Automated Discovery of Innovative Designs of Mechanical Components Using Evolutionary Multi-Objective Algorithms

Kalyanmoy Deb and Shamik Chaudhuri
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
PIN 208016, India
{deb,shamik}@iitk.ac.in
http://www.iitk.ac.in/kangal/

KanGAL Report Number 2004006

Abstract

Automated design of different mechanical components, particularly in terms of their shapes and sizes, has been an important task in engineering design. In this chapter, we consider the design task as a simultaneous optimization of two or more conflicting objectives and use an evolutionary algorithm (EA) for finding multiple trade-off optimal solutions. An analysis of the optimized solutions reveals innovative design principles for solutions to become optimum, a task which is otherwise difficult to achieve by other means. The proposed methodology is generic, and its usefulness is demonstrated by applying it to a number of mechanical component shape design problems.

1 Introduction

Design of mechanical components, particularly for their shapes and sizes, has received plenty of attention in the past. Various methods have been used for this purpose. However, due to the advancement of genetic algorithms (GAs) – search and optimization algorithms motivated by the principles of natural evolution and genetics, the shape optimization of mechanical components have been dealt by discretizing the design domain in a number of finite sized elements and by forming a shape by deciding whether to keep or not keep each element [11, 5]. In this chapter, we use this strategy as a mode of developing optimal shapes for different applications.

Although most earlier studies on automated shape design seem to have concentrated in finding a shape which will minimize or maximize a particular objective (or a goal) of design, here we look beyond and attempt to unveil important properties related to the design which any optimum solution may possess. In this chapter, we restrict our computations for two conflicting objectives, although the proposed methods can also be extended to more than two conflicting objectives. We treat a design problem as a simultaneous optimization of two conflicting objectives and use an evolutionary multi-objective optimization (EMO) procedure to find the resulting Pareto-optimal (or non-inferior) solutions [2, 13]. The reason for using an evolutionary algorithm for the task is straightforward. A pair of conflicting objectives will give rise to a multitude of optimum solutions (known as Pareto-optimal solutions) and an evolutionary algorithm (including a GA) is an optimization algorithm which deals with more than one solution in an iteration. Thus, a GA is an ideal candidate for the task of finding more than one solution in one single simulation. After a set of trade-off optimal solutions are obtained, they will be analyzed to decipher useful design principles which all or most of the optimized solutions may possess. Such information, if known are useful to designers.

In the remainder of this chapter, we describe the EMO-based procedure and demonstrate the working principles by applying on four different mechanical engineering design problems. The suggested procedure is generic and can also be applied to similar other engineering design activities.

2 Evolutionary Multi-Objective Optimization (EMO)

The practice of evolutionary algorithms (EAs) in engineering applications, although a recent phenomenon, has been well demonstrated. In most applications, EAs are applied in finding an optimum or a near-optimum solution of a single-objective optimal design problem. For the past five years or so, EAs are increasingly being used in solving multi-objective optimization problems, involving more than one objective of design, such as minimization of cost, maximization of reliability, and others. It is intuitive that a low-cost solution is less-reliable and vice versa. Therefore, while optimizing both cost and reliability, a designer faces with a dilemma: Which of the two objectives is to be optimized? Theoretically, such multi-objective optimization problems give rise to a set of optimal solutions, each trading off the two objectives in a different manner. It then becomes a task on the designer to choose one such solution by using some other *higher-level* information. Although the literature on multi-objective optimization using classical methods is comparatively old and growing, the research and application of evolutionary multi-objective optimization (EMO) introduces a completely different way of solving such problems. By first finding a well-distributed set of Pareto-optimal solutions, EMO provides useful information about the working range of each objective, which in turn should help a designer to choose a particular solution at the end of optimization process.

Based on the above principle of evolutionary multi-objective optimization, there now exist a number of efficient algorithms, such as non-dominated sorting GA (or NSGA-II) [4], strength Pareto EA (or SPEA) [6], Pareto-envelope evolution strategy (or PESA) [1] and others. Interested readers may refer to a recent introductory text [2] by the first author on these algorithms and the working principles of EMO methodologies. In this chapter, we use NSGA-II [4] as a representative EMO methodology, a brief discussion of which is given in the next section.

3 An Evolutionary Multi-Objective Optimizer: NSGA-II

Contrary to the classical approaches, evolutionary multi-objective optimization (EMO) procedures aim at finding a finite, representative set of Pareto-optimal solutions of a problem. Figure 1 illustrates this aspect on a hypothetical three-objective minimization problem. The task of an EMO procedure is to *identify* the Pareto-optimal front (shown separately as a surface) out of the entire feasible search space. This is, by no means, an easy task. Various optimization concepts must have to be used to clearly identify the true Pareto-optimal front. In the next section, we shall demonstrate how different optimization concepts can be used to discover the Pareto-optimal front for a number of mechanical component design problems.

Although one optimal solution would be chosen at the end of the optimization task, a knowledge of the entire range of Pareto-optimal solutions is helpful in (i) choosing a particular optimal solution and (ii) getting useful insights about trade-off among objectives of the problem [3]. Thus, the task of an EMO is to (i) converge to the true Pareto-optimal front and (ii) maintain a good distribution of solutions on the entire front. Here we describe the elitist non-dominated sorting GA or NSGA-II procedure [4] for achieving both the above tasks in a multi-objective optimization problem.

Like in a genetic algorithm (GA), NSGA-II starts with a population of N_{pop} random solutions. In the N-th iteration of NSGA-II, the offspring population Q_N is first created by using the parent population P_N and the usual genetic operators – reproduction, recombination, and mutation [8].

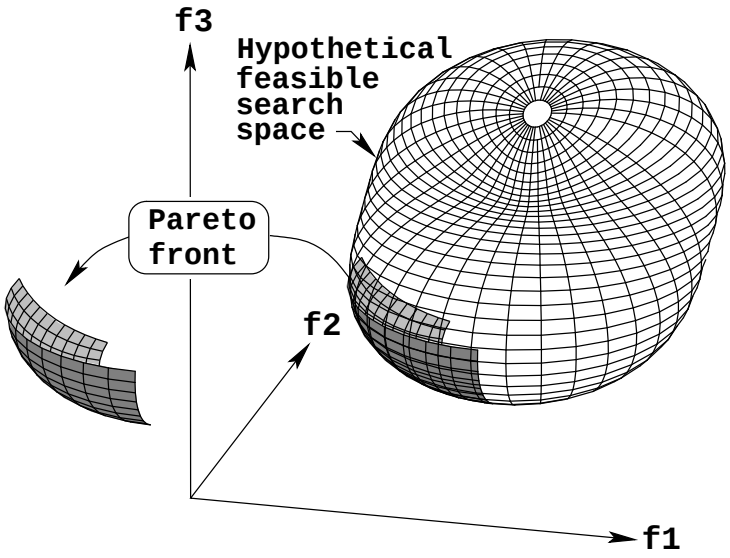

Figure 1: A three-objective feasible region and the corresponding Pareto-optimal front are shown.

Thereafter, both populations are combined together to form R_N of size $2N_{\text{pop}}$. Then, a non-dominated sorting procedure [2] is applied to classify the entire population R_N into a number of hierarchical non-dominated fronts. Figure 2 shows a schematic of one iteration of NSGA-II. Fronts are marked as 1,2,... in the decreasing level of non-domination of solutions. A property of such fronts is that a solution lying on a better non-dominated front (say $\mathcal{F}$) does not get dominated by any solution of a front worse than $\mathcal{F}$. For M objectives, there exists an algorithm requiring $O(N \log^{M-1} N)$ computations to make such a non-dominated sorting of the entire population of size N [12].

Once the non-dominated sorting of the set R_N is over, the new population is filled with solutions of different non-dominated fronts, one at a time. The filling starts with the best non-dominated front (marked as front 1) and continues with solutions of the second non-dominated front, followed by the third non-dominated front, and so on. Since the overall population size of R_N is $2N_{\text{pop}}$, not all fronts may be accommodated in N slots available in the new population. This scenario is shown in Figure 2. All fronts which could not be accommodated (such as, fronts 4, 5 and 6 in the figure) are simply deleted. When the last allowed front (front 3 in the figure)is being considered, there may exist more solutions in the front than the remaining slots in the new population. Instead of arbitrarily discarding some members from the last front, the solutions which will make the *diversity* of the selected solutions the largest are chosen. In this step, a *crowded-sorting* of the solutions of front i (the last front which could not be accommodated fully) is performed by using a *crowding distance metric* and the adequate number of solutions are picked from the top of the list. The crowding distance of a solution in a non-dominated front is a measure of crowding by other members of the front. In the NSGA-II implementation, a metric totalling the objective-wise distances between neighboring solutions is used. For details, readers are encouraged to refer to the original NSGA-II study [4]. The reproduction operator uses a *crowded tournament* selection method in which two solutions are compared and the winner is declared as the one residing on a better non-dominated front, or the one having a larger crowding distance value. The computational complexity of one iteration of NSGA-II is dictated by the non-dominated sorting complexity mentioned above.

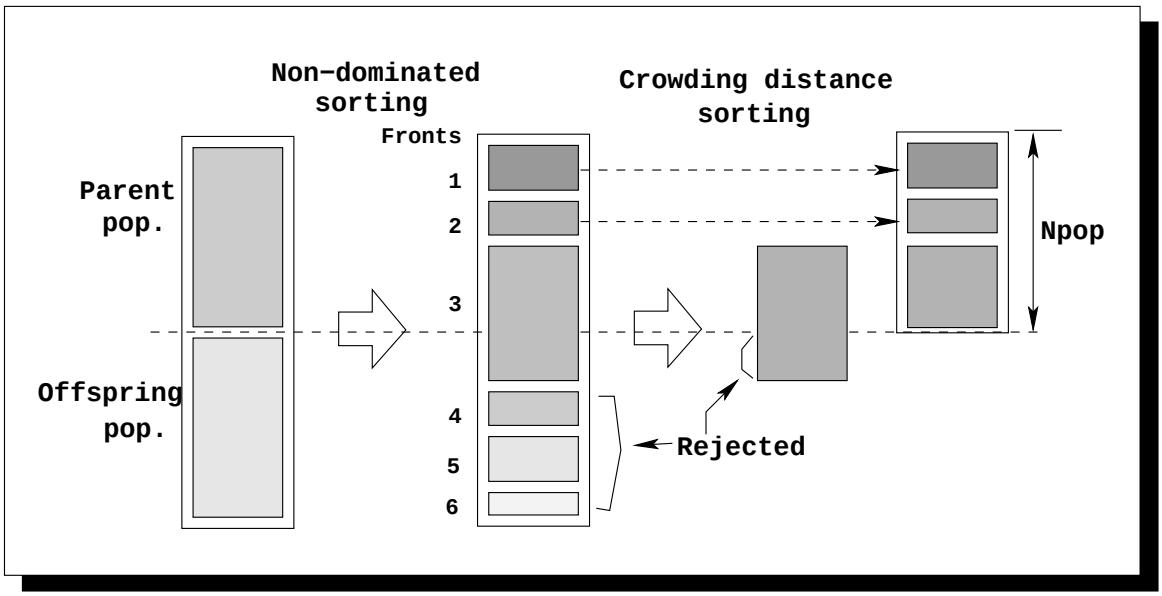

Figure 2: Schematic of the NSGA-II procedure.

3.1 Constraint-Handling in NSGA-II

The constraint handling method modifies the crowded tournament selection operator of NSGA-II by checking the feasibility of two participating solutions [2]. In the presence of constraints, each solution can be either feasible or infeasible. Thus, there may exist at most three situations: (i) both solutions are feasible, (ii) one is feasible and other is not, and (iii) both are infeasible. We consider each case by simply redefining the domination principle as follows for any two solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$).

A solution $\mathbf{x}^{(i)}$ is said to 'constrain-dominate' a solution $\mathbf{x}^{(j)}$, if any of the following conditions are true:

1. Solution $\mathbf{x}^{(i)}$ is feasible and solution $\mathbf{x}^{(j)}$ is not.
2. Solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are both infeasible, but solution $\mathbf{x}^{(i)}$ has a smaller constraint violation.
3. Solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are feasible and solution $\mathbf{x}^{(i)}$ dominates solution $\mathbf{x}^{(j)}$ in the usual sense.

The highlight of this procedure is that it makes a simple adjustment on the definition of the domination and does not require any additional parameter, such as the penalty parameter. This is because feasible solutions are always better than infeasible solutions and are sorted into different non-dominated levels in the usual way. However, the infeasible solutions get sorted depending on their aggregate normalized constraint violation values – an infeasible solution having a larger constraint violation lies on a worse non-dominated front.

4 EMO for Discovering Useful Design Variants

In this section, we describe how the EMO methodology discussed above is used for the task of finding useful design variants for engineering shape design problems. We divide the task into two phases: (i) finding a set of trade-off optimal solutions using an EMO and (ii) analyzing the obtained solutions for useful insights.

4.1 Phase I: Finding a Set of Pareto-Optimal Solutions

In principle we can simply use an existing EMO, such as NSGA-II described in the previous section, for achieving the first task mentioned above. Here, we employ a hybrid search strategy of using an EMO with a *local-search* method to make sure that the obtained solutions are indeed very close to the real optima of the problem. Such a hybrid strategy was first suggested elsewhere [2, 5] and is also a quick and reliable procedure to reach near optima. However, a number of other hybrid EMO approaches are also suggested elsewhere [9, 10].

In this approach, first, an EMO is applied without the use of any preference information, such as a weight-vector often used to scalarize multiple objectives. Here, we have used NSGA-II [4] for this purpose. After some generations are elapsed (thereby allowing the EMO procedure to reach near to the true Pareto-optimal front), a local-search algorithm is applied from each of the obtained EMO solutions. This local-search algorithm can be a variant of classical hill-climbing procedures [14, 15]. However, since a single-objective function is required for the hill-climbing procedure, the multi-objective problem is reduced to a single-objective problem by using a weighted-sum of different objectives. The scaled single-objective function is given below:

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

where, $f_j(x)$ is the j^{th} objective function and $\bar{w}_j^x$ is the weight to the j^{th} objective function. In this approach, the weight vector decides the importance of different objectives. In other words, the weight-vector provides a direction of the local-search procedure in the objective space. In the original study, the weights were assigned with the knowledge of a solution's location on the Pareto-optimal front [5]. When a solution is near the minimum value of a objective function, the weight of that function will get a large value. The weight calculation can be performed as follows [5]:

$$\bar{w}_j = \frac{(f_j^{max} - f_j^{(x)})/(f_j^{max} - f_j^{min})}{\sum_{k=1}^{N}(f_k^{max} - f_k(x))/(f_k^{max} - f_k^{min})}, \tag{2}$$

where, $\bar{w}_j$ is the weight for the j^{th} objective, f_j^{max} is the scaled maximum value of the j^{th} function in a EMO population, $f_j(x)$ is the scaled value of the j^{th} function, and f_j^{min} is the scaled minimum value of the j^{th} function in the population.

It is noteworthy that the use of above weighted approach may not be useful in problems having non-convex Pareto-optimal front and other difficult problems. In such problems, the methodology suggested for a local-search technique can be replaced by using a Tchebycheff metric [2, 13] or by using Benson's method [7]. The essential idea is to employ a local-search method from each optimized solution obtained using the EMO procedure and obtain the corresponding optimum solution. Thus, at the end of the first phase, we are expected to obtain a set of near Pareto-optimal or Pareto-optimal solutions.

4.2 Phase II: Analyzing Optimized Solutions for Useful Properties

The obtained set of Pareto-optimal solutions can then be analyzed for finding possible *commonality* principles among all such solutions. Although not intuitive, such commonality principles are expected to be present in most engineering design problems. Qualitatively, since all these solutions are optimal (and are definitely not arbitrary solutions in the search space), they may follow certain properties for being optimal. Quantitatively, for any multi-objective optimization problem, all Pareto-optimal solutions must satisfy the Fritz-John optimality conditions given below [13]:

Theorem 1 *(Fritz–John necessary condition) A necessary condition for* $\mathbf{x}^*$ *to be Pareto-optimal is that there exist vectors* $\boldsymbol{\lambda} \geq \mathbf{0}$ *and* $\mathbf{u} \geq \mathbf{0}$ *(where* $\boldsymbol{\lambda} \in \mathbb{R}^M$, $\mathbf{u} \in \mathbb{R}^J$ *and* $\boldsymbol{\lambda}, \mathbf{u} \neq \mathbf{0}$*) such that the following conditions are true:*

1. $\sum_{m=1}^{M} \lambda_m \nabla f_m(\mathbf{x}^*) - \sum_{j=1}^{J} u_j \nabla g_j(\mathbf{x}^*) = \mathbf{0}$, *and*
2. $u_j g_j(\mathbf{x}^*) = 0$ *for all* $j = 1, 2, \ldots, J$.

Although the above set of conditions require that the objective functions and constraints are required to be differentiable, they collectively indicate that for a solution to be Pareto-optimal, it must satisfy certain optimality conditions. Now, it is very well be true that for every Pareto-optimal solution such conditions give rise to a different set of properties and when all Pareto-optimal solutions are considered together there is no commonality principle among all Pareto-optimal solutions. Although this is a possibility theoretically, it is unlikely for most engineering problems due to the underlying scientific laws which the engineering systems abide by. In any case, for a particular problem of interest, it is worth investigating whether there exist such commonality principles among all Pareto-optimal solutions.

If in a problem there exists some commonality principles, they are undoubtedly important to the decision-maker or the user. Once such principles are known, the user can easily change the existing operating optimal solution to another by maintaining the commonality principles identical between them but simply changing the properties which must vary among them. Moreover, the knowledge of such commonality principles will provide important insights about the engineering system under consideration.

In short, after a set of Pareto-optimal solutions are found in Phase I, their decision variable vectors can be investigated for the existence of any *common* principles and *principles* which make them lie on different parts of the Pareto-optimal front. In all problems here, we make a visual investigation for finding such principles (and in most cases such a visual inspection may be enough); a more systematic investigation can also be used. We are currently devising an automatic method of deciphering such common principles from the obtained Pareto-optimal set by using machine learning principles, results of which will be communicated at a later date.

We now suggest a systematic multi-objective optimization procedure used in this study for solving engineering design problems having multiple objectives.

5 Proposed Multi-Objective Optimization Procedure

The methodology described in Phase I is extended to develop an efficient optimization procedure:

1. **Varying the grid size:** Here, we suggest varying the grid (or mesh) size from coarse to fine. We recognize that at the start of an optimization process, it is not necessary to evaluate a solution very precisely. However, as iterations progress, more precise evaluations are necessary. Many researchers implement this aspect in many different ways so as to reduce the overall computational efficiency, but without sacrificing the solution quality. Here, we simply use a coarse grid to start with and as iterations proceed, we make the grid finer and finer by simply reducing the grid size in each dimension to half.

2. **Improved crossover operator:** The standard row or column-wise crossover operator may be used for a crude mesh. But for a fine mesh, swapping rows or columns is too generic to mix the properties of two parents to come up with a good offspring solution. Not only that, the operator is also discrete in nature, that is, the rows or columns, which are interchanged may not be contiguous. For this reason, here we swap a contiguous rectangular region (not necessarily spanning to a complete row or a column) of `1` and `0` between two parents to create two offsprings.

3. **A fast local-search operator:** The local-search algorithm used in a past study [5] was not a particularly efficient one. It starts searching the domain by mutating each and every bit of the string under consideration. If the element (representing a void) under mutation is far away from the main body of the design, application of the mutation operator will turn it into a material element from a void element. This newly-formed material element will not be a part of the main cluster and will not alter the mesh used in the FEM analysis. Thus, there will not be any improvement in the scaled objective function. Due to the above reason, the search effort will be wasted. Here, we only restrict the mutation operator (or the local-search operator) to boundary elements dictated by the shape. Figure 3 shows the domain for mutation operator (shaded elements). For element A, only eight elements around it will be attempted to mutate. Elements, such as B, will not be mutated because of non-existence of any material member around it.

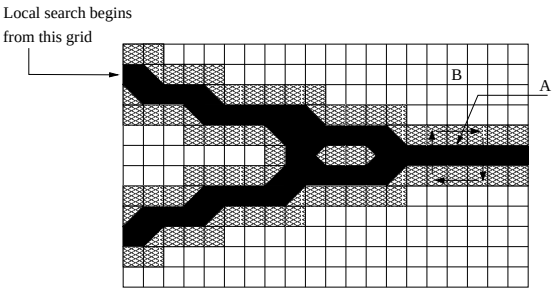

Figure 3: The fast local-search procedure.

4. **Distributed computing:** The EMO algorithm used in the earlier study [5] was a serial one. The time required to complete one simulation of the optimization procedure on a Pentium II, 350 MHz serial computer was around 2 days. To reduce the computing time, we use a distributed computing machine and population members are computed in a parallel fashion, thereby reducing the overall computational time within a few hours.

   After the EMO run, the non-dominated solutions are clustered and representatives are sent for further improvement using the local-search procedure. At this stage, processing of different solutions are independent of each other. So parallelization can also be applied in this stage. The maximum speed up that can be available is $T_{\text{local-search}}/N$ (assuming that each solution takes an identical processing time). Here, N is the number of solutions in local-search and $T_{\text{local-search}}$ is the time required for a single processor to process N solutions. In this module, a parallelization is implemented in a different way. Here, all the individuals are queued up. If there are N processors, then processing starts with the first N solutions. Whenever processing on a particular processor is over, a queued solution is sent for processing to that processor.

5.1 Iterative local-search based EMO

Initially, the grid size (that is, the size of the square elements) of the search domain is large, so the NSGA-II procedure can remove large chunks of material from the initial random shape within a few generations, if needed. If the grid size is large, the required computation time will be small. The non-dominated front after the NSGA-II run is clustered and the representative solutions are taken for further refinement. To make the solutions smoother, the mesh is refined by dividing each square of the crude mesh into four equal squares. Thereafter, a fast local-search algorithm is applied to a non-dominated solution with a composite objective function, formed by the location of the solution on the non-dominated front [5]. Using this level of refined mesh, the search moves

further and ultimately until no further improvement in the combined objective value is obtained. At this stage, the mesh is refined again in the same way. The local-search is applied again and the shapes are made smoother. The step-by-step procedure is illustrated in Figure 4.

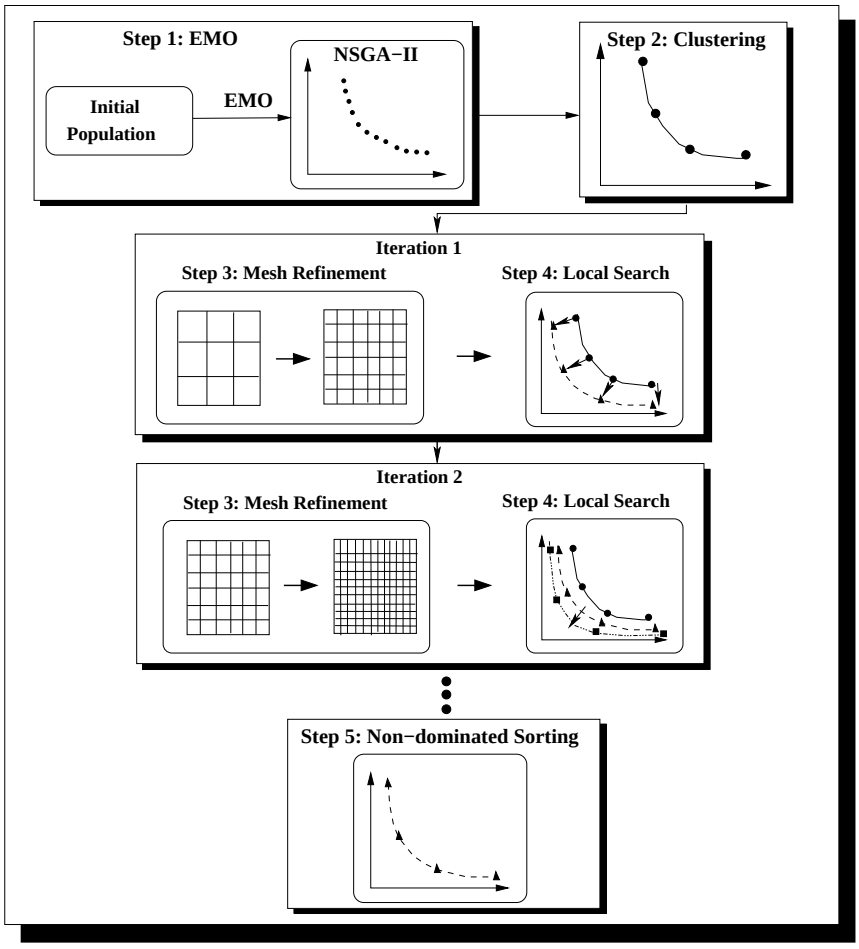

Figure 4: A flowchart of the proposed local-search based EMO procedure (Phase I).

6 Simulation Results

In this section, we present simulation results for a number of mechanical component design problems.

6.1 A Cantilever Plate

The cantilever plate is designed with an end load of P = 100 kN as shown in Figure 5. The thickness of the plate is 20 mm. For the NSGA-II simulation in the first phase, the grid size taken is chosen to be (6×10), such that a solution is represented by a 60 bit binary string. Thereafter, the grids are increased to 12×20 for the first local-search. Grids are again increased to 24×40 before the second local-search is begun. Geometry constraint is implemented by keeping at least one material element at the left-most column of the grid and also at the elements adjacent to the node where the force is applied. The different parameters used in the analysis are given below:

Material Data :

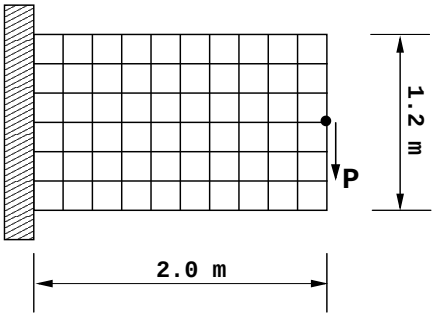

Figure 5: Loading and support of the cantilever plate.

Density of Material	: 7800 kg/m^3
Yield strength	: 150 MPa
Young's Modulus	: 200 GPa
Poisson's ratio	: 0.25

NSGA-II Parameters :

Population Size	: 54
Number of Generations	: 100
Crossover Probability	: 0.95
Mutation Probability	: 1/(string length)

The solutions on the non-dominated front after the NSGA-II run (after Step 1) is shown in the Figure 6. The non-dominated solutions (marked using diamonds) are plotted along with the initial population (marked using '+'). Nine well-distributed solutions are picked up from the entire non-dominated front for further refinement. These representative solutions are shown in Figure 7. The actual geometrical shapes of these solutions after the NSGA-II run are shown in Figure 8. The solutions are numbered in a matrix notation. The top left figure shows the minimum-weight structure having a large displacement and the same at the bottom right is the minimum-deflection solution having a large weight. It is clear from the figure that NSGA-II is capable of developing crude structures in only 100 generations using a coarse mesh. These representative shapes are further processed by the local search procedure. Figure 7 showed the position of these solutions after the first local-search (Step 4 of the first iteration). The movements of the solutions in the objective space from NSGA-II solutions to local-search solutions are marked with the help of arrows. It is evident that remarkable improvements have taken place by the local-search operation.

Figure 9 shows the new topology of the structures. At this stage, the grid size is refined from 6×10 to 12×20. The minimum-weight solution for this problem is likely to have a reducing width towards the end load. To have the widest base, at the support, the obtained minimum-weight solution spreads two arms to the maximum extent possible. It is interesting to note that without any such information hard-coded, the optimization procedure adopted here arrives at such a solution in successive iterations of the local-search procedure. Figure 10 represents the plot of the non-dominated fronts after the second local-search procedure (Step 4 of iteration 2). Here also the movements of the solutions are marked by arrows. The minimum-weight solution of first local-search shown at the top left position in Figure 9 becomes a dominated solution after the second local-search. This is because of the zigzag shape of the two arms of the parabola. As the local-search process tries to remove material from the arms, the stress generated inside the structure exceeds the allowable load. The shape at the next position moving horizontally from the top left corner, is much smoother and symmetric and after second local-search the solution placed at (1,2) dominates the solution placed at (1,1). At the end, the final non-dominated front consists of eight non-dominated solutions. Figure 11 shows the final optimized configurations

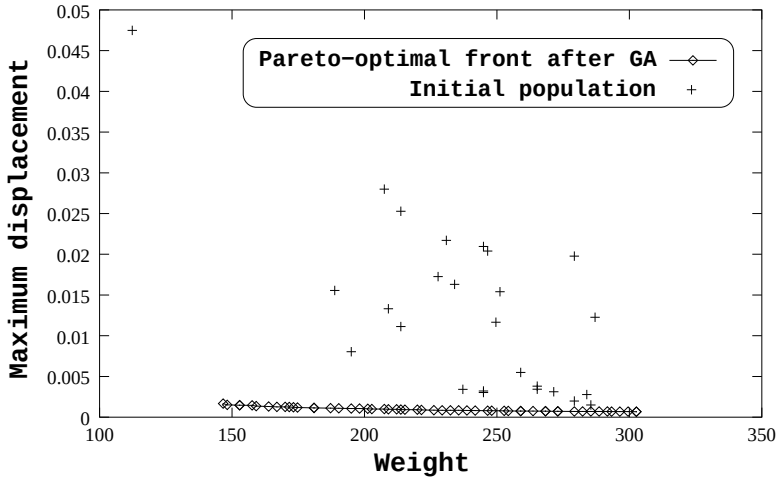

Figure 6: Non-dominated solutions after NSGA-II simulation on the coarse grid.

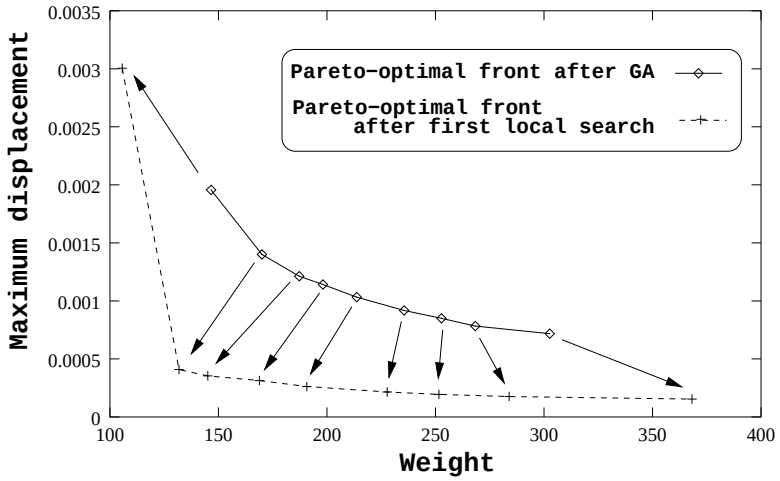

Figure 7: The non-dominated solutions move during the first local-search.

(after Step 5) of the cantilever plate problem. The shapes are presented in the same fashion, that is, in order of minimum-weight to minimum-deflection solutions. The minimum-weight solution at position (1,1) closely resembles the intuitively smallest weight solution. The next solution at (1,2) has a vertical stiffener joining the two long arms of the minimum-weight solution. Moreover, due to the stiffener the arm thickness is also reduced at the tip. The shape at (1,3) is the next stiffened solution having no stiffener. Here, the arms are made thicker providing a larger strength than that of solution (1,2). Solutions at (2,1), (2,2), (2,3) and (3,1) rediscover the stiffener and are of the same kind. The arm and the stiffener thickness increase gradually and hence the corresponding deflection is reduced. Finally the complete plate with right-most top and bottom ends chopped off as shown at (3,2), is the minimum-deflection solution. The achievement of an intuitive, thin, two-armed shape to the innovation of adding stiffeners to the complete covering of almost the entire plate and importantly the symmetry about the horizontal plane in the middle of the plate in all obtained solutions are not explicitly coded or programmed. The procedure adopted here has the adequate power to discover interesting and important design variants adaptively through an efficient optimization procedure.

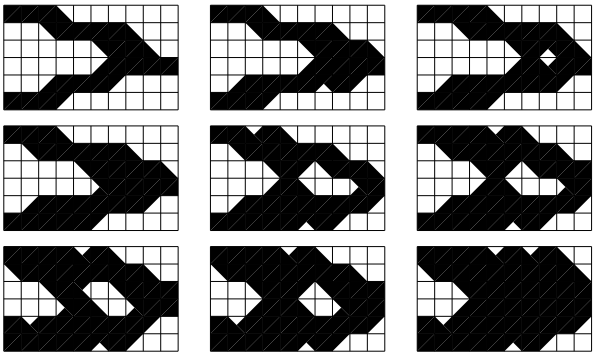

Figure 8: Non-dominated solutions after NSGA-II run.

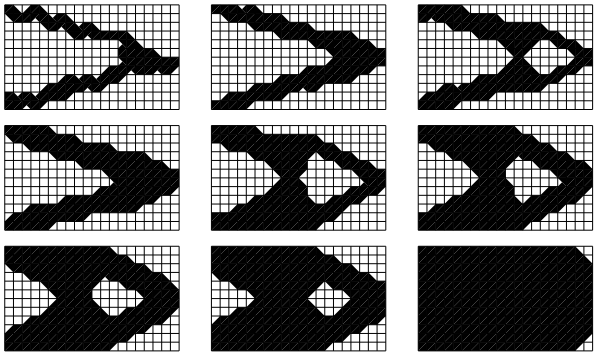

Figure 9: Non-dominated solutions after the first local-search procedure.

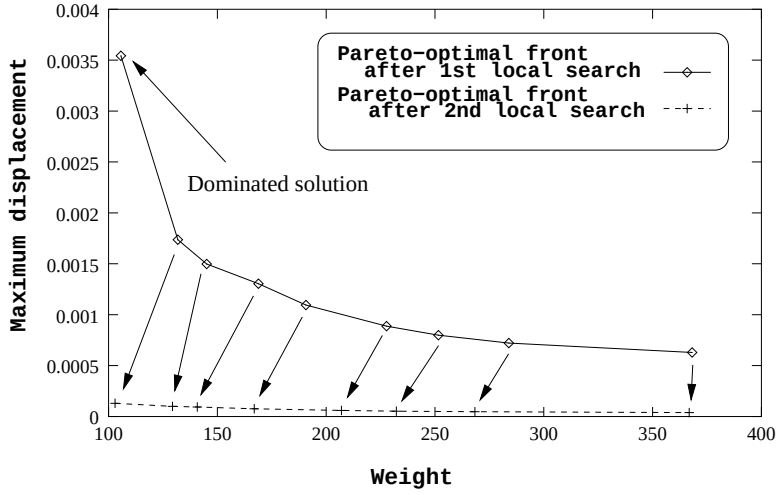

Figure 10: Non-dominated solutions move after the second local-search.

6.1.1 Overall function evaluations:

The number of function evaluations required to form the above shapes from random shapes is given in the Table 1. It is interesting to note that the second local-search requires more evaluations than the first local-search. With successive local-searches, the mesh gets more refined, meaning that there are more elements involved in the decision-making. With more variables, it takes more iterations of the local-search procedure to arrive at an optimum solution. The total function evaluations required by the NSGA-II procedure are 5,400. Thus, the total number of function evaluations for the design of the cantilever plate including the local-searches are 23,281. The processing time is 57.06 hours on a 18-processor Pentium-III (1 GHz) PC cluster.

6.2 Comparison with NSGA-II Alone

A comparative study with NSGA-II is done on the basis of the above data. To do so, the shape optimization problem is run only using NSGA-II for the same number of function evaluations. The final non-dominated front obtained using NSGA-II is compared with the same obtained using the local-search based NSGA-II procedure described above. Figure 12 shows the non-dominated front of the two runs. It is clear from Figure 12 that only NSGA-II could not reach the obtained

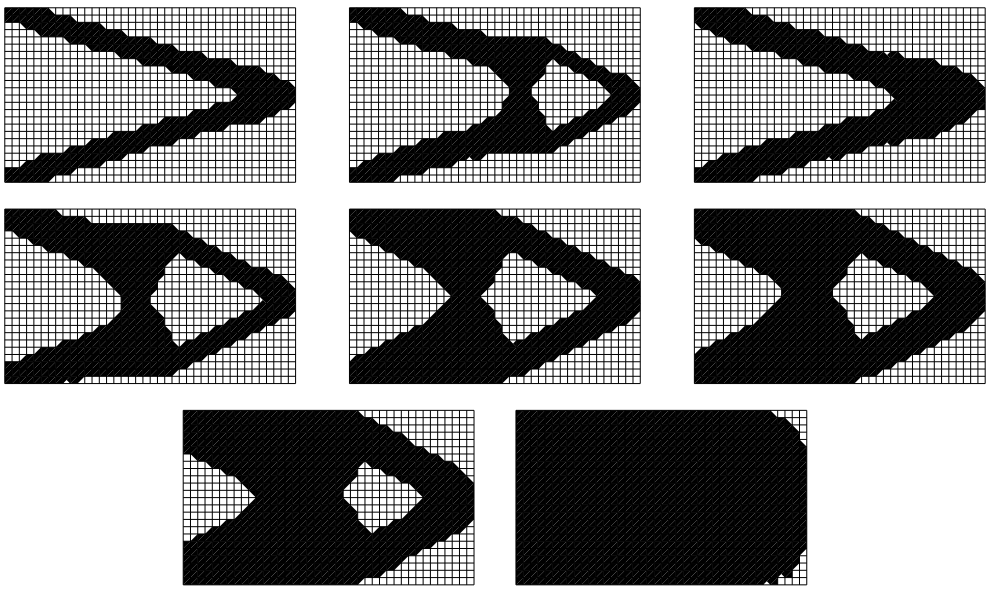

Figure 11: Non-dominated solutions after the second local-search procedure (at the end of Step 5).

Table 1: Function evaluation for the design of the cantilever plate. The 'Dom. Sol.' represents a dominated solution, shown in Figure 10.

Shape No. →	1	2	3	4	5	6	7	8	Dom. Sol.
1st local-search	696	755	672	740	411	653	358	198	410
2nd local-search	2,187	2,423	1,480	2,003	1,804	1,424	1,013	324	330
Total	2,883	3,178	2,152	2,743	2,215	2,077	1,371	522	740

non-dominated front after the same number of function evaluations. The extent of prematurity in obtained solutions can by found by investigating the shapes generated using only NSGA-II, shown in Figure 13. The solutions are certainly inferior to the solutions developed by the local search based NSGA-II (shown in Figure 11). Here, all solutions seem to show a similar shape with internal voids. The NSGA-II makes a decision of removing a material element from the topology on the basis of minimizing the maximum deflection and the weight of the structure. Since the mesh size is small, the contribution of one element to the overall deflection is also small. So it is difficult for NSGA-II to generate practically viable structures. The local-search based method makes a balance between a population-based EMO procedure and the point-based local-search method. The overall procedure is capable of arriving at multiple optimal shapes (shown in Figure 11) in a systematic manner.

6.3 A Simply-Supported Plate

The next problem is the design of a simply-supported plate, with a load applied on the top middle part on the plate, as shown in Figure 14. In this case, the plate thickness is 20 mm. The bottom left corner of the plate is pinned and the bottom right has a roller support. To create a valid shape, material elements are imposed at the extreme left and right elements of the bottom row as shown in the figure. The material properties and NSGA-II parameters used here remain the same as those in the previous case. The non-dominated front after the NSGA-II run is presented with the initial population in Figure 15. It is interesting to note that the initial generation had

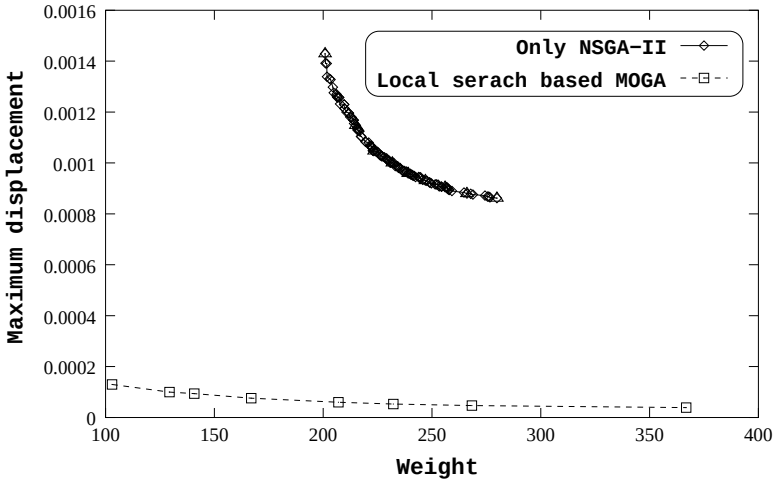

Figure 12: Non-dominated front obtained by the NSGA-II run alone.

only one feasible solution (shown by a '+') on the window used to plot the figure. The constraint-handling method used with NSGA-II [2] drifts infeasible solutions towards the feasible region in a systematic fashion. Figure 16 presents the representative non-dominated front obtained after the first local-search procedure. It is clear from the figure that the movements of the solutions are not as regular as in the case of the cantilever plate design application. The direction of movement of the solutions depends on the weight vector associated with the solutions and also with the complexity of the problem. Besides the non-linearity in the objectives and constraints, the arbitrary location of the material elements introduces further complexity. If material is added near the support locations, maximum deflection of the structure decreases. But if material is added near the load point, due to the self-weight of the added material, the maximum deflection increases. Due to the above reason, zigzag movements of the solutions in the objective space may take place. Figure 17 plots the non-dominated front after the first and second local-searches. In this case also the movements of the solutions are erratic in nature. All nine solutions obtained by the complete local-search based EMO procedure (that is, at the end of Step 4 of Figure 4) are presented in Figure 19. However, all of them are not non-dominated solutions, as evident from Figure 17. The completely-filled solution (shown in position (3,3) in Figure 19) is a dominated solution. Though it possess a large weight, the self-weight deflection of this solution is more than the deflection of a lesser-weight solution shown at (2,2), thereby making the completely-filled solution dominated. It is intuitive that the optimal shape for this type of a problem would be a truss-like structure. The general tendency of the obtained solutions are truss-like (although the joints of the bars are not pinned). The final four non-dominated solutions are presented in Figure 20. The plot of final non-dominated front showing four solutions in the objective space is given in Figure 18. Only truss-like shapes emerge at the end of the local-search based EMO procedure. The minimum weight solution (1,1) is a convex structure with a connection from the main body to the load point. The convex shape of the main body gives more strength than a straight horizontal connection between the supports. The second solution (1,2) is a triangle with a vertical connection between the top vertex and the load point. This shape looks like a truss (only from the view point of the shape). This is a symmetric structure, which has evolved without any prior information about symmetry. The next solution (at (1,3)) is a variant of the previous solution, having more material at the support position. The minimum-deflection shape is also a triangle with a connection from the base to the left inclined link. This extra link provides more rigidity and hence this is the minimum-deflection solution obtained by the proposed EMO

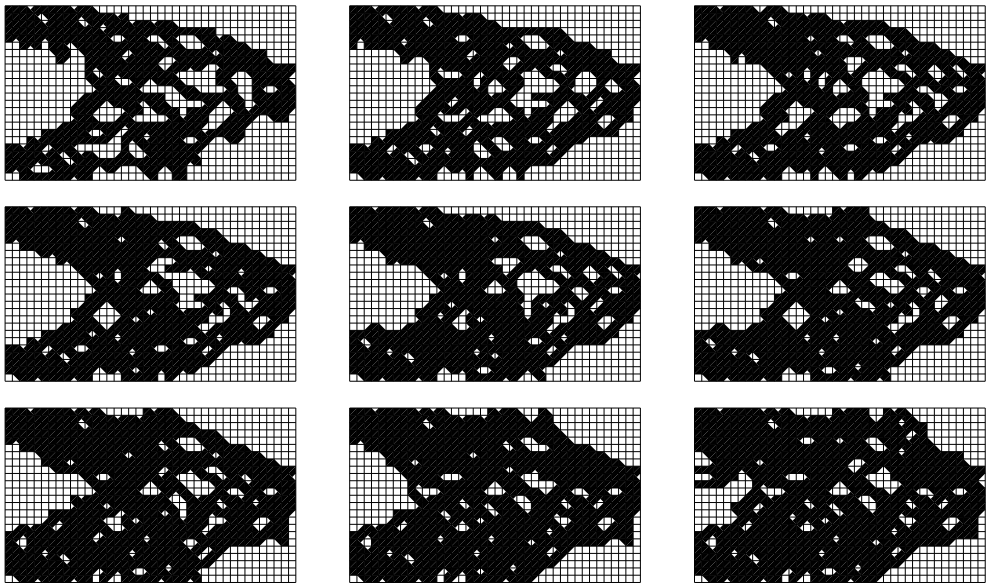

Figure 13: Nine trade-off shapes from NSGA-II run alone.

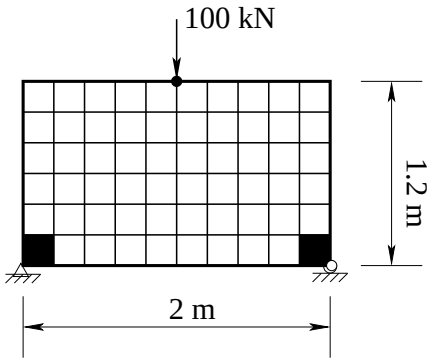

Figure 14: Boundary and loading conditions for simply-supported plate.

procedure.

Figure 18 suggests that the second-best weight solution ((1,2) in Figure 20) which is a truss-

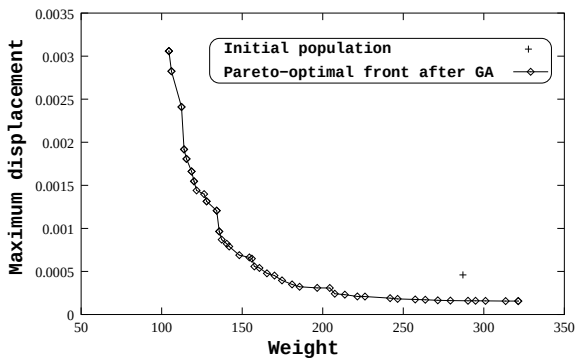

Figure 15: Non-dominated front after the NSGA-II run.

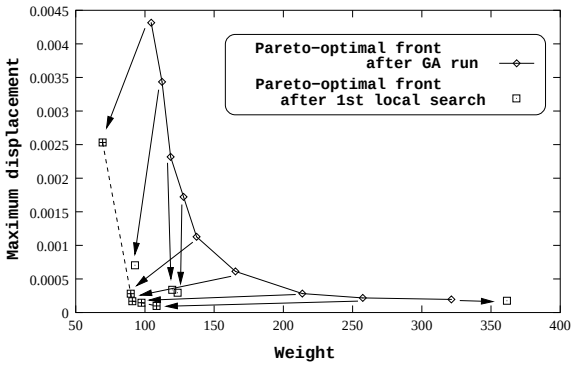

Figure 16: Non-dominated front after the first local-search.

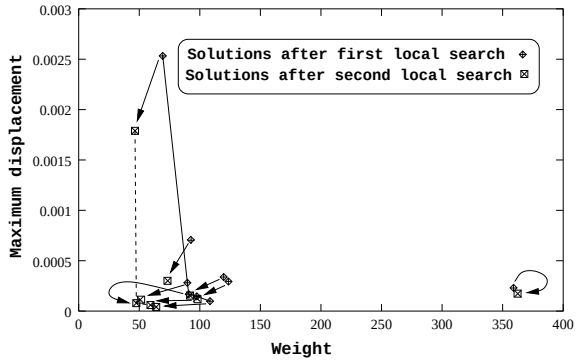

Figure 17: Non-dominated front after the second local-search.

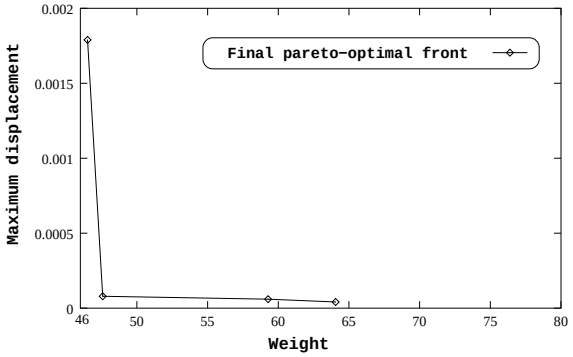

Figure 18: Non-dominated front at the end of local-search based EMO procedure.

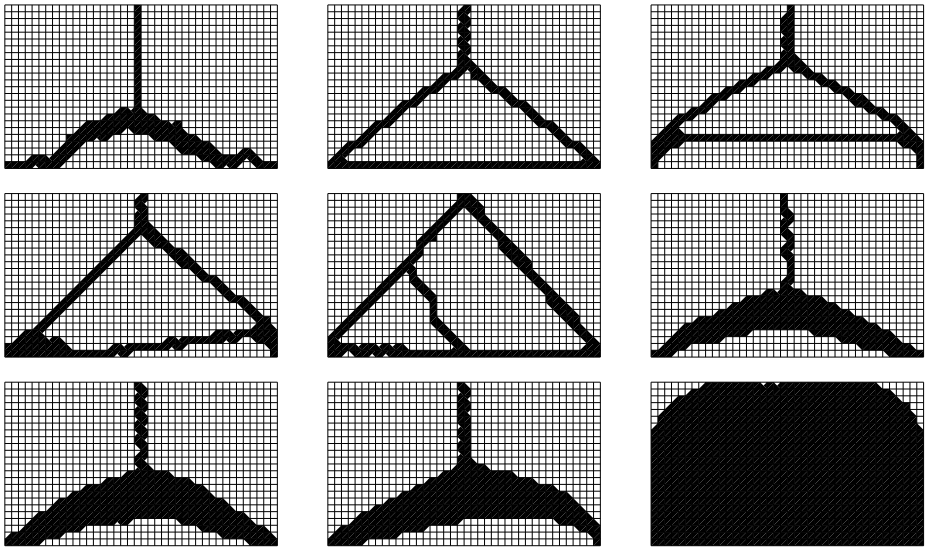

Figure 19: Nine trade-off shapes for the design of the simply-supported plate after Step 4.

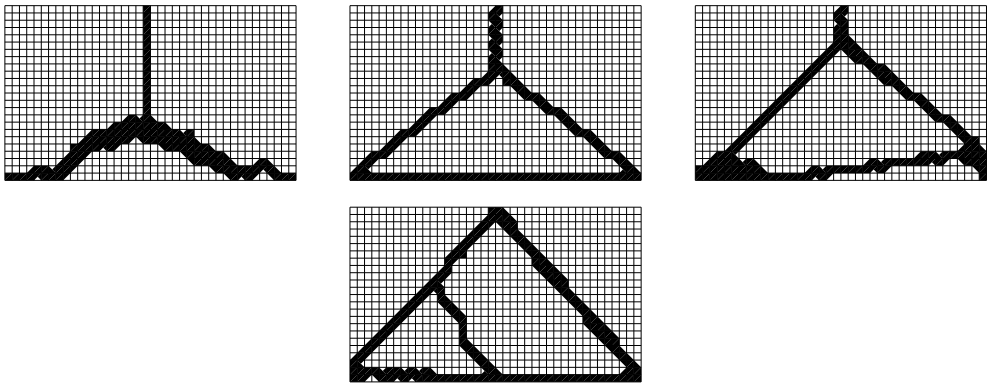

Figure 20: Final non-dominated shapes (after Step 5).

like symmetric structure is a 'knee' solution, meaning that a small improvement in an objective requires a large sacrifice in the other objective. Thus, there is not much motivation of choosing solutions other than the one in position (1,2). It is interesting that the proposed local-search based EMO procedure is able to discover such a well-engineered solution without any engineering knowledge and purely from an optimization process applied to a set of random shapes.

6.3.1 Function evaluations:

The number of function evaluations required to develop the above shapes are given in Table 2. A similar trend requiring an increasing number of evaluations for later local-searches is observed

Table 2: Function evaluations needed for the simply-supported plate design.

Shape No. →	1	2	3	4	5	6	7	8	9
First local-search	197	944	1,119	1,444	754	405	309	421	293
Second local-search	663	1,392	3,310	1,854	1,747	666	1,799	1,137	448
Total	860	2,336	4,429	3,298	2,501	1,071	2,108	1,558	741

here as well. The total function evaluations needed for the design of simply-supported plate are 24,302, of which 5,400 evaluations are taken by the NSGA-II procedure. The total processing time is 74.25 hours on the same 18-node PC cluster.

6.4 A Tower Plate

The schematic diagram of the tower design problem is given in Figure 21. In the two-dimensional

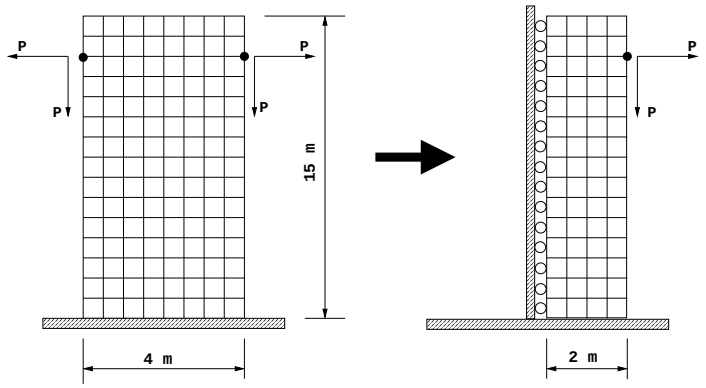

Figure 21: Boundary and loading conditions for the tower design problem.

model, the ground nodes are fixed and cable load is applied in the two nodes at the top. Loads are applied both in horizontal and vertical directions, as shown in the figure. It is clear from the above diagram that we have treated the problem to be symmetric about the vertical mid-plane of the structure. To keep symmetry forcibly, one half of the structure is taken for analysis. To impose appropriate boundary conditions for the half problem, the nodes at the left side of the domain is not allowed to move in the horizontal direction. However, they are allowed to move in the vertical direction. The nodes at the bottom are fixed in both horizontal and vertical directions.

Figure 22 plots the non-dominated front obtained using NSGA-II procedure (at the end of Step 1) with a randomly-created initial population. Also in this case, the number of feasible solutions in the initial population is small. Figure 23 presents the non-dominated front after the first local-search. The non-dominated front after the NSGA-II run is also presented on the

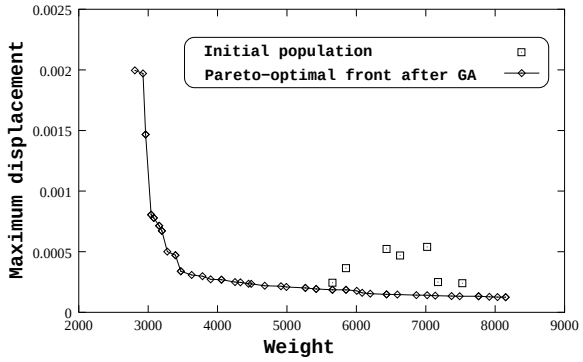

Figure 22: Non-dominated front after the NSGA-II run.

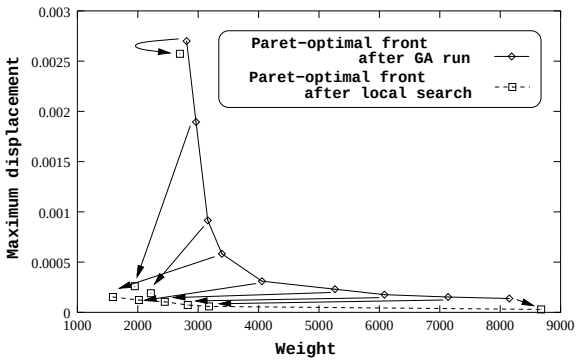

Figure 23: Non-dominated front after the local-search.

same plot to show the movements of the solutions. The first three small-weight solutions become dominated after the local-search procedure. In this case, the movements of solutions is not that erratic during the local-search operation. The final non-dominated front for the tower design problem is again plotted in Figure 24 which shows the objective values of the final solutions obtained after Step 5. The shapes of these six solutions are shown in Figure 25. The minimum-

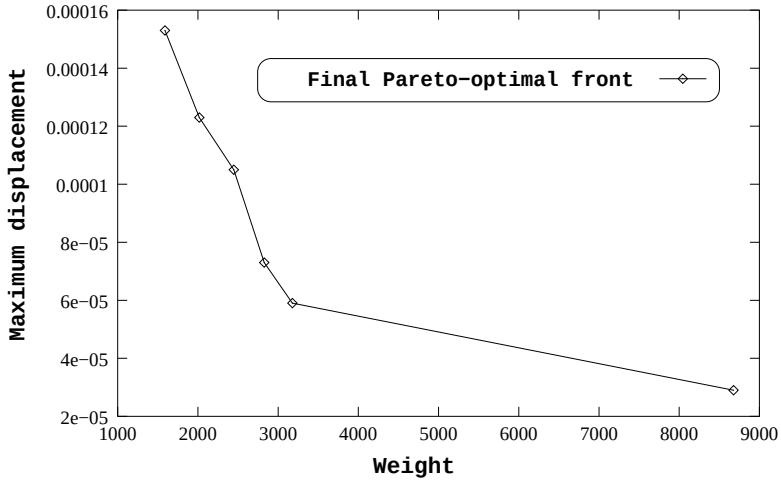

Figure 24: Final non-dominated front at the end of the local-search based EMO procedure.

weight solution connects the ground support and the load point by slender plates. Such shapes are often observed in optimally-designed transmission towers. To increase strength in the horizontal direction, a horizontal connection is formed in between the two vertical plates. This fix-up is common in structure design activities. The next solution has more horizontal stiffeners than the first solution. In the next solution, the thickness of the arms increases to provide more strength. Next two solutions are of similar type. They are rectangular in nature with cross-links extending from the ground to the vertical members. More connections to the ground increases the rigidity. The minimum-deflection solution is a fully solid plate with a hole at the center. The small hole does not reduce the deflection property of the plate too much, but helps reduce the weight of the plate.

The total function evaluations required for the design of the tower plate were 10,457, of which NSGA-II took 5,400 evaluations. Both local-search operations for each of the nine solutions

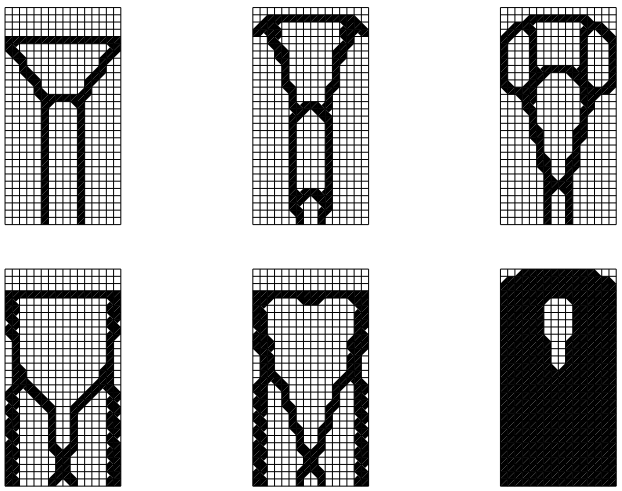

Figure 25: Six trade-off shapes for the design of the tower plate.

(obtained after Step 2) took on an average 562 evaluations. The total processing time was 31 minutes on the same 18-node PC cluster.

6.5 A Connecting Plate

The final problem considered in this study is the shape design of a connecting plate, shown in Figure 26. The size of the holes at the big end and small end are known beforehand. The square

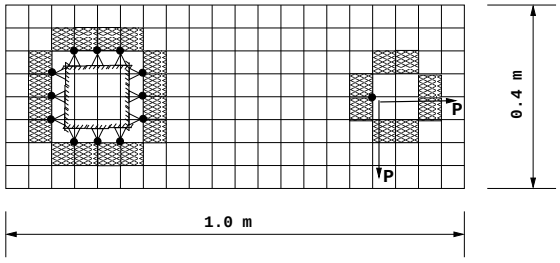

Figure 26: Geometric and loading conditions for the connecting plate.

elements at the hole positions are forcibly made material-free to create holes. To make a well-defined boundary around the holes, the elements surrounding the holes are always turned into material elements. This problem information helps the evolutionary algorithm to create realistic designs. The task of the search algorithm is to find the optimal link between the big and small ends. In this problem, a symmetry is imposed forcibly. To do so, the domain is divided into two equal parts about the horizontal mid-plane. Void or material elements are placed on one half of the domain. Then mirror image is taken about the horizontal mid-plane to create the entire shape. The complete structure generated by the above process is sent for the FEM analysis. Crossover and mutation operