In light of proposition 5.3 and corollary 3.4, a justifiable second order criterion is the number of edges between the conflict nodes of compression candidates. As previously observed, the compression of two vertices having the minimal value for this particular criterion is a greedy step towards introducing the fewest possible new triangles in the compressed graph (see proposition 5.3). In turn, the number of triangles influences the number of odd cycles on which the chromatic number depends (see corollary 3.4). As well, another theoretical justification for this criterion will be found in the upcoming perfect class discussion and the conflict free lemma (see lemma 5.6). Finally, empirical evidence will show that this method provides better results at the cost of a larger time complexity.

Because of the lure of improved performance from the second order algorithm, a natural goal is to search for a first order criterion which is an approximation to the second order one. Such an algorithm would have the time complexity of the first order affinity and conflict calculations while targeting performance results near those of the second order. That it precisely what WWI on conflicts sets out to achieve.

For the number of edges between the conflict nodes of two vertices v_i and v_j there is a clear upper bound at $\gamma(v_i, v_j) \cdot \gamma(v_j, v_i)$ as demonstrated in proposition 5.4. Since this bound is based on first order values of conflict it can be used as a rapid indicator of the maximal number new triangles inserted by a compression. Furthermore if edges are uniformly distributed on the graph then the product of conflicts is proportional to the number of new triangles created via a compression. And therefore $\gamma(v_i, v_j) \cdot \gamma(v_j, v_i)$ is used as a first order approximation to the aforementioned second order criterion.

At each compression step the criteria for WWI on conflicts are:

1. If there are possible compressions which include the last vertex compressed then only consider compression candidates including that vertex. Otherwise consider all candidates. This is a localized algorithm.
2. Compress the nodes v_i and v_j with the smallest conflict product: $\gamma(v_i, v_j) \cdot \gamma(v_j, v_i)$. At the same time this serendipitously gives priority to (optimal) zero conflict compressions.

3. If several pairs of vertices share the minimal value of conflict product then select amongst them the pair with the highest affinity.
4. If there is still no outstanding pair of vertices then select a pair with smallest sum of conflicts amongst the competing pairs remaining after 2 and 3.

The algorithm which uses the product of conflicts is the same as that in the WWI on affinities discussion except that the sub-algorithm of section A.2 was adjusted for the new criteria. Since the product of conflicts also stems from a first order calculation the complexity of the algorithm remains at $O(|V|^3)$. Empirical evidence will be provided to show that the first order approximation achieves its objective of emulating the second order.

5.2.3 Discussion on Algorithms

Tseng's algorithm [TsSi86] discussed in chapter 2 has metrics similar to affinity and conflict in its operation. Their algorithm addresses the related problem of clique partitioning. The algorithms differ as theirs is localized of the graphs whereas WWI on affinities is globalized and WWI on conflicts is localized, the node selection criterion differs, the theoretical justifications and outcomes of the algorithms also differ. Finally, the WWI algorithms operate in the worst case time order of $O(|V|^3)$ whereas the algorithm in [TsSi86] is of order $O(|V|^2|E^c|)$. However, using the same technique as in appendix A, Tseng's algorithm could also be implemented in $O(|V|^3)$. The method is universal to all conflict and affinity calculations. The benchmarks will demonstrate that the WWI algorithms actually operate in a time proportional to $O(|V||E^c|)$ (this becomes intuitively obvious in appendix A when it is shown that only the edge complement needs to be traversed at each compression step).

5.3 An Example of WWI on Affinities at Work

Figure 5.2 shows an example of a graph being colored with the WWI on affinities algorithm. Figure 5.2 (a) shows the graph to color and the affinity-conflict table for that particular graph. Since there are no conflicts with zero value in the graph the algorithm turns to

finding the pair of vertices with the highest possible affinity. Three pairs of nodes have the highest affinity, namely $\{v_3, v_6\}, \{v_3, v_7\}, \{v_6, v_7\}$, however only one has the minimal sum of conflicts and that is $\{v_3, v_6\}$ with $\gamma(v_3, v_6) + \gamma(v_6, v_3) = 2$. Therefore v_3 and v_6 are compressed and the resulting graph is shown in figure 5.2 (b) along with its affinity-conflict table. At that stage nodes v_7 and $v_{3,6}$ are compressed since they have the highest affinity of the graph and no zero conflicts are present. The resulting graph is shown in figure 5.2 (c). The affinity conflict table is uniform throughout because of the symmetry between elements of the edge complement. Clearly it does not matter which pair is selected at this point but the algorithm proceeds by arbitrarily selecting the pair v_5 and v_9 . This yields the graph of figure 5.2 (d). From which the selection of nodes v_4 and v_8 is optimal since these nodes are in a zero conflict situation (in fact both $\gamma(v_4, v_8) = 0$ and $\gamma(v_8, v_4) = 0$). Compressing v_4 and v_8 produces another symmetric graph, that of figure 5.2 (e). Compressing the nodes v_2 and $v_{5,9}$ returns the graph of figure 5.2 (f). Finally compressing nodes v_1 and $v_{4,8}$ results in the complete graph of figure 5.2 (g). Therefore the graph of figure 5.2 (a) can be colored with 3 colors with $C(v_1) = C(v_4) = C(v_8) = 2$, $C(v_2) = C(v_5) = C(v_9) = 1$ and $C(v_3) = C(v_6) = C(v_7) = 0$.

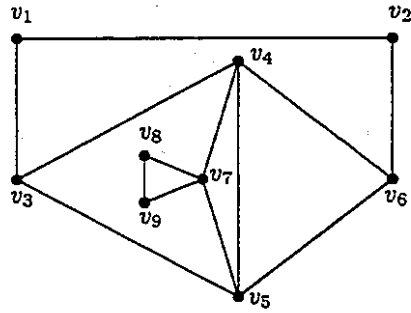
5.4 Implications of WWI on affinities

The following results provide theoretical indication that WWI on affinities is well suited for graphs of low edge density and small chromatic number.

Theorem 5.3 *Let $G = (V, E)$ be a graph such that its chromatic number is $\chi(G) \leq 2$. The WWI on affinities algorithm returns an optimal coloring for G .*

Proof The algorithm proceeds by first finding pairs of nodes v_i and v_j such that $\gamma(v_i, v_j) = 0$ or $\gamma(v_j, v_i) = 0$. If such a pair of nodes exists then it compresses both nodes to a common color. As demonstrated in theorem 5.2 this assignment is optimal and thus the proof only needs to concentrate on the cases that such a pair does not exist.

In the case that zero conflict pairs do not exist then the algorithm selects two nodes $v_p, v_q \in G$ which have the highest affinity and it then compresses them into the same

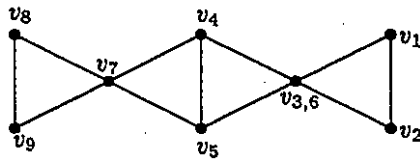


E^c	v_a	v_1	v_1	v_1	v_1	v_1	v_1	v_2	v_2
	v_b	v_4	v_5	v_6	v_7	v_8	v_9	v_3	v_4
$\alpha(v_a, v_b)$		1	1	1	0	0	0	1	1
$\gamma(v_a, v_b)$		1	1	1	2	2	2	1	1
$\gamma(v_b, v_a)$		3	3	2	4	2	2	2	3

E^c	v_a	v_2	v_2	v_2	v_2	v_3	v_3	v_3	v_3
	v_b	v_5	v_7	v_8	v_9	v_6	v_7	v_8	v_9
$\alpha(v_a, v_b)$		1	0	0	0	2	2	0	0
$\gamma(v_a, v_b)$		1	2	2	2	1	1	3	3
$\gamma(v_b, v_a)$		3	4	2	2	1	2	2	2

E^c	v_a	v_4	v_4	v_5	v_5	v_6	v_6	v_6
	v_b	v_8	v_9	v_8	v_9	v_7	v_8	v_9
$\alpha(v_a, v_b)$		1	1	1	1	2	0	0
$\gamma(v_a, v_b)$		3	3	3	3	1	3	3
$\gamma(v_b, v_a)$		1	1	1	1	2	2	2

(a)

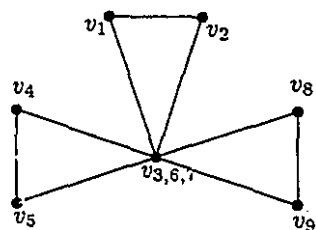


E^c	v_a	v_1	v_1	v_1	v_1	v_1	v_2	v_2	v_2	v_2
	v_b	v_4	v_5	v_7	v_8	v_9	v_4	v_5	v_7	v_8
$\alpha(v_a, v_b)$		1	1	0	0	0	1	1	0	0
$\gamma(v_a, v_b)$		1	1	2	2	2	1	1	2	2
$\gamma(v_b, v_a)$		2	2	4	2	2	2	2	4	2

E^c	v_a	v_2	v_4	v_4	v_5	v_5	v_7	v_8	v_9
	v_b	v_9	v_8	v_9	v_8	v_9	$v_{3,6}$	$v_{3,6}$	$v_{3,6}$
$\alpha(v_a, v_b)$		0	1	1	1	1	2	0	0
$\gamma(v_a, v_b)$		2	2	2	2	2	2	2	2
$\gamma(v_b, v_a)$		2	1	1	1	1	2	4	4

(b)

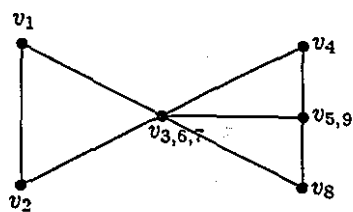
Figure 5.2: WWI on affinities example



E^c	v_a	v_1	v_1	v_1	v_1	v_2	v_2
	v_b	v_4	v_5	v_8	v_9	v_4	v_5
$\alpha(v_a, v_b)$		1	1	1	1	1	1
$\gamma(v_a, v_b)$		1	1	1	1	1	1
$\gamma(v_b, v_a)$		1	1	1	1	1	1

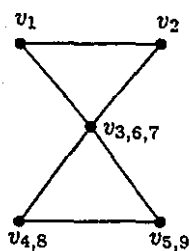
E^c	v_a	v_2	v_2	v_4	v_4	v_5	v_5
	v_b	v_8	v_9	v_8	v_9	v_8	v_9
$\alpha(v_a, v_b)$		1	1	1	1	1	1
$\gamma(v_a, v_b)$		1	1	1	1	1	1
$\gamma(v_b, v_a)$		1	1	1	1	1	1

(c)



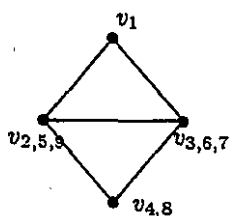
E^c	v_a	v_1	v_1	v_1	v_2	v_2	v_2	v_4
	v_b	v_4	v_8	$v_{5,9}$	v_4	v_8	$v_{5,9}$	v_8
$\alpha(v_a, v_b)$		1	1	1	1	1	1	2
$\gamma(v_a, v_b)$		1	1	1	1	1	1	0
$\gamma(v_b, v_a)$		1	1	2	1	1	2	0

(d)



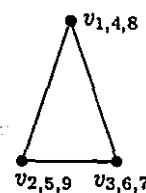
E^c	v_a	v_1	v_1	v_2	v_2
	v_b	$v_{4,8}$	$v_{5,9}$	$v_{4,8}$	$v_{5,9}$
$\alpha(v_a, v_b)$		1	1	1	1
$\gamma(v_a, v_b)$		1	1	1	1
$\gamma(v_b, v_a)$		1	1	1	1

(e)



E^c	v_a	v_1
	v_b	$v_{4,8}$
$\alpha(v_a, v_b)$		2
$\gamma(v_a, v_b)$		0
$\gamma(v_b, v_a)$		0

(f)



(g)

Figure 5.2: WWI on affinities example (continued)

color.

If $\alpha(v_p, v_q) > 0$ then it will be shown that v_p and v_q must have the same color in any optimal coloring and that the assignment in WWI on affinities is therefore correct. Suppose that v_p and v_q have different colors in an optimal coloring C for G . Since $\chi(G) \leq 2$ and v_p and v_q are of different colors then it must be that all other nodes in the optimal color assignment C are of the color of v_p or v_q so that the number of colors does not exceed 2. But since $\alpha(v_p, v_q) > 0$ there exists a node v_r which is adjacent to both v_p and v_q . For C to be a valid vertex coloring it must be that v_r is a third color from that of v_p and v_q . This contradicts that C is an optimal coloring. Therefore it must be that v_p and v_q are assigned the same color if $\alpha(v_p, v_q) > 0$ and this is precisely what WWI on affinities does.

