Multi-Objective Evolutionary Algorithms for Engineering Shape Design

Kalyanmoy Deb and Tushar Goel
Kanpur Genetic Algorithms Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur, India
{deb,tusharg}@iitk.ac.in
http://www.iitk.ac.in/kangal

KanGAL Report No. 200003

Abstract

Evolutionary optimization algorithms work with a population of solutions, instead of a single solution. Since multi-objective optimization problems give rise to a set of Pareto-optimal solutions, evolutionary optimization algorithms are ideal for handling multi-objective optimization problems. Over many years of research and application studies have produced a number of efficient multi-objective evolutionary algorithms (MOEAs), which are ready to be applied to real-world problems. In this paper, we propose a practical approach, which will enable an user to move closer to the true Pareto-optimal front and simultaneously reduce the size of the obtained non-dominated solution set. The efficacy of the proposed approach is demonstrated in solving a number of mechanical shape optimization problems, including a simply-supported plate design, a cantilever plate design, a hoister design, and a bicycle frame design. The results are interesting and suggest immediate application of the proposed technique in more complex engineering design problems.

1 Introduction

For last decade or so, a number of multi-objective optimization techniques using evolutionary algorithms are suggested [21, 9, 13, 4, 25, 16]. The outcome of these studies is that different multi-objective optimization problems are possible to solve for the purpose of finding multiple Pareto-optimal solutions in one *single* simulation run. Classical means of finding one solution at a time with a weight vector or with a similar approach requires a priori knowledge of weight vector and need to be run many times, hopefully finding a different Pareto-optimal solution each time. In addition to converging close or on the true Pareto-optimal set, multi-objective evolutionary algorithms (MOEAs) are capable to finding a widely distributed set of solutions.

In this paper, we suggest a hybrid technique to take evolutionary multi-objective optimization procedures one step closer to practice. Specifically, in a real-world problem, we would like to ensure a better convergence to the true Pareto-optimal front and would also like to reduce the size of obtained non-dominated solutions to a reasonable number. The solutions obtained by an MOEA are modified using a local search method, in which a weighted objective function is minimized. The use of a local search method from the MOEA solutions will allow a better convergence to the true Pareto-optimal front. A clustering method is suggested in general to reduce the size of the obtained set of solutions. For finite search space problems, the local search approach may itself reduce the size the the obtained set.

A specific MOEA—elitist non-dominated sorting GA or NSGA-II—and a hill-climbing local search method are used together to solve a number of engineering shape optimization problems

for two objectives. Minimizing the weight of a structure and minimizing the maximum deflection of the structure have conflicting solutions. When these two objectives are considered together in a design, a number of Pareto-optimal solutions result. By representing presence and absence of small constituting elements in a binary string [2, 3, 8], NSGA-II uses an innovative crossover operator which seems to help in combining good partial solutions together to form bigger partial solutions. The finite element method is used to evaluate a string representing a shape. The paper shows how the proposed hybrid technique can find a number of solutions with different trade-offs between weight and deflection. On a cantilever plate design, a simply-supported plate design, a hoister plate design, and a bicycle frame design problem, the proposed technique finds interesting and well-engineered solutions. These results indicate that the proposed hybrid technique is ready to be applied to more complex engineering shape design problems.

2 Multi-Objective Optimization and Pareto-Optimality

The principles of multi-criterion optimization are different from that of a single-objective optimization. The main goal in a single-objective optimization is to find the global optimal solution. However, in a multi-criterion optimization problem, there are more than one objective function, each of which may have a *different* individual optimal solution. If there is sufficient difference in the optimal solutions corresponding to different objectives, the objective functions are often known as *conflicting* to each other. Multi-criterion optimization with such conflicting objective functions gives rise to a set of optimal solutions, instead of one optimal solution. The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all objective functions. These optimal solutions have a special name—Pareto-optimal solutions. Let us illustrate this aspect with a hypothetical example problem shown in Figure 1. The figure considers two objectives—cost and accident rate—both of which are to be

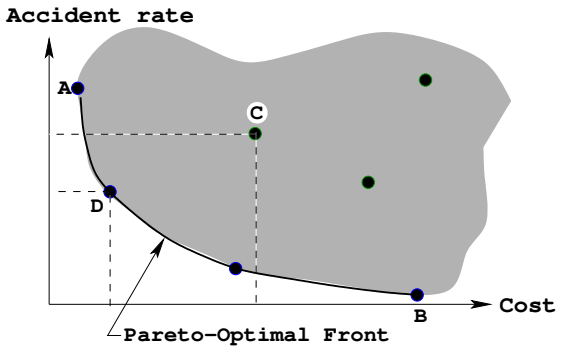

Figure 1: The concept of Pareto-optimal solutions is illustrated.

minimized. The point A represents a solution which incurs a near-minimal cost, but is highly accident-prone. On the other hand, the point B represents a solution which is costly, but is near least accident-prone. If both objectives (cost and accident rate) are important goals of design, one cannot really say whether solution A is better than solution B, or vice versa. One solution is better than other in one objective, but is worse in the other. In fact, there exist many such solutions (like solution D) which also belongs to the Pareto-optimal set and one cannot conclude about an absolute hierarchy of solutions A, B, D, or any other solution in the set without any further information. All these solutions (in the front marked by a thick solid line) are known as *Pareto-optimal* solutions.

Looking at the figure, we also observe that there exists non-Pareto-optimal solutions, like the point C. If we compare solution C with solution A, we again are in a fix and cannot say whether

one is better than the other in both objectives. Does this mean that solution C is also a member of the Pareto-optimal set. The answer is no. This is because there exists another solution D in the search space, which is better than solution C in *both* objectives. That is why solutions like C are known as *dominated* solutions or *inferior* solutions.

It is now clear that the concept of optimality in multi-criterion optimization deals with a number (or a set) of solutions, instead of one solution. Based on the above discussions, we first define conditions for a solution to become dominated with respect to another solution and then present conditions for a set of solutions to become a Pareto-optimal set.

For a problem having more than one objective function (say, f_j, $j = 1, \ldots, M$ and $M > 1$), any two solutions $x^{(1)}$ and $x^{(2)}$ (having P decision variables each) can have one of two possibilities—one dominates the other or none dominates the other. A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both the following conditions are true [22]:

1. The solution $x^{(1)}$ is no worse (say the operator $\prec$ denotes worse and $\succ$ denotes better) than $x^{(2)}$ in all objectives, or $f_j(x^{(1)}) \not\prec f_j(x^{(2)})$ for all $j = 1, 2, \ldots, M$ objectives.

2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective, or $f_{\bar{j}}(x^{(1)}) \succ f_{\bar{j}}(x^{(2)})$ for at least one $\bar{j} \in \{1, 2, \ldots, M\}$.

If any of the above condition is violated, the solution $x^{(1)}$ does not dominate the solution $x^{(2)}$. If $x^{(1)}$ dominates the solution $x^{(2)}$, it is also customary to write $x^{(2)}$ is dominated by $x^{(1)}$, or $x^{(1)}$ is non-dominated by $x^{(2)}$, or, simply, among the two solutions, $x^{(1)}$ is the non-dominated solution. Although the above is a standard definition of domination among two solutions, Parmee et al. [18] suggested a weighted dominance relation based on relative importance of objectives. In problems where a weight information is available, such modified definitions may be useful.

The following definitions ensure whether a set of solutions belong to a local or global Pareto-optimal set, similar to the definitions of local and global optimal solutions in single-objective optimization problems:

Local Pareto-optimal Set: If for every member x in a set $\underline{P}$, there exist no solution y satisfying $\|y - x\|_\infty \leq \epsilon$, where ϵ is a small positive number (in principle, y is obtained by perturbing x in a small neighborhood), which dominates any member in the set $\underline{P}$, then the solutions belonging to the set $\underline{P}$ constitute a local Pareto-optimal set.

Global Pareto-optimal Set: If there exists no solution in the search space which dominates any member in the set $\bar{P}$, then the solutions belonging to the set $\bar{P}$ constitute a global Pareto-optimal set.

We would like to highlight here that there exists a difference between a non-dominated set and a Pareto-optimal set. A non-dominated set is defined in the context of a sample of the search space. In a sample of search points, solutions that are not dominated (according to the above definition) by any other solution in the sample space are non-dominated solutions. A Pareto-optimal set is a non-dominated set, when the sample is the entire search space.

From the above discussions, we observe that there are primarily two goals that a multi-criterion optimization algorithm must try to achieve:

1. Guide the search towards the global Pareto-optimal region, and

2. Maintain population diversity in the Pareto-optimal front.

The first task is a natural goal of any optimization algorithm. The second task is unique to multi-criterion optimization. Since no one solution in the Pareto-optimal set can be said to be better than the other, what an algorithm can do best to find as many different Pareto-optimal solutions as possible.

The classical ways of tackling multi-objective optimization problems is straightforward: Convert multiple objectives into one objective. There exists a number of conversion methods [1, 17, 20]—weighted sum approach, ϵ-perturbation method, Tchybeshev method, min-max method, goal programming method and others. Since multiple objectives are converted into one objective, the resulting solution to the single-objective optimization problem is usually subjective to the parameter settings chosen by the user. Moreover, since usually a classical optimization method is used, only one solution (hopefully a Pareto-optimal solution) can be found in one simulation run. Thus, in order to find multiple Pareto-optimal solutions, the chosen optimization algorithm must have to be used a number of times. Furthermore, the classical methods have been found to be sensitive to the convexity and continuity of the Pareto-optimal region.

3 Elitist Non-dominated Sorting GA (NSGA-II)

The details of NSGA-II algorithm appears elsewhere [5]. Initially, a random parent population P_0 is created. The population is sorted based on the non-domination. A special book-keeping procedure is used in order to reduce the computational complexity down to $O(N^2)$. Each solution is assigned a fitness equal to its non-domination level. Binary tournament selection, recombination, and mutation operators are used to create a child population Q_0 of size N. Thereafter, we use the following algorithm in every generation.

$R_t = P_t \cup Q_t$
$\mathcal{F} =$ `fast-nondominated-sort(`R_t`)`
$P_{t+1} = \emptyset$ and $i = 1$
until $|P_{t+1}| + |\mathcal{F}_i| \leq N$
    `crowding-distance-assignment(`$\mathcal{F}_i$`)`
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$
    $i = i + 1$
Sort$(\mathcal{F}_i, \prec_n)$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$
$Q_{t+1} =$ `make-new-pop(`P_{t+1}`)`
$t = t + 1$

First, a combined population $R_t = P_t \cup Q_t$ is formed. This allows parent solutions to be compared with the child population, thereby ensuring elitism. The population R_t is of size $2N$. Then, the population R_t is sorted according to non-domination [22]. The sorting procedure classifies the population into several non-dominated fronts, F_1, F_2, and so on. The new parent population P_{t+1} is formed by adding solutions from the first front F_1 and continuing to other fronts successively till the size exceeds N. Thereafter, the solutions of the last accepted front are sorted according to a crowded distance metric and a total of N points are picked. The crowded distance is measured as the distance of the maximum box containing the two neighboring solutions in the objective space, as depicted in Figure 2. For details, please refer to the original study [5]. Since the diversity among the solutions is important, we define a *crowded comparison operator* using a partial order relation $\prec_n$ as follows:

Definition 1 *Solution i is better than solution j in relation $\prec_n$ if ($i_{rank} < j_{rank}$) or (($i_{rank} = j_{rank}$) and ($i_{distance} > j_{distance}$)).*

In the tournament selection, we use the above crowded comparison operator to choose one of the two solutions. Between two solutions with differing non-domination ranks in a tournament, we prefer the solutions with the lower rank. Otherwise, if both the points belong to the same front then we prefer the point which is located in a region with lesser number of points (or with

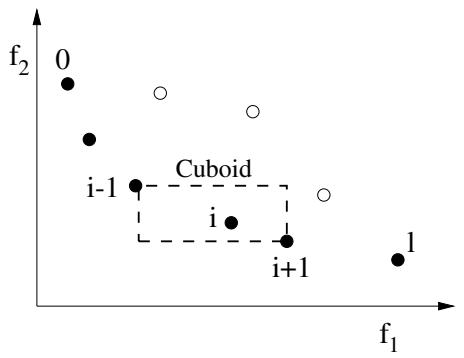

Figure 2: The crowding distance calculation is shown.

larger crowded distance). This way solutions from less dense regions in the search space are given importance in deciding which solutions to choose from R_t. This constructs the population P_{t+1}. This population of size N is now used for selection, crossover and mutation to create a new population Q_{t+1} of size N. We use a binary tournament selection operator but the selection criterion is now based on the crowded comparison operator $\prec_n$. The above procedure is continued for a specified number of generations.

It is clear from the above description that NSGA-II uses (i) a faster non-dominated sorting approach, (ii) an elitist strategy, and (ii) no niching parameter. It has been shown elsewhere [4] that the above procedure has $O(MN^2)$ computational complexity.

3.1 Two Test Problems

We illustrate the working of NSGA-II on two difficult test problems. The function ZDT2 has an non-convex Pareto-optimal region, whereas the second function KUR has a discontinuous Pareto-optimal region:

$$\text{ZDT2:} \quad \begin{cases} f_1(\mathbf{x}) = x_1, \\ f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - (x_1/g(\mathbf{x}))^2\right], \\ g(\mathbf{x}) = 1 + 9 \left(\sum_{i=2}^{30} x_i\right)/29, \\ 0 \le x_i \le 1, \quad i = 1, 2, \ldots, 30. \end{cases} \tag{1}$$

$$\text{KUR:} \quad \begin{cases} f_1(\mathbf{x}) = \sum_{i=1}^{3} \left(-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right), \\ f_2(\mathbf{x}) = \sum_{i=1}^{3} \left(|x_i|^{0.8} + 5 \sin x_i^3\right) \\ -5 \le x_i \le 5, \quad i = 1, 2, 3. \end{cases} \tag{2}$$

Figure 3 shows the obtained non-dominated set of solutions for ZDT2. All solutions lie on the true Pareto-optimal front. Binary-coded GAs with a single-point crossover operator ($p_c = 0.9$) and a bit-wise mutation operator ($p_m = 1/600$) are used. For each variable, we use 20 bits. A population of size 100 is chosen. NSGA-II is run for 250 generations. It is clear that NSGA-II is able to maintain a wide spread of solutions on the Pareto-optimal front, which is non-convex. The search space lies above the Pareto-optimal front shown in the figure.

Figure 4 shows that NSGA-II is also able to find a well-distributed set of solutions in all three discontinuous regions of Pareto-optimal front for KUR. Here, a real-coded GA is used. Variables are coded directly and the simulated binary crossover operator with $p_c = 0.9$ [7] and a polynomial mutation operator with $p_m = 1/3$ are used. NSGA-II with a population size of 100 is run for a maximum of 250 generations. Despite disconnected regions, the NSGA-II can find well-distributed solutions in all regions.

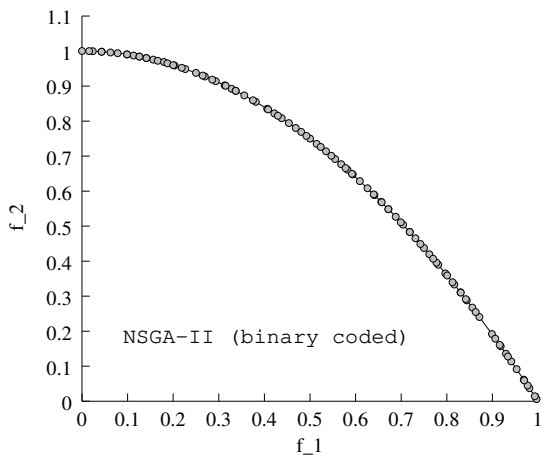

Figure 3: Non-dominated solutions with NSGA-II (binary-coded) on ZDT2.

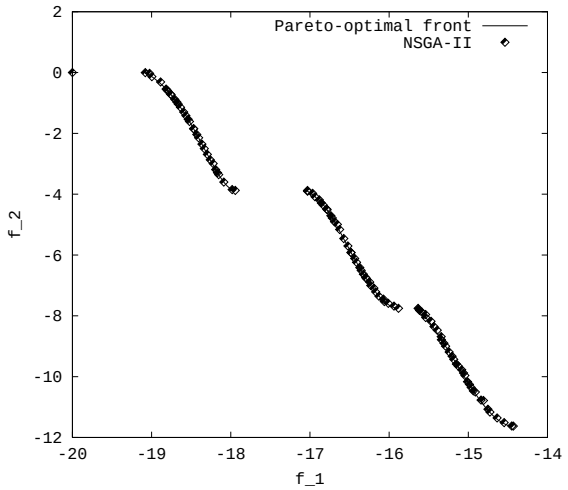

Figure 4: Non-dominated solutions with NSGA-II (real-coded) on KUR.

3.2 Constraint handling

The usual way of handling constraints using penalty functions is criticized for their sensitivity on the so-called penalty parameters. Here, we suggest a more elegant approach for constraint handling as follows. We simply change the definition of domination between two solutions as follows:

Definition 2 *A solution i is said to constrained-dominate a solution j, if any of the following conditions is true:*

1. *Solution i is feasible and solution j is not.*
2. *Solutions i and j are both infeasible, but solution i has a smaller constraint violation.*
3. *Solutions i and j are feasible and solution i dominates solution j.*

This way, feasible solutions *constrained-dominate* any infeasible solution and two infeasible solutions are compared based on their constraint violations only. However, when two feasible solutions are compared, they are checked based on their domination level.

3.2.1 Simulation Results

We illustrate the working of the constraint handling strategy on the following problem:

$$\text{TNK:} \begin{cases} f_1(\mathbf{x}) = x_1, \\ f_2(\mathbf{x}) = x_2, \\ g_1(\mathbf{x}) = -x_1^2 - x_2^2 + 1 + 0.1\cos(16\arctan x/y) \leq 0, \\ g_2(\mathbf{x}) = (x-0.5)^2 + (x_2 - 0.5)^2 \leq 0.5, \\ 0 \leq x_i \leq \pi, \quad i = 1, 2. \end{cases} \tag{3}$$

Figure 5 shows the NSGA-II population after 100 generations. Real-coded NSGA-II with identical parameter setting as in KUR is used here. It is clear that NSGA-II is able to distribute population members on the critical constraint boundary, portions of which form the constrained Pareto-optimal set. Regions which are not Pareto-optimal are not found by the NSGA-II. This shows the ability of NSGA-II in converging and in distributing solutions on the true Pareto-optimal regions.

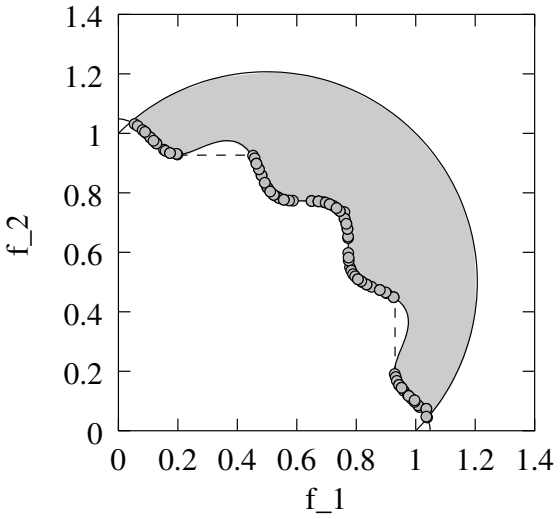

Figure 5: Obtained non-dominated solutions with NSGA-II on the constrained problem TNK.

4 Hybrid Approach

It is clear from the previous section that NSGA-II is an efficient procedure of finding a wide-spread as well as well-converged set of solutions in a multi-objective optimization problem. All problems considered in the previous section and in earlier studies are test problems where exact location of Pareto-optimal solutions is known. Such test problems are necessary to evaluate the working of any MOEA. However, in this section we would like to take MOEAs a step closer to practice by

1. ensuring convergence closer to the true Pareto-optimal front, and
2. reducing the size of the obtained non-dominated set.

We illustrate both the above issues in the following subsections.

4.0.2 Converging better

In a real-world problem, the knowledge of the Pareto-optimal front is usually not known. Although NSGA-II has demonstrated good convergence properties in test problems, we enhance the probability of its true convergence by using a hybrid approach. A local search strategy is suggested from each obtained solution of NSGA-II to find a better solution. Since a local search strategy requires a single objective function, a weighted objective or a Tchybeshev metric or any other metric which will convert multiple objectives into a single objective can be used. In this study, we use a weighted objective:

$$F(\mathbf{x}) = \sum_{j=1}^{M} \bar{w}_j^{\mathbf{x}} f_j(\mathbf{x}), \tag{4}$$

where weights are calculated from the obtained set of solutions in a special way. First, the minimum $f_j^{\min}$ and maximum $f_j^{\max}$ values of each objective function f_j are noted. Thereafter, for any solution $\mathbf{x}$ in the obtained set, the weight for each objective function is calculated as follows:

$$\bar{w}_j^{\mathbf{x}} = \frac{(f_j^{\max} - f_j(\mathbf{x}))/(f_j^{\max} - f_j^{\min})}{\sum_{k=1}^{M}(f_k^{\max} - f_k(\mathbf{x}))/(f_k^{\max} - f_k^{\min})}. \tag{5}$$

In the above calculation, minimization of objective functions is assumed. When a solution $\mathbf{x}$ is close to the individual minimum of the function f_j, the numerator becomes one, causing a large value of the weight for this function. For an objective which has to be maximized, the term $(f_j^{\max} - f_j(\mathbf{x}))$ needs to be replaced with $(f_j^{\mathbf{x}} - f_j^{\min})$. The division of the numerator with the denominator ensures that the calculated weights are normalized or $\sum_{j=1}^{M} \bar{w}_i^{\mathbf{x}} = 1$.

In order to distinguish the above calculated weight from the usual user-specified weight needed in weighted multi-objective optimization algorithms, we call these calculated weights $\bar{\mathbf{w}}$ as pseudo-weights.

Once the pseudo-weights are calculated, the local search procedure is simple. Begin the search from each solution $\mathbf{x}$ independently with the purpose of optimizing $F(\mathbf{x})$. Figure 6 illustrates this procedure. Since, the pseudo-weight vector $\bar{\mathbf{w}}$ dictates roughly the priority of different objective

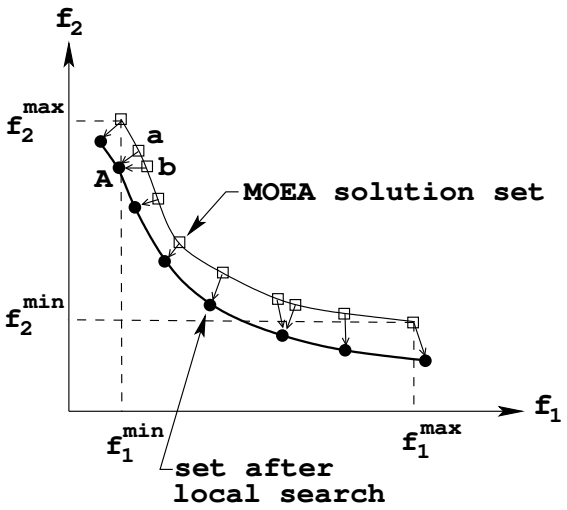

Figure 6: The local search technique may find better solutions.

functions at that solution, optimizing $F(\mathbf{x})$ will produce a Pareto-optimal or a near Pareto-optimal solution. This is true for convex Pareto-optimal regions. However, for non-convex Pareto-optimal regions, there exists no weight vector corresponding to Pareto-optimal solutions in certain regions. Thus, a different metric, such as Tchybeshev metric can be used in those cases. Nevertheless, the overall idea is that once NSGA-II finds a set of solutions close to the true Pareto-optimal region, we use a local search technique from each of these solutions with a differing emphasis of objective functions in the hope of better converging to the true Pareto-optimal front. Since independent local search methods are tried from each solution obtained using an MOEA, all optimized solutions obtained by the local search method need not be non-dominated to each other. Thus, we find the non-dominated set of solutions from the obtained set of solutions before proceeding further.

The complete procedure of the proposed hybrid strategy is shown in Figure 7. Starting from the MOEA results, we first apply a local search technique, followed by a non-domination check. After non-dominated solutions are found, a clustering technique is used to reduce the size of the optimal set, as discussed in the next subsection.

4.1 Reducing the size of non-dominated set

In an ideal scenario, an user is interested in finding a good spread of non-dominated solutions closer to the true Pareto-optimal front. From a practical standpoint, the user would be interested in a handful of solutions (in most cases, 5 to 10 solutions are probably enough). Interestingly,

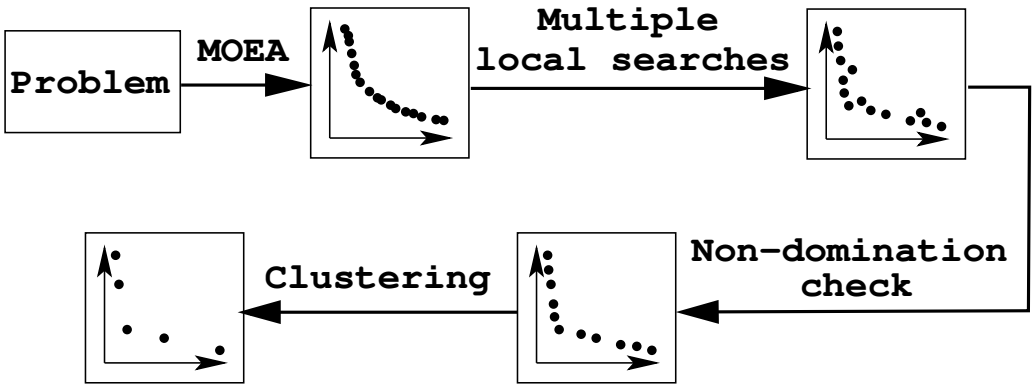

Figure 7: The proposed hybrid procedure of using a local search technique, a non-domination check, and a clustering technique is illustrated.

most MOEA studies use a population of size 100 or more, thereby finding about 100 different non-dominated solutions. The interesting question to ask is 'Why are MOEAs set to find many more solutions than desired?'

The answer is fundamental to the working of an EA. The population size required in an EA depends on a number of factors related to the number of decision variables, the complexity of the problem, and others [10, 12]. The population cannot be sized according to the desired number of non-dominated solutions in a problem. Since in most interesting problems, the number of decision variables are large and are complex, the population sizes used in solving those problems can be in hundreds. Such a population size is mandatory for the successful use of an EA. The irony is that when an MOEA works well with such a population size N, eventually it finds N different non-dominated solutions, particularly if the niching mechanism used in the MOEA is good. Thus, we need to devise a separate procedure of identifying a handful of solutions from the large obtained set of non-dominated solutions.

One approach would be to use a clustering technique similar to that used in [23] for reducing the size of the obtained non-dominated set of solutions. In this technique, each of N solutions is assumed to belong to a separate cluster. Thereafter, the distance d_c between all pairs of clusters is calculated by first finding the centroid of each cluster and then calculating the Euclidean distance between the centroids. Two clusters having the minimum distance are merged together into a bigger cluster. This procedure is continued till the desired number of clusters are identified. Finally, with the remaining clusters, the solution closest to the centroid of the cluster is retained and all other solutions from each cluster are deleted. This is how the clusters can be merged and the cardinality of the solution set can be reduced. Figure 8 shows the MOEA solution set in open boxes and the reduced set in solid boxes. Care may be taken to choose the extreme solutions in the extreme clusters.

However, in many problems the local search strategy itself can reduce the cardinality of the obtained set of non-dominated solutions. This will particularly happen in problems with a discrete search space. For two closely located solutions, the pseudo-weight vectors may not be very different. Thus, when a local search procedure is started from each of these solutions (which are close to each other) with a $F(\mathbf{x})$ which is also similar, the resulting optimum solutions may be identical in a discrete search space problem. The solutions a and b in Figure 6 are close and after the local search procedure they may converge to the same solution A. Thus, for many solutions obtained using NSGA-II, the resulting optimum obtained using the local search method may be the same. Thus, the local search procedure itself may reduce the size of the obtained

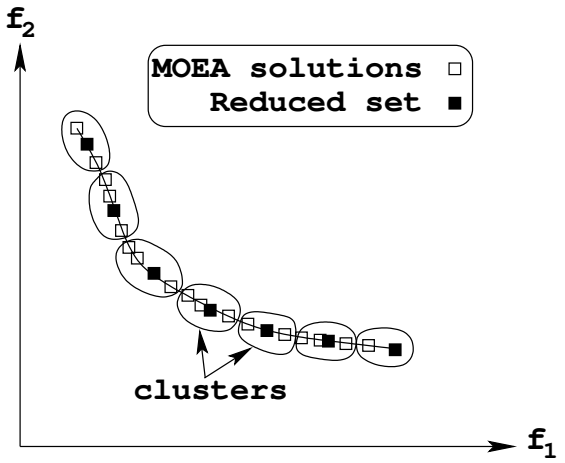

Figure 8: The clustering method of reducing the set of non-dominated solutions is illustrated.

non-dominated solutions in problems with a finite search space. Figure 7 shows that clustering is the final operation of the proposed hybrid strategy.

5 Optimal Shape Design

Designing shape of engineering components is not a new activity. The use optimization in engineering shape design has also received a lot of attention. However, in optimal shape design problems, the most popular approach has been to pre-define a parametric mathematical function for the boundary describing the shape and use an optimization technique to find the optimal values of the parameters describing the mathematical function. Although this procedure requires prior knowledge of the shape, the classical optimization methods facilitated the optimization of parameters describing the mathematical shape functions.

With the advent of evolutionary algorithm as an alternate optimization method, there exist a number of applications of optimal shape design, where shapes are evolved by deciding presence or absence of a number of small elements [2, 3, 14, 19, 11]. A predefined area (or volume) is divided into a number of small regular elements. The task of an evolutionary optimization procedure is to find which elements should be kept and which should be thrown away so that the resulting shape is optimal with respect to an objective function. This procedure has a number of advantages:

1. The use of numerical finite element method (or boundary element method) is an usual method of analyzing an engineering component. Since finite element method procedure requires the component to be divided into a number of small elements, this approach reduces one computation step and is complimentary to the usual finite element method.

2. Since no a priori knowledge about the shape is required, this method does not have any bias from the user.

3. By simply using three-dimensional elements, the approach can be extended to three-dimensional shape design problems.

4. The number and shape of holes in a component can evolve naturally without explicitly fixing them by the user.

Most studies of this method, including the studies with evolutionary algorithms, have concentrated on optimizing a single objective. In this study, we apply this evolutionary procedure for multiple conflicting objectives.

5.1 Representation

In this study, we consider two-dimensional shape design problems only. However, the procedure can be easily extended to three-dimensional shape design problems as well. We begin with a rectangular plate, describing the maximum overall region, where the shapes will be confined. Thereafter, we divide the rectangular plate into a finite number of small elements (refer to Figure 9). We consider here square elements, although any other shape including triangular or

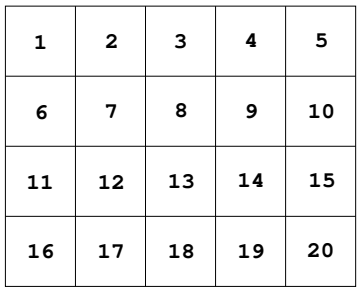

Figure 9: Rectangular plate divided into small elements.

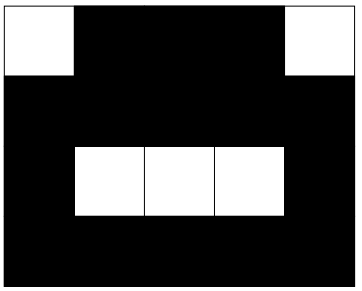

Figure 10: The skeleton of a shape.

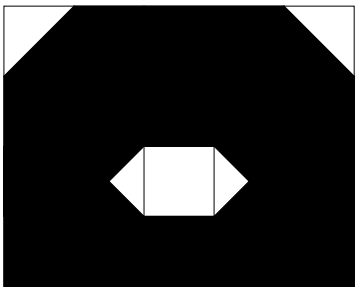

Figure 11: Final smoothen-ed shape.

rectangular elements can also be considered. Since the presence or absence of every element is a decision variable, we use a binary coding describing a shape. For the shape shown in Figure 10, the corresponding binary coding is as follows:

```
01110 11111 10001 11111
```

The presence is denoted by a `1` and the absence is shown by a `0`. A left-to-right coding procedure as shown in Figure 9 is adopted here. In order to smoothen the stair-case like shape denoted by the basic skeleton representation, we add triangular elements (shown shaded) for different cases in Figure 12. The resulting skeleton shape shown in Figure 10 represents the true shape shown in Figure 11.

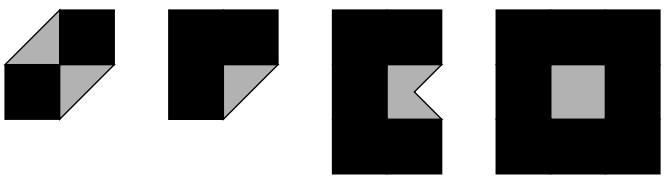

Figure 12: Different cases of smoothening through triangular elements.

5.2 Evaluation

When the shape is smoothened, the shape is further divided into smaller elements. All interior rectangular elements are divided into two triangles and all boundary elements (including elements around a hole) are divided into four small triangles. Even the boundary triangles used for

smoothening is divided into smaller triangles. The shape is evaluated by finding the maximum stress and deflection developed at any point in the component by the application of the specified loads. Since no connectivity check is made while creating a new string or while creating the initial random population, a string may represent a number of disconnected regions in the rectangle. In this case, we proceed with the biggest cluster of connected elements (where two elements are defined to be connected if they have at least one common corner). The string is repaired by assigning a `0` at all elements which are not part of the biggest cluster.

In all applications here, two conflicting objectives are chosen: weight and deflection. These two objectives are conflicting because a minimum weight design is usually not stiff and produces a large deflection, whereas a minimum deflection design has densely packed elements, thereby causing a large weight of the overall component. The maximum stress and deflection values are restricted to lie within specified limits of the design by using them as constraints.

6 Simulation Results

To show the efficacy of the proposed hybrid multi-objective optimization procedure in solving optimal shape design problems, we use a number of mechanical component design problems. In all cases, we use NSGA-II as the multi-objective optimizer. Since binary-coded strings are used to represent a shape, we use a bit-wise hill-climbing strategy as the local search operator. The procedure is simple. Starting from the left of the string, every bit is flipped to see if it improves the design. If it does, the flipped bit is retained, else the bit is unchanged. This procedure is continued until no bit-flipping over the the entire string length has resulted an improvement.

Since the shapes are represented in a two-dimensional grid, we introduce a new crossover operator which respects the rows or columns of two parents. Whether to swap rows or columns are decided with with a probability 0.5. Each row or column is swapped with a probability $0.95/d$, where d is the number of rows or columns, as the case may be. This way on an average one all most one row or column will get swapped between the parents. A bit-wise mutation with a probability of 1/string-length are used. NSGA-II is continued till 150 generations. It is important to highlight that NSGA-II does not require any extra parameter setting. In all problems, a population of size 30 is used.

For all problems, we use the following material properties:

Plate thickness	: 10 mm
Yield strength	: 150 MPa
Young's modulus	: 200 GPa
Poisson's ratio	: 0.25

6.1 Cantilever Plate Design

First, we consider a cantilever plate design problem, where an end load $P = 10$ kN is applied as shown in Figure 13. The rectangular plate of size 60×100 mm^2 is divided into 60 small rectangular elements. Thus, 60 bits are used construct a binary string representing a shape.

Figure 14 shows the four steps of the proposed hybrid method in designing the cantilever plate. First plot shows the non-dominated solutions obtained using NSGA-II. Since the population size is 30, NSGA-II is able to find 30 different non-dominated solutions. Thereafter, the local search method is applied from each non-dominated solution and new and improved set of solutions are obtained. The third plot is the result of the non-dominated check of the solutions obtained after the local search method. Three dominated solutions are eliminated by this process. The final plot is obtained after the clustering operation with a choice of nine solutions. The plot shows how

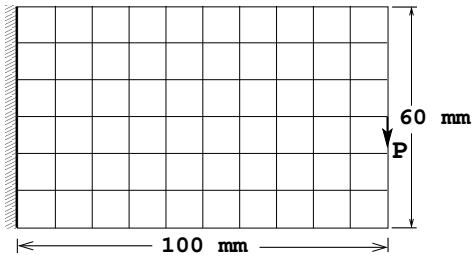

Figure 13: The loading and support of the cantilever plate are shown.

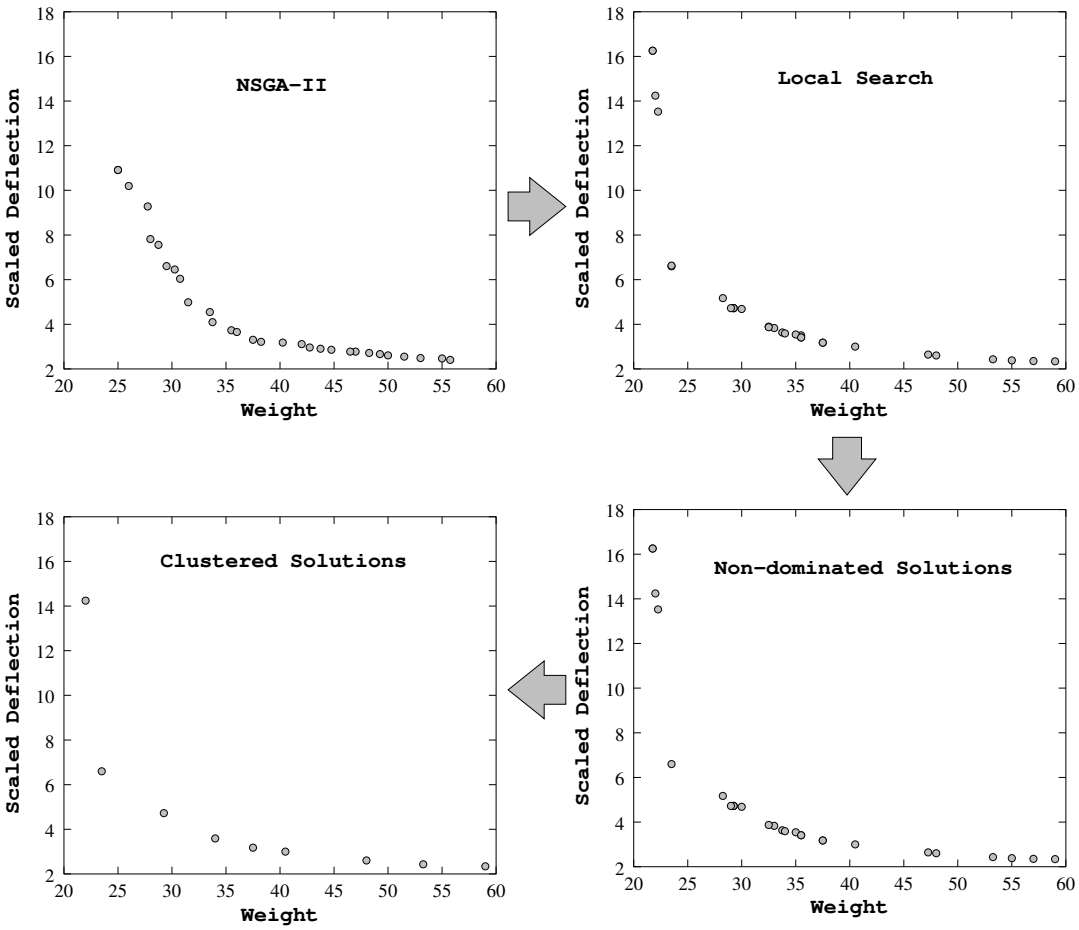

Figure 14: Hybrid procedure to find nine trade-off solutions for the cantilever plate design problem.

nine well distributed set of solutions are found from the third plot of 27 solutions. If fewer than nine solutions are desired, the clustering mechanism can be set accordingly.

In order to visualize the obtained set of nine solutions having a wide range of trade-offs in the weight and scaled deflection values, we show the shapes in Figure 15. It is clear that starting from a low-weight solution (with large deflection), how large-weight (with small deflection) shapes are found by the hybrid method. It is interesting to note that the minimum weight solution eliminated one complete row (the bottom-most row) in order to reduce the overall weight. The second solution (the element (1,2) in the above 3×3 matrix) corresponds to the second-best weight solution. It is well known that for an end load cantilever plate, a parabolic shape is optimal. Both shapes

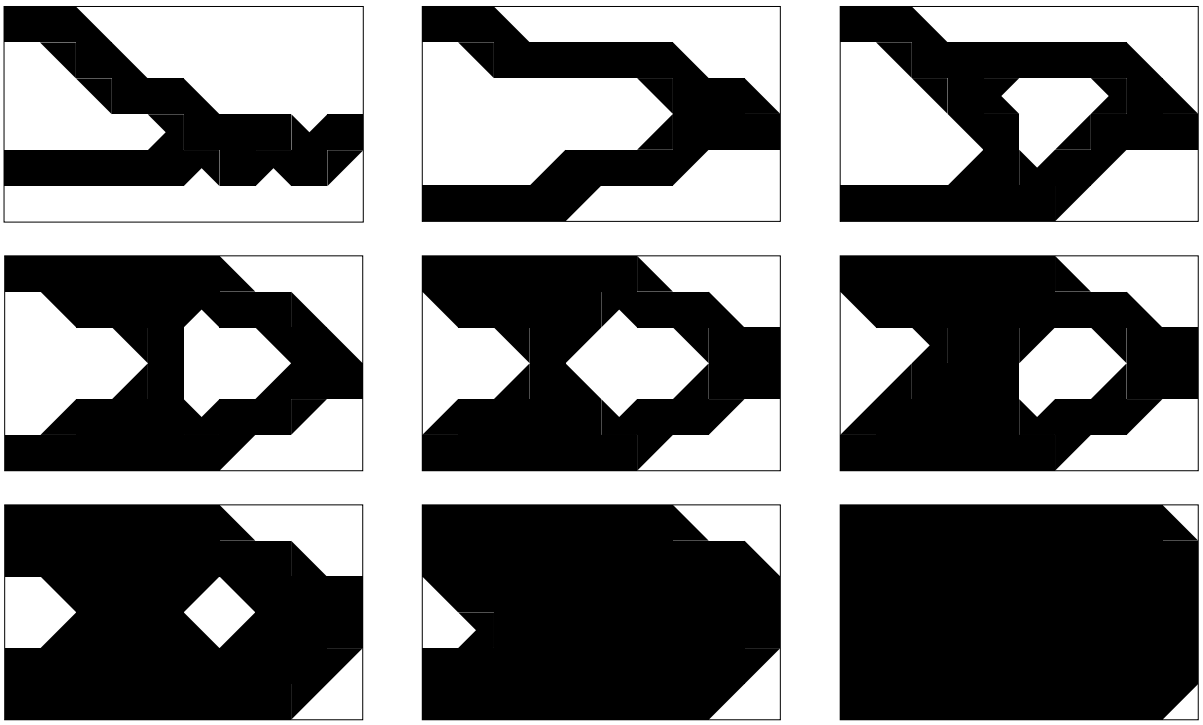

Figure 15: Nine trade-off shapes for the cantilever plate design.

(elements (1,1) and (1,2)) exhibits a similar shape. As the importance of deflection increases, the shapes tend to more and more elements, thereby making the plate rigid enough to have smaller deflection. In the middle, the development of vertical stiffener is interesting. This is a compromise between the minimum weight solution and a minimum deflection solution. By adding a stiffener the weight of the structure does not increase much, whereas the stiffness of plate increases (hence the deflection reduces). Finally, the complete plate with right top and bottom ends chopped off is the minimum deflection solution.

We would like to reiterate here that the above nine solutions are not results of multiple runs of a multi-objective optimization algorithm. All nine solutions (and if needed, more can also be obtained) with interesting trade-offs between weight and deflection are obtained using in one simulation run of the hybrid method.

6.2 Simply-Supported Plate Design

Next, we consider a simply-supported plate design, starting from a rectangular plate of identical dimension as in the previous design. The plate is supported on two supports as shown in Figure 16 and a vertical load $P = 10$ kN is acted on the top-middle node of the plate.

Figure 17 shows the obtained non-dominated solutions using NSGA-II. After local search method, the obtained non-dominated solutions have a wider distribution. The number of solutions have been reduced from 30 solutions to 22 solutions by the non-dominated checking. Finally, the clustering algorithm finds nine widely separated solutions from 22 non-dominated solutions. The shape of these nine solutions are shown in Figure 18. The minimum weight solution tends to use one row (the top-most row) less, but since the load is acting on the top of the plate, one element is added to have the load transferred to the plate. The third solution (shown in the (1,3)-th position in the matrix) is interesting. A careful look at Figure 17 reveals that this solution is a 'knee' solution. To achieve a small advantage in weight-loss, a large sacrifice in the deflection-gain is evident. Similarly, to achieve a small advantage in deflection-loss, a large sacrifice in weight is

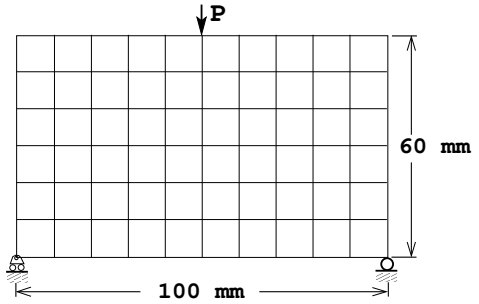

Figure 16: The loading and support of the simply-supported plate are shown.

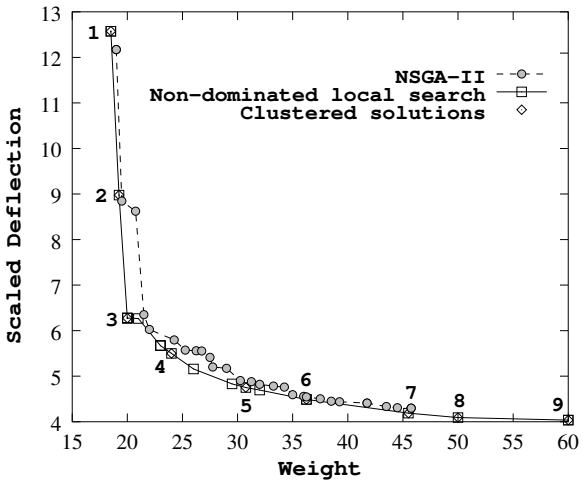

Figure 17: Hybrid procedure finds nine trade-off solutions for the simply-supported plate design problem.

needed. Shapes in position (1,2) and (2,1) can be compared with respect to the shape in position (1,3). Shape in position (3,1) or solution 7 is also interesting. In order to have further reduction in deflection stiffening of the two slanted arms is needed. Finally, the absolute minimum deflection shape is the complete rectangle with maximum possible weight.

Starting with the minimum weight design having two slim slanted legs down to thickening the legs to make them stiff, followed by joining the legs with a stiffener, and finally finding the complete rectangular plate having minimum deflection are all intuitive trade-off solutions. In the absence of any such knowledge, it is interesting how the hybrid procedure with NSGA-II is able to find the whole family of different trade-off solutions.

6.3 Hoister Plate Design

Here, we attempt to design a hoister, that lifts a weight. Once again, a rectangular plate is used, but this time we use a 80×60 mm^2 plate with 48 elements. Thus, the string to represent a hoister is 48 bits long. The loading and the support conditions are shown in Figure 19. The vertical load is 5 kN and the horizontal loadings are 2.5 kN distributed over the length of the element. In order apply the load, two center elements are forced to made absent (that is, the corresponding bits in a string are always set of zero).

Figure 20 shows 30 non-dominated solutions obtained using NSGA-II. The figure shows a

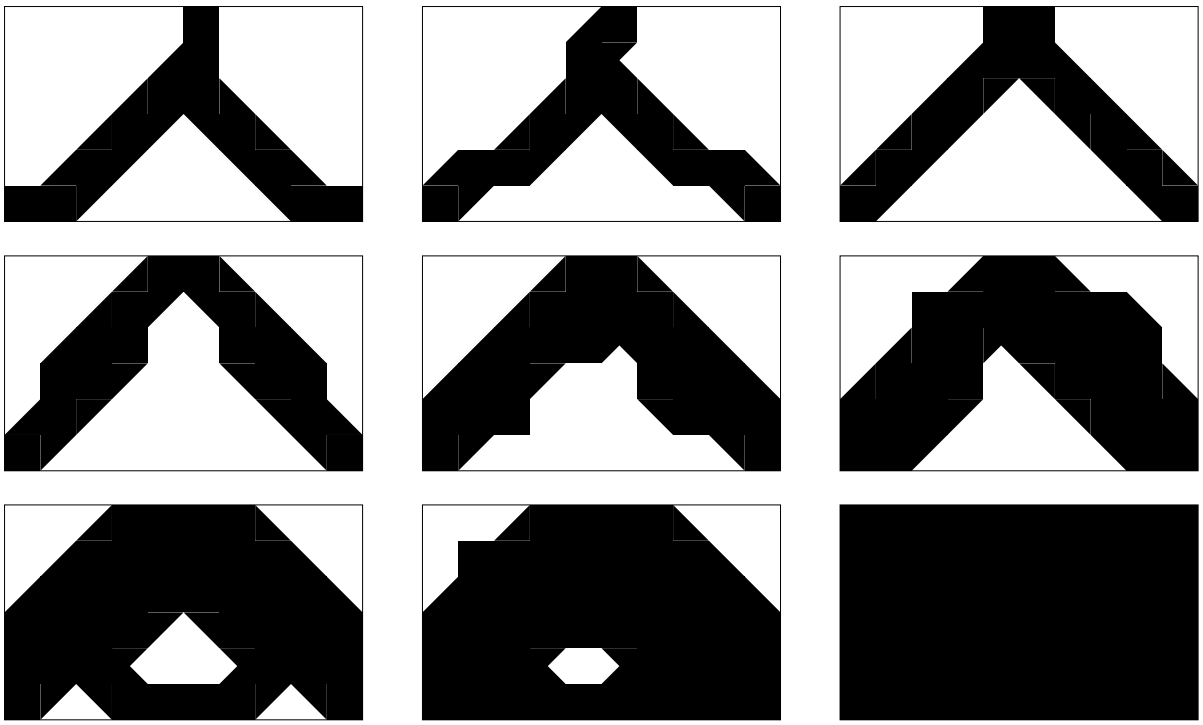

Figure 18: Nine trade-off shapes for the simply-supported plate design.

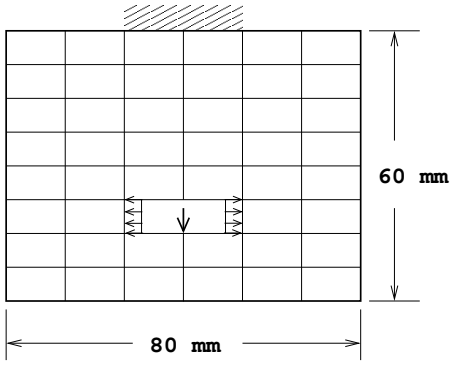

Figure 19: The loading and support of the hoister plate are shown.

good spread of trade-off solutions. When the local search method is applied from each solution and a non-domination check is made, there exists only 14 solutions, as shown in Figure 21. Moreover, most solutions concentrate near the middle of the previously-obtained front. Although two extreme solutions are found, the diversity among solutions is somewhat lost. Many solutions near the minimum deflection region in Figure 20 converge to an identical solution after the local search method. Although this allows the procedure to converge to a reduced set of non-dominated solutions, some important trade-off solutions may get lost by this process. This happens mainly due to the way we have allocated a pseudo-weight to a solution.

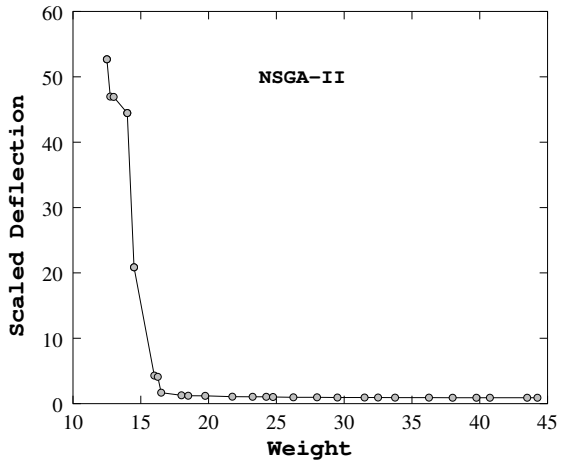

Figure 20: NSGA-II solutions for the hoister plate design problem.

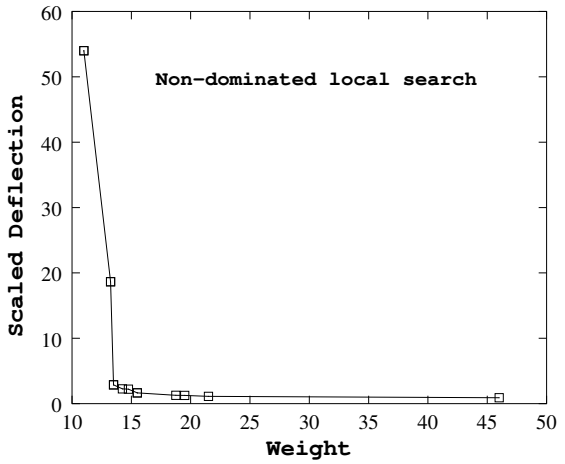

Figure 21: Solutions obtained after the local search method for the hoister plate design problem.

6.3.1 Continuously updated pseudo-weight vector

We have highlighted earlier that the pseudo-weight vector computation using equation 5 provides an approximate idea of the true weights associated with objectives at any solution. In order to get a better approximation of the true weight, we propose a continuously-updated pseudo-weight vector, where solutions are not assigned a fixed weight vector according to equation 5, instead the optimum found after each local search is used to update the weight vector.

Let us illustrate the procedure with the help of Figure 22 and for two objectives. For more than two objectives, a similar procedure can be adopted easily. First, the extreme solutions A

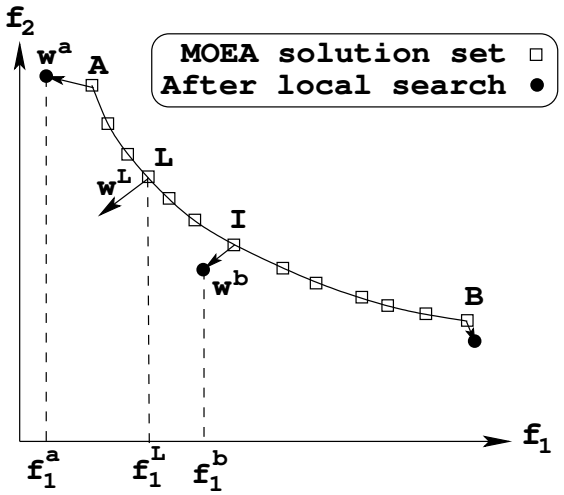

Figure 22: The continuously-updated weighted approach is illustrated.

and B are assigned a weight vector according to equation 5. A local search procedure is used to find the corresponding optimum solutions. Based on the new values of $f_j^{\min}$ and $f_j^{\max}$ values, an intermediate solution I is assigned a weight vector. Preferably, this intermediate solution I

can be chosen as the one having maximally away from the extreme solutions. The local search procedure can be used with this intermediate solution and the optimum solution can be found. Now we consider each of the two intervals: region between A and I and the region between I and A. An intermediate solution (preferably the solution in the middle of the range) is chosen in each range. Say the solution L is chosen in the range between A and I. The pseudo-weight vector for this solution is calculated as follows.

$$\bar{w}_j^{\mathrm{L}} = \bar{w}_j^{\mathrm{a}} \frac{f_j^{\mathrm{b}} - f_j^{\mathrm{L}}}{f_j^{\mathrm{b}} - f_j^{\mathrm{a}}} + \bar{w}_j^{\mathrm{b}} \frac{f_j^{\mathrm{L}} - f_j^{\mathrm{a}}}{f_j^{\mathrm{b}} - f_j^{\mathrm{a}}}, \tag{6}$$

where solutions a and b are extreme solutions for the range under consideration. Thus, for solution L, solution a is A and solution b is I. The parameters $\bar{w}_j^{\mathrm{a}}$ and $\bar{w}_j^{\mathrm{b}}$ are pseudo-weights associated with the extreme solutions. When pseudo-weights are calculated for objective functions, they can be normalized. This procedure can be repeated till the local search is applied to all solutions. This update of pseudo-weights with a local search procedure is advantageous in two ways:

1. The true optimum solution corresponding to a weight vector can be found, and
2. The calculated weights for each optimum solution is close to their true weights.

Figure 23 shows the obtained non-dominated solutions using the hybrid method with the above continuously updated pseudo-weight vector calculation. The figure shows that a good

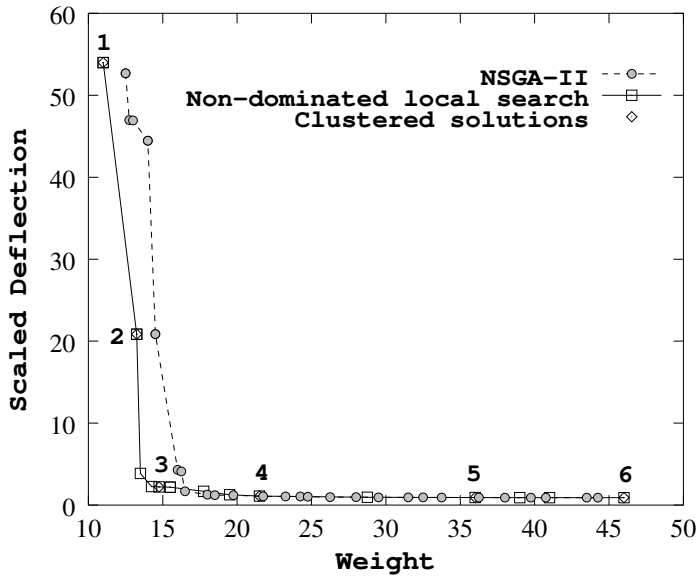

Figure 23: Hybrid procedure finds six trade-off solutions for the hoister plate design problem.

distribution of solutions is now possible to obtain with the continuously updated scheme. There are 17 solutions after the local search method, but importantly they are well distributed.

During clustering, we choose to have only six solutions, which are marked in the figure. The figure shows the spread of obtained solutions. It is interesting to note here that solutions 4, 5 and 6 have very similar deflection, but their weights are very different. Figure 24 shows the shape of six trade-off solutions. The minimum weight solution is our familiar hook, often used for hoisting purposes. What is interesting is that the opening in the left side of the hook evolves without any pre-defined knowledge. The minimization of overall weight finds the hook as the optimum

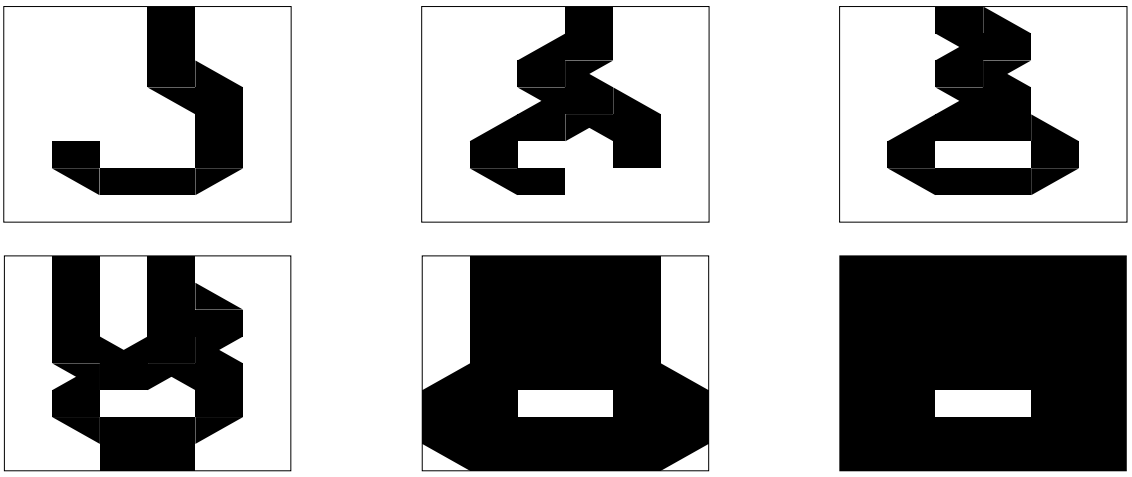

Figure 24: Six trade-off shapes for the hoister plate design.

design. It is also interesting that this design does not use two columns (left-most and right-most) and one row (bottom-most), in order to minimize the weight. From the support in the middle of the plate, the shape turns to accommodate the loading nodes, just like a human designer would do in designing a hook. As more stiff designs are desired, the gap in the hook closes and in the third solution, the loop is closed. This way, a much stiffer hoister plate is designed. Other three designs are different ways to make the plate more stiff, but mainly by significantly increasing the weight for a small gain in the stiffness.

6.4 Bicycle Frame Design

Finally, we attempt to design a bicycle frame for a vertical load of 10 kN applied at A in Figure 25. The specifications are similar to that used elsewhere [15]. The plate is 20 mm thick and is restricted to be designed within in the area shown in Figure 25. The frame is supported at two places B and

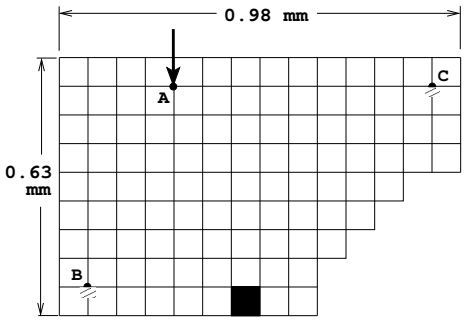

Figure 25: The hybrid procedure is illustrated for the bicycle frame design.

C. The point B marks the position of the axle of the rear wheel and the point C is the location of the handle support. The filled element is the location of the pedal assembly and is always present. The material yield stress is 140 MPa, Young's modulus is 80 GPa and Poisson's ratio is 0.25. The maximum allowed displacement is 5 mm.

Figure 26 shows the NSGA-II solutions and corresponding solutions obtained by the hybrid approach. Here, we are interested in finding four different trade-off solutions, marked in the figure.

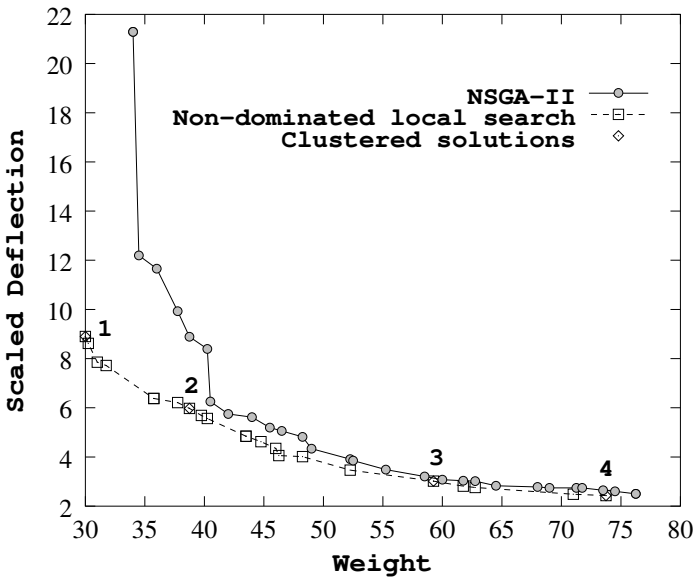

Figure 26: The hybrid procedure is illustrated for the bicycle frame design.

These four solutions are shown mounted on a sketch of a bicycle in Figure 27. The top-left

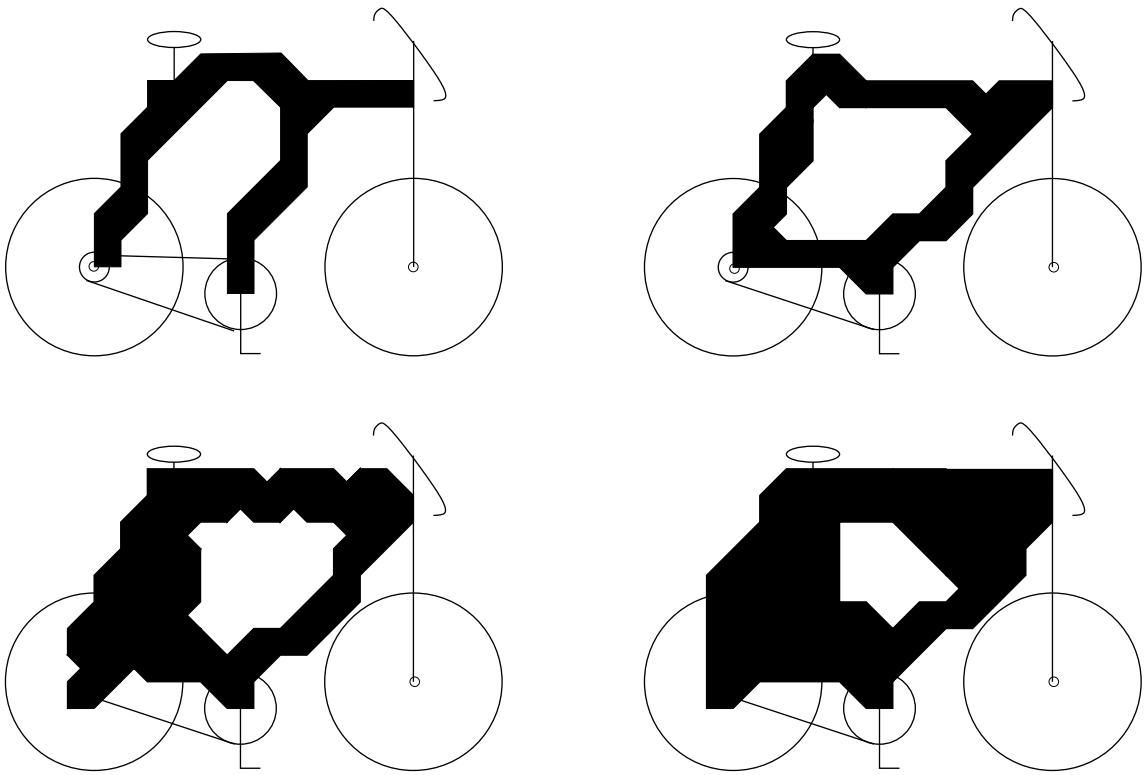

Figure 27: Four trade-off shapes for the bicycle frame design.

solution is the minimum weight design. The second solution joins the two vertical legs to make the structure more stiff. The other two solutions make the legs more thick in order to increase

the stiffness of the frame. The interior hole and absence of top-left elements are all intuitive. The proposed hybrid approach can evolve such solutions without these knowledge and mainly by finding and maintaining trade-off solutions among weight and deflection. The presence of many such solutions with different trade-offs between weight and stiffness provides a plethora of information about various types of design.

7 Conclusion

The hybrid multi-objective optimization technique proposed in this paper uses a combination of an multi-objective evolutionary algorithm (MOEA) and a local search operator. The proposed technique ensures a better convergence of MOEAs to the true Pareto-optimal region and helps in finding a small set of diverse solutions for practical reasons.

The efficacy of the proposed technique is demonstrated by solving a number of engineering shape design problems for two conflicting objectives—weight of the structure and maximum deflection of the structure. In all cases, the proposed technique has been shown to find a set of four to nine diverse solutions better converged than an MOEA alone. The results are encouraging and take the evolutionary multi-objective optimization approach much closer to practice.

Acknowledgments

The authors wish to thank Ministry of Human Resources Development (MHRD) for supporting this research.

References

[1] Chankong, V., and Haimes, Y. Y. (1983). *Multiobjective decision making theory and methodology*. New York:North-Holland.

[2] Chapman, C. D., Jakiela, M. J. (1996). Genetic algorithms based structural topology design with compliance and topology simplification considerations. *ASME Journal of Mechanical Design, 118*, 89–98.

[3] Chapman, C. D., Saitou, K., Jakiela, M. J. (1994). Genetic algorithms as an approach to configuration and topology design. *ASME Journal of Mechanical Design, 116*, 1005–1012.

[4] Deb, K., Pratap, A., Agrawal, S. and Meyarivan, T. (2000). A fast and elitist multi-objective genetic algorithm: NSGA-II. Technical Report No. 2000001. Kanpur: Indian Institute of Technology Kanpur, India.

[5] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. (2000). A Fast Elitist Non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 849–858.

[6] Deb, K. (1999) Multi-objective genetic algorithms: Problem difficulties and construction of test Functions. *Evolutionary Computation, 7*(3), 205–230.

[7] Deb, K. and Agrawal, R. B. (1995) Simulated binary crossover for continuous search space. *Complex Systems, 9* 115–148.

[8] Duda, J. W. and Jakiela, M. J. (1997). Generation and classification of structural topologies with genetic algorithm speciation. *ASME Journal of Mechanical Design, 119*, 127–131.

[9] Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms.* 416–423.

[10] Goldberg, D. E., Deb, K., and Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems, 6*, 333–362.

[11] Hamada, H. and Schoenauer, M. (2000). Adaptive techniques for evolutionary optimum design. *Proceedings of the Evolutionary Design and Manufacture.*, pp. 123–136.

[12] Harik, G., Cantu-paz, E. Goldberg, D. E., and Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation, 7*(3), 231–254.

[13] Horn, J. and Nafploitis, N., and Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multi-objective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation.* 82–87.

[14] Jakiela, M. J., Chapman, C., Duda, J., Adewuya, A., abd Saitou, K. (2000). Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*, 339–356.

[15] Kim, H., Querin, O. M., and Steven, G. P. (2000). Post-processing of the two-dimensional evolutionary structural optimization topologies. In I. Parmee (Ed.) *Evolutionary Design and Manufacture*, London: Springer. pp. 33–44.

[16] Knowles, J. and Corne, D. (1999) The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscataway: New Jersey: IEEE Service Center, 98–105.

[17] Miettinen, K. (1999) *Nonlinear multiobjective optimization*, Boston: Kluwer.

[18] Parmee, I. C., Cvetkovic, D., Watson, A. H., and Bonham, C. R. (2000). Multiobjective satisfaction within an interactive evolutionary design environment. *Evolutionary Computation, 8*(2), 197–222.

[19] Sandgren, E., Jensen, E, and Welton, J. (1990). Topological design of structural components using genetic optimization methods. *Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers*, pp. 31–43.

[20] Sen, P. and Yang, J.-B. (1998) *Multiple criteria decision support in engineering design.* London: Springer.

[21] Srinivas, N. and Deb, K. (1995). Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*(2), 221–248.

[22] Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and application.* New York: Wiley.

[23] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications. *Doctoral thesis ETH NO. 13398*, Zurich: Swiss Federal Institute of Technology (ETH), Aachen, Germany: Shaker Verlag.

[24] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation Journal, 8*(2), 173–196.

[25] Zitzler, E. and Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. *Technical Report No. 43 (May 1998).* Zürich: Computer Engineering and Networks Laboratory, Switzerland.