A non–dominated sorting hybrid algorithm for multi–objective optimization of engineering problems

Hossein Ghiasi (Corresponding Author)
PhD Student
Department of Mechanical Engineering, McGill University
Macdonald Engineering Building, Room 368
817 Sherbrooke West
Montreal, QC, Canada H3A 2K6
Phone: +1(514) 398–6292
Fax: +1(514) 398–7365
Email: hossein.ghiasi@mail.mcgill.ca

Damiano Pasini
Assistant Professor
Department of Mechanical Engineering, McGill University
Macdonald Engineering Building, Room 372
817 Sherbrooke West
Montreal, QC, Canada H3A 2K6
Phone: +1(514) 398–6295
Fax: +1(514) 398–7365
Email: damiano.pasini@mcgill.ca

Larry Lessard
Associate Professor
Department of Mechanical Engineering, McGill University
Macdonald Engineering Building, Room 362
817 Sherbrooke West
Montreal, QC, Canada H3A 2K6
Phone: +1(514) 398–6305
Fax: +1(514) 398–7365
Email: larry.lessard@mcgill.ca

A non–dominated sorting hybrid algorithm for multi–objective optimization of engineering problems

Hossein Ghiasi, Damiano Pasini, Larry Lessard

Department of Mechanical Engineering, McGill University
Macdonald Engineering Building, 817 Sherbrooke West
Montreal, QC, Canada H3A 2K6

Abstract

Among numerous multi–objective optimization algorithms, the Elitist Non–dominated Sorting Genetic Algorithm (NSGA–II) is one of the most popular methods due to its simplicity, effectiveness and minimum involvement of the user. In this paper, we develop a multi–objective variation of the Nelder–Mead simplex method which is then hybridized with NSGA–II to improve the convergence rate and ability of NSGA–II to capture a wide extent of the Pareto front. The proposed hybrid algorithm, called Non–dominated Sorting Hybrid Algorithm (NSHA), is compared to NSGA–II on several constrained and unconstrained mathematical test functions. The higher convergence rate and wider spread of solutions obtained with NSHA makes this algorithm a good candidate for engineering problems that require time–consuming simulation and analysis. To demonstrate this point, NSHA is applied to the design of a carbon fiber bicycle stem simultaneously optimized for strength, weight and processing time.

Keywords: multi–objective optimization; genetic algorithm; non–dominated sorting; hybrid algorithm

1 Introduction

Optimizing multiple performance criteria is a goal governing the design of engineering products in today's competitive and demanding market. A good example of a multi–criterion problem is designing with composite materials, where the structural and the manufacturing parameters are often strongly coupled (Le Riche *et al.*, 2003; Park *et al.*, 2005; Ghiasi *et al.*, 2008) and must be optimized simultaneously. Here the challenge consists in solving an optimization problem with multiple conflicting objectives. The solution of such a multi–objective optimization problem is a set of optimum solutions, representing the trade–off among objectives. The set of solutions is called a *Pareto optimal set* or a *Pareto front* (Deb, 2001).

Because of using a population–based approach, *evolutionary algorithms* (EAs) have a good potential to be used for multi–objective optimization. Contrary to classical optimization methods that provide only one optimum for each run, EAs have the potential to achieve multiple optimum solutions just in one run. In their previous works, the authors (Ghiasi *et al.*, 2008, 2009b) showed that using a globalized form of a classical simplex method and a multi–objective optimization approach can be more efficient than an evolutionary method, when one or a small number of Pareto solutions is required. When a large number of Pareto solutions for multi–objective optimization problems is required, EAs are better suited than classical optimization methods (Ghiasi et al., 2009b). Because of their simplicity, robustness and independency on gradient information, EAs have received significant attention from the researchers in this field. Several EAs have been developed to solve multi–objective problems, examples of which are: Pareto–archived evolution strategy (PAES; Sayin and Karabati, 1999), strength Pareto–evolutionary algorithm (SPEA2; Zitzler *et al.*, 2001) and multi–objective differential evolution (MODE; Babu and Anbarasu, 2005). An overview of the earlier works using EAs for multi–objective optimization can be found in Fonseca and Fleming (1995) and in Coello (1999, 2002), who provided a comprehensive survey and a critical review of the evolutionary–based multi–objective algorithms.

Among numerous multi–objective optimization methods, some listed in preceding paragraph, *genetic–based algorithms* (GAs) have attracted the most attention (Coello, 1999) because of their ability in addressing discontinuous, nondifferentiable and nonconvex functions having multiple peaks and supporting parallel computation. The potential of GAs in multiobjective optimization was

initially hinted by Rosenberg in the 1960s, and later by Goldberg (1989). But, this research area remained unexplored until recently when many GA–based methods have been developed. Due to the large number and diversity of applications, this paper can not include a comprehensive review and comparison of GA–based methods, thus only the most popular multi–objective genetic algorithms are briefly explained in the following paragraphs.

Hajela and Lin's genetic algorithm (HLGA; Hajela and Lin, 1992) used the weighted–sum approach to assign fitness to each individual. The weighting coefficients were included in the chromosome, thus GA evolves solutions and weight combinations simultaneously. Difficulty in determining appropriate weights is the main drawback of this method (Bingul, 2006). Vector evaluated genetic algorithm (VEGA; Schaffer, 1984) avoids this problem by creating a number of sub–populations and performing the selection according to each objective function in turn. In this method no weighting coefficient is required; however, the population may tend to split into different species, each of them particularly strong in one of the objectives.

As Opposed to the mentioned methods, recent works on multi–objective optimization are generally based on the definition of Pareto optimality. Multi–objective genetic algorithm (MOGA; Fonseca and Fleming, 1993) uses a rank–based fitness assignment procedure, in which each individual is ranked by number of individuals dominating the selected solution. Fitness sharing in the objective value domain, rather than the decision variable domain, and only between pairwise non–dominated individuals, was used to evolve a uniformly distributed representation of the global trade–off surface. This method was later improved to a real–coded algorithm called rMOGA (Purshouse and Fleming, 2001). Niched Pareto genetic algorithm (NPGA; Horn *et al.*, 1994) also uses Pareto domination tournament. The

tournament selection included picking two (or more) candidate solutions at random and comparing them with a random comparison set. The candidate solution which is not dominated by the comparison set is selected for reproduction. If the candidate solutions are either both dominated or both non–dominated, fitness sharing was applied. Fonseca and Fleming's multi–objective genetic algorithm (FFGA; Fonseca and Fleming, 1993) also uses niche formation methods but with a modified fitness assignment which allows intervention of an external decision maker. The details about how the decision maker can interact with the genetic algorithm can be found in (Fonseca and Fleming, 1993). Non–dominated sorting genetic algorithm (NSGA; Srinivas and Deb, 1994) and its improved form, elitist non–dominated sorting genetic algorithm (NSGA–II; Deb *et al.*, 2002a) are the other examples of GA–based methods using Pareto optimality to rank individuals within the population.

More information about VEGA, MOGA, NPGA, NSGA and their comparative strengths and weaknesses can be found in Coello (2002). Also, Zitzler et al. (2000) provided a comparison among a number of multi–objective evolutionary methods using six test functions. They ranked algorithms regarding the distance of the final solutions to the Pareto front. Several algorithms were applied to six test functions, and the algorithms were ranked as follow: NSGA, VEGA, HLGA, NPGA and FFGA. In this research we used an improved version of the first algorithm in this list, called NSGA–II. This algorithm is explained in detail in section two.

In this paper, NSGA–II is hybridized with a multi–objective adaptation of the Nelder–Mead simplex method (NM). The proposed hybrid method, called *non–dominated sorting hybrid algorithm* (NSHA), maintains all above–mentioned features of NSGA–II, while it improves the convergence rate and the scope of solutions. Using

the same number of function evaluations, the proposed hybrid algorithm is able to achieve a better quality of the results than the ones obtained by NSGA–II.

The remainder of this paper is organized as follows: First, the NSGA–II is explained and the corresponding modifications proposed in the literature are reviewed. Most of these modifications can be equally applied to the proposed hybrid algorithm. The next section describes the proposed hybrid method called NSHA. Section 4 compares NSHA with NSGA–II on some mathematical unconstrained and constrained test problems. After demonstrating the superiority of NSHA to NSGA–II, the hybrid algorithm is applied to a composite design problem in Section 5. Section 6 concludes the paper.

2 Non–dominated Sorting GA

The elitist non–dominated sorting GA (NSGA–II), proposed by Deb *et al.* (2002a), has been demonstrated to be one of the most efficient and popular algorithms for multi–objective optimization. Its performance has been proved through mathematical test functions to be superior to that of other evolutionary multi–objective methods (Deb *et al.*, 2002a). After providing a few necessary definitions, this section briefly describes NSGA–II and reviews the modifications and improvements suggested in the literature.

Definition 1 (*Domination*): Considering a general multi–objective optimization problem formulated as:

$$\begin{aligned} &\min_{\vec{x}} \ \{ f_i(\vec{x}) : R^n \mapsto R; \ \ i = 1,...,m; \ m \geq 2 \} \\ &\text{s.t.} \begin{cases} g_j(\vec{x}) \geq 0; \ \ g_j(\vec{x}) : R^n \mapsto R; \ \ j = 1,...,J; \ J \geq 0 \\ h_k(\vec{x}) = 0; \ \ h_k(\vec{x}) : R^n \mapsto R; \ \ k = 1,...,K; \ K \geq 0 \end{cases} \end{aligned} \tag{1}$$

a feasible solution $\vec{x}_1$ is called *dominating* a feasible solution $\vec{x}_2$, if solution $\vec{x}_1$ is no worse than $\vec{x}_2$ in all objectives and the solution $\vec{x}_1$ is strictly better than $\vec{x}_2$ in at least one objective.:

$$x_1 \text{ dominate } x_2 \Leftrightarrow \begin{cases} \forall i : 1 \le i \le m;\ f_i(\vec{x}_1) \le f_i(\vec{x}_2) \\ \exists j : 1 \le j \le m;\ f_j(\vec{x}_1) < f_j(\vec{x}_2) \end{cases} \qquad (2)$$

A set of solutions is said to be at the same *non–domination front*, if none of which dominates or is dominated by any other solution in the set.

Definition 2 (*Crowding distance***;** Deb *et al.*, 2002a): Crowding distance is a measure of the density of the solutions in the neighbourhood of a selected solution. Crowding distance is calculated as the summation of the major dimensions of the cuboid formed by using the nearest neighbours of the selected solution as the vertices. For instance, the crowding distance for a problem with two objectives is the summation of the length and width of the rectangle with two vertices located at the two solutions on either side of the selected solution (shown in Figure 1). For a problem with three objectives, the crowding distance is the summation of length, width and height of the cube formed around the candidate solution. This parameter is used in order to maintain the diversity of the solutions; therefore, the genetic selection operator gives higher chance of selection to the individual with a higher crowding distance than the one with the lower crowding distance.

2.1 NSGA–II

Fast and elitist non–dominated sorting genetic algorithm or NSGA–II, proposed by Deb *et al.* (2002a), is a multi–objective evolutionary algorithm that uses non–dominated sorting and *crowded–comparison* approach to find a set of evenly distributed solutions to a multi–objective optimization problem. NSGA–II was proposed to reduce the computational complexity, to improve the diversity of the

solutions and to add elitism to its ancestor called non–dominated sorting genetic algorithm (NSGA) (Srinivas and Deb, 1994). Simplicity, effectiveness, modularity and independency on user–defined parameters, are the main factors determining the popularity of NSGA–II among multi–objective optimization methods (Deb, 2008).

As other GA–based methods, NSGA–II starts with a random population of solutions (or individuals). The initial population is then sorted by the non–domination front. In this ranking procedure, all non–dominated solutions are ranked "1" and are temporarily removed from the population. The next set of non–dominated solutions in the population is then defined and ranked "2". The procedure is continued until all the solutions are ranked. To achieve a better computational performance, the actual ranking procedure is different than what is explained here, for which the details can be found in Deb *et al.* (2002a). A population of solutions, called parents, is generated by applying a binary tournament selection to the current population. The binary tournament selection randomly picks two solutions from the current population and selects the better solution with respect to the non–domination rank. Solutions at the same non–domination front are compared by the crowding distance. The genetic operators (i.e. recombination and mutation) are then applied to the population of parents to create a population of off–springs. The next population is formed by taking the best solutions from the combined population of parents and off–springs. The selection procedure is based on the non–domination rank and then the crowding distance. The procedure is terminated when a user–defined maximum number of generations is reached.

Deb (2008) provided a functional decomposition of NSGA–II into three main operations: (i) elitism to achieve fast and reliable convergence towards better solutions, (ii) non–domination sorting to emphasize non–dominated solutions and

achieve a progress towards the entire Pareto front, and (iii) crowding distance to put emphasis on less crowded solutions for maintaining the diversity of the solutions. Over the years, various extensions of NSGA–II are proposed through the modification of each of these three aspects. These modifications are studied in this section, according to the part of the algorithm they are targeting.

2.1.1 NSGA–II with modified genetic operators

Genetic operators in NSGA–II are in charge of generating new solutions (e.g. mutation and recombination) and preserving the fittest individuals (i.e. selection and elitism operators). These operators play a key role in the performance of NSGA–II, thus their modification may change the performance significantly. Deb *et al.* (2002b) reported improvement in convergence of NSGA–II by proposing a Parent–centric based recombination (PCX) operator, which uses more than two parents to create one descendant. Iorio and Li (2004) replaced the real–coded crossover and mutation with a differential evolution scheme that uses the difference between solutions to perturb the population. Another modified operator is the jumping gene operator, in which a randomly selected part of the chromosome is replaced by a new randomly generated set of binary numbers (Agarwal and Gupta, 2008; Kasat and Gupta, 2003). To improve the diversity of the solutions, Yijie and Gongzhang (2008) forced the crossover operator to be more likely performed on genes located far from each other within the design space. Murugana *et al.* (2009) recommended a controlled elitist and a virtual mapping procedure for the same purpose. Methods with modified operators have been tested on a number of test problems, for which, in most cases, mixed improved convergence rate and spread of solutions were reported. Not a noticeable improvement in convergence and spread was reported for a general test problem.

Not only the genetic operators, but also the structure of the chromosome and the population has been altered to improve the performance of NSGA–II. Maneeratana *et al.* (2005) and Praveen Kumar *et al.* (2007) proposed a co–evolution of multiple species by splitting the population into a number of sub–populations or species that share a gene similarity. Hierarchical genotype encoding proposed by Kumar *et al.* (2009) is another modification. Tran (2005) suggested running multiple populations with different population sizes simultaneously, in order to automatically select the population size. In some cases, an improvement was reported in the quality of the solutions; however, additional computation due to working with multiple populations is the main drawback of these methods.

2.1.2 NSGA–II with modified non–dominated sorting

Altering the non–dominated sorting procedure affects the progress towards the Pareto–optimal front. Using a more severe definition for domination may speed up the search process (Deb, 2008). Examples of efforts in modifying the non–dominated sorting procedure in NSGA–II are ε–MOEA, a proper domination, fuzzy domination (Deb, 2008) and quick sort (Zheng *et al.*, 2004). The aims of these improvements were to reduce the time to converge to the Pareto front and to reduce the computational complexity, but the subsequent penalty is that part of the real Pareto front may be excluded.

2.1.3 NSGA–II with modified crowding distance

The diversity preservation procedure may also be altered in order to achieve a better distribution of solutions or to emphasise specific part of the Pareto front. Clustered NSGA–II (Deb, 2008) claimed to find a better distribution of points by replacing the crowding distance operator with a K–mean clustering approach. Li *et al.* (2008) tried to achieve the same goal by replacing the crowding distance operator with an

algorithm based on minimum spanning tree (MST). In another attempt, the selection procedure was modified to accept a point to be in the new population only if its distance to all current points in the new population is greater than a user–defined value (Ghomsheh *et al.*, 2007). Other similar attempts are reviewed by Deb (2008), including: projection–based diversity preservation, niching, Omni–optimizer, extreme–point preference, and other methods. Although these methods generally require a longer computational time, the modified algorithm can find a better distribution of Pareto optimal solutions than the original NSGA–II. The crowding distance may also be altered in order to find important points of the Pareto front, such as knee points (Deb, 2008), where a small gain in one objective requires a large penalty in at least one of the other objectives.

2.2 NSGA–II in hybrid algorithms

Even an improved version of NSGA–II usually requires a very large number of generations to approach the Pareto front. The population initially moves fast towards the real Pareto front but slows down further into the process and finally approaches the Pareto front only asymptotically (Lahanas *et al.*, 2003). In addition to the low convergence rate, which may not meet the speed requirement for complicated industrial problems, NSGA–II may not efficiently generate a Pareto set that covers the entire true Pareto front within a reasonably low number of generations (Gao *et al.*, 2008). The deficiency is due to the fact that the crossover and mutation operators do not allow intensifying the search sufficiently. One promising approach to increase convergence rate and solution diversity is to hybridize this method with a local search. The local search operator replaces or follows the mutation operator and helps to intensify the search in various areas pointed by the genetic mechanisms. This type of

hybrid algorithm is called a *Memetic Algorithm* (MAs; Moscato, 1989) and they have been shown to be more efficient than a genetic algorithm (Hart and Belew, 1996).

Many hybrid algorithms with different combinations of genetic algorithm and local search methods have been reported in literature. A review of some of these hybrid methods can be found in El–Mihoub et al. (2006) and the cited references in this paper. This section provides a summary of the hybrid multi–objective optimization tools that uses NSGA–II as the global optimizer.

Sequential quadratic programming (SQP) is the most common gradient–based local search method hybridized with NSGA–II. Hu *et al.* (2003) used a SQP algorithm by means of the modified ε–constraint method. The hybrid form was judged to be successful regarding the convergence rate, and not deteriorating in terms of diversity of solutions. Another hybrid form was proposed by Kumar *et al.* (2007), who resort to SQP in order to locally improve one objective in the set of non–dominated solutions obtained. The mixed performance was reported considering the convergence and diversity of the solutions. Gao *et al.* (2008) also proposed a hybrid form, where NSGA–II and SQP run almost independently, but with some exchange of information. The SQP module sought a Pareto set using the weighted sum approach with equal weight for all objectives, while the NSGA–II module generated another set of Pareto points with an even distribution of solutions. Then, the SQP module updated its set of weights by using results from NSGA–II, and NSGA–II used the results from the SQP as elites. The proposed hybrid form was found more efficient than NSGA–II in terms of convergence, particularly in the earlier steps.

SQP requires calculation of the function gradient and the optimum step length in each iteration, which can be costly for a practical engineering problem. To reduce the computational cost, Hernandez–Diaz *et al.* (2008) proposed using the gradient

information only at the beginning of the search process. The steepest descent method was adapted to generate a number of non–dominated points which formed the initial population for NSGA–II. Lahanas *et al.* (2003) used a similar approach but with a different local search called L–BFGS. The proposed methods by Hernandez–Diaz *et al.* (2008) and Lahanas *et al.* (2003) required considerably less computational time but presented a lower performance than the previously mentioned hybrid forms.

One way to avoid time–consuming calculation of gradient information is to opt for a direct search method, which needs no gradient information. An example of hybrid methods with direct local search is PHC–NSGA–II by Bechikh *et al.* (2008) that uses Pareto Hill Climbing (PHC) as a local search. Using a mutation operator, PHC generates several solutions at the neighbourhood of the selected solution. A solution that is not dominated by any other solution in this neighbourhood replaces the original solution. A higher convergence rate was reported compared to the original NSGA–II; however, not a significant improvement in the diversity of the solutions was achieved. Another similar hybrid form is S–MOGLS by Ishibuchi and Narukawa (2004), who used the *r–opt* algorithm as a local search. The r–Opt is a heuristic optimization method that improves the current solution by sequentially replacing one, two or three adjacent genes in the chromosome. Xu *et al.* (2008) also used a similar hybrid form, but the r–Opt was used only during the initialization process. Although faster in convergence; these algorithms do not perform an efficient local search, because of using a heuristic local search method. A more efficient local search is suggested to better exploit the computational resources.

The Nelder–Mead (NM) (Nelder and Mead, 1965) simplex method is the most common direct search method. Effective in producing a rapid initial improvement in the objective function values (Lagarias *et al.*, 1998), NM has been extensively applied

to different single–objective problems. Numerous hybrid forms of this simplex method with genetic algorithm are proposed for solving single–objective optimization problems (e.g.: Yen *et al.*, 1998, Chelouah and Siarry, 2003; Hongfeng *et al.*, 2009); however, application of such a hybrid form for multi–objective optimization is less studied due to the fact that using NM requires aggregating multiple objectives into one single objective. Koduru *et al.* (2005) used the concept of fuzzy dominance to solve this problem and to assign a single measure of fitness to each individual considering multiple objectives. In their proposed hybrid algorithm, called FSGA, K–means clustering was used to break up the population into closely spaced clusters. Some sufficiently populated clusters were picked, from which a simplex was randomly selected for the application of the NM method. The major difficulty associated with this method is the computational time required for the clustering procedure, which requires calculation of the distance matrix among the individuals in the current population. The second problem is the shape of the initial simplex, which may be poorly scaled, for the simplex is randomly selected and no control can be applied to its geometry. Finally, in this method NM is being called in every generation of NAGA–II. Since the difference between the two subsequent populations is usually very small, a local search may not lead to a considerable improvement when it is applied every generation.

Martinez and Coello (2008) alleviated the last problem by performing the local search only after a certain number of generations by NSGA–II. In their proposed method, called NSS–GA, solutions with the best value for each objective were selected to be locally improved with respect to that objective. If the problem has only one objective, golden section search was used as the local search, but if more than one objective was involved NM, was used as the local search. To avoid the tendency

toward the anchor points, using an aggregating objective, which minimizes a weighted sum of all the objectives, was suggested. Although effective, the use of one aggregating objective is not sufficient to compensate the tendency toward the anchor points and provide an even distribution of solutions on the Pareto front. In addition, since the efficiency of the NM method is strongly dependent on the number of design variables (Han and Neumann, 2004), this method is not efficient for problems with a large number of design variables.

The hybrid form presented in this paper also uses NM method as the local search due to its good convergence rate and simplicity of the algorithm. NM method typically requires only one or two function evaluations per iteration (except in shrink, which is rare in practice), while many other direct search methods that use a finite–difference approximation of the function gradient, such as *derivative free conjugate direction method*, *model–based methods*, *implicit filtering*, etc. (Nocedal and Wright, 2006) require O(n) or $O(n^2)$ function evaluations per iteration. This is very important in applications where the function evaluation is expensive or time–consuming. Pattern search methods, such as *coordinate search* (Nocedal and Wright, 2006), *Hooke and Jeeves method* (Hooke and Jeeves, 1961) or *Method of Rosenbroke* (Nocedal and Wright, 2006), also require more function evaluations than NM method, because of the line search that must be performed at each iteration.

In the next section, we propose a hybrid algorithm that improves the performance of NSGA–II and does not suffer from the shortcomings of the previous hybrid algorithms. The main features and benefits of the proposed hybrid algorithm include: 1) the use of a multi–objective form of NM method as a local search, 2) not having a tendency toward a certain region(s) of the Pareto front, 3) using the full benefit of the local search by its proper initialization, 4) maintaining the efficiency of

the local search even for high–dimensional problems, 5) maintaining the simplicity of NSGA–II, 6) not requiring additional user–defined parameters that need prior insight into the problem, and, finally, 7) using the same building blocks as NSGA–II and preserving the similar modular aspect of this popular method.

3 Non–dominated Sorting Hybrid Algorithm (NSHA)

The hybrid algorithm proposed in this paper integrates the Nelder–Mead (NM) simplex method into NSGA–II in order to improve the quality of the solutions and to accelerate the advancement of the non–dominated front toward the true Pareto front. The non–domination rank, which is assigned by the same sorting algorithm as the one used in NSGA–II, is considered as the objective to be minimized. The local search, which is performed after a certain number of generations by NSGA–II, is applied only to a part of the best individuals of the current population. In addition, rather than the whole set of design variables, only a random subset of design variables is considered for the local search in order to maintain the high performance of the local search for high–dimensional problems. In order to avoid a poorly scaled initial simplex, the NM method is initialized around the selected solution by a regular hyper–polygon. Next subsection describes the main optimization loop of NSHA in more details.

3.1 The main optimization loop

As shown in Figure 2, NSHA starts with a randomly generated population with a user–defined size. Using this initial population, only a few generations are proceeded by NSGA–II. The number of generations proceeded by NSGA–II before calling the local search is defined by the number of non–dominated solutions in the current population. The local search will start only after all the individuals of the current population are located at the first non–domination front.

When the local search is called, a subset of α_l percent of the current population is selected. The selected solutions are improved by the local search algorithm, called non–dominated sorting Nelder–Mead (NSNM), explained in the next section. Only a randomly selected subset of three to five design variables are used in the local optimization algorithm, all other variables are kept constant at their current value. The reduced number of design variables helps to increase the efficiency of the local search algorithm. For each selected solution, the local search generates an initial simplex, a regular hyper–polygon with a predefined size, a, where the selected solution is located at one of its vertices. The local optimization process is terminated when either a user–defined maximum number of function evaluations, $nf_{l,\max}$, is reached or any other stopping criteria indicating convergence to a local optimum is satisfied. The selected solutions in the current population are then replaced by the improved solutions found by the local search, creating a locally improved population.

The locally improved population is used as an initial population for the next few generations by NSGA–II. The number of generations performed by NSGA–II before calling the local search is defined by the maximum number of function evaluations allowed for NSGA–II. This parameter called $nf_{g,\max}$ is defined by the user at the onset of the process. The values for $nf_{l,\max}$ and $nf_{g,\max}$ specify the share of NSGA–II and NSNM in the optimization process. It also specifies the frequency and duration for which each algorithm is performed. The value of $nf_{g,\max}$ should be selected sufficiently large to give NSGA–II the chance to improve the population before the next local search is called. On the other hand, $nf_{l,\max}$ must be selected large enough to allow convergence of the local search. The main optimization loop is

repeated until either the total maximum number of function evaluations, nf_t , or any other user–defined stopping criterion is reached.

3.2 The local search algorithm (NSNM)

The local optimization method used in NSHA is called non–dominated sorting Nelder–Mead (NSNM) method. NSNM (shown in Figure 3) differs from the original NM method in the number of objective functions it can handle. NSNM deals with multiple objectives by using the non–domination rank as a single objective to be optimized. To sort the points within the simplex, the non–domination sorting procedure of NSGA–II is adapted. As such, the most dominated point (i.e. the worst solution) within the simplex is reflected with respect to the centroid of the other points.

Crowding distance may not be used to rank the solutions located at the same non–domination front, because this parameter cannot be defined for more than one of the points within the simplex, which is only possible when all the points within the simplex lie on the same non–domination front. Here, solutions in the same non–domination front are ranked by preserving their original order in the simplex. This method of ranking was found to be efficient in reaching a diverse set of solutions on the Pareto front when it is performed several times from random initial points. The problem of partially retrieving the Pareto front, reported in some other hybrid methods (e.g. Martinez and Coello, 2008), is not encountered with NSNM.

The local optimization algorithm is terminated if one of the following stopping criteria is met: all of the points within the simplex are located at the same non–domination front, the simplex size becomes smaller than a pre–defined value, or a user–defined maximum number of function evaluations is reached.

3.3 Selective use of design variables

Han and Neumann (2004), who studied the effect of dimensionality on the performance of the Nelder–Mead method, showed that the performance of NM deteriorates when number of design variables increases. The NM method is not a good choice for design problems with more than approximately ten design variables, while NSGA–II can handle problems with more design variables. To achieve the best performance from NSHA for high–dimensional problems, the number of design variables involved in the local search is limited to five. If the optimization problem has more than five design variables, only a random subset of three to five design variables are used during the local search. All other variables are kept constant at their current value. Numerical results presented in section 4 show the effectiveness of this approach.

3.4 Constraint handling method

Both the NM method and GAs were originally proposed for solving unconstrained optimization problems; however, for most practical problems, the design variables are bounded in a specified range, expressed as box constraints. If a design variable assumes a value out of the specified range, its value is forced to take that of the upper or lower limit defined for the corresponding variable.

Other constraints, generally referred to as non–linear constraints, are formulated as inequality constraints that must be greater than or equal to zero. If there is any equality constraint involved, it may be taken into account by reformulating the objective function (if a closed form expression is available), or by expressing the equality constraint as two inequality constraints. In order to handle these inequality constraints, the concept of constrained domination (Deb, 2008) is applied to the sorting algorithm. In this method, all infeasible solutions assume a non–domination rank higher than the last feasible solution within the population. A crowding distance

is assigned to the feasible solutions, while a value equal to the sum of all violated constraints is assigned to the infeasible solutions. Since the constraints are formulated as inequality equations which must be positive or zero, the sum of the violated non–linear constraints is a negative value that shows the extent of the constraint violation. Solutions lying at a certain non–domination front are sorted in descending order either by the crowding distance if the solutions are feasible, or by the extent of constraint violation if the solutions are infeasible.

In contrast to the projection method that projects infeasible solutions to the boundary of the feasible region (Ghiasi *et al.*, 2008) and reshapes the simplex, this constraint handling method avoids infeasible solutions without affecting the simplex shape. Therefore, reaching a poorly scaled simplex, which significantly deteriorates the performance of NM method, is less probable as opposed to what can often take place with the projection method. The other advantage of this constraint handling method is that no additional function evaluations are required. A more elaborate algorithm that also resorts to non–domination ranking for infeasible solutions can be used; however, Deb *et al.* (2002a) showed that the procedure explained in the previous paragraph is generally more effective.

4 Mathematical Test Problems

NSHA is here applied to a set of constrained and unconstrained test problems from the literature, and its performance is compared with the real coded NSGA–II, considering two performance measures. This section describes these performance metrics, the test problems and the numerical results.

4.1 Performance measures

As defined by Zitzler *et al.* (2000), three factors should be examined to assess the performance of a multi–objective optimization algorithm. 1) *Convergence*, the solutions should be as close as possible to the true Pareto front. 2) *Spread*, the distribution of the solutions should be uniform along the Pareto front. 3) *Scope*, the whole extent of the Pareto front should be captured.

Usually performance parameters are classified in three categories (Deb *et al.*, 2001): metrics that measure the convergence, metrics that measure the spread, and those evaluating both convergence and spread. In this research we select the two performance metrics used by Deb *et al.* (2002a) to compare NSGA–II with other evolutionary multi–objective methods. These metrics measure the extent of achieving the first two goals. Finally, to compare the scope of the solutions, visual inspection of the solutions is used.

The first metric, γ, measures the extent of convergence to a known set of Pareto–optimal solutions, measured by the average of the minimum distance of the solutions from the Pareto front. To determine the minimum distance from the Pareto front, a grid of uniformly distributed points on the Pareto front is generated. The minimum distance of a solution from one of the points in this grid is used as the minimum distance from the Pareto front. The convergence metric γ is defined as:

$$\gamma = \frac{1}{p}\sum_{i=1}^{p} d_i; \;\; d_i = \min_j \left\| \vec{f}_i - \vec{f}_j^{\,*} \right\| \qquad (3)$$

In this expression, $\| \|$ shows the Euclidian distance between the two points in the criterion space. p in this equation is equal to the number of points in the population. Figure 4 (a) illustrates this metric for a bi–criterion problem.

The second metric, Δ, provides information about the extent of spread achieved by the solutions. As mentioned, the set of solutions is desired to span the

entire Pareto front and be uniformly distributed along it. The following equation is used to calculate this metric for a bi–criterion problem:

$$\Delta = \frac{l_0 + l_p + \sum_{i=1}^{p-1} (l_i - \bar{l})}{l_0 + l_p + (p-1)\bar{l}} \qquad (4)$$

$$\bar{l} = \frac{1}{p-1} \sum_{i=1}^{p-1} l_i \qquad (5)$$

where p shows the number of points in the population. l_0 and l_p in the above equation are respectively the Euclidian distances between the extreme solutions and the anchor points, as shown in Figure 4(b), and l_i is the Euclidian distance between two solutions. This metric is zero if the solutions are equally spaced and include both anchor points. Therefore, this metric not only measures the spread of the solutions, but also provides some information about their scope. In the next section, these performance metrics and visual inspection of the results are used to compare NSHA with NSGA–II.

4.2 Test problems

Several test problems from the literature are selected to compare the performance of NSHA to NSGA–II. Table 1 shows the list of unconstrained problems used for this purpose. The first six test problems have two objectives with known, continuous Pareto fronts. MOP4 has a discontinuous 2D Pareto front, whose closed–form solution is not available to the authors. In addition, since the Pareto front of DTZL1 is three–dimensional, the performance metrics may not be properly calculated. Therefore, for MOP4 and DTZL1, a visual comparison of the results is discussed.

A set of constrained test problems are also used to assess the performance of NSHA on this type of problem. Table 2 shows the list of selected constrained test problems. The first test problem is chosen due to its continuous 2D Pareto front, which enables using the performance metrics for a precise comparison of the two optimization algorithms. The second test problem is more complex because six design

variables and six constraints need to be considered. The theoretical Pareto optimal front of this test problem consists of discontinuous broken–lines.

A constant population size of 100 individuals is used for each test case and the optimization process is conducted for up to 25,000 function evaluations, except for DTZL1, for which 100,000 function evaluations are performed. The maximum number of points participating in the local improvement process is set to be 20% of the population size. Size of the initial simplex for the local optimization algorithm is set to be 10% of the smallest edge of the hyper–cube surrounding the design domain, while the minimum simplex size at which the local search is terminated is chosen to be 0.1% of this value. The maximum number of function evaluations for the local search, $nf_{l,\max}$, and for NSGA–II, $nf_{g,\max}$, are respectively limited to 100 and 2000 function evaluations. These choices provide almost equal contribution of the local search (i.e. 20×100=2000 function evaluations) and NSGA–II. This ratio is kept constant in all test cases solved in this paper (including the composite design problem); half of the total number of function evaluations is used by NSGA–II and half by NSNM. Due to the stochastic nature of the algorithm, each test problem is solved several times and the averaged performance metrics are compared. The performance parameters are determined each time that the algorithm switches from the local search to NSGA–II or vice versa.

4.3 Impact of the selective use of design variables

In this section, it is shown that selecting a subset of design variables for the local search improves the convergence rate of the hybrid algorithm. The second test problem, ZDT1, with 30 design variables is chosen for this purpose. Three different cases are examined. In the first case only NSGA–II is used. In the two other cases the hybrid algorithm is used for optimization. In the second case (NSHA–all) the whole

set of design variables is used in the local search, whereas, in the third case (NSHA), only a random subset of three to five design variables are selected and used in the local optimization process.

The convergence measure, γ, is plotted versus the number of function evaluations performed by each method in Figure 5. NSHA is shown to achieve a better convergence (lower γ) than the two other methods, all along the optimization process. When all 30 design variables of the ZDT1 are used during the local search (NSHA–All), the convergence rate of the local search is very slow, for the performance of NM is dependent on the dimension of the problem. The local search is unable to locally improve the solutions in the current population; therefore, the function evaluations performed during the local search are ineffective.

4.4 Unconstrained test problems

This section reports the results of applying the NSHA on selected unconstrained mathematical test problems shown in Table 1. The results are reported for each function, individually.

The performance metrics averaged during ten trials of NSGA–II and NSHA on FON test function is shown in Figure 6. Since this test problem has only three design variables, both algorithms are expected to find solutions very close to the real optima after performing 25,000 function evaluations. This is a fairly easy problem without irregularity in function shape, therefore the performance of the two algorithms was similar and the solutions found by both algorithms are close to the real Pareto front of the problem.

Similarly, Figure 7 shows the performance parameters averaged over ten trials of solving the ZDT1 test problem. This test problem is more complex than the previous one due to the larger number of design variables. This figure shows that

regardless of the number of function evaluations performed, NSHA achieved a better convergence than NSGA–II after the local search is activated in NSHA. For instance, the convergence metric achieved by NSGA–II after 12,000 function evaluations is achieved by NSHA with only 7,000 function evaluations. As seen in this figure, NSHA continues the progress toward the Pareto front even after 20,000 function evaluations, while NSAG–II does not achieve a noticeable progress after around 12,000 function evaluations. Both algorithms performed similarly regarding the spread of the solutions, because both methods use the same technique, crowding–comparison, to preserve the spread of the solutions.

The visual comparison of the solutions in Figure 8 shows that the two algorithms also performed similarly regarding the scope of the solutions, with slightly better performance for NSHA. This figure shows the solutions found in one of the trials whose performance is close to the average performance of the ten trials.

Figure 9 to Figure ***12***, respectively, show the performance parameters for ZDT2, ZDT3, ZDT4, and ZDT6 test problems, averaged over ten runs of NSHA and NSGA–II. In Figure 9 to Figure 11, except at the beginning of the optimization process, the convergence metric, γ, achieved by NSHA after certain number of function evaluations is smaller than the one achieved by NSGA–II with the same number of function evaluations. This metric shows that NSHA could obtain solutions closer to the Pareto front than the one obtained by NSGA–II. The improvement observed confirms that the integrated local search is efficiently incorporated into the optimization process and increases the rate of convergence toward the Pareto front. For ZDT6 test function, for which the results are shown in Figure 12, none of the two algorithms clearly outperformed the other; however, looking at the numerical values of the convergence metric, γ, shows that both algorithms could obtain solutions very

close to the Pareto front. The corresponding test function lacks the trigonometric term and thus includes fewer local minima; therefore the simplex local optimizer can make a significant improvement.

NSGA–II showed higher rate of convergence in the first few iterations on ZDT3 in Figure 10. The reason is that this test problem has numerous local optima, which make the local search inefficient at the early iterations where most of the GA's solutions are far from the Pareto front. For this test problem, the authors expect an improvement in the performance of NSHA if the activation of the local search is postponed. As the population moves closer to the Pareto front, the local search becomes more effective and its contribution to the search becomes more evident. In all test problems presented in these figures, NSHA yielded solutions closer to the Pareto front than those provided by NSGA–II at the end of the process.

In all the six unconstrained test problems, the two algorithms found the same overall spread of solutions, because the crowding–comparison operator, which is responsible for achieving an even spread of solutions, is similar in both algorithms. Since during the local search no crowding distance is calculated, the distribution of the solutions may decline, but it is improved as soon as a few generations of NSGA–II are performed. In order to avoid a poor spread of solutions in the final set of solutions, the recommendation is to terminate the optimization process with a few generations of NSGA–II.

In FON and all the ZDT test problems, the solutions found by both algorithms include the entire Pareto front and, as it was shown for ZDT1 in Figure 8, the difference between the two algorithms is very small. The reason is that the Pareto fronts of these test problems are smooth continuous curves, with no discontinuity or

irregularity. Effect of the local search in improving the scope of the solution is more evident in cases with discontinuous Pareto front and constrained problems.

In order to demonstrate the performance of NSHA on problems with a piecewise Pareto front, both algorithms are applied to the MOP4 test problem. As explained previously, since the performance metrics are not suited for problems with piecewise Pareto fronts, the results for MOP4 are visually compared in Figure 13. The non–dominated solutions obtained by the two algorithms are illustrated for three levels of total function evaluations, namely 2500, 5000 and 25000.

Both algorithms achieved almost the same set of final solutions after 25,000 function evaluations; however, NSHA reaches this solution faster than NSGA–II. The faster convergence is confirmed when the solutions found after 2500 function evaluations are compared; the solutions of NSHA are closer to the real Pareto front and, compared to the solutions of NSGA–II, they cover a wider portion of the Pareto front.

The last unconstrained test problem solved here is DTZL1, which is selected to demonstrate and compare the performance of NSHA on problems with more than two objectives. The set of solutions obtained by the two algorithms after 5000 and 100,000 function evaluations is illustrated in Figure 14, which shows that NSHA has a higher convergence rate than NSGA–II, since the solutions obtained by NSHA are closer to the origin and dominating the ones obtained by NSGA–II. At 5000 iterations, the spread of the solutions achieved by NSGA–II is better than the one by NSHA; however, the quality of the solutions by NSHA is better. Further into the process, the spread of the solution is improved and becomes closer to the uniform distribution. Generally, both algorithms perform poorly, as also previously reported by Deb (2008), who suggested improving the solutions by using ε–domination in NSGA–II.

Since NSHA uses the same non–domination sorting algorithm, this modification can also be applied to NSHA to improve the distribution of the solutions for this test problem.

4.5 Constrained test problems

Two constrained test problems are studied in this section to compare the performance of NSHA and NSGA–II. The first test problem is CONSTR (Table 2), whose Pareto front and the feasible criterion space are shown in Figure 15. The averaged performance parameters over three trials of NSGA–II and NSHA are shown in Figure 16. The two algorithms performed similarly, although the performance metrics were less stable for NSHA. The significant difference between the two algorithms is observed in the scope of solutions. For a number of function evaluations less than approximately 5,000, NSHA performed better than NSGA–II and provided a wider scope of solutions. The solutions obtained by NSHA and NSGA–II after 2,000 and 15,000 function evaluations are compared in Figure 15.

The other selected constrained test problem is COK (Table 2), which has a piecewise broken–line Pareto front shown in Figure 17. The complexity of this problem is higher because six design variables and six constraints need to be considered. Since the closed–form representation of the Pareto front is not available, we decided to run both algorithms for up to 100,000 function evaluations and use the best solutions found to calculate the convergence metric γ. Figure 17 shows the approximate Pareto front and the average convergence metric, γ, averaged over ten trials for each algorithm.

Regarding the convergence metric, shown in Figure 17, except for the last few generations, a slightly better convergence is achieved by NSGA–II; however, a remarkable advantage of NSHA can be reported if the solutions are inspected as

shown in Figure 18. Even though the solutions obtained by NSGA–II are very close to the real Pareto front, they do not cover the entire Pareto front. Solutions found by NSHA cover a wider portion of the front.

Figure 18 shows the progress of the solutions toward the real Pareto front in one of the representative trials. Whereas NSGA–II could not retrieve a wide portion of the Pareto front in a number of trials, NSHA was successful in finding the entire Pareto front in all ten trials. This confirms that the local search algorithm has a strong influence in intensifying the search capability of the algorithm. The local search is particularly beneficial in problems with irregular Pareto fronts, because the local search enables the algorithm to reach solutions close to the knee points and discontinuities, which are often located at the corners of the feasible region. Such solutions cannot be easily reached by pure evolutionary operators.

4.6 Sensitivity to the user–defined parameters

One important advantage of NSGA–II is the small number of parameters that must be defined by the user at the onset of the optimization process. Introduced in this work, NSHA uses the same modules used in NSGA–II (e.g. the non–domination sorting algorithm, the crowding distance calculation procedure), thereby it also requires definition of a limited number of user–defined parameters. The additional parameters that must be defined by the user include the frequency of calling the local search and the number of individuals participating in the local search. In this section the test problem ZDT1 is used to show the effect of these two parameters on the performance of NSHA.

To examine the sensitivity of the convergence rate to the frequency of calling the local search, the convergence metric γ is plotted in Figure 19. Here, the local search is called every 5, 10, or 20 generations. On the other hand, Figure 20 plots the

convergence metric for different numbers of solutions participating in the local search, namely 10, 20 and 40 percent of the current population. The results presented in these figures confirm that the user–defined parameters have a minor effect on the convergence of NSHA. For instance, calling the local search every 5 or 20 generations does not significantly change the convergence rate (Figure 19). In addition, changing the number of solutions participating in the local search from 10% to 40% of the population also does not yield a major difference in convergence rate (Figure 20). Therefore, although problem–dependent, assigning a reasonable value to these two parameters is not a difficult task and does not require a significant insight into the problem. For a general problem, we suggest calling the local search every 10 generations and assigning the percentage of 20 to the number of solutions participating in the local search.

5 Design of a Composite Part

Design and manufacturing of a composite part is used in this section as an example to demonstrate the ability of the proposed hybrid algorithm in solving a practical engineering problem. Composite materials are well–known for having many design variables, which gives the designer more opportunities to tailor the material for the intended application; however it requires solving a more complex design problem. Different optimization algorithms have been used to solve the corresponding design optimization problem (Ghiasi *et al.*, 2009). In this section, the application of NSHA in solving a multi–objective design problem, which takes into account both manufacturing and structural parameters, is demonstrated.

5.1 Composite test problem

The part studied in this section is a carbon fiber bicycle stem, part of a bicycle that connects the handlebar to the fork, as shown in Figure 21. Bladder assisted resin transfer moulding (RTM) is used as the manufacturing method, because of the small size of the part, its complicated geometry with hollow sections, high level of surface finish and fast production cycle required. In this manufacturing method, first, a dry preform consisted of four layers of continuous braided carbon fiber sleeves is placed around an inflatable bladder and is positioned inside a solid airtight mould, as shown in Figure 22. After inflating the bladder, an injection pump is used to push the resin through the fibers which are compressed between the bladder and the mould surface. The air exits through the strategically placed vents as the resin front advances. The part is ready to be demoulded when the cure procedure is completed. This manufacturing process results in a strong interconnection between the structural and manufacturing parameters; therefore, a multi–objective optimization method is required to solve the corresponding design problem.

5.2 Optimization problem

The corresponding optimization problem consists in finding the minimum weight and mould filling time and the maximum strength or minimum inverse Tsai–Wu strength ratio. The design variables consist of four braid diameters, injection pressure and bladder pressure. The optimization problem is formulated as follows:

$$\min_{d_i, P_{inj}, P_{bladder}} \begin{cases} W(d_i, P_{inj}, P_{bladder}) \\ T(d_i, P_{inj}, P_{bladder}) \; ; \; i = 1,...,4 \\ S((d_i, P_{inj}, P_{bladder}) \end{cases}$$

$$s.t. \begin{cases} 38.1mm < d_i < 63.5mm; \; i = 1,...,4 \\ 100kPa < P_{inj} < 450kPa \\ 150kPa < P_{bladder} < 500kPa \\ P_{bladder} - P_{inj} \geq 50kPa \end{cases} \qquad (6)$$

where W, T and S in this equation represent weight, filling time and the inverse Tsai–Wu strength ratio, respectively. d_i is the braid diameter, and p_{inj} and $P_{bladder}$ stand for injection and bladder pressures. The flowchart in Figure 23 shows the interconnection between the design variables and the objectives and the method to calculate each parameter. Function evaluation process here consists of a structural analysis and a flow simulation. A finite element analysis in ANSYS® is employed for structural analysis, while the flow simulation is performed using a MATLAB® code developed by the first author. The function evaluation process for this problem is time consuming and takes about 100 seconds on a computer with Intel–Pentium4, 3.2GHZ, 2GB RAM. Therefore, the number of function evaluations must be kept as low as possible. As a result, the higher convergence rate of the hybrid optimization algorithm introduced in this paper is an important asset for situations similar to this test problem.

5.3 Results and discussion

NSHA is applied to the stem design problem using a population of 40 individuals. The optimization process is terminated after 5,000 function evaluations. Figure 24 shows the final solutions in the 3D criterion space and in the pair–wise 2D spaces. This figure gives insight into the problem by presenting the achievable values for the objectives. For instance, part (c) in this figure shows that there is no trade–off between strength and filling time, thus one may find a solution that has the minimum production time and the maximum strength at the same time; however, simultaneously achieving the maximum strength and minimum weight (Figure 24–b) or minimum weight and minimum filling time (Figure 24–d) is impossible. In addition to the presence and significant effect of the coupling terms, this figure also provides the numerical values for the achievable objectives; for instance, part (b) in this figure

shows that for a part which weights 65 grams, the minimum achievable inverse Tsai–Wu strength ratio is 0.78.

In addition, having an image of the Pareto front of the problem helps finding the critical design points such as knee points or points of discontinuity in the Pareto front. For the stem design problem, Figure 24(b) shows a knee point, marked by letter (A). This point shows that decreasing the inverse Tsai–Wu strength ratio from 1.1 to 0.78 is associated with a smaller penalty in weight than improving the strength of the part beyond this value.

The solid circle shown in Figure 24 part (b) to (c) is the experimental design found in our laboratory after months of trial and error (Thuoin, 2004). This point is very close to the Pareto front regarding the weight and filling time (part (d) in this figure); however, it is not close to the Pareto front regarding the trade–off between weight and strength. The results found in this research can be used to find a solution better than the one found by experiments, while saving the large amount of time usually spent on designing a part by experiment.

Figure 24 shows a fairly even distribution of the solutions within the criterion space; however, such good spread may not be observed within the design space. The spread of the solutions within the design space is not related to the performance of the optimization algorithm, but to the non–linearity of the objectives and constraints that map the design space to the criterion space. Since the position of a solution within the design space shows the values for the design variables, it provides a physical understanding of the problem from the engineering point of view. Therefore, the spread of the solutions within the design space is also studied for this test case.

In order to illustrate the six–dimensional design space of the stem problem, the design variables of the same nature are collectively considered as shown in Figure 25

and Figure ***26***. The best design with respect to each objective (i.e. an anchor point) is shown with solid symbols, while the other solutions are shown with non–solid circles. A tendency toward the solution with the best strength is observed among the results, while such a tendency does not exist toward the best design with respect to the filling time. The distribution of the solutions shows that the coupled designs are dominated by the structural parameters, and a coupled design is closer to the best structural design than the best design for manufacturing. Note that the proximity to the structural design is with respect to the design variables; therefore, it is not in conflict with the reported strong coupling in performances. The domination by structural design was also previously reported by Henderson *et al.* (1999) for a blade–stiffened composite panel.

To summarize the results of the practical design problem, the following three points are noted: 1) the trade–off between structural performance and manufacturing concerns can be captured and qualified using this multi–objective optimization method. 2) The solutions obtained by this method can help distinguishing the feasible combination of performances and finding the critical designs. 3) A coupled design with a reasonable combination of structural and manufacturing performances can be achieved by making only small changes in the best structural design.

6 Conclusions

An efficient multi–objective optimization algorithm is required to solve multi–objective design problems. Evolutionary algorithms are a popular tool for this purpose, because of their ability in achieving more than one solution in a single run. Among these methods, NSGA–II has received a significant attention and popularity; however, like other evolutionary algorithms, it is slow in convergence and usually

needs several function evaluations before a reasonably good set of solutions is reached.

In this paper, NSGA–II was hybridized with a local search algorithm based on the Nelder–Mead (NM) simplex method to improve its convergence rate and quality of the solutions. In order to make NM capable of handling multi–objective problems, we used the non–dominated sorting algorithm used in NSGA–II. The constrained–domination was used to handle the inequality constraints in constrained optimization problems.

The proposed algorithm called non–dominated sorting hybrid algorithm (NSHA) was compared with NSGA–II on eight unconstrained and two constrained test problems from the literature. The performance of the two algorithms was compared using two performance metrics measuring the convergence to the real Pareto front and spread of the solutions. The mathematical test problems presented in this paper showed that the hybrid algorithm significantly increases the convergence rate and the extent of solutions. It was observed that to obtain the same quality of the results, NSHA requires a smaller number of function evaluations than NSGA–II. The local search integrated into the algorithm significantly intensifies the ability of the algorithm in searching the design space and reaching the critical solutions such as knee points or discontinuities in the Pareto front. Therefore, the hybrid algorithm was shown to be able to obtain a wider portion of the Pareto front, when irregularities exist in the Pareto front of the problem. Since the diversity preservation mechanism is only applied within NSGA–II, the similar spread of solutions was observed for both NSHA and NSGA–II.

To demonstrate the capability in handling practical engineering problems, NSHA was applied to the optimum design of a composite bicycle stem. Finite element

analysis and resin flow simulation were required to evaluate the three objectives of this problem, which made the function evaluation process time consuming. The optimum solutions found by NSHA reveals the trade–off among conflicting objectives, the extreme values for each objective, and the critical design points. The results were found close to the solutions found by experiment and trial and error.

By intensifying the search and reducing the computational time, NSHA provides an effective tool that benefits from the robustness of the genetic algorithms and the high convergence rate of the Nelder–Mead method. This optimization tool is suitable for solving complex time consuming multi–objective optimization problems. One of the major advantages of NSHA is that it maintains the simplicity and modular aspect of NSGA–II, while it achieves a wider scope and a higher quality of the solutions. Most modifications proposed in the literature to adapt NSGA–II to special applications, can be applied to NSHA to further improve its performance for a particular application.

7 References

Agarwal, A. and Gupta, S.K., 2008. Jumping gene adaptations of NSGA–II and their use in the multi–objective optimal design of shell and tube heat exchangers. *Chemical Engineering Research and Design*, 86, 123–139.

ANSYS theory reference, 2007. *Release 11.0: documentation for ANSYS*. SAS IP, Inc.

Babu, B.V. and Anbarasu, B., 2005. Multi–objective differential evolution (MODE): An evolutionary algorithm for multi–objective optimization problems. *Proceedings of The Third International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS–2005)*, Singapore, December 13–16.

Bechikh, S., Belgasmi, N., Said, L.B. and Ghédira, K., 2008. PHC–NSGA–II: A novel multi–objective memetic algorithm for continuous optimization. *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008)*, November 3–5, Dayton, Ohio, USA, 180–189.

Bingul Z, 2006. Adaptive genetic algorithms applied to dynamic multiobjective problems. *Applied Soft Computing*, 7, 791–799.

Chelouah R. and Siarry P., 2003. Genetic and Nelder-Mead algorithm hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operation Research*, 148, 335-348.

Coello, C.A., 1999. A comprehensive survey of evolutionary–based multi–objective optimization techniques. *Int J Knowledge and Information Systems*, 1(3), 269–308.

Coello, C.A., 2002. Evolutionary multi-objective optimization: a critical review. In: M. Ehrgott and X. Gandibleux, Editors, *Multiple Criteria Optimization, State of the Art*, Annotated Bibliographic Surveys, Kluwer Academic Publishers.

Deb, K., 2001. *Multiobjective optimization using evolutionary algorithms*, Chichester, UK: John Wiley and Sons Ltd.

Deb, K., 2008. A robust evolutionary framework for multi–objective Optimization. *Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO'08)*, July 12–16, Atlanta, Georgia, USA, 633–640.

Deb, K., Anand, A. and Joshi, D., 2002b, A computationally efficient evolutionary algorithm for real–parameter optimization. *Evolutionary Computation Journal*, 10(4), 371–395.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan T., 2002a. Fast and elitist multi–objective genetic algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2).

El-Mihoub, T.A., Hopgood A.A., Nolle L. and Battersby A., 2006. Hybrid genetic algorithms: a review. *Engineering Letters*, 13(2), 124–137.

Fonseca, C.M. and Fleming, P.J., 1993. Genetic algorithms for multi–objective optimisation: formulation, discussion and generalisation. *Proceedings of the 5th International Conference on Genetic Algorithms* (Ed. Forrest S), Urbana, Illinois, 416–423.

Fonseca, C.M. and Fleming, P.J., 1995. An overview of evolutionary algorithms in multi–objective optimization. *Journal of Evolutionary Computation*, 3, 1–16.

Gao, X., Chen, B., He, X., Qiu, T., Li, J., Wang, C. and Zhang, L., 2008. Multi–objective optimization for the periodic operation of the naphtha pyrolysis process using a new parallel hybrid algorithm combining NSGA–II with SQP. *Computers and Chemical Engineering*, 32, 2801–2811.

Ghiasi, H., Pasini, D. and Lessard, L., 2008. Constrained globalized Nelder–Mead method for simultaneous structural and manufacturing optimization of a composite bracket, *Composite Materials*, 42(7), 717–736.

Ghiasi, H., Pasini, D. and Lessard, L., 2009a. Optimum stacking sequence design of composite materials part I: constant stiffness design. *Composite Structures*, 90(1), 1–11.

Ghiasi, H., Pasini, D. and Lessard, L., 2009b. Pareto frontier for simultaneous structural and manufacturing optimization of a composite part. Accepted by *Structural and Multidisciplinary Optimization, doi: 10.1007/s00158-009-0366-4, Available online Feb 2009.*

Ghomsheh, V., Khanehsar, M.A. and Teshnehlab, M., 2007. Improving the non–dominate sorting genetic algorithm for multi–objective optimization. *International Conference on Computational Intelligence and Security Workshops (CISW 2007)*, Heilongjiang, China, 89–92.

Hajela, P. and Lin, C.Y., 1992. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4, 99–107.

Han, L. and Neumann, M., 2006. Effect of dimensionality on the Nelder–Mead simplex method. *Optimization Methods and Software*, 21(1), 1–16.

Hart, W. and Belew, R., 1996. Optimization with genetic algorithm hybrids that use local search. *Adaptive individuals in evolving populations: Models and*

algorithms, Santa Fe Institute Studies in the Science of Complexity Vol. 26, Addison–Wesley.

Henderson, J.L., Gürdal, Z. and Loos, A.C., 1999. Combined structural and manufacturing optimization of stiffened composite panels. *J. of Aircraft*, 36(1), 246–254.

Hernandez–Diaz, A.G., Coello, C.A., Perez, F., Caballero, R., Molina, J. and Santana–Quintero, L.V., 2008. Seeding the initial population of a multi–objective evolutionary algorithm using gradient–based information, *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 1617–1624.

Hongfeng X., Guanzheng T. and Jingui H., Large scale function optimization or high-dimension function optimization in large using simplex-based genetic algorithm, GEC'09, June 12–14, 2009, Shanghai, China.

Hooke, R. and Jeeves, T.A., 1961. Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8, 212–229

Horn, J., Nafpliotis, N. and Goldberg, D.E., 1994. A niched Pareto genetic algorithm for multi–objective optimization. *Proceeding of the 1st IEEE Conference on Evolutionary Computation*, 82–87.

Hu, X., Huang, Z. and Wang, Z., 2003. Hybridization of the multi–objective evolutionary algorithms and the gradient–based algorithms. *Proceedings of the IEEE Congress on Evolutionary Computation*, 870–877.

Iorio, A.W. and Li, X., 2004. Solving rotated multi–objective optimization problems using differential evolution, *Proceeding of AI 2004: Advances in Artificial Intelligence*, Springer–Verlag, LNAI, 3339, 861–872.

Ishibuchi, H. and Narukawa, K., 2004. Performance evaluation of simple multi–objective genetic local search algorithms on multi–objective 0/1 Knapsack problems. *Congress on Evolutionary Computation, CEC2004*, 19–23 June, 1, 441– 448.

Kasat, R.B. and Gupta, S.K., 2003. Multi–objective optimization of an industrial fluidized–bed catalytic cracking unit (FCCU) using genetic algorithm with the jumping gene operator. *Comput Chem Eng*, 27(12), 1785–1800.

Koduru, P., Das, S., Welch, S., LRoe, J., and Lopez–Dee, Z.P., 2005. A Co–evolutionary hybrid algorithm for multi–objective optimization of gene regulatory network models. *2005 Genetic and Evolutionary Computation Conference (GECCO'05)*, June 25–29, Washington, DC, USA, 393–399.

Kumar, A., Sharma, D. and Deb, K., 2007. A hybrid multi–objective optimization procedure using PCX based NSGA–II and sequential quadratic programming. *Special Session & Competition on Performance Assessment of Multi–Objective Optimization Algorithms, CEC–07*, Singapore, September 25–28, 2007.

Kumar, R., Izui, K., Yoshimura, M. and Nishiwaki, S., 2009. Multi–objective hierarchical genetic algorithms for multi level redundancy allocation optimization. *Reliability Engineering and System Safety*, 94, 891–904.

Lagarias, J.C., Reeds, J.A., Wright, M.H. and Wright, P.E., 1998. Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM J of Optimtimization*, 9(1), 112–147.

Lahanas, M., Baltas, D. and Zamboglou, N., 2003. A hybrid evolutionary algorithm for multi–objective anatomy–based dose optimization in high–dose–rate brachytherapy. *Physics in Medicine and Biology*, 48, 399–415.

Le Riche, R., Saouab, A. and Breard, J., 2003. Coupled compression RTM and composite layup optimization. *Composite Science and Technology*, 63(15), 2277–2287.

Li, M., Zheng, J. and Wu, J., 2008. Improving NSGA–II algorithm based on minimum spanning tree. *Lecture Notes in Computer Science*, Springer–Verlag Berlin Heidelberg, LNCS 5361, 170–179.

Maneeratana, K., Boonlong, K. and Chaiyaratana, N., 2005. Co–operative co–evolutionary genetic algorithms for multi–objective topology design. *Computer–Aided Design & Applications*, 2(1–4), 487–496.

Martinez, S.Z. and Coello, C.A., 2008. A proposal to hybridize multi–objective evolutionary algorithms with non–gradient mathematical programming techniques. *Lecture Notes in Computer Science (Eds. G. Rudolph et al.)*, Springer–Verlag, Berlin Heidelberg, PPSN X, LNCS 5199, 837–846.

Moscato, P., 1989. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program*, C3P Report 826, 1989.

Murugan, P., Kannan, S. and Baskar, S., 2009. NSGA–II algorithm for multi–objective generation expansion planning problem. *Electric Power Systems Research*, 79, 622–628.

Nelder, J.A. and Mead, R., 1965. Simplex method for function minimization. *Computer Journal*, 7(4), 308–313.

Nocedal, J. and Wright, S.J., 2006. *Numerical Optimization*, Springer series in operations research, Second edition, United State of America: Springer Science+Business Media, LLC.

Park, C.H., Lee, W.I., Han, W.S. and Vautrin, A., 2005. Multiconstraint optimization of composite structures manufactured by resin transfer moulding process. *Composite Material*, 39(4), 347–374.

Praveen Kumar, K., Sharath, S., D'Souza, G.R. and Chandra, S.K., 2007. Memetic NSGA – a multi–objective genetic algorithm for classification of microarray data. *Proceeding of 15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, 18–21 Dec, Guwahati, India, 75–80.

Purshouse, R.C. and Fleming, P.J., 2001. The multi–objective genetic algorithm applied to benchmark problems–an analysis. *Research Report No. 796, Department of Automatic Control and Systems Engineering University of Sheffield*, Sheffield, S1 3JD, UK, 2001.

Sayin, S. and Karabati, S., 1999. A bicriteria approach to the two–machine flow shop scheduling problem. *European Journal of Operational Research,* 113(2), 435–449.

Schaffer, J.D., 1984. *Some experiments in machine learning using vector evaluated genetic algorithms*. Ph.D. Thesis, Vanderbilt University, Nashville, TN, USA.

Srinivas, N. and Deb, K., 1994. Multi–objective function optimization using non–dominated sorting genetic algorithms. *Evolutionary Computation*, 2(3), 221–248.

Thouin M., 2004. Design of a carbon fiber bicycle stem using an internal bladder and resin transfer molding. M.Eng. Thesis, McGill University, QC, Canada.

Tran, K.D., 2005. Elitist Non–Dominated Sorting GA–II (NSGA–II) as a parameter–less multiobjective GA. Presented at *IEEE SoutheastCon 2005*, Fort Lauderdale, Florida, USA.

Xu, H., Fan, W., Wei, T. and Yu, L., 2008. An or–opt NSGA–II algorithm for multiobjective vehicle routing problem with time windows. *4th IEEE*

Conference on Automation Science and Engineering, Key Bridge Marriott, Washington DC, USA, August 23–26, 309–314.

Yen J., Liao, J.C., Lee, B., Randolph, D., 1998. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 28(2): 173–191.

Yijie, S. and Gongzhang, S., 2008. Improved NSGA–II multi–objective genetic algorithm based on hybridization–encouraged mechanism. *Chinese Journal of Aeronautics*, 21, 540–549.

Zheng, J., Ling, C., Shi, Z., Xue, J. and Li, X., 2004. A multi–objective genetic algorithm based on quick sort. *Canadian AI 2004 (Ed. Tawfik AY and Goodwin SD)*, Springer–Verlag Berlin Heidelberg, LNAI 3060, 175–186.

Zitzler, E., Deb, K. and Thiele, L., 2000. Comparison of multi–objective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2), 173–195.

Zitzler, E., Deb, K., Thiele, L., Coello, A.C. and Corne, D., 2001. *Proceedings of the First International Conference on Evolutionary Multi–Criterion Optimization (EMO 2001),* Lecture Notes in Computer Science (LNCS, Vol. 1993). Heidelberg: Springer.

Table 1 Unconstrained test problems used to compare the performance of NSHA and NSGA–*II*

Problem	n	$x_i \in$	Objective functions	Optimal solution	Reference
FON	3	$[-4,4]$	$f_1 = 1-\exp(-\sum_{i=1}^{3}(x_i-\frac{1}{\sqrt{3}})^2)$ $f_2 = 1-\exp(-\sum_{i=1}^{3}(x_i+\frac{1}{\sqrt{3}})^2)$	$x_1 = x_2 = x_3 \in [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]$	Deb *et al.*, 2002a
ZDT1	30	$[0, 1]$	$f_1 = x_1$ $f_2 = g(x)[1-\sqrt{x_1/g(x)}]$ $g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2,...,n$	Deb *et al.*, 2002a
ZDT2	30	$[0, 1]$	$f_1 = x_1$ $f_2 = g(x)[1-(x_1/g(x))^2]$ $g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2,...,n$	Deb *et al.*, 2002a
ZDT3	30	$[0, 1]$	$f_1 = x_1$ $f_2 = g(x)[1-\sqrt{x_1/g(x)}-\frac{x_1}{g(x)}\sin(10\pi x_1)]$ $g(x) = 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2,...,n$	Deb *et al.*, 2002a
ZDT4	10	$x_1 \in [0, 1],$ $x_i \in [-5, 5]$ $i = 2,...,n$	$f_1 = x_1$ $f_2 = g(x)[1-\sqrt{x_1/g(x)}]$ $g(x) = 1+10(n-1)+\sum_{i=2}^{n}(x_i^2-10\cos(4\pi x_i))$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2,...,n$	Deb *et al.*, 2002a
ZDT6	10	$[0, 1]$	$f_1 = x_1$ $f_2 = g(x)[1-(f_1(x)/g(x))^2]$ $g(x) = 1+9[\sum_{i=2}^{n} x_i/(n-1)]^{0.25}$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2,...,n$	Deb *et al.*, 2002a
MOP4	3	$[-5, 5]$	$f_1 = \sum_{i=1}^{n-1}\left(-10\exp(-0.2\sqrt{x_i^2+x_{i+1}^2})\right)$ $f_2 = \sum_{i=1}^{n}\left(\lvert x_i\rvert^{0.8}+5\sin(x_i)^3\right)$	a piecewise curve	Yijie and Gongzhang, 2008
DTZL1	7	$[0, 1]$	$f_1 = \frac{1}{2}x_1x_2g(x)$ $f_2 = \frac{1}{2}x_1(1-x_2)g(x)$ $f_3 = \frac{1}{2}(1-x_1)g(x)$ $g(x) = 1+100[5+\sum_{i=3}^{7}[(x_i-0.5)^2-\cos(20\pi(x_i-0.5))]$	A linear optimal front	Deb *et al.*, 2001

Table 2 Constrained test problems used to compare the performance of NSHA and NSGA–II

Problem	n	$x_i \in$	Objective functions	Constraints	Reference
CONSTR	2	$x_1 \in [0.1, 1]$ $x_2 \in [0,5]$	$f_1 = x_1$ $f_2 = (1 + x_2)/x_1$	$g_1(x) = 9x_1 + x_2 - 6 \geq 0$ $g_2(x) = 9x_1 - x_2 - 1 \geq 0$	Deb *et al.*, 2002a
COK	6	$x_1, x_2, x_6 \in [0, 5]$ $x_3, x_5 \in [1, 5]$ $x_4 \in [0, 6]$	$f_1(x) = -25(x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2$ $f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$	$g_1(x) = x_1 + x_2 - 2 \geq 0$ $g_2(x) = 6 - x_1 - x_2 \geq 0$ $g_3(x) = 2 + x_1 - x_2 \geq 0$ $g_4(x) = 2 - x_1 + 3x_2 \geq 0$ $g_5(x) = 4 - (x_3 - 3)^2 - x_4 \geq 0$ $g_6(x) = (x_5 - 3)^2 + x_6 - 4 \geq 0$	Yijie and Gongzhang, 2008

Figure 1 Crowding distance calculation for the solutions located at the same non–domination front

Figure 2 Pseudo code of the main loop in NSHA

Figure 3 Pseudo code of the local simplex optimizer NSNM

Figure 4 (a) Convergence metric γ, (b) diversity metric, Δ (Deb et al., 2002a)

Figure 5 Convergence measure, γ, versus number of function evaluations performed when NSGA–II, NSHA, and NSHA–All are applied to the ZDT1 test problem. The results are averaged over ten trials with random initial populations.

Figure 6 Performance parameters, γ and Δ for FON, Averaged over ten trials

Figure 7 Performance parameters, γ and Δ for ZDT1, averaged over ten trials

Figure 8 Solutions close to the two anchor points, found by NSHA and NSGA–II after 25,000 function evaluations for ZDT1

Figure 9 Average performance parameters, γ and Δ for ZDT2 test function after ten trials

Figure 10 Average performance parameters, γ and Δ for ZDT3 after ten trials

Figure 11 Average performance parameters, γ and Δ for ZDT4 test function after ten trials

Figure 12 Average performance parameters, γ and Δ for ZDT6 test function after ten trials

Figure 13 Non–dominated solutions obtained by NSGA–II and NSHA for MOP4 test problem, the solid line shows the real Pareto front

Figure 14 Non–dominated solutions obtained by NSGA–II (left) and NSHA (right) for DTZL1 test function after performing 5000 (top) and 100,000 (bottom) function evaluations

Figure 15 Solutions found by NSGA–II and NSHA for CONSTR after 2,000 and 15,000 function evaluations, the solid lines show the boundary of the feasible region

Figure 16 Average performance metrics measured on five trials of solving CONSTR test problem with NSGA–II and NSHA.

Figure 17 Non–dominated solutions found for COK after 100,000 function evaluations with NSGA–II and NSHA (left).Convergence metric, γ, averaged over ten trials of solving COK with NSHA and NSGA–II (right)

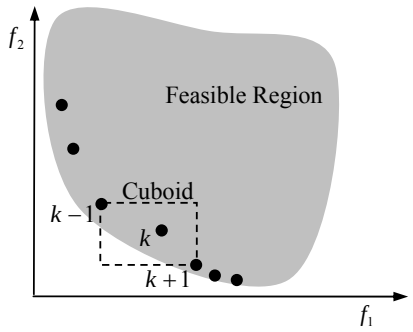


Figure 1 *Crowding distance calculation for the solutions located at the same non–domination front*

n = number of design variables,
$nf_{l,\max}, nf_{g,\max}, nf_t$ = maximum number of function evaluations for local search, genetic algorithm, and total
a = size of the initial hyper–polygon for the local search,
n_{pop} = population size for genetic algorithm,
n_l = maximum number of points within the population that can be improved by the local search,
randomly initialize the first population, P_0
$k = 0$
while there is more than one level of non-domination in P_k
    $P_{k+1} = NSGA-II(P_k)$ improve current population with NSGA–II
    $k = k + 1$
end
nf = number of function evaluations performed by NSGA-II
while $nf < nf_t$
    $j = 1$
    $nf_{l,\max} = \min\{nf_{l,\max}, nf_t - nf\}$
    while $(j < n_l) \,\&\, (P_k(j)\_rank = 1) \,\&\, (nf < nf_t)$
        if $n > 5$, randomly select a subset of three to five variables to be optimized by the local search
        create a hyper–polygon simplex S with the size a based on $P_k(j)$
        $nf_l = 0$
        while $(nf_l < nf_{l,\max}) \,\&\, (no\ other\ stopping\ criteria\ for\ NSNM\ is\ met)$
            Improve S using NSNM
            $nf_l = nf_l + 1$
        end
        $nf = nf + nf_l$
        replace $P_k(j)$ with the best point in S
        set $j = j + 1$
    end
    set $nf_{g,\max} = \min\{nf_{g,\max}, nf_t - nf\}$
    $nf_g = 0$
    while $nf_g < nf_{g,\max}$
        $P_{k+1} = NSGA-II(P_k)$ improve current population with NSGA–II
        $k = k + 1$
        $nf_g = nf_g +$ number of function evaluations performed by NSGA-II
    end
    $nf = nf + nf_g$
end

Figure 2 *Pseudo code of the main loop in NSHA*

S_0 = a given initial simplex

$nf_{l,\max}$ = maximum number of function evaluations for local search

$a_{\min}$ = minimum simplex size

$S = S_0$

$k = 0$

while (there is at least one solution in S dominating another solution in S) and ($k < nf_{l,\max}$) and (simplex size < $a_{\min}$)

    Improve S using NM optimization (Nelder and Mead, 1965) using the sorting procedure as below:

        sort selected solutions using the non-dominate sorting (Deb et al., 2002a)

        maintain the order of the points if more than one point in each domination level

    end

    $k = k +$ number of function evaluations used during NM optimization

end

Figure 3 *Pseudo code of the local simplex optimizer NSNM*

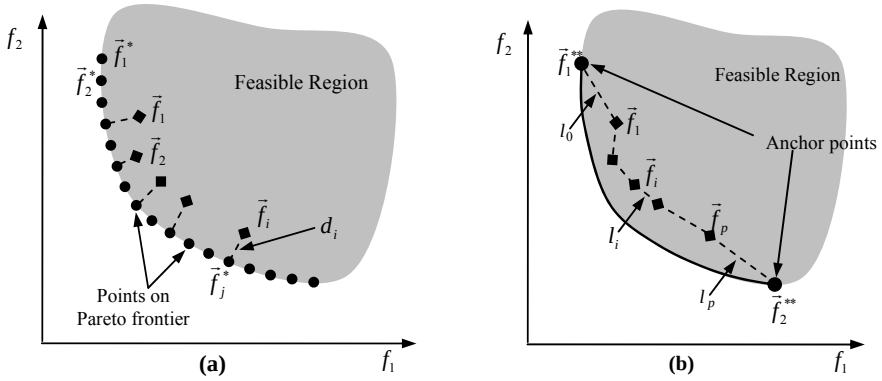


Figure 4 *(a) Convergence metric γ , (b) diversity metric, Δ (Deb et al., 2002a)*

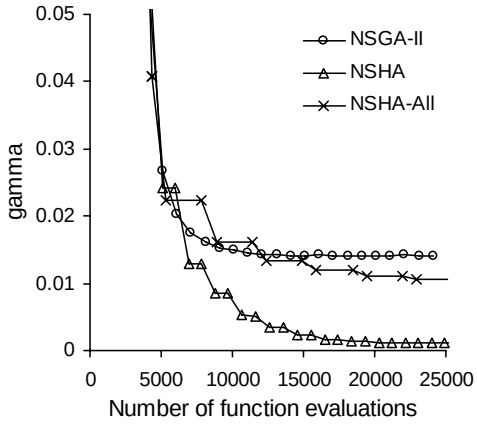


Figure 5 *Convergence measure, γ , versus number of function evaluations performed when NSGA–II, NSHA, and NSHA–All are applied to the ZDT1 test problem. The results are averaged over ten trials with random initial populations.*

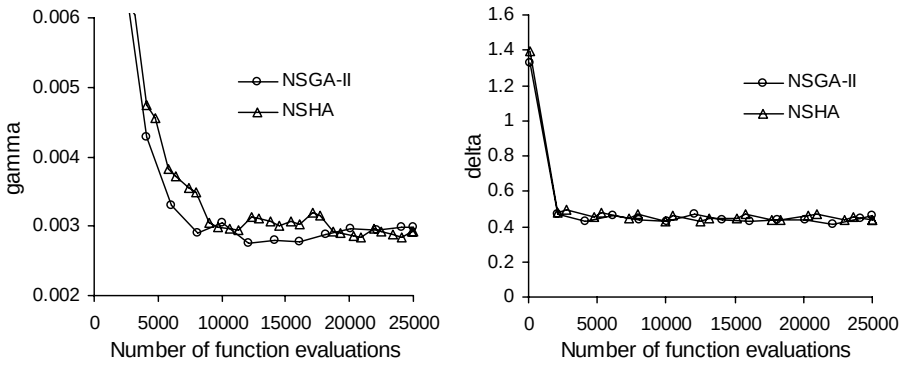


Figure 6 *Performance parameters γ and Δ for FON, averaged over ten trials*

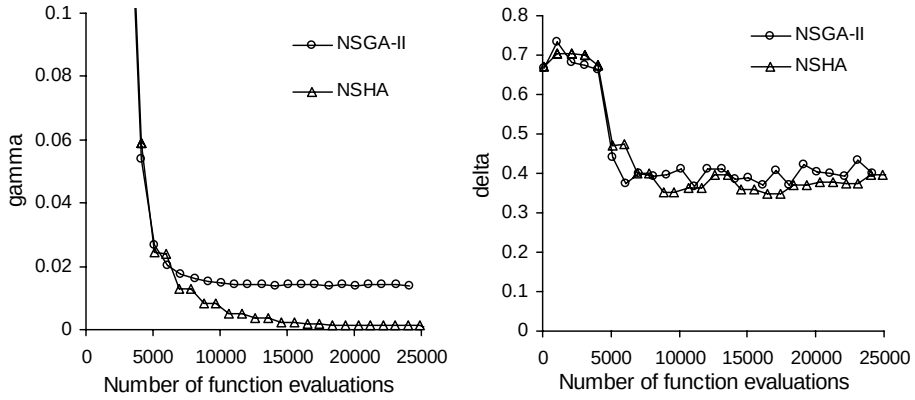
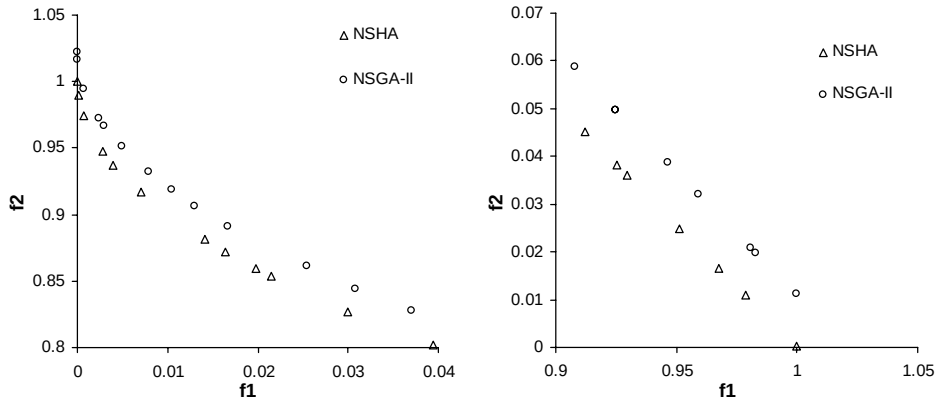


Figure 7 *Performance parameters γ and Δ for ZDT1, averaged over ten trials*



(a) *(b)*

Figure 8 *Solutions close to the two anchor points (i.e. a: best f_1, b: best f_2) found by NSHA and NSGA–II after 25,000 function evaluations for ZDT1*

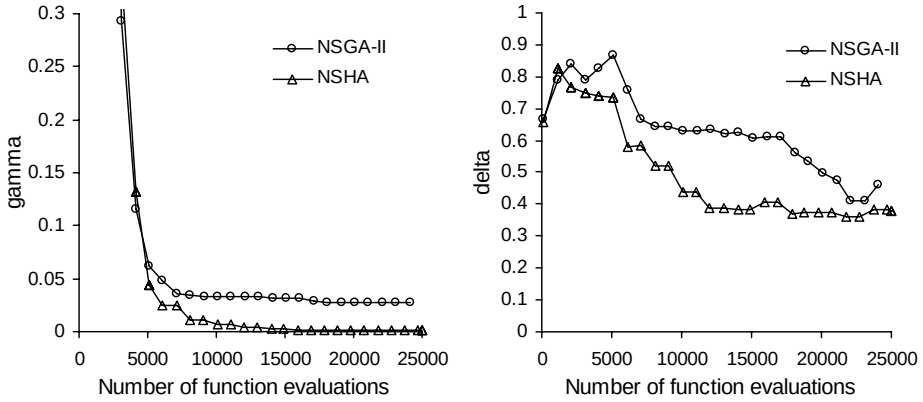


Figure 9 *Average performance parameters γ and Δ for ZDT2 test function after ten trials*

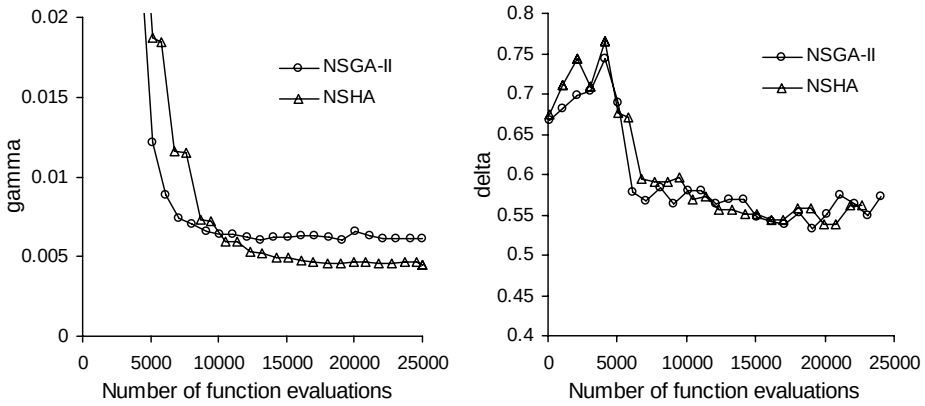


Figure 10 *Average performance parameters γ and Δ for ZDT3 after ten trials*

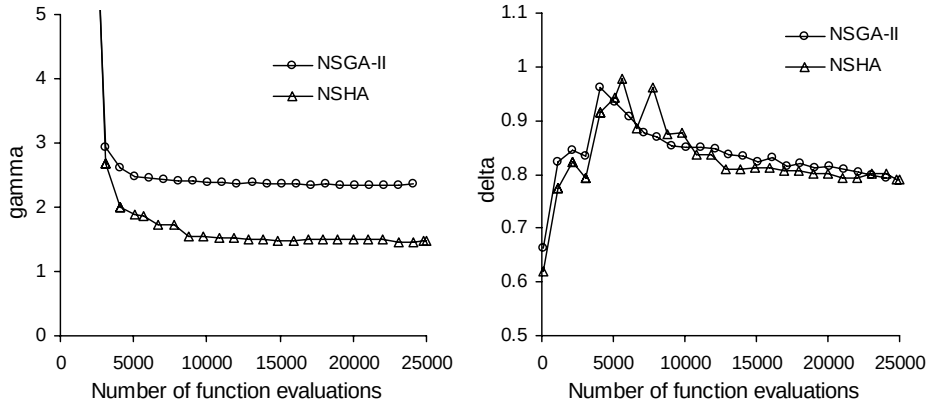


Figure 11 *Average performance parameters γ and Δ for ZDT4 test function after ten trials*

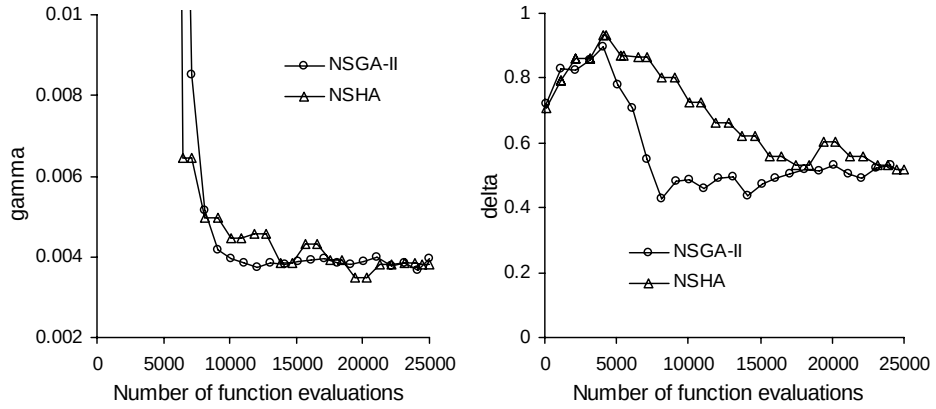


Figure 12 *Average performance parameters γ and Δ for ZDT6 test function after ten trials*

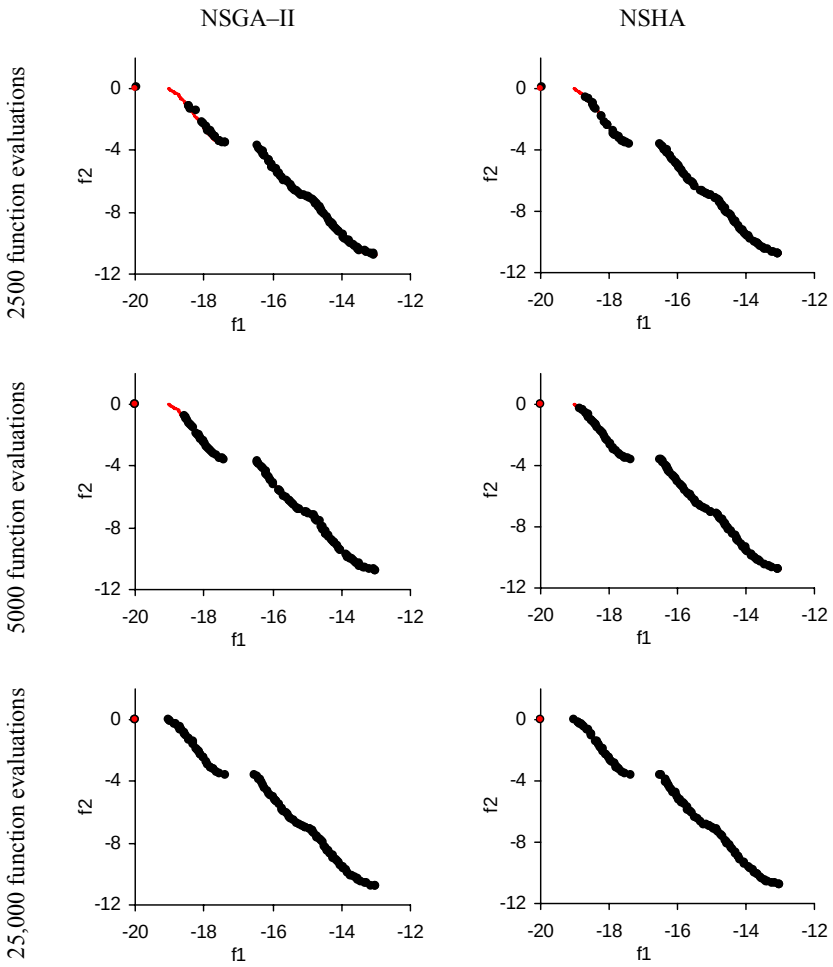


Figure 13 *Non–dominated solutions obtained by NSGA–II and NSHA for MOP4 test problem, the solid line shows the real Pareto front*

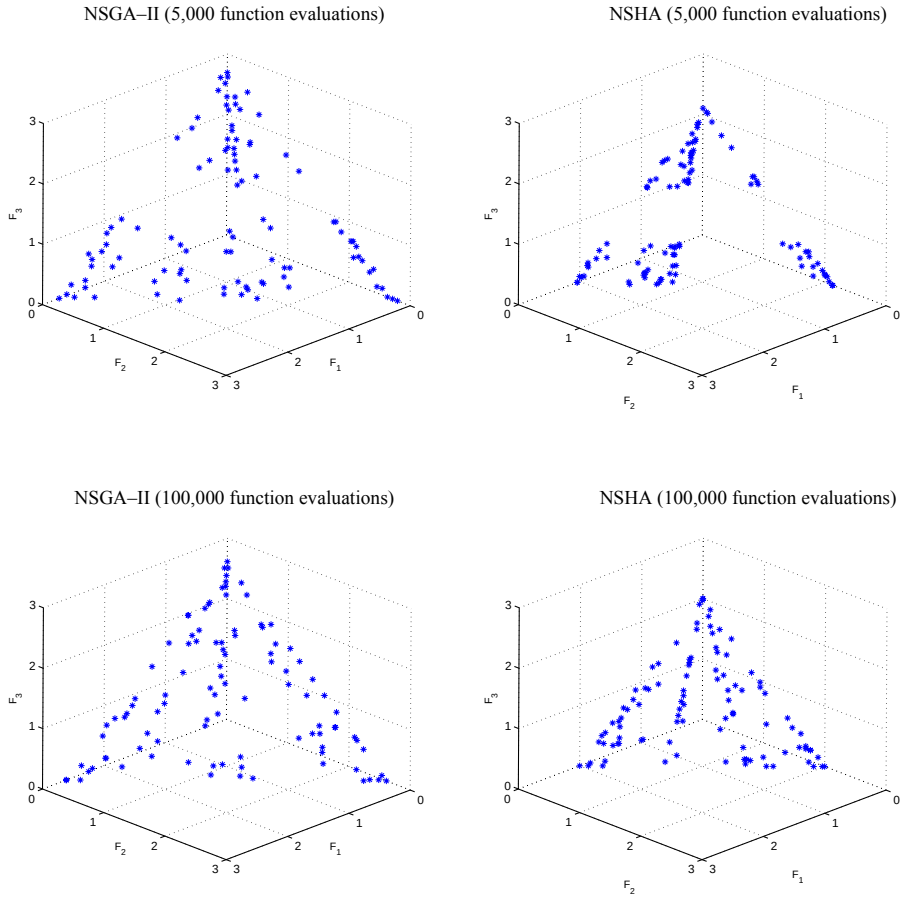


Figure 14 *Non–dominated solutions obtained by NSGA–II (left) and NSHA (right) for DTZL1 test function after performing 5000 (top) and 100,000 (bottom) function evaluations*

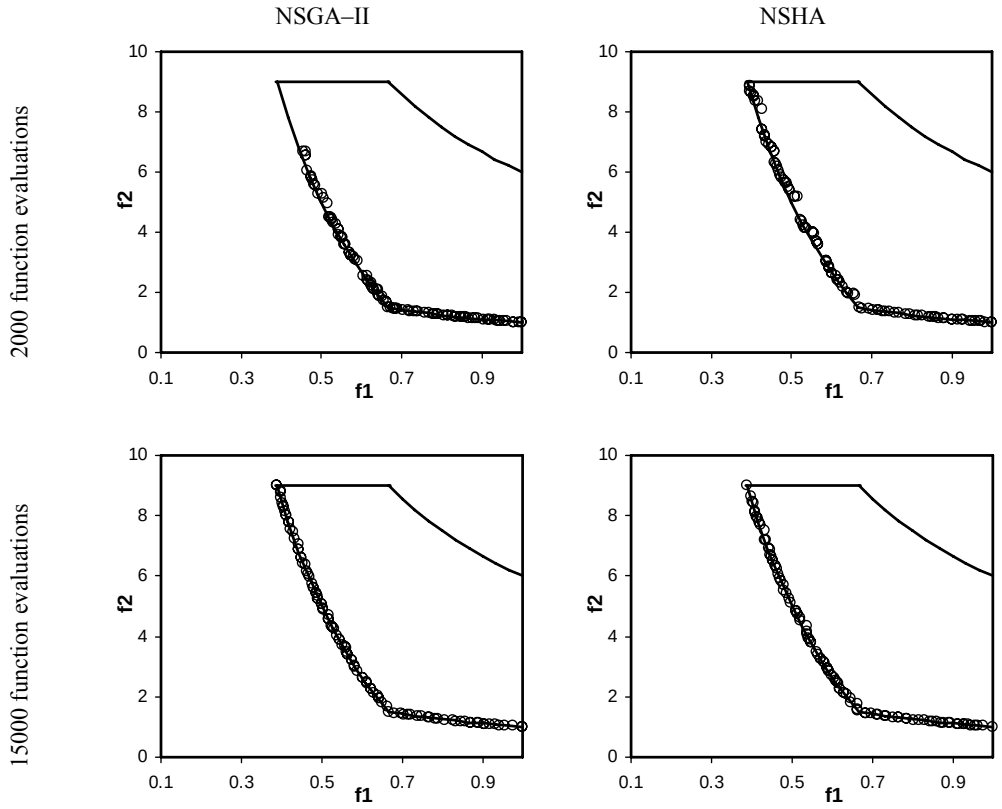


Figure 15 *Solutions found by NSGA–II and NSHA for CONSTR after 2,000 and 15,000 function evaluations, the solid lines show the boundary of the feasible region*

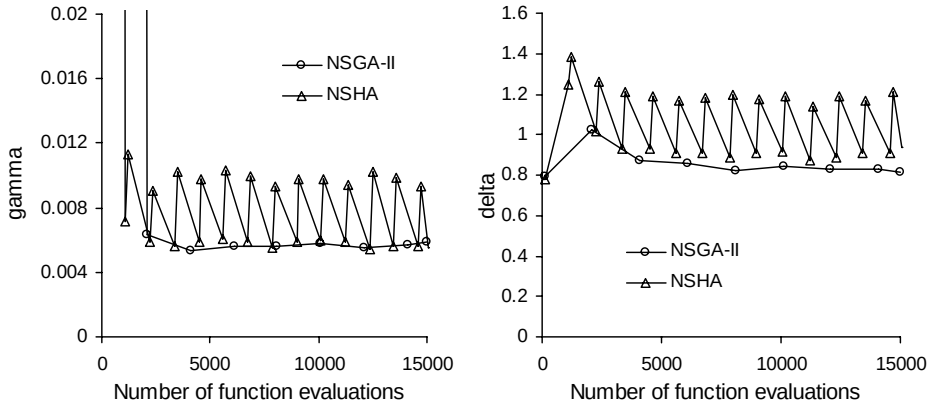


Figure 16 *Average performance metrics measured on five trials of solving CONSTR test problem with NSGA–II and NSHA.*

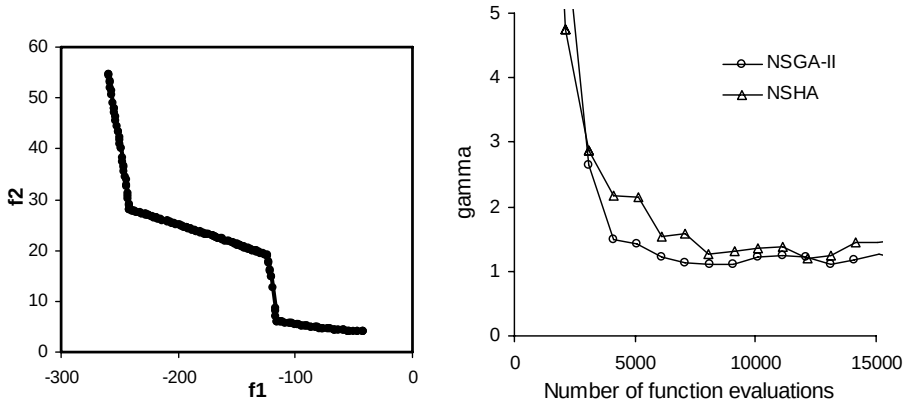


Figure 17 *Non–dominated solutions found for COK after 100,000 function evaluations with NSGA–II and NSHA (left).Convergence metric, γ , averaged over ten trials of solving COK with NSHA and NSGA–II (right)*

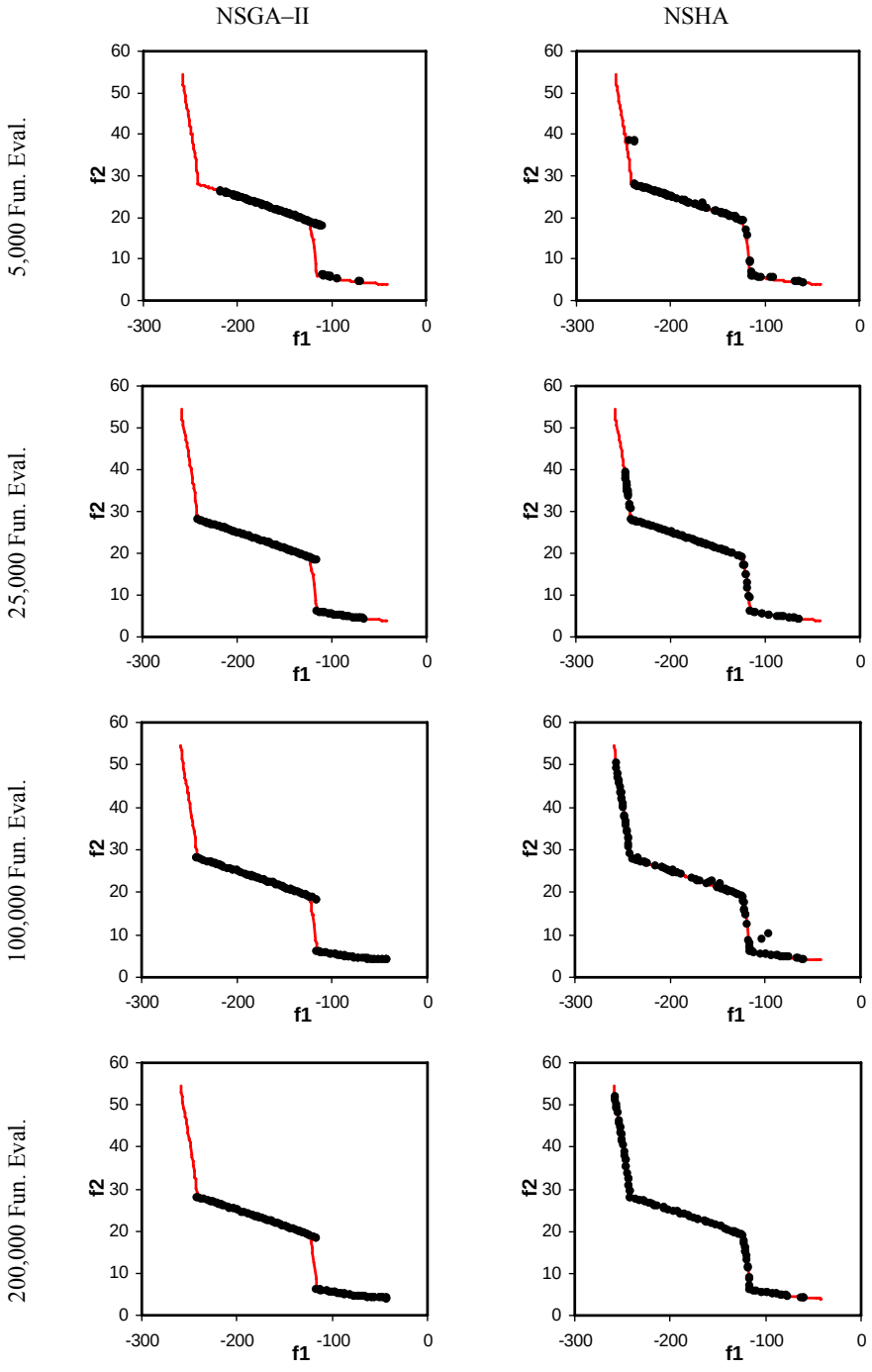


Figure 18 *Progress of the solutions found by NSGA–II and NSHA toward the Pareto front of COK*

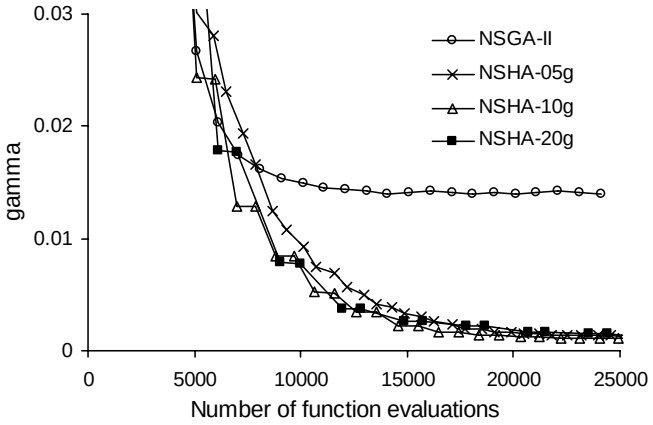


Figure 19 *Convergence metric,γ, for ZDT1,averaged over five trials, while the local search inside NSHA is called every 5, 10, or 20 generations*

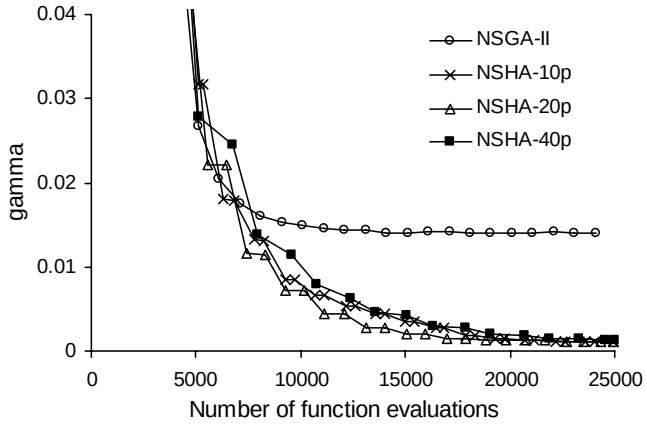


Figure 20 *Convergence metric,γ, for ZDT1 (averaged over five trials),while 10, 20, or 40 percent of the current population participated in the local search*

Figure 21 *Stem is part of a bicycle that connects the handlebar to the fork*

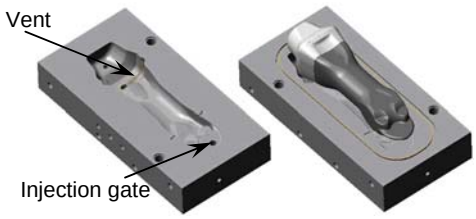


Figure 22 *The mould designed and used for the production of the stem body*

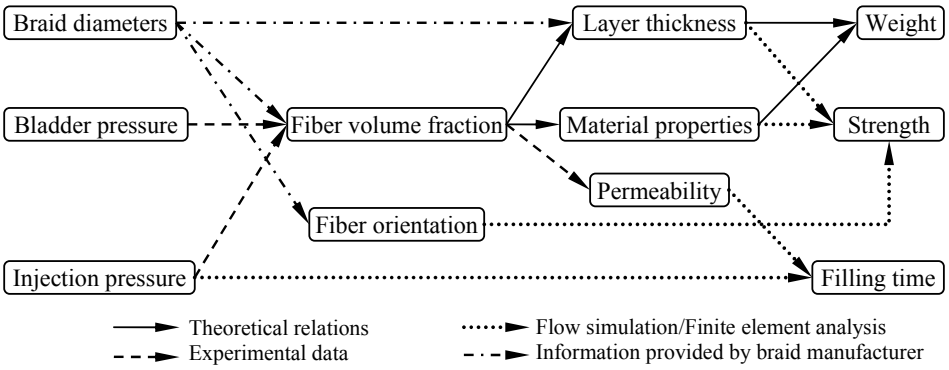


Figure 23 *Interconnection among design variables, intermediate parameters and the objectives of the stem design problem*

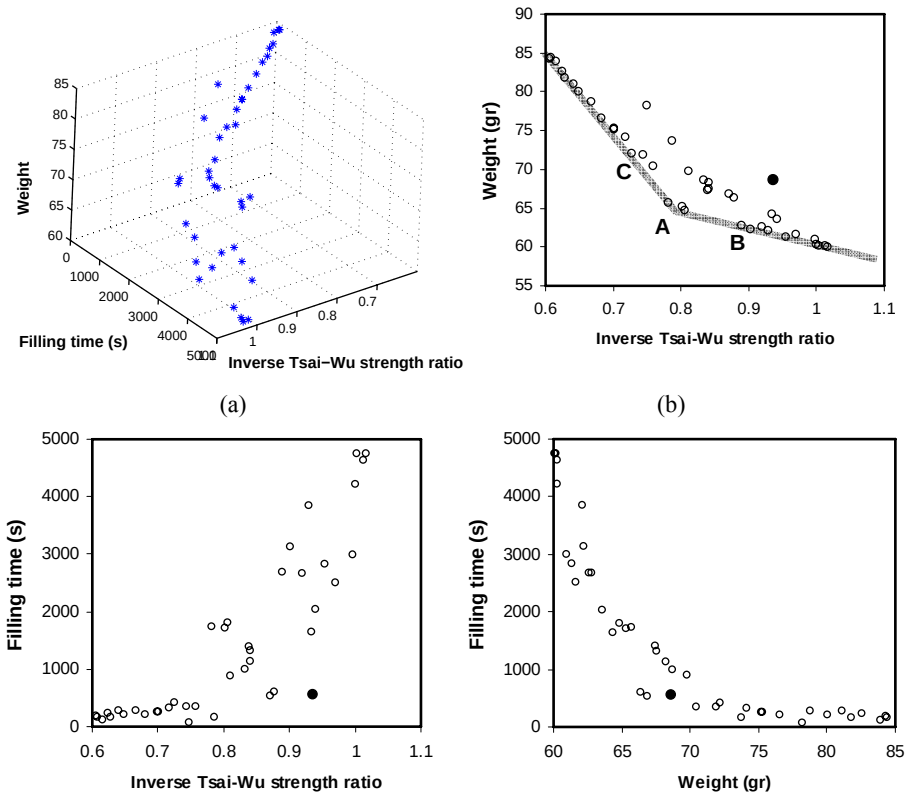


Figure 24 *Solutions found by NSHA for the Stem design problem after 5,000 function evaluations, (a) solutions in 3D criterion space (b–d) pair wise 2D plots of the solutions*

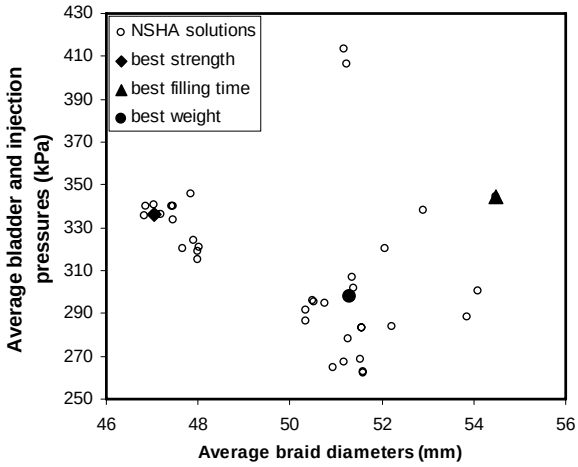


Figure 25 *Illustration of the solutions of the stem design problem within the design space of average braid diameters and average of the injection and bladder pressures*

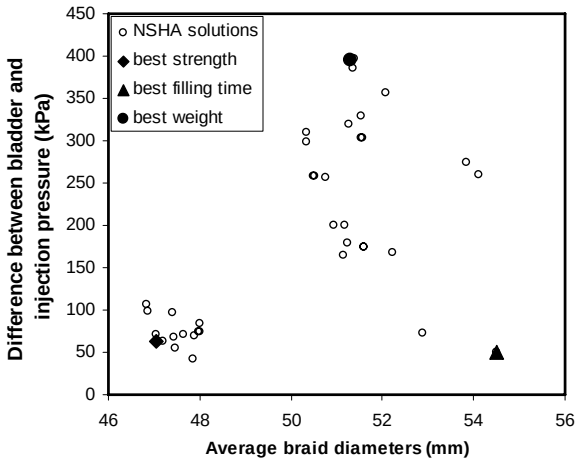


Figure 26 *Illustration of the solutions of the stem design problem within the design space of average braid diameters and difference between the injection and bladder pressures*