If $\alpha(v_p, v_q) = 0$ then it can be shown that there exists an optimal coloring with v_p and v_q having the same color. Only the case of $\gamma(v_p, v_q) > 0$ or $\gamma(v_q, v_p) > 0$ must be considered; otherwise, by virtue of theorem 5.2, the algorithm would have compressed the two nodes to a same color by its first selection criterion. Since $\gamma(v_p, v_q) > 0$ there exists at least one node v_α which is adjacent to v_p but not v_q . Similarly there is at least one node v_β adjacent to v_q but not v_p . This situation is illustrated in the partial graph of figure 5.3. Now suppose that v_p has another neighbor v_γ different from v_α . Then either v_γ is also

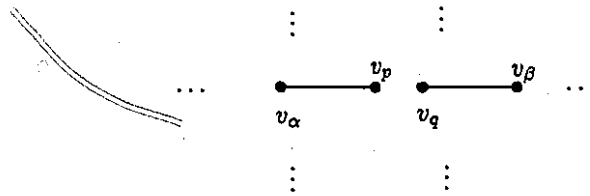


Figure 5.3: A partial graph with dependencies between $v_p, v_q, v_\alpha, v_\beta$

connected to v_α or it is not. The two situations are illustrated in figure 5.4 (a) and (b). The scenario in figure 5.4 (a) is impossible since $\alpha(v_\alpha, v_\gamma) \geq 1$ and this would contradict that the maximum affinity in the graph is 0. And the scenario in (b) cannot hold either since the subgraph shown requires at least three colors and this would contradict $\chi(G) \leq 2$. Hence v_p cannot have any other neighbors and $\gamma(v_p, v_q) = 1$. Similarly v_q cannot have any

other adjacencies than v_β . Also v_α cannot have other adjacencies because if another vertex

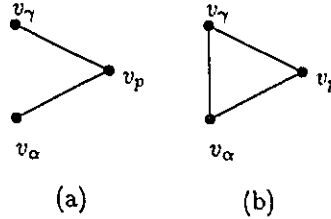


Figure 5.4: Two possibilities with v_γ

v_δ would be adjacent to v_α then v_p and v_δ would share v_α in their affinity and this would contradict that the maximal affinity in the graph would be 0. By a similar argument v_β cannot have any other adjacencies. Therefore vertices v_α, v_p and v_β, v_q form subgraphs of size 2 disconnected from the rest of G as shown in figure 5.3. And clearly there must exist an optimal coloring with v_p and v_q having the same color. Hence the theorem stands for all possible cases. \square

A trivial consequence of theorem 5.3 is that if WWI on affinities returns a coloring of three colors for any given graph then that is an optimal coloring for the graph. In addition WWI on affinities colors graphs consisting of disconnected cycles optimally (cycles which do not have nodes in common):

Lemma 5.3 *Cycles are optimally colored by WWI on affinities.*

Proof Even cycles require 2 colors and by invoking theorem 5.3 they will be optimally colored. Odd cycles require three colors and a proof by induction is used to show that they will be colored optimally.

Firstly note that an odd cycle of length $2(1) + 1$ (a triangle) is colored optimally by WWI on affinities. Secondly, assume that all odd cycles of length $2(k) + 1$ are colored optimally by the algorithm. We now proceed to show that odd cycles of length $2(k+1) + 1$ are colored optimally.

As a preliminary, define the operator \oplus so that $a \oplus b = (a + b) \bmod (2(k + 1))$. Label the nodes in the cycles so that the cycle is $(w_0, w_1, \dots, w_{2(k+1)}, w_0)$. For an odd cycle the maximum affinity between any pair of non-adjacent vertices is 1 and there are no situations of zero conflict. An affinity of 1 exists between the following pairs of vertices: w_i and $w_{i \oplus 2} \forall 0 \leq i \leq 2(k + 1)$. Hence, for some $0 \leq j \leq 2(k + 1)$, WWI on affinities will compress vertices w_j and $w_{j \oplus 2}$. Once w_j and $w_{j \oplus 2}$ are compressed, vertex $w_{j \oplus 1}$ will have only one adjacency: $w_{j \oplus 3}$. Furthermore $w_{j \oplus 1}$ will have zero conflict with $w_{j \oplus -1}$ and $w_{j \oplus 3}$. Because of the zero conflict, the algorithm's next step will be to compress $w_{j \oplus 1}$ with $w_{j \oplus -1}$ or $w_{j \oplus 1}$ with $w_{j \oplus 3}$. In both cases the resulting graph is an odd cycle of length $2(k) + 1$ which WWI on affinities will color optimally by the induction hypothesis. \square

Theorem 5.4 *Any graph consisting of the union of disconnected cycles is colored optimally by WWI on affinities.*

Proof The algorithm first reduces each cycle to a clique before compressing nodes of different cycles since there is 0 affinity between nodes on disconnected cycles. By lemma 5.3, each cycle is optimally reduced to a clique of size 2 or 3. The algorithm will then proceed to compress pairs of cliques until only one clique remains. It is easy to verify that the compression steps of two disconnected cliques into a single clique are optimal. Hence the resulting coloring will always be optimal. \square

5.4.1 A Heuristic

Although the WWI algorithms are quite efficient, they are only heuristics and they do not necessarily provide optimal graph colorings. To demonstrate this we use the interesting graph of figure 5.5 (a). As shown on the graph it is possible to color the graph with three colors and since cycles of length three are within the graph then it must be that $\chi = 3$. This graph has an interesting property that if a coloring C is an optimal coloring then $C(v_2) \neq C(v_5)$ and $C(v_6) = C(v_8)$ hold true.

If one performs WWI on affinities for this graph then there is the possibility that the optimal coloring will not be obtained. From the affinity and conflict table of figure 5.5 (b)

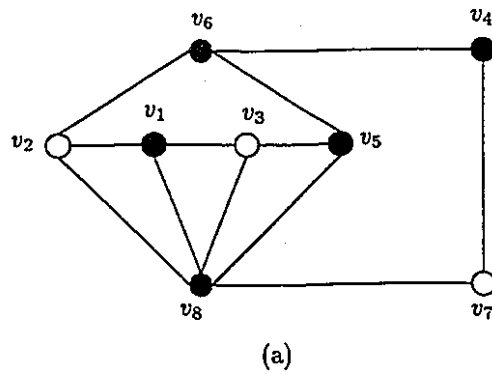
one can notice that there are no 0 conflicts in the graph. So the algorithm will proceed selecting a pair of vertices with the largest affinity, in this case 2. There are four pairs with affinity 2 but the pair of v_6 and v_8 is dropped since it has a conflict sum higher than the other three pairs. This leaves the algorithm with three equally rated pairs in a tie for compression. The equally rated pairs best satisfying the criteria of WWI on affinities are highlighted in the table. Unfortunately the pair of v_2 and v_5 is amongst the choices with $\alpha(v_2, v_5) = 2$, $\gamma(v_2, v_5) = 1$ and $\gamma(v_5, v_2) = 1$. If the pair of v_2 and v_5 is selected then the resulting graph is shown in figure 5.5 (c). From there the compression of v_6 and v_7 is ranked highest with $\alpha(v_6, v_7) = 1$, $\gamma(v_6, v_7) = 1$ and $\gamma(v_7, v_6) = 1$. This results in the graph of figure 5.5 (d) which has two equally good candidates for a compression: v_4 and v_8 for $\gamma(v_4, v_8) = 0$ or v_4 and $v_{2,5}$ for $\gamma(v_4, v_{2,5}) = 0$. The graph in figure 5.5 (e) shows the result of compressing v_4 and $v_{2,5}$. From that point both $\gamma(v_{6,7}, v_1) = 0$ and $\gamma(v_{6,7}, v_3) = 0$. Compressing $v_{6,7}$ and v_3 results in the complete graph of figure 5.5 (f). At this point WWI on affinities reports 4 colors and a coloring C with $C(v_1) = 3$, $C(v_2) = C(v_4) = C(v_5) = 2$, $C(v_3) = C(v_6) = C(v_7) = 1$ and $C(v_8) = 0$. Clearly the number of colors required by WWI on affinities is not optimal for this particular case.

5.5 Benchmarks

Two sets of extensive benchmarks are provided to demonstrate the efficiency of the algorithms. The first concerns the widely benchmarked *random graphs* while the other is on the hard to color *k-colorable random graphs*.

5.5.1 Random Graphs

To appraise the performance of the WWI algorithms we use random graphs to collect statistics [AlSE92, Boll88, BoEr76]. A random graph $G(n, p)$ is generated given a number of nodes n and a probability p of an edge between any pair of nodes. The tables in figure 5.6 and 5.7 are a compilation of averages on the number of colors, the number of zero conflict compressions, the fraction of zero conflict compressions over the number of compressions,



E^c	v_a	v_1	v_1	v_1	v_1	v_2	v_2	v_2	v_2
	v_b	v_4	v_5	v_6	v_7	v_3	v_4	v_5	v_7
$\alpha(v_a, v_b)$	0	2	1	1	2	1	2	1	1
$\gamma(v_a, v_b)$	3	1	2	2	1	2	1	2	2
$\gamma(v_b, v_a)$	2	1	2	1	1	1	1	1	1

E^c	v_a	v_3	v_3	v_3	v_4	v_4	v_5	v_6	v_6
	v_b	v_4	v_6	v_7	v_5	v_8	v_7	v_7	v_8
$\alpha(v_a, v_b)$	0	1	1	1	1	1	1	1	2
$\gamma(v_a, v_b)$	3	2	2	1	1	2	2	2	1
$\gamma(v_b, v_a)$	2	2	1	2	4	1	1	1	3

(b)

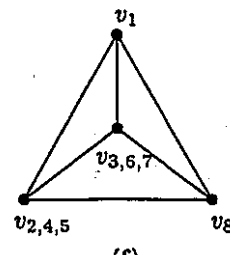
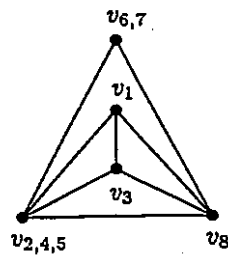
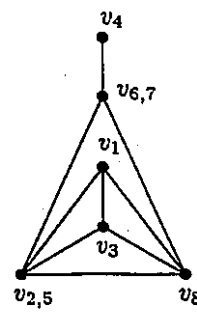
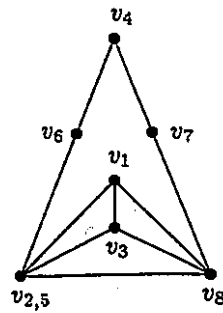


Figure 5.5: Diamond Graph

the number of conflict free compressions detected, the fraction of conflict free compressions over the total number of compressions, and the average execution times used by WWI on affinities and WWI on conflicts. The exact meaning of these specialized compressions will be introduced in section 5.6 but for the present purpose it suffices to state that they are known to preserve the chromatic number or the maximal clique number.

In addition the number of colors used by the widely known greedy [Bigg85, Gibb85] algorithm is also supplied in figure 5.6. The greedy algorithm is used as a comparison since it is often benchmarked against and thus it is a useful yardstick. In figure 5.7 the number of colors for Tseng's algorithm is given (see chapter 2). The results were averaged over 100 graphs for each value of (n, p) and were computed on Sun SparcStations 10 model 41 with 32MB RAM and 1MB cache running SunOS 4.1.3. The WWI algorithms are fastest on complete graphs and are slowest when there are no edges. The execution times listed include the operating system's maintenance time and can be improved if performed on machines with lesser loads than the ones available to us. The execution times also include several consistency checks and each coloring undergoes a post-processing verification. From the tables one can extrapolate that the execution time is about $C \cdot (1 - p) \cdot |V|^3$ where C is a constant dependent on the computer architecture. For our particular architecture $C \approx 1\mu s$ for WWI on affinities, and for a Gateway 2000 4DX2-66V personal computer with 12MB RAM, 256KB cache running the Linux 1.0 operating system $C \approx 1.4\mu s$. If p is interpreted as the edge density then it can be replaced by $p \approx \frac{2|E|}{|V| \cdot (|V| - 1)}$ and the formula becomes a useful predictor of the execution time. As implemented, the WWI algorithms operate in a time proportional to $O(|V||E^c|)$.

For graphs of a small number of nodes there is a good number of perfect graphs which are reported to be optimally colored by the WWI algorithms since conflict free compressions are used throughout the coloring process. However, as the number of nodes increases, these graphs become less frequent and quickly get skewed out in the extremal edge density regions (such as shown in figure 5.6 and 5.7 for $n = 25$ at $p = 0.05$ and $p = 0.95$).

Furthermore, from the benchmarks for the low values of p ($p = 0.05, 0.25$) it becomes apparent that there can be a large disparity between the number of conflict free compressions

and zero conflict compressions. Conflict free compressions involve a costly second order check which is a superset to the conditions of theorem 5.2. Instead of aiming the search criteria at conflict free compressions directly, the WWI algorithms only verifies for conflict free compressions once compression pairs have already been selected on the basis of other criteria. This is in order to keep the time complexity at $O(|V|^3)$. Therefore, if one is willing to tradeoff on the overall time complexity of the algorithms, there can be a definite advantage in a direct search for conflict free compressions instead of performing the *a posteriori* detection currently implemented in the WWI algorithms; especially in the case of low edge densities.

It is clear throughout the benchmarks that WWI on conflicts provides a superior performance to Tseng's algorithm and that WWI on affinities does provide better results at lower edge densities than WWI on conflicts. However, random graphs do not make a particularly rigid test despite their popular use in benchmarks. The random graph benchmarks do provide a distinction between WWI on affinities, WWI on conflicts and Tseng's algorithm but the differences are sometimes minor, especially in the very high edge density cases of $p = 0.95$. The next subsection will provide a more stringent set of benchmarks which will result in substantial differences between the algorithms.

5.5.2 k -Colorable Random Graphs

It has been shown that random graphs [AlSE92, Wilf84, Boll88, BoEr76] are relatively easy to color and that even the greedy algorithm will perform relatively well on them. As a more stringent test on the algorithm, we use k -colorable random graphs as proposed by Turner [Turn88] to benchmark WWI. A k -colorable random graph $G(n, k, p)$ consists of a random experiment which first generates a k -coloring on n nodes and then produces edges with a probability of p between nodes of different colors.

Figures 5.8 (a),(b),(c),(d) plot the performance of five algorithms for $p = 0.5$. Fifty k -colorable graphs were generated for each value of k plotted and the average ratio of the number of colors returned by the algorithms over k is given. Clearly the WWI algorithms outperform the greedy algorithm and the Brélaz algorithm by a wide margin (for the latter

$p = 0.05$	WWI on Affinities	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Greedy # Colors
n	# Colors						
25	2.35	19.56	86.3	22.5	99.3	0.015	2.87
100	4.1	42.35	44.1	68.62	71.6	1.124	5.29
250	6.35	76.81	31.5	117.24	48.1	14.37	8.43
500	9.57	119.45	24.3	174.34	35.5	120.7	12.65
1000	14.96	192.14	19.5	256.5	26.0	911.2	19.62

$p = 0.25$	WWI on Affinities	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Greedy # Colors
n	# Colors						
25	4.33	10.85	52.5	14.8	71.6	0.017	5.31
100	9.76	26.19	29.0	33.26	36.8	0.828	12.46
250	18.08	48.02	20.7	56.88	24.5	10.19	23.05
500	30.44	79.15	16.9	88.84	18.9	87.19	38.39
1000	52.74	130.77	13.8	142.42	15.0	748.2	64.94

$p = 0.50$	WWI on Affinities	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Greedy # Colors
n	# Colors						
25	6.70	9.00	49.1	10.85	59.2	0.008	8.09
100	17.05	21.23	25.6	24.10	29.0	0.472	21.37
250	34.10	40.84	18.9	44.22	20.5	7.03	42.19
500	59.73	68.59	15.3	72.64	16.1	61.20	72.64
1000	106.83	117.60	13.2	121.83	13.7	482.3	126.68

Figure 5.6: Benchmark results for random graphs and WWI on affinities

$p = 0.75$	WWI on Affinities	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Greedy # Colors
n	# Colors						
25	10.21	8.91	60.2	9.68	65.4	0.004	11.78
100	27.38	18.78	25.8	20.46	28.1	0.210	33.37
250	57.17	37.28	19.3	38.95	20.1	3.17	68.47
500	102.16	65.83	16.5	67.62	17.0	26.09	121.5
1000	186.15	112.04	13.7	113.8	14.0	239.4	216.63

$p = 0.95$	WWI on Affinities	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Greedy # Colors
n	# Colors						
25	17.19	7.77	99.5	7.78	99.6	0.001	17.81
100	47.65	21.38	40.8	22.16	42.3	0.047	54.9
250	100.85	42.83	28.7	43.77	29.3	0.651	119.36
500	180.48	78.94	24.7	79.68	24.9	5.26	216.33
1000	331.68	97.3	14.5	98.29	14.7	39.34	389.56

Figure 5.6: Benchmark results for random graphs and WWI on affinities (*continued*)

$p = 0.05$	WWI on Conflicts # Colors	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Tseng # Colors
n							
25	2.35	19.94	88.0	22.47	99.1	0.016	2.56
100	4.3	41.17	43.0	59.99	62.7	1.08	4.77
250	7.14	40.66	16.7	62.92	25.9	15.45	7.38
500	10.66	43.35	8.85	64.21	13.1	125.1	11.02
1000	16.13	47.35	4.81	67.31	6.84	980.2	16.76

$p = 0.25$	WWI on Conflicts # Colors	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Tseng # Colors
n							
25	4.56	10.91	53.2	13.44	65.6	0.012	4.80
100	10.49	13.92	15.5	16.85	18.8	0.722	10.92
250	19.37	17.69	7.67	20.83	9.03	10.9	19.87
500	31.55	25.04	5.34	27.82	5.94	91.6	32.32
1000	52.99	35.90	3.79	38.94	4.11	744.1	53.92

$p = 0.50$	WWI on Conflicts # Colors	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Tseng # Colors
n							
25	6.79	8.42	46.2	9.63	52.8	0.008	7.26
100	17.3	12.44	15.0	13.84	16.7	0.463	18.37
250	33.96	20.13	9.3	21.64	10.0	7.31	35.77
500	57.91	31.21	7.05	32.55	7.36	61.8	60.63
1000	101.15	50.0	5.56	51.37	5.71	504.2	104.72

Figure 5.7: Benchmark results for random graphs and WWI on conflicts

$p = 0.75$	WWI on Conflicts # Colors	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Tseng # Colors
n							
25	10.08	9.5	63.63	10.15	68.01	0.005	10.46
100	27.17	13.65	18.7	14.96	20.5	0.228	28.79
250	54.95	25.71	13.17	26.84	13.75	4.93	57.90
500	96.47	43.11	10.68	44.26	10.97	28.34	101.08
1000	172.09	69.43	8.39	70.47	8.51	267.11	179.27

$p = 0.95$	WWI on Conflicts # Colors	Zero Conflict Comp.	Zero Conflict %	Conflict Free Comp.	Conflict Free %	Exec. Time (secs)	Tseng # Colors
n							
25	17.13	7.81	99.3	7.85	99.7	0.002	17.19
100	47.28	23.19	43.9	24.02	45.5	0.052	48.09
250	99.25	39.04	25.9	39.90	26.5	0.766	99.62
500	175.38	68.33	21.0	69.30	21.3	5.56	178.42
1000	321.89	92.19	13.6	93.25	13.7	47.35	321.80

Figure 5.7: Benchmark results for random graphs and WWI on conflicts (*continued*)

we have taken the statistics from [Turn88]). WWI on conflicts clearly outperforms all other algorithms by a large margin. WWI on affinities, designed for lower edge densities, becomes outperformed by Tseng's algorithm as n gets large.

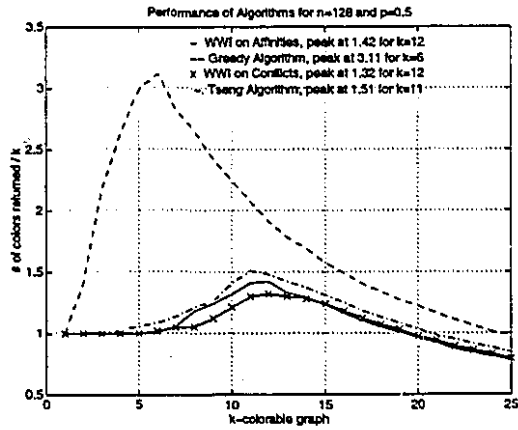
Figures 5.9 and 5.10 show the effect of varying p for $n = 128$ and $n = 256$. Reducing p shifts the worst case performance peak to smaller k while thinning it out. Eventually at $p = 0$ the peaks will vanish and the curves will behave as $\lfloor \frac{1}{k} \rfloor$. Increasing p shifts the worst case performance peak to higher values of k while flattening it out. Ultimately, at $p = 1$ the performance ratio will never exceed 1 and all graphs will be colored optimally.

The benchmarks of figures 5.8 and 5.9 and 5.10 show that WWI on conflicts is clearly the better algorithm with exception to the lower edge densities when WWI on affinities performs better. At lower edge densities, $p = 0.15$ and $p = 0.3$, WWI on conflicts does not provide a significant improvement over Tseng's algorithm; however, the performance difference becomes clearly apparent when p increases.

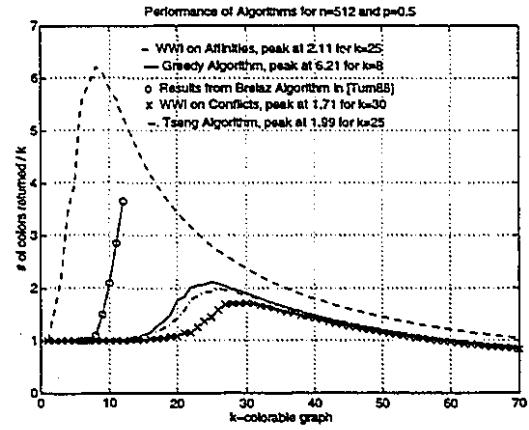
5.5.3 The first order approximation to the second order

The graph of figure 5.11 provides a sample of the 2nd order algorithm minimizing the new triangles introduced at each compression step and the extent of the approximation achieved with WWI on conflicts. Although it does not provide results which are identical, the curve for WWI on conflicts tightly follows that of the second order algorithm. With lower edge probability the bound of proposition 5.4 becomes less tight and consequentially the approximation becomes a bit less accurate. This is another indicator as to why WWI on conflicts does better at higher edge densities. As a possible improvement a better first order approximation would involve a mechanism to keep track of the probabilistic distribution of edges between conflict nodes.

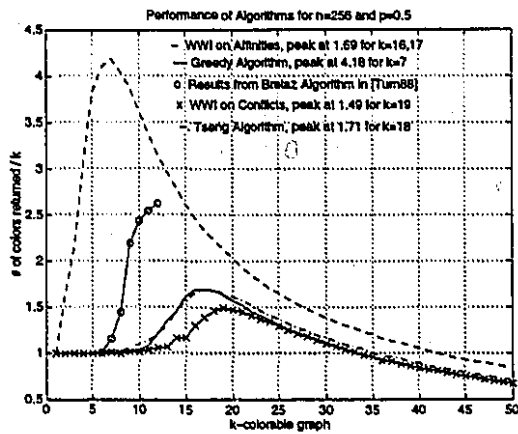
The implementation of the second order algorithm is identical to that of WWI on affinities found in appendix A except that the sub-algorithm of section A.2 was modified to count the number of edges between the conflict nodes. Furthermore, if two pairs of nodes present the same number of edges between their conflict nodes then the pair with the highest affinity is selected.



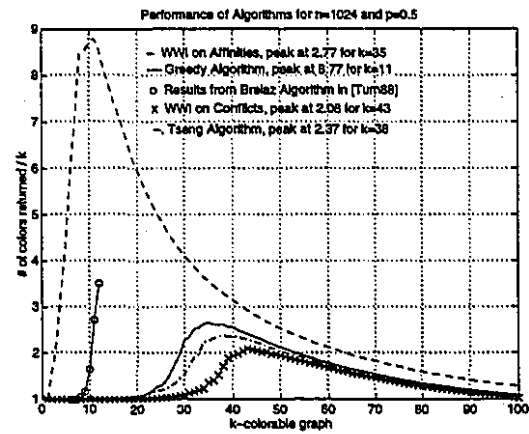
(a)



(c)

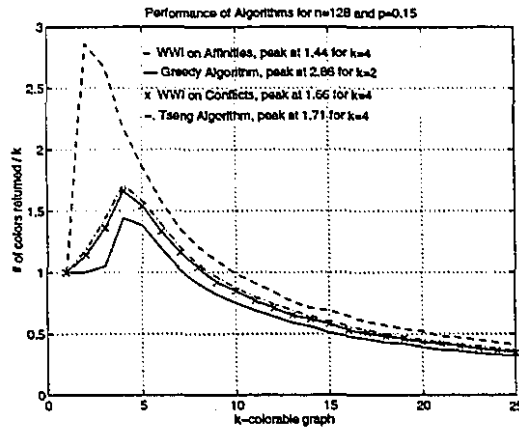


(b)

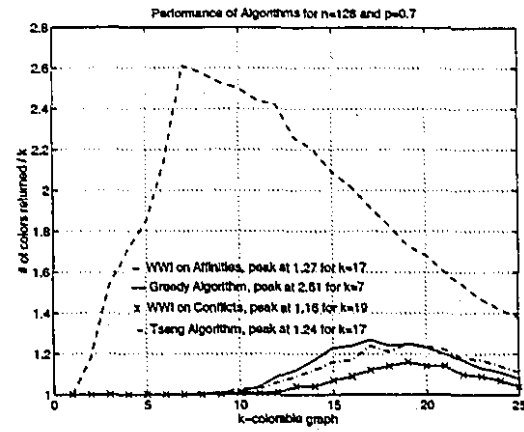


(d)

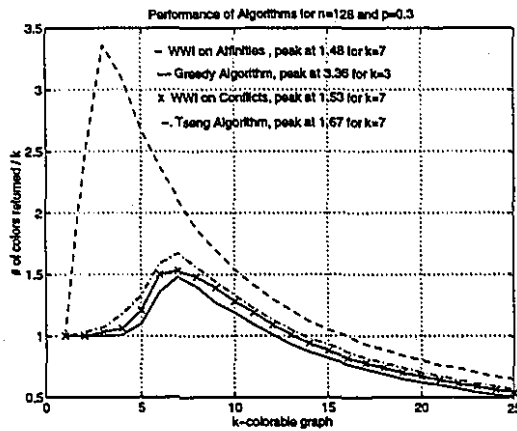
Figure 5.8: Benchmark results for $n = 128, 256, 512, 1024$ and $p = 0.5$.



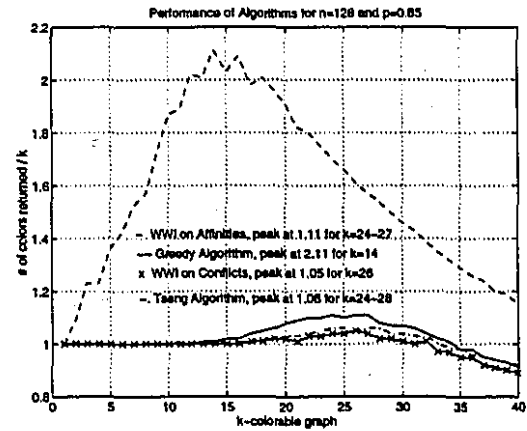
(a)



(c)

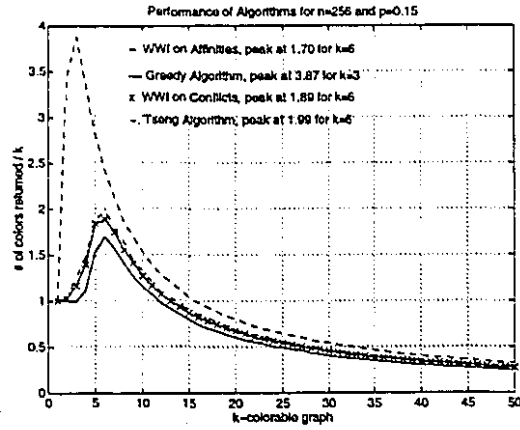


(b)

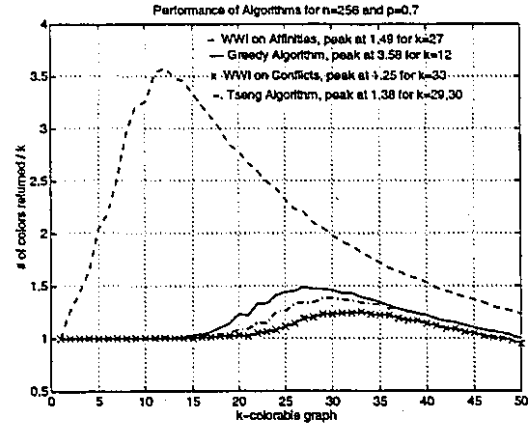


(d)

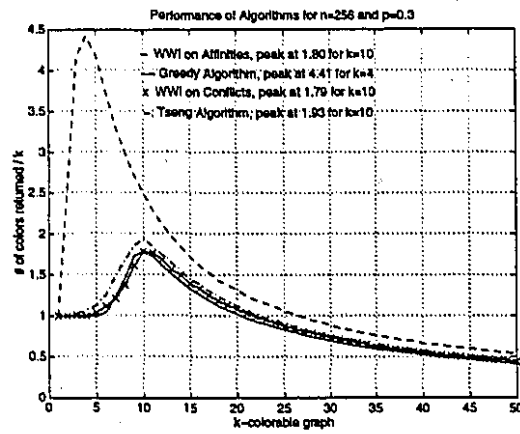
Figure 5.9: Benchmark results for $n = 128$ and $p = 0.15, 0.3, 0.70, 0.85$



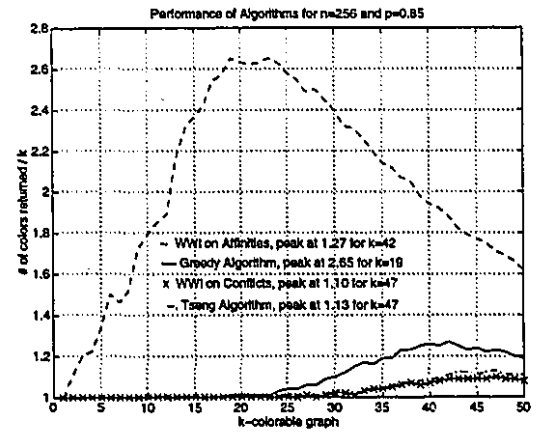
(a)



(c)



(b)



(d)

Figure 5.10: Benchmark results for $n = 256$ and $p = 0.15, 0.3, 0.70, 0.85$

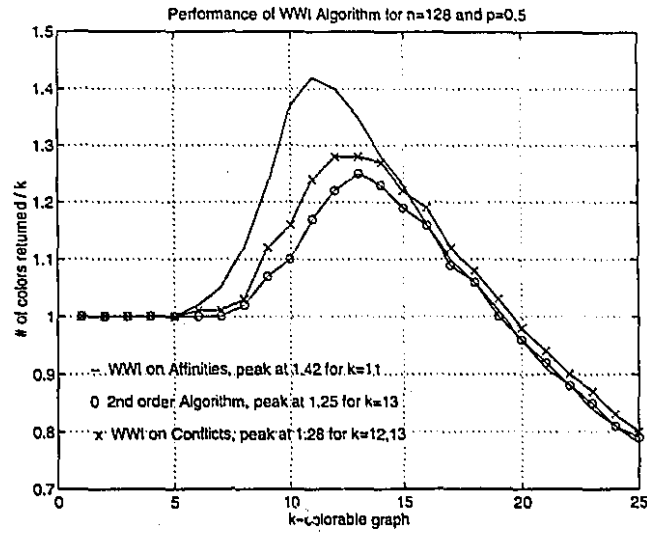


Figure 5.11: 2nd order approximation on 128 nodes and $p = 0.5$

5.6 Perfect class

This section devotes itself to defining a class of perfect graphs which lends itself well to compression style coloring algorithms. The following lemmas and definitions are key in characterizing the class.

Lemma 5.4 Suppose a graph $G = (V, E)$ which is not complete. A compression of two non-adjacent vertices in G to a graph $G' = (V', E')$ has the following clique number relationship:

$$\omega(G) \leq \omega(G') \leq \omega(G) + 1$$

If v_i and v_j are the two compressed vertices and $\omega(G') = \omega(G) + 1$ then $v_{i,j}$ is in all maximal cliques of G' .

Proof Let $v_i, v_j \in V$ be the two compressed vertices in G and let $v_{i,j}$ be the compressed vertex in G' ($\succ (G : v_i, v_j; G' : v_{i,j})$). First it is shown that $\omega(G') \geq \omega(G)$ by dividing the proof into three distinct cases:

1. If there is a clique W_1 of maximum size in G with $v_i \notin W_1$ and $v_j \notin W_1$ then W_1 is also a clique in G' since all vertices in W_1 are preserved in G' as well as the edges by the definition of a compression. So for this case we are ensured that there will be a clique of size $\omega(G)$ in G' .
2. If there is a clique W_2 of maximum size in G such that $v_i \in W_2$ then there will also be a clique of size $\omega(G)$ in G' . Clearly $v_j \notin W_2$ since a compression of v_i and v_j implies that $\{v_i, v_j\} \notin E$ and this would contradict that W_2 is a clique. We now argue that the set of vertices $W'_2 = (W_2 \setminus \{v_i\}) \cup \{v_{i,j}\}$ is a clique in G' . This is so because all vertices in $W'_2 \setminus \{v_{i,j}\}$ remain fully interconnected in G' and vertex $v_{i,j}$ has all the adjacencies of v_i by the definition of compression. Since $|W'_2| = |W_2| = \omega(G)$ it follows that G' has a clique of size $\omega(G)$ for this particular case.
3. If there is a maximal size clique including vertex v_j in G then the argument given in case 2 with the roles of v_i and v_j reversed ensures that graph G' has a clique of size $\omega(G)$.

The three cases cover all possibilities for the location of a maximal clique in G and therefore one is always ensured that there will be clique of size $\omega(G)$ in G' . Hence

$$\omega(G) \leq \omega(G')$$

Now suppose there is a clique W' of size $|W'| > \omega(G)$ in G' . It must be that $v_{i,j} \in W'$ otherwise W' would also be a clique in G since all the nodes and the edges connecting vertices different from $v_{i,j}$ in G' are inherited from G (by the definition of compression). And if W' is a clique in G then this would contradict that $|W'| > \omega(G)$. However note that $W = W' \setminus \{v_{i,j}\}$ is a clique in G since the vertices in W are fully interconnected in G' and this implies that they must be fully interconnected in G by the definition of a compression. Therefore there is always a clique W in G with $|W| + 1 = |W'|$ if $|W'| > \omega(G)$. Hence

$$\omega(G') \leq \omega(G) + 1$$

This completes the proof. □

Definition 5.1 (Zero Conflict Compression) Let $G = (V, E)$ be a graph which is not complete. A compression on two non-adjacent vertices v_i and v_j is a **zero conflict compression** if and only if

$$\gamma(v_i, v_j) = 0 \vee \gamma(v_j, v_i) = 0$$

Lemma 5.5 Suppose a graph $G = (V, E)$ is not complete. If a zero conflict compression exists on two non-adjacent vertices in G then a compression of the two vertices yields a graph $G' = (V', E')$ with the following clique number relation with G :

$$\omega(G') = \omega(G)$$

Proof Since a zero conflict compression exists there are two vertices v_i and v_j in G such that $\gamma(v_i, v_j) = 0$ or $\gamma(v_j, v_i) = 0$. Let $v_{i,j}$ be the compressed vertex in graph $G' = (V', E')$ obtained from a compression v_i and v_j ($\succ (G : v_i, v_j; G' : v_{i,j})$). Suppose there is a clique W' of size $|W'| > \omega(G)$ in G' . By lemma 5.4, it must be that $v_{i,j} \in W'$.

Now notice that if $\gamma(v_i, v_j) = 0$ then the set of nodes adjacent to $v_{i,j}$ in G' is identical to the set of nodes adjacent to v_j in G . Therefore it can only be that $W = (W' \setminus \{v_{i,j}\}) \cup \{v_j\}$ is a clique in G since all nodes which are different from $v_{i,j}$ in W' exist in G and they are fully interconnected by the definition of compression. Similarly if $\gamma(v_j, v_i) = 0$ then $W = (W' \setminus \{v_{i,j}\}) \cup \{v_i\}$ is a clique in G .

Therefore if $\gamma(v_i, v_j) = 0$ or $\gamma(v_j, v_i) = 0$ then there is a clique W in graph G such that $|W| = |W'|$. This contradicts that a clique W' with $|W'| > \omega(G)$ can exist in G' . Hence

$$\omega(G') \leq \omega(G)$$

Lemma 5.4 demonstrates that $\omega(G') \geq \omega(G)$ so it must be that

$$\omega(G') = \omega(G)$$

□

Definition 5.2 (Zero Conflict Graph) Let $G^0 = (V, E)$ be a graph. If there exists a sequence of compressions from G^0 :

$$\begin{aligned} &> (G^0 : v_{i_0}, v_{j_0}; G^1 : v_{i_0, j_0}) \\ &> (G^1 : v_{i_1}, v_{j_1}; G^2 : v_{i_1, j_1}) \\ &\quad \vdots \\ &> (G^{k-2} : v_{i_{k-2}}, v_{j_{k-2}}; G^{k-1} : v_{i_{k-2}, j_{k-2}}) \\ &> (G^{k-1} : v_{i_{k-1}}, v_{j_{k-1}}; G^k : v_{i_{k-1}, j_{k-1}}) \end{aligned}$$

which leads to a complete graph G^k and for which each compression is a zero conflict compression:

$$\bigwedge_{m=0}^{k-1} (\gamma(v_{i_m}, v_{j_m}) = 0 \vee \gamma(v_{j_m}, v_{i_m}) = 0)$$

then G^0 is a zero conflict graph.

For example the graph of figure 2.4 (a) is a zero conflict graph.

Definition 5.3 (Zero Conflict Class) \mathcal{Z} is the class of all zero conflict graphs:

$$\mathcal{Z} = \{ G \mid G \text{ is a zero conflict graph} \}$$

Theorem 5.5 All graphs in the zero conflict class \mathcal{Z} are perfect.

Proof Let $G \in \mathcal{Z}$. Consequently there must exist a sequence of zero conflict compressions from G such that a complete graph G'' is ultimately reached. By repeated use of lemma 5.5, one for each compression of the sequence, it must be that

$$\omega(G) = \omega(G'') \quad (5.2)$$

For any graph it must be that the chromatic number is larger than the clique number i.e. for graph G this implies $\chi(G) \geq \omega(G)$. By equation 5.2 we conclude that $\chi(G) \geq \omega(G'')$. But since G'' is a complete graph it follows that $\chi(G'') = \omega(G'')$. So it follows that

$$\chi(G) \geq \chi(G'') \quad (5.3)$$

But lemma 5.2 implies that a compressed graph can only have a chromatic number greater or equal to the chromatic number of the graph prior to compression. Repeated use of lemma 5.2, one for each compression in the sequence, implies that

$$\chi(G) \leq \chi(G'') \quad (5.4)$$

From equations 5.3 and 5.4 it must be that $\chi(G) = \chi(G'')$. But as it has already been argued $\chi(G'') = \omega(G'')$ and $\omega(G'') = \omega(G)$. Hence it follows that

$$\chi(G) = \omega(G)$$

and G is therefore a perfect graph. □

The WWI algorithms keep track of all zero conflict compressions and declare a graph of class \mathcal{Z} if all compressions performed were of zero conflict type. If a graph was not fully compressed with zero conflict compressions but a high percentage of the compressions were of zero conflict type then the resulting coloring can only be off the optimal goal by a few colors in the worst case.

The proof of theorem 5.5 purposely avoids the result of corollary 5.1 so that the proof becomes generic to any type of compression which preserves the clique number of the graph. This generic property is now used to prove a class of perfect graphs which is more general than \mathcal{Z} .

Definition 5.4 (Conflict Free Compression) *Let $G = (V, E)$ be a graph which is not complete. Consider the compression of two non-adjacent vertices v_i and v_j . Let*

$$\Gamma_{i,j} = \{\gamma \mid \{v_i, \gamma\} \in E \wedge \{v_j, \gamma\} \notin E\}$$

$$\Gamma_{j,i} = \{\gamma \mid \{v_j, \gamma\} \in E \wedge \{v_i, \gamma\} \notin E\}$$

The compression of v_i and v_j is a conflict free compression if and only if

$$\{\gamma_i, \gamma_j\} \notin E \quad \forall \gamma_i \in \Gamma_{i,j}, \gamma_j \in \Gamma_{j,i}$$

Lemma 5.6 Suppose a graph $G = (V, E)$ which is not complete and two non-adjacent vertices v_i and v_j such that the compression $\succ (G : v_i, v_j; G' : v_{i,j})$ to a graph G' is a conflict free compression. Then

$$\omega(G') = \omega(G)$$

Proof Define $\Gamma_{i,j}$ and $\Gamma_{j,i}$ as in definition 5.4. Also define the set of all vertices adjacent to both v_i and v_j :

$$A_{i,j} = \{\alpha \mid \{v_i, \alpha\} \in E \wedge \{v_j, \alpha\} \in E\}$$

Since the compression of v_i and v_j is a conflict free compression there cannot be an edge between a vertex of $\Gamma_{i,j}$ and one in $\Gamma_{j,i}$. Also notice that the adjacencies of v_i in G are represented by the set $\Gamma_{i,j} \cup A_{i,j}$ and the adjacencies of v_j are $\Gamma_{j,i} \cup A_{i,j}$.

Now suppose there is a clique W' in G' with a size $|W'| > \omega(G)$. By lemma 5.4, it must be that $v_{i,j} \in W'$. Since $v_{i,j} \in W'$ it must be that $W' \subseteq \Gamma_{i,j} \cup \Gamma_{j,i} \cup A_{i,j} \cup \{v_{i,j}\}$. Otherwise a node outside the superset would not be adjacent to $v_{i,j}$ (by definition of compression) and this would contradict that W' is a clique. Furthermore since no edges exist between elements of $\Gamma_{i,j}$ and $\Gamma_{j,i}$ it follows that W' cannot have vertices in both sets. Otherwise would contradict that W' is a clique. Therefore it must be that $W' \subseteq \Gamma_{i,j} \cup A_{i,j} \cup \{v_{i,j}\}$ or $W' \subseteq \Gamma_{j,i} \cup A_{i,j} \cup \{v_{i,j}\}$.

If $W' \subseteq \Gamma_{i,j} \cup A_{i,j} \cup \{v_{i,j}\}$ then it must be that $W_1 = (W' \setminus \{v_{i,j}\}) \cup \{v_i\}$ is a clique in G because as previously noted v_i is adjacent to all nodes of $\Gamma_{i,j} \cup A_{i,j}$ in G and all edges existing between vertices of $\Gamma_{i,j} \cup A_{i,j}$ in G' are also in present in G (this follows from the definition of compression). Clearly $|W_1| = |W'|$. Similarly if $W' \subseteq \Gamma_{j,i} \cup A_{i,j} \cup \{v_{i,j}\}$ then $W_2 = (W' \setminus \{v_{i,j}\}) \cup \{v_j\}$ is a clique in G . And once again $|W_2| = |W'|$.

Therefore in all cases it has been shown that a clique of size $|W'|$ exists in G . This contradicts that $|W'| > \omega(G)$. This implies that

$$\omega(G') \leq \omega(G)$$

Lemma 5.4 demonstrates that $\omega(G') \geq \omega(G)$ so it must be that

$$\omega(G') = \omega(G)$$

□

Definition 5.5 (Conflict Free Graph) A graph G^0 is a *conflict free graph* if there exists a sequence of compressions from G^0 :

$$\begin{aligned}
 & \succ (G^0 : v_{i_0}, v_{j_0}; G^1 : v_{i_0, j_0}) \\
 & \succ (G^1 : v_{i_1}, v_{j_1}; G^2 : v_{i_1, j_1}) \\
 & \quad \vdots \\
 & \succ (G^{k-2} : v_{i_{k-2}}, v_{j_{k-2}}; G^{k-1} : v_{i_{k-2}, j_{k-2}}) \\
 & \succ (G^{k-1} : v_{i_{k-1}}, v_{j_{k-1}}; G^k : v_{i_{k-1}, j_{k-1}})
 \end{aligned}$$

which leads to a complete graph G^k and for which each compression in the sequence is a conflict free compression.

Definition 5.6 (Conflict Free Class) \mathcal{F} is the class of all conflict free graphs:

$$\mathcal{F} = \{G \mid G \text{ is a conflict free graph} \}$$

Theorem 5.6 All graphs in the conflict free class \mathcal{F} are perfect.

Proof Let $G \in \mathcal{F}$ and emulate the proof of theorem 5.5 but use lemma 5.6 instead of lemma 5.5 to argue that $\omega(G) = \omega(G'')$. □

Since all zero conflict compressions also satisfy the definition of a conflict free compression, it follows that $\mathcal{Z} \subseteq \mathcal{F}$. An example which illustrates that $\mathcal{Z} \neq \mathcal{F}$ is the graph of figure 5.5 (a). Clearly from the table in figure 5.5 (b) this graph does not belong to class \mathcal{Z} . However it does belong to class \mathcal{F} . The compression of v_3 and v_7 is conflict free since no edges can be found between the vertices of $\Gamma_{3,7} = \{v_1, v_5\}$ and $\Gamma_{7,3} = \{v_4\}$. Figure 5.12 shows a remaining sequence of conflict free compressions which lead to a 3-clique.

In comparison to zero conflict compressions, conflict free compressions take longer to identify since edges incident to the conflict vertices of the compression candidates must also be examined. Because of the additional time complexity they introduce, conflict free

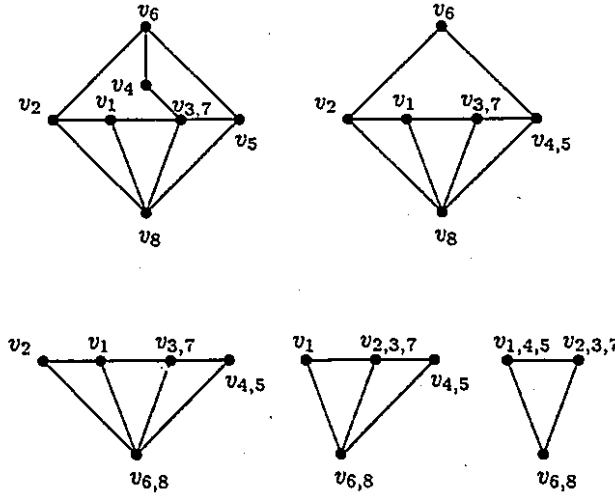


Figure 5.12: A sequence of conflict free compressions

compressions are not directly sought after by the WWI algorithms. Instead, the current implementations of WWI only identify conflict free compressions once they have occurred. At the cost of missing several conflict free compressions, this has the benefit that no additional time complexity is introduced. If a graph is colored exclusively with conflict free (and zero conflict) compressions then the WWI algorithms will declare the graph to be optimally colored.

Let G be a graph to be colored and G'' be the clique resulting from successive compressions using a WWI algorithm. An easily verified formula is $\chi(G'') - \chi(G) \leq C - F$ where C is the number of compressions performed by WWI, F is the number of conflict free compressions, and $\chi(G'')$ is the number of colors returned by the WWI heuristic. For a graph chiefly compressed with conflict free compressions this gives a tight lower bound on the chromatic number. However, in general, this formula has only been found to be useful on graphs of very small size.

5.7 Improving the WWI Algorithms

As of yet, the development of the WWI algorithms has led to a first order compression which preserves the chromatic number and the clique number of a graph (zero conflict compressions). And we have only found a second order compression (conflict free compression) which preserves the clique number of a graph. Finding the existence of other first order and second order compressions (or even higher order compressions) which are chromatic number and clique number preserving would be an asset to any graph coloring algorithm since they permit problem reduction without loss (or the identification of perfect graphs). Identifying optimal or profitable compressions at each step is the guiding design principle behind the WWI algorithms as opposed to algorithms designed with an overall performance bound in mind.

Finally, there are necessary characteristics of optimality which an optimal coloring must obey. It is possible to run some post-processing checks on these characteristics and then improve the coloring if they are not met. A few checks which can be run in $O(|V|^2)$ time complexity are presented in appendices B and C. The WWI heuristics do not always meet these characteristics and applying the checks does enhance a small percentage of the colorings returned.

5.8 Conclusion

This chapter has presented fast and efficient algorithms for the vertex coloring of graphs. The approach used was to take advantage of bounds on the chromatic number (bounds on the number of edges, the degree of nodes, the number of odd cycles) and compression steps which are guaranteed to preserve the chromatic number or the maximal clique number of a graph. Some theoretical outcomes of the algorithm, such as a class of perfect graphs, were discussed. It must also be stressed that a good part of the algorithm's efficiency is due to its adaptive nature which enables it to take into account the important structural changes which occur as the assignment of colors (compressions) progresses. Finally, a set of benchmarks was presented to illustrate the practicality and efficiency of the algorithms

on classes random graphs.

Chapter 6

Conclusion

In retrospect, this dissertation examined the vertex coloring problem via necessary characteristics of optimality, bounds on the chromatic number, a structural study of the solution space and practical coloring heuristics. Chapter 1 introduced the problem from the perspective of optimization issues confronting digital systems design and practical occurrences of vertex coloring in synthesis. Chapter 2 formally defined the problem and provided known results in complexity, algorithms and characteristics of vertex coloring. The compression mechanism used by the WWI algorithms was first detailed in chapter 2 and a general classification of compressions based on adaptability, order and localization was presented.

Chapter 3 established characteristics of optimality on a property of fundamental nodes which all optimal colorings must satisfy. This led to a strict set of bicolored paths which must lie between the fundamental nodes of all optimal colorings. In turn, a cubic bound relating the chromatic number and the number of odd cycles in a graph was obtained. This bound eventually played an important role in the design of the WWI on conflicts algorithm of chapter 5. Derivations of the characteristics were achieved through constructive methods which had a direct algorithmic applicability as coloring refinements to heuristics. Appendix B exemplifies such a refinement.

Chapter 4 gave a transformation of the vertex coloring problem into a continuous variable mathematical formulation. The resulting solution space did prove to have nice prop-

erties as it consisted of the union of disjoint convex regions. However, subsequent results were indicators of the complexity of vertex coloring. It was shown that there can be a great disparity between the local minima of the solution space since the ratio between the number of colors at a local minimum and the chromatic number can be directly proportional to the graph instance sizes. Furthermore, there can be an exponential number of local minima. An examination of active constraints on the mathematical programs led to the formulation of color stratifications and a characteristic of optimality closely related to the Gallai-Roy theorem. The characteristic dictates a set of colored paths which must be present within an optimal coloring. The constructive proof provided for that particular characteristic brought the algorithmic check of appendix C. Additional observations resulted into a complete characterization of the local minima of the real valued problem.

Lastly, chapter 5 yielded two vertex coloring heuristics based upon the first order properties of conflicts and affinities. The algorithms were founded on theoretical bounds of the chromatic number, maximal clique number and chromatic number preserving compressions, and approximation arguments over a second order criterion. Appendix A supplied an efficient implementation of the algorithms due to a rapid method to re-evaluate conflicts and affinities after each compression. Then followed a set of benchmarks on random graphs and k -colorable random graphs which justified some claims of efficiency and generated favorable comparisons with existing algorithms. WWI on affinities was shown to perform well on graphs of low edge density whereas WWI on conflicts was superior for the remaining graphs. Finally, a class of perfect graphs was identified.

6.1 Possible directions

From this work there remains several avenues to be explored. From the algorithmic perspective, the WWI algorithms could be parallelized because of the vectorial (matrix) and independent nature of some of their operations. With the use of parallelism an even more efficient version of WWI could be achieved. As well, a computationally efficient implementation of the second order algorithm of chapter 5 could be obtained with the aid of parallelism.

An other direction to pursue is the search for new chromatic number and maximal clique number preserving compression. These would result in an immediate retribution as they provide a means of optimal problem reduction. Given the generic proof mechanism of section 5.6, the discovery of any such compression also expands the space of algorithmically recognizable perfect graphs. Furthermore, the characterization of such compressions also provides directions in which to develop new heuristics. The availability of a larger set of such compressions can only result in a better overall performance.

The search for higher order bounds, such as the quadratic bound of theorem 5.1 and the cubic bound of corollary 3.4, results in the identification of significant factors on which the chromatic number of a graph is dependent and supplies guidance in the selection of heuristics. Efforts to find such bounds proved to be profitable and are a good direction to undertake. Similarly the search for constructive proofs of necessary characteristics of optimality yields algorithmic mechanisms to refine sub-optimal colorings. For example, an implementation of the proof of theorem 3.2 on bicolored paths is an obvious direction to explore.

Bibliography

- [AhSU86] A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, Reading, MA, USA, 1986. (Chapter 9).
- [AhHU82] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, USA, 1982. (Section 6.6).
- [AlSE92] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, NY, 1992. (pp. 86-89, 113-114, 146-149).
- [ApHa77] K. Appel and W. Haken. "Every planar map is four-colorable". *Illinois Journal of Mathematics*, 21, 1977. (Parts I and II).
- [AsDN92] P. Ashar, S. Devadas, and A. R. Newton. *Sequential Logic Synthesis*. Kluwer Academic Publishers, Boston, MA, 1992.
- [BaGS94] D. Bacon, S. Graham, and O. Sharp. "Compiler Transformations for High-Performance Computing". *ACM Computing Surveys*, 26(4):345-420, December 1994.
- [BeSu93] M. Bellare and M. Sudan. "Improved Non-Approximability Results". *26th ACM Symposium on Theory of Computing*, pages 184-193, 1993.
- [Berg73] C. Berge. *Graphs and Hypergraphs*. North-Holland Publishing Company, 1973. (Chapter 15).
- [Bigg85] N. Biggs. *Discrete Mathematics*. Oxford University Press, Oxford, Great Britain, 1985. (Chapter 8).

- [BiLW76] N. Biggs, K. Lloyd, and R. Wilson. *Graph Theory 1736-1936*. Oxford University Press, Oxford, England, 1976. (Chapters 6,7 and 9).
- [Blum94] A. Blum. "New Approximation Algorithms For Graph Coloring". *Journal of the Association of Computing Machinery*, 41(3):470-516, May 1994.
- [Boll88] B. Bollobás. "The Chromatic Number of Random Graphs". *Combinatorica*, 8(1):49-55, 1988.
- [BoEr76] B. Bollobás and P. Erdős. "Cliques in Random Graphs". *Mathematical Proceedings of the Cambridge Philosophical Society*, 80:419-427, 1976.
- [BrBr87] G. Brassard and P. Bratley. *Algorithmics: Theory and Practice*. Prentice Hall, Englewood Cliffs, NJ, USA, 1987. (Section 3.4.1).
- [Ca1545] G. Cardano. *Artis Magnae, sive de regulis algebraicis (The Great Art, or the Rules of Algebra)*. M.I.T. Press, Cambridge, Mass., 1968.
- [CACC81] G. Chaitin, M. Auslander, A. Chandra, J. Cocke, M. Hopkins, and P. Markstein. "Register Allocation via Coloring". *Computer Languages*, 6:47-57, January 1981.
- [ChNS80] N. Chiba, T. Nishizeki, and N. Saito. *A linear algorithm for five-coloring a planar graph*, volume 108 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.
- [GDWL92] D. Gajski, N. Dutt, A. Wu, and S. Lin. *High-Level Synthesis, Introduction to Chip and System Design*. Kluwer Academic Publishers, MA, USA, 1992.
- [GaKu83] D. Gajski and R. Kuhn. "Guest Editors Introduction: New VLSI Tools". *IEEE Computer*, 6(12):11-14, 1983.
- [GaJo79] M. Garey and D. Johnson. *Computers and Intractability, A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, USA, 1979.
- [Gavr72] F. Gavril. "Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independence Set of a Chordal Graph". *SIAM Journal of Computation*, pages 180-187, 1972.

- [GeEl90] C. Gebotys and M. Elmsary. "A Global Optimization Approach for Architectural Synthesis". *International Conference on Computer Aided Design*, pages 258–261, 1990.
- [GhDN92] A. Ghosh, S. Devadas, and A. R. Newton. *Sequential Logic Testing and Verification*. Kluwer Academic Publishers, MA, USA, 1992.
- [Gibb85] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, Cambridge, Great Britain, 1985. (Chapter 7).
- [Glas92] R. Glass. *Building Quality Software*. Prentice Hall, NJ, 1992. (sections 1.1, 1.5, 1.6).
- [Golu80] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.
- [GrWa77] J. Graver and M. Watkins. *Combinatorics with the emphasis on the theory of graphs*. Springer-Verlag, 1977. (Chapter 7).
- [Hall93] M. M. Halldórson. "A still better performance guarantee for approximate graph coloring". *Information Processing Letters*, 45:19–23, 1993.
- [HaSt71] A. Hashimoto and J. Stevens. "Wire Routing by Optimizing Channel Assignment Within Large Apertures". *8th Design Automation Workshop*, 1971.
- [TCHu69] T. C. Hu. *Integer Programming and Network Flows*. Addison Wesley, 1969. (Chapter 15, section 3).
- [John74] D. S. Johnson. "Worst case behaviour of graph coloring algorithms". *Proceedings of the 5th Southeastern conference on combinatorics*, pages 513–527, 1974.
- [KaMS95] D. Karger, R. Motwani, and M. Sudan. "Approximate Graph Coloring by Semidefinite Programming". *Personal communication*, 1995. (karger@cs.stanford.edu).
- [Ke1879] A. Kempe. "On the Geographical Problem of the Four Colours". *American Journal of Mathematics*, pages 193–200, 1879.

- [Khac79] L. Khachian. "A Polynomial Algorithm In Linear Programming 20". *Soviet Mathematics Doklady*, pages 191-194, 1979.
- [KhLS92] S. Khana, N. Linial, and S. Safra. "On the Hardness of Approximating the Chromatic Number". *Proceedings of the 2nd Israeli Symposium on Theory and Computing Systems*, pages 250-260, 1992.
- [KuDe92] D. Ku and G. DeMicheli. *High Level Synthesis of ASICs Under Timing and Synchronization Constraints*. Kluwer Academic Publishers, Norwell, MA, USA, 1992. (Chapter 1).
- [KuPa87] F. Kuhard and A. Parker. "REAL: A Program for REGISTER ALlocation". *Proceedings of the 24th ACM/IEEE Design Automation Conference*, pages 210-215, 1987.
- [LaTh89] E. Lagnese and D. Thomas. "Architectural Partitioning for System Level Design". In *Proceedings of the 26th Design Automation Conference*, pages 62-67. ACM/IEEE, 1989.
- [Lawl76] E. Lawler. "A note on the complexity of the chromatic number problem". *Rapport de Recherche de l'IRIA*, (204), December 1976.
- [CLiu68] C. L. Liu. *Introduction to Combinatorial Mathematics*. McGraw-Hill, New York, NY, 1968. (Chapter 9).
- [Lova79] L. Lovasz. *Combinatorial Problems and Exercises*. North-Holland, Amsterdam, Holland, 1979. (Section 9).
- [LuYa94] C. Lund and M. Yannakakis. "On the Hardness of Approximating Minimization Problems". *Journal of the Association of Computing Machinery*, 41(5):960-981, 1994.
- [LuYa93] C. Lund and M. Yannanakis. "On the Hardness of Approximating Minimization Problems". In *25th ACM Annual Symposium on the Theory of Computing*, pages 286-293. ACM, 1993.
- [Matu87] D. Matula. "Expose-And-Merge Exploration and The Chromatic Number of a Random Graph". *Combinatorica*, 7(3):275-284, 1987.

- [Merm92] J. Mermet, editor. *VHDL for Simulation, Synthesis and Formal Proofs of Hardware*. Kluwer Academic Publisher, 1992.
- [MiLD92] P. Michel, U. Lauther, and P. Duzy. *The Synthesis Approach To Digital System Design*. Kluwer Academic Publishers, Boston, MA, USA, 1992. (Chapter 6).
- [Naza87] J. L. Nazareth. *Computer Solution of Linear Programs*. Monographs on Numerical Analysis. Oxford University Press, Oxford, England, 1987. (Chapter 1).
- [NeWo88] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988.
- [PaWi88] P. Papalambros and D. Wilde. *Principles of Optimal Design, Modeling and Computation*. Cambridge University Press, 1988. (chapters 1 and 7).
- [PaRo87] P. Pardalos and J. Rosen. *Constrained Global Optimization: Algorithms and Applications*, volume 268 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987. (Chapter 3).
- [Pa1896] V. Pareto. *Cours d'Économie Politique*. Rouge, Lausanne, Switzerland, 1896.
- [Perr91] D. Perry. *VHDL*. McGraw-Hill, 1991.
- [PFTV88] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1988. (Chapter 10).
- [Sher93] N. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, Boston, MA, USA, 1993. (Chapter 3).
- [SpTh94] D. Springer and D. Thomas. "Exploiting the Special Structure of Conflict and Compatibility Graphs in High-Level Synthesis". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 13(7):843–856, July 1994.
- [Stra69] V. Strassen. "Gaussian elimination is not optimal". *Numerische Mathematik*, 13:354–356, 1969.

- [TaLe91] Y. Takefuji and K. Lee. "Artificial Neural Networks for the 4 colouring Map problem and the K-colourability problem". *IEEE Transactions on Circuits and Systems*, 38, 1991. (pp. 326-333).
- [THKR83] D. Thomas, C. Hitchcock, T. Kowalski, J. Rajan, and R. Walker. "Automatic Data Path Synthesis". *IEEE Computer Magazine*, 1983.
- [Tome85] I. Tomescu. *Problems in Combinatorics and the Theory of Graphs*. Discrete Mathematics. Wiley Interscience, 1985.
- [TsSi86] C. Tseng and D. Siewiorek. "Automated Synthesis of Data Paths on Digital Systems". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-5(3):379-395, July 1986.
- [Turn88] J. S. Turner. "Almost All k -Colorable Graphs Are Easy to Color". *Journal of Algorithms*, 9:63-82, 1988.
- [Vizi64] V. G. Vizing. "On an estimate of the chromatic number of a p -graph". *Diskret. Analiz.*, 3:25-30, 1964.
- [Wels94] S. Welstead. *Neural Network and Fuzzy Logic Applications*. John Wiley and Sons, 1994. (Chaper 1).
- [Wigd83] A. Widgerson. "Improving the Performance Guarantee for Approximate Graph Coloring". *Journal of the Association of Computing Machinery*, 10:729-735, 1983.
- [Wilf84] H. Wilf. "Backtrack: An $O(1)$ expected time algorithm for the graph coloring problem". *Information Processing Letters*, 18:119-121, March 1984.
- [Zlob78] S. Zlobec. *Lecture Notes in Mathematical Programming*. Department of Mathematics, McGill University, 1978. (chapter 3).

Appendix A

WWI Implementation

Our implementation of the WWI on affinities algorithm is described in this appendix. It will be evident in the upcoming discussion that it is possible to attain a more efficient implementation at the expense of implementation obscurity. As for the implementation of WWI on conflicts, it is identical with exception to the criteria of section A.2 which must reflect those of section 5.2.2.

The implementation depends on four main data structures generated each time a graph $G = (V, E)$ with $V = \{v_1, v_2, \dots, v_n\}$ and $|V| = n$ is passed as an instance to the algorithm.

The first is the $n \times n$ adjacency matrix A . Each element of the adjacency matrix is initially defined as follows:

$$A(i, j) = \begin{cases} 0 & \text{if } \{v_i, v_j\} \notin E \\ 0 & \text{if } i = j \\ 1 & \text{if } \{v_i, v_j\} \in E \end{cases}$$

After each compression the adjacency matrix is adjusted to reflect the edges in the compressed graph.

The second structure is C , the set of colors used by the algorithm. C is implemented as a linked list of integers. For each compression a color is removed from C to indicate that the two compressed nodes have been assigned a common color. Initially, each node has a

different color corresponding to its node index. Figure A.1 (a) shows the initial assignment of C . When two nodes v_i and v_j are compressed, it is the highest index which is removed from the list. If $i < j$ then j would be removed from the list. For that reason the list of colors can also be interpreted as a list of indices for the vertices remaining in the graph. Each time two vertices v_i and v_j with $i < j$ are compressed, they are replaced by a vertex v'_i in the compressed graph and v'_i has the adjacencies of v_i and v_j in the graph prior to compression. For example the graph of figure 2.4 (c) would have the color list shown in figure A.1 (b).

The third structure is the n -entry vector K which keeps track of the color of each node in the initial graph.

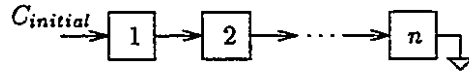
$$K(i) = k \text{ if the color of vertex } v_i \text{ is } k$$

For the graph of figure 2.4 (c) the color vector K is shown in figure A.1 (c).

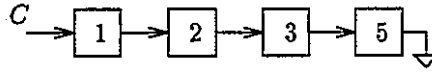
The last data structure is the list EC which is used to store the edge complement. Each $ec \in EC$ represents a pair in the edge complement and it has five parameters associated with it: $ec.lo$, $ec.hi$, $ec.aff$, $ec.con_lh$, $ec.con_hl$. $ec.lo$ and $ec.hi$ are the indices of a pair of non-adjacent vertices in a graph. $ec.aff$, $ec.con_lh$ and $ec.con_hl$ represent the affinity and conflicts between the vertices. Suppose two non-adjacent nodes v_4 and v_5 with $\alpha(v_4, v_5) = 3$, $\gamma(v_4, v_5) = 1$ and $\gamma(v_5, v_4) = 2$. The entry ec for v_4 and v_5 in EC would have $ec.lo = 4$, $ec.hi = 5$, $ec.aff = 3$, $ec.con_lh = 1$, $ec.con_hl = 2$. The distinction between lo and hi indicates that we sort the vertices of a pair based on the indices. However, this is not of importance to the proper behavior of the algorithm. Examples for EC are the tables in figure 5.2 and figure 5.5 (b) with the exception that vertices with a composite index such as $v_{3,6}$ would be represented by a single index, in this particular case 3 would be used for $v_{3,6}$.

Finally the algorithm uses an integer to count the number of colors required by the algorithm. It is represented by the variable χ . The following gives a general description of the WWI algorithms. Subsequently a detailed description will be given along with a complexity analysis of the algorithm.

0. Assume a simple undirected graph $G = (V, E)$



(a) Initial Color List C



(b) Color list for figure 2.4 (c)

$K(1)$	$K(2)$	$K(3)$	$K(4)$	$K(5)$	$K(6)$
1	2	3	3	5	3

(c) Node colors for the graph of figure 2.4 (c)

Figure A.1: Data structure examples

1. Initialize C , K , EC , A , χ as defined above
2. while ($EC \neq \emptyset$) do
 - 2.1 Find the best compression $v_{\text{opt.lo}}$, $v_{\text{opt.hi}}$
 - 2.2 Adjust EC for the compression
 - 2.3 Adjust the adjacency matrix A
 - 2.4 Remove opt.hi from list C
 - 2.5 Color nodes of opt.hi color to opt.lo in K
 - 2.6 $\chi \leftarrow \chi - 1$
2. end while
3. Return χ, C, K

The initialization step colors each node with a different color, it finds all the affinities and conflicts between non-adjacent pairs of a graph $G = (V, E)$. The while loop is then iterated once for each compression performed on the graph. For each iteration the two best nodes for compression are selected according to the criteria discussed in section 5.2. The edge complement list is then modified to reflect the new affinities and conflicts after the compression has taken place. The adjacency matrix is also adjusted, the color list is reduced by one color, and the number of colors required by the algorithm is diminished by

one each time a compression is performed.

A.1 Initialization

The following describes in detail the initialization stage:

```
// Initialize  $C$ ,  $K$  and  $A$ 
 $C \leftarrow \emptyset$ 
for  $i \leftarrow 1$  to  $|V|$  do
     $C \leftarrow C \cup \{i\}$ 
     $K(i) \leftarrow i$ 
     $A(i, i) \leftarrow 0$ 
end for

for  $i \leftarrow 1$  to  $|V| - 1$  do
    for  $j \leftarrow i + 1$  to  $|V|$  do
        if  $\{v_i, v_j\} \in E$  then
             $A(i, j) \leftarrow 1$ 
             $A(j, i) \leftarrow 1$ 
        else
             $A(i, j) \leftarrow 0$ 
             $A(j, i) \leftarrow 0$ 
        end if
    end for
end for

// Initialize  $EC$ 
 $EC \leftarrow \emptyset$ 
for  $i \leftarrow 1$  to  $|V| - 1$  do
    for  $j \leftarrow i + 1$  to  $|V|$  do
        if  $A(i, j) = 0$  then
             $ec.lo \leftarrow i$ 
             $ec.hi \leftarrow j$ 
             $ec.aff \leftarrow 0$ 
             $ec.con\_lh \leftarrow 0$ 
             $ec.con\_hl \leftarrow 0$ 
            for  $k \leftarrow 1$  to  $|V|$  do
                if  $A(i, k) = 1 \wedge A(j, k) = 1$  then
                     $ec.aff \leftarrow ec.aff + 1$ 
                else if  $A(i, k) = 1 \wedge A(j, k) = 0$  then
```



```

        ec.con_lh  $\leftarrow$  ec.con_lh + 1
    else if  $A(i, k) = 0 \wedge A(j, k) = 1$  then
        ec.con_hl  $\leftarrow$  ec.con_hl + 1
    end if
end for
 $EC \leftarrow EC \cup \{ec\}$ 
end if
end for
end for

```

```

// Each node starts with a distinct color
 $\chi \leftarrow |V|$ 

```

When analyzing the complexity of the initialization stage it is clear that the attention should be concentrated on the triple-nested loop which generates EC . Ignoring the body within the second loop one can notice that the two external loops will iterate in the order of $O(|V|^2)$. And clearly the body within the conditional of the second loop will execute $|EC|$ times. For each iteration of the conditional body there will be a traversal of the $|V|$ vertices to find the affinity and conflicts of the two non-adjacent vertices. Hence the execution of the initialization is within the order of $O(|V|^2 + |EC| \cdot |V|)$ where $0 \leq |EC| \leq \frac{|V| \cdot (|V|-1)}{2}$.

Our actual implementation differs as it takes advantage of the symmetry of the adjacency matrix and the mechanism to calculate the affinities and conflicts is slightly more efficient as it takes advantage of the possible sparsity of the adjacency matrix. However, in the worst case, it preserves the same complexity. There is a more efficient technique which we have not exercised for reasons of implementation ease. First define the matrix \overline{A} :

$$\overline{A}(i, j) = \begin{cases} 0 & \text{if } A(i, j) = 1 \\ 1 & \text{if } A(i, j) = 0 \end{cases}$$

and then consider the following matrix products (T denotes the matrix transpose operation):

$$F = A \cdot A^T = A \cdot A = A^2$$

$$G = A \cdot \overline{A}^T = A \cdot \overline{A}$$

It is easy to verify the following relations:

$$\begin{aligned}\alpha(v_i, v_j) &= F(i, j) \quad \text{if } \{v_i, v_j\} \notin E \\ \gamma(v_i, v_j) &= G(i, j) \quad \text{if } \{v_i, v_j\} \notin E \\ d(v_i) &= F(i, i) \quad \forall v_i \in V\end{aligned}$$

Therefore it is possible to calculate the affinities and conflicts through matrix multiplications. Several algorithms have recently been published to multiply matrices in efficient times [BrBr87]. For example, using Strassen's algorithm one could calculate the affinities and conflict in the order of $O(|V|^{2.81})$ [Stra69].

A.2 Finding the best compression

We now discuss algorithm to find the best possible compression in the graph according to the criteria of section 5.2. The algorithm traverses list *EC* and stores the indices of the best compression in variables *opt_lo* and *opt_hi*. In this particular case the conditions are for WWI on affinities.

```
// Finding the best compression
max_affinity ← -1
for each ec ∈ EC do
  if ec.con_lo = 0 ∨ ec.con_hi = 0 then
    opt_lo ← ec.lo
    opt_hi ← ec.hi
    break out of for loop
  else if ec.aff > max_affinity then
    max_affinity ← ec.aff
    min_conflict ← ec.con_lo + ec.con_hi
    opt_lo ← ec.lo
    opt_hi ← ec.hi
  else if ec.aff = max_affinity ∧
    min_conflict > ec.con_lo + ec.con_hi then
    min_conflict ← ec.con_lo + ec.con_hi
    opt_lo ← ec.lo
    opt_hi ← ec.hi
  end if
end for
```

Since the body of the loop only executes $O(1)$ type operations it is clear that the algorithm to find the best compression is $O(|EC|)$ where $0 \leq |EC| \leq \frac{|V| \cdot (|V|-1)}{2}$. For WWI on conflicts simply replace the criteria with those of section 5.2.2 and ensure that the search is localized.

A.3 Adjusting EC

Once the two nodes v_{opt_lo} and v_{opt_hi} have been selected to be compressed, WWI proceeds by adjusting EC to reflect the new affinities and conflicts in the compressed graph. Elements of EC are separated into six disjoint cases and each case is treated differently (see figures A.2, A.3 for diagrams of the cases as they are described):

1. $ec \in EC$ such that $ec.lo = opt_hi$ or $ec.hi = opt_hi$: ec will be deleted from EC since v_{opt_hi} will be compressed with v_{opt_lo} and we therefore only need to retain the non-adjacencies with v_{opt_lo} .
2. $ec \in EC$ such that $ec.lo = opt_lo$, $ec.hi \neq opt_hi$ and $A(ec.hi, opt_hi) = 1$: ec will be deleted since the compression of v_{opt_lo} and v_{opt_hi} will place an edge between $v_{ec.lo}$ and $v_{ec.hi}$. See figure A.2 (a).
3. $ec \in EC$ such that $ec.hi = opt_lo$, $ec.lo \neq opt_hi$ and $A(ec.lo, opt_hi) = 1$: ec will be deleted for the same reason as in case 2. See figure A.2 (b).
4. $ec \in EC$ such that $ec.lo = opt_lo$, $ec.hi \neq opt_hi$ and $A(ec.hi, opt_hi) = 0$: ec is preserved in the compressed graph (see figure A.2 (c)). However the affinity and conflicts of $v_{ec.lo}$ and $v_{ec.hi}$ need to be recalculated. This is done by traversing the adjacencies of $v_{ec.lo}$ and $v_{ec.hi}$ in the compressed graph.
5. $ec \in EC$ such that $ec.hi = opt_lo$, $ec.lo \neq opt_hi$ and $A(ec.lo, opt_hi) = 0$: ec will be preserved (see figure A.2 (d)) but the affinity and conflicts need to be recalculated as in case 4.

6. $ec \in E$ such that both $ec.lo$ and $ec.hi$ are different from $opt.lo$ and $opt.hi$: E^c will be preserved and the affinity and conflicts between $v_{ec.lo}$ and $v_{ec.hi}$ need to be recalculated. Figure A.3 shows that there are 16 possible cases each identifiable by the presence or absence of edges and that the affinities and conflicts can be adjusted with simple increment and decrement operations. Based on the values of $A(ec.lo, opt.lo)$, $A(ec.lo, opt.hi)$, $A(ec.hi, opt.lo)$ and $A(ec.hi, opt.hi)$ which distinguish the cases, it is easy to build a truth table or a Karnaugh map to generate conditions that will implement the appropriate increment and decrement operations. As it will be demonstrated, this observation yields a substantial performance gain with respect to recalculating the affinity and conflicts by traversing the adjacencies of $v_{ec.lo}$ and $v_{ec.hi}$.

The following is the algorithm to adjust EC :

```
//Readjust EC for the compression
//of  $v_{opt.lo}$  and  $v_{opt.hi}$ 
for each  $ec \in EC$  do

    // Case 1.
    if  $ec.lo = opt.hi \vee ec.hi = opt.hi$  then
         $EC \leftarrow EC \setminus \{ec\}$ 

    // Case 2.
    elseif  $ec.lo = opt.lo \wedge A(ec.hi, opt.hi) = 1$  then
         $EC \leftarrow EC \setminus \{ec\}$ 

    // Case 3.
    elseif  $ec.hi = opt.lo \wedge A(ec.lo, opt.hi) = 1$  then
         $EC \leftarrow EC \setminus \{ec\}$ 

    // Case 4.
    elseif  $ec.lo = opt.lo \wedge A(ec.hi, opt.hi) = 0$  then
         $ec.aff \leftarrow 0$ 
         $ec.con_{lh} \leftarrow 0$ 
         $ec.con_{hl} \leftarrow 0$ 
        // Traverse the compressed nodes
        for each  $c \in C$  do
            if  $A(ec.hi, c) = 1 \wedge$ 
```

```

        ( $A(\text{opt\_lo}, c) = 1 \vee A(\text{opt\_hi}, c) = 1$ ) then
             $ec.\text{aff} \leftarrow ec.\text{aff} + 1$ 
        else if  $A(ec.\text{hi}, c) = 1 \wedge$ 
            ( $A(\text{opt\_lo}, c) = 0 \wedge A(\text{opt\_hi}, c) = 0$ ) then
             $ec.\text{con\_hl} \leftarrow ec.\text{con\_hl} + 1$ 
        else if  $A(ec.\text{hi}, c) = 0 \wedge$ 
            ( $A(\text{opt\_lo}, c) = 1 \vee A(\text{opt\_hi}, c) = 1$ ) then
             $ec.\text{con\_lh} \leftarrow ec.\text{con\_lh} + 1$ 
        end if
    end for

```

//Case 5.

```

else if  $ec.\text{hi} = \text{opt\_lo} \wedge A(ec.\text{lo}, \text{opt\_hi}) = 0$  then
     $ec.\text{aff} \leftarrow 0$ 
     $ec.\text{con\_lh} \leftarrow 0$ 
     $ec.\text{con\_hl} \leftarrow 0$ 
    // Traverse the compressed nodes
    for each  $c \in C$  do
        if  $A(ec.\text{lo}, c) = 1 \wedge$ 
            ( $A(\text{opt\_lo}, c) = 1 \vee A(\text{opt\_hi}, c) = 1$ ) then
             $ec.\text{aff} \leftarrow ec.\text{aff} + 1$ 
        else if  $A(ec.\text{lo}, c) = 1 \wedge$ 
            ( $A(\text{opt\_lo}, c) = 0 \wedge A(\text{opt\_hi}, c) = 0$ ) then
             $ec.\text{con\_lh} \leftarrow ec.\text{con\_lh} + 1$ 
        else if  $A(ec.\text{lo}, c) = 0 \wedge$ 
            ( $A(\text{opt\_lo}, c) = 1 \vee A(\text{opt\_hi}, c) = 1$ ) then
             $ec.\text{con\_hl} \leftarrow ec.\text{con\_hl} + 1$ 
        end if
    end for

```

// Case 6.

```

else
    //  $x_1, x_2, x_3, x_4, y_1, y_2$  are booleans
     $x_1 \leftarrow A(ec.\text{lo}, \text{opt\_lo}) = 1$ 
     $x_2 \leftarrow A(ec.\text{lo}, \text{opt\_hi}) = 1$ 
     $x_3 \leftarrow A(ec.\text{hi}, \text{opt\_lo}) = 1$ 
     $x_4 \leftarrow A(ec.\text{hi}, \text{opt\_hi}) = 1$ 
     $y_1 \leftarrow x_1 \wedge x_2$ 
     $y_2 \leftarrow x_3 \wedge x_4$ 
    if  $(x_2 \wedge x_3 \wedge \sim (x_1 \vee x_4)) \vee$ 
         $(x_1 \wedge x_4 \wedge \sim (x_2 \vee x_3))$  then
         $ec.\text{aff} \leftarrow ec.\text{aff} + 1$ 
    end if

```

```

    ec.con_lh ← ec.con_lh - 1
    ec.con_hl ← ec.con_hl - 1
  else if  $y_1 \wedge (\sim y_2)$  then
    ec.con_lh ← ec.con_lh - 1
  else if  $(\sim y_1) \wedge y_2$  then
    ec.con_hl ← ec.con_hl - 1
  else if  $y_1 \wedge y_2$  then
    ec.aff ← ec.aff - 1
  end if
end if
end for

```

A complexity analysis of the adjustment of $|EC|$ follows. Clearly if one was to ignore the nested loops within cases 4 and 5 then the re-adjustment would be within the order of $|EC|$. But the nested loops must be taken into account. For that reason the attention of the reader is shifted to two observations. Since node $v_{\text{opt_hi}}$ cannot be non-adjacent to more than $|V| - 1$ other nodes then case 1 cannot be satisfied more than $|V| - 1$ times when the re-adjustment proceeds. Similarly because $v_{\text{opt_lo}}$ can only be non-adjacent to at most $|V| - 2$ nodes different from $v_{\text{opt_hi}}$, cases 2,3,4 and 5 put together can only account for at most $|V| - 2$ iterations of the main loop. Therefore $|V| - 2$ is an upper bound on the number of times the loops of cases 4 and 5 will be visited. Since the loops of cases 4 and 5 require a traversal of the vertices and there are at most $|V|$ vertices at any time, the time spent within these loops is within the order of $O(|V|^2)$. Therefore the full re-adjustment of the whole of $|EC|$ is in the order of $O(|EC| + |V|^2)$. Since $0 \leq |EC| \leq \frac{|V| \cdot (|V| - 1)}{2}$ it follows that the full re-adjustment is the order of $O(|V|^2)$. Since cases 1 to 5 can only be visited at most $2|V| - 3$ times during the whole execution of the main loop it follows that most of the burden will usually be carried out by case 6. Fortunately the technique used to adjust the affinity and conflicts in case 6 is within the order of $O(1)$, thus resulting in an efficient adjustment for the whole of EC .

A.4 Adjusting A, C, K, χ

Adjusting A, C, K, χ is straightforward and it can be done in the order of $O(|V|)$:

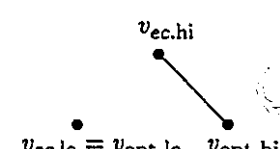
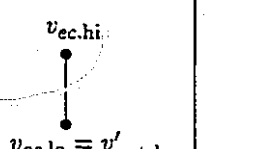
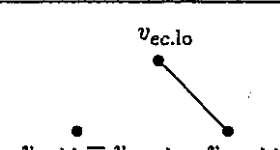
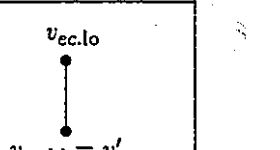
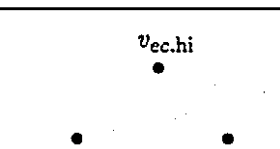
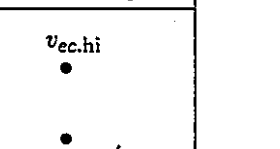
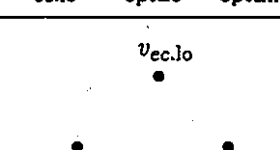
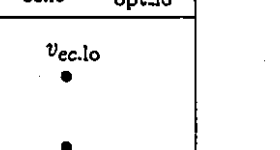
	Before Compression	After Compression
(a)		
(b)		
(c)		
(d)		

Figure A.2: Cases 2,3,4 and 5.

Before Compression	After Compression	Effect
		No effect
		No effect
		No effect
		dec ec.con_lh
		No effect
		No effect
		inc ec.aff dec ec.con_lh dec ec.con_hl
		dec ec.con_lh

inc = increment dec = decrement

Before Compression	After Compression	Effect
		No effect
		inc ec.aff dec ec.con_lh dec ec.con_hl
		No effect
		dec ec.con_lh
		dec ec.con_hl
		dec ec.con_hl
		dec ec.con_hl
		dec ec.aff

inc = increment dec = decrement

Figure A.3: Case 6.


```
// Adjusting  $A, C, K, \chi$ 
```

```
// For  $A$ 
```

```
for each  $c \in C$  do
```

```
  if  $A(\text{opt\_hi}, c) = 1$  then
```

```
     $A(\text{opt\_lo}, c) \leftarrow 1$ 
```

```
     $A(c, \text{opt\_lo}) \leftarrow 1$ 
```

```
  end if
```

```
end for
```

```
// For  $C$ 
```

```
for each  $c \in C$  do
```

```
  if  $c = \text{opt\_hi}$  then
```

```
     $C \leftarrow C \setminus \{c\}$ 
```

```
    break out of loop
```

```
  end if
```

```
end for
```

```
// For  $K$ 
```

```
for  $i \leftarrow 1$  to  $|V|$  do
```

```
  if  $K(i) = \text{opt\_hi}$  then
```

```
     $K(i) \leftarrow \text{opt\_lo}$ 
```

```
  end if
```

```
end for
```

```
// For  $\chi$ 
```

```
 $\chi \leftarrow \chi - 1$ 
```

A.5 Complexity of the implementation

The overview of the algorithm is revisited and it is annotated with the complexities of the steps within the while loop:

Get a graph $G = (V, E)$ to color

Initialize C, K, EC, A, χ

while ($EC \neq \emptyset$) do

$O(|EC|)$ Find the best compression $v_{\text{opt_lo}}, v_{\text{opt_hi}}$

```

 $O(|V|^2)$   Adjust  $EC$  for the compression
 $O(|V|)$     Adjust the adjacency matrix  $A$ 
 $O(|V|)$     Remove  $opt\_hi$  from list  $C$ 
 $O(|V|)$     Color nodes of  $opt\_hi$  color to  $opt\_lo$  in  $K$ 
 $O(1)$        $\chi \leftarrow \chi - 1$ 
           end while

```

Return χ, C, K

Clearly the step that re-adjusts $|EC|$ at $O(|V|^2)$ dominates the body of the loop and therefore one iteration can be executed within the order of $O(|V|^2)$. Notice that the loop cannot have more than $|V|$ iterations since each iteration corresponds to a compression and there cannot be more than $|V| - \chi(G)$ compressions for a graph. Otherwise would contradict that the chromatic number is $\chi(G)$. Hence the execution of the loop is of the order of $O(|V|^3)$. Since the proposed initialization is of the order of $O(|V|^2 + |EC| \cdot |V|)$ it follows that the loop is the dominant factor and the WWI algorithms are of order $O(|V|^3)$. However since the maximum possible value of $|EC|$ is $\frac{|V| \cdot (|V|-1)}{2}$ the time spent initializing the structures can be comparable to the time spent performing all the compressions. Our experiments validate this and that is why techniques improving the initialization complexity are of importance. Since the complexity of the algorithm strongly depends on $|EC|$, it is apparent that the algorithm should perform best when $|EC|$ is smallest. This in turn implies that WWI will perform at its best when the graph has a high edge population and it will perform at its worst in a graph void of any edges. This also has been verified experimentally. A final note pertains to the first two steps within the body of the loop (2.1 and 2.2). Both of these steps are closely related as they require a traversal of EC . To save on execution time we have folded both of these steps into a single iteration over EC in our actual implementation.

Appendix B

Fundamental Node Refinement

This appendix presents a simple algorithm which refines a coloring so that the optimality characteristic of theorem 3.1 is met. The algorithm requires a graph $G = (V, E)$ and a coloring C using the set of colors S as input. It returns a refined coloring C' using the set of colors S' . We need three specialized structures: L, R and adj_color . $L(s)$ represents the set of nodes assigned to color s ; $R(v)$ is a color to which vertex v can be recolored or has value *none* if no such color exists; and $adj_color(t)$ is either *true* or *false* if a particular vertex under study is adjacent to a node of color t . The algorithm proceeds as follows:

```
// Obtain the input
Get A graph  $G = (V, E)$ ,
    A coloring  $C$  using the set of colors  $S$ ;
// Copy over the coloring
 $C' \leftarrow C$ 
 $S' \leftarrow S$ 
// Regroup nodes with same color
for each  $s \in S$  do
     $L(s) \leftarrow \emptyset$ 
end for
for each  $v \in V$  do
     $L(C'(v)) \leftarrow L(C'(v)) \cup \{v\}$ 
end for
//Recolor the nodes
for each  $s \in S$  do //  $S$ , not  $S'$ 
     $fundamental\_s \leftarrow false$ 
```

```

for each  $v \in L(s)$  do
  for each  $t \in S'$  do
     $adj\_color(t) \leftarrow false$ 
  end for
   $adj\_color(s) \leftarrow true$ 
  for each  $w \in V$  do
    if  $v \neq w \wedge \{v, w\} \in E$  then
       $adj\_color(C'(w)) \leftarrow true$ 
    end if
  end for
   $R(v) \leftarrow none$ 
  for each  $t \in S'$  do
    if  $adj\_color(t) = false$  then
       $R(v) \leftarrow t$ 
    end if
  end for
  if  $R(v) = none$  then
     $fundamental\_s \leftarrow true$ 
    break // out of for loop
  end if
end for
if  $fundamental\_s = false$  then
  for each  $v \in L(s)$  do
     $C'(v) \leftarrow R(v)$ 
     $L(R(v)) \leftarrow L(R(v)) \cup \{v\}$ 
  end for
   $S' \leftarrow S' \setminus \{s\}$  //  $S'$  is a subset to  $S$ 
end if
end for
// Return the result
return  $C', S'$ 

```

It is easy to verify that if a coloring C of a graph $G = (V, E)$ already meets the characteristic of optimality of theorem 3.1 then the refinement will be achieved in time complexity of order $O(|V|^2)$. This is because the two outer loops of the recoloring code will only traverse each vertex once while investigating the color of its neighbors. However, if the characteristic of optimality is not met then vertices will be recolored and therefore some nodes might be traversed more than once. For the algorithm to remain of order $O(|V|^2)$ for all input instances the following rule must be imposed:

- Loops which traverse the elements of S' must do so in the same order in which the elements appear in the loop over S .

This guarantees that a node will never be visited more than twice. A node v of color s_1 will be visited twice if it is recolored to color s_2 when s_1 precedes s_2 in the order that the elements of S are traversed. If, when the vertices of color s_2 are visited, vertex v needs to be recolored once more, then it can only be to a color s_3 which precedes s_2 in the ordering. This is because the loop which assigned $R(v)$ to s_2 in the first recoloring ensured that s_2 was the highest possible color to which v was not adjacent (under the condition that the elements of S' were traversed in the same order as they appear in the loop which traverses S). Therefore v is adjacent to all colors beyond s_2 and it must be reassigned to a color prior to s_2 if it is to be recolored. Hence, v can only be visited twice and the algorithm is $O(|V|^2)$ for all input instances. The ordering constraint implies the most natural way in which to implement the algorithm but it is important that it be specified otherwise the algorithm would be of order $O(|V|^3)$. The worst case occurs when a graph with no edges is presented with a different color for each node. In that particular case all nodes but one will be visited twice. Finally, to color a graph from scratch simply provide a coloring with each node a different color.

Appendix C

Colored Path Refinement

This appendix presents an algorithm which ensures that a graph has a colored path which traverses the colors of a coloring in the precise order of a color permutation prescribed to it. Three arguments are expected as input: a graph $G = (V, E)$, a coloring C and a permutation Π of the colors in C . And two outputs are produced: a new coloring C' and an integer χ which represents the number of colors in C' . C' is guaranteed to have a colored path which traverses the colors in the order of Π and it will use fewer colors than C if the latter does not have such a path to start with. Otherwise C' will require the same number of colors as C .

Only one specialized structure is used by the algorithm and that is structure L which is used to represent the colored stratification $\mathcal{L}(G, C, \Pi)$. Each $L(s)$ represents the nodes of a specific stratum, i.e. all the nodes which share color s in the set of colors S . The following is the algorithm in its entirety:

```
// Obtain the input
Get A graph  $G = (V, E)$ ,
    A coloring  $C$  using the set of colors  $S$ ,
    A color ordering  $\Pi = (\pi_1, \pi_2, \dots, \pi_{|S|})$ ;
// Initialize number of colors
 $\chi \leftarrow |S| + 1$ 
// Regroup the nodes of same color in  $C$ 
for each  $s \in S$  do
```

```

     $L(s) \leftarrow \emptyset$ 
end for
for each  $v \in V$  do
     $L(C(v)) \leftarrow L(C(v)) \cup \{v\}$ 
end for
// Create the path by altering the coloring
do
     $\chi \leftarrow \chi - 1$ 
    for  $i = 2$  to  $\chi$  do
        for each  $v \in L(\pi_i)$  do
             $lower\_v \leftarrow true$ 
            for each  $w \in L(\pi_{i-1})$  do
                if  $\{v, w\} \in E$  then
                     $lower\_v \leftarrow false$ 
                end if
            end for
            if  $lower\_v = true$  then
                 $L(\pi_i) \leftarrow L(\pi_i) \setminus \{v\}$ 
                 $L(\pi_{i-1}) \leftarrow L(\pi_{i-1}) \cup \{v\}$ 
            end if
        end for
    end for
while ( $L(\pi_\chi) = \emptyset$ )
// Create the new coloring  $C'$ 
for  $i = 1$  to  $\chi$  do
    for each  $v \in L(\pi_i)$  do
         $C'(v) \leftarrow \pi_i$ 
    end for
end for
// Return the result
return  $C', \chi$ 

```

Each iteration of the **do-while** loop removes one color off the original coloring (the highest ranked color in Π which remains). The iterations terminate when the graph has a colored path in the order of the colors of Π . The triply nested loop **for** loop traverses the stratification (L) in a manner similar to that of the proof of theorem 4.3. The outermost **for** loop traverses the strata one by one starting from the 2nd lowest to the highest ranked stratum. The middle loop traverses each vertex of a particular stratum while the innermost loop verifies that it is not adjacent to any vertices of the stratum below. If such is the case

then the vertex is automatically recolored to the color of the stratum below.

Although that would be sufficient, the algorithm does not limit itself to recoloring nodes which stem a longest colored path to the highest ranked color. Instead, it proceeds by changing the color of all nodes which are not adjacent to nodes of the previous stratum. By doing so, nodes stemming the longest colored path will be included in the recolored nodes and the algorithm remains simple and efficient.

For each iteration of the **do** loop each vertex will be traversed once and the nodes on the strata below will be examined for adjacencies. Therefore the algorithm is of the complexity $O(|V|^2)$ for each color it removes.

C.1 Greedy Algorithm

The greedy algorithm (as described in section 2.5.1) and the sequential algorithm [Gibb85] both share a property with respect to colored paths. If they produce a coloring using the range of colors from 1 to k then it is certain that they will return a coloring which has a colored path on the color permutation $\Pi = (1, 2, \dots, k)$. This is because the coloring mechanisms of the algorithms attribute color j to a node if and only if it is adjacent to at least one from each color in 1 to $j - 1$. Therefore if one selects a node of color k then it must be adjacent to at least one node of color $k - 1$ which in turn must be adjacent to one node of color $k - 2$ and so on until color 1 is reached; thus forming a colored path over the permutation $\Pi = (1, 2, \dots, k)$ of colors. Hence it is pointless to run a refinement over the permutation $\Pi = (1, 2, \dots, k)$ if the graph is originally colored with the greedy algorithm. Interestingly, by theorem 4.4 this also implies that the greedy algorithm will always result in a coloring which is a local minimum of its feasible region. But the emplacement of that feasible region can be quite undesirable. Given the poor efficiency of the greedy algorithm [BrBr87] it is best to avoid the greedy heuristic to initially color a graph if possible.

However the greedy heuristic can be used as a refinement to ensure that a colored path does indeed exist for a particular graph coloring and a permutation of its colors. And as

with the previous algorithm if the path does not exist then the coloring returned by the greedy refinement is certain to require fewer colors. Our constructive proof of this resembles that of theorem 4.3 and relies on the properties of the greedy algorithm. The proof given in theorem 4.3 was preferred since it constructs a coloring in the same feasible region as the original coloring of our programming formulation and that was needed for the proof of the subsequent result which characterizes the local minima (theorem 4.4). The greedy refinement is very likely to produce a coloring in another feasible region.

To use greedy heuristic as a refinement on a coloring C and a coloring permutation $\Pi = (\pi_1, \pi_2, \dots, \pi_k)$ it suffices modify the algorithm in section 2.5.1 so that the nodes are traversed in the order of their colors in the permutation: nodes of color π_1 first, then π_2 , ..., and finally π_k (no importance needs to be attached on the order amongst nodes of the same color). The colors must be traversed in the order of the permutation as well. This results in an $O(|V|^2)$ refinement regardless of the number of colors removed by the algorithm. However the greedy refinement traverses the adjacencies of all colors for each vertex unlike the previous algorithm which only concerns itself with the nodes of the color strata below it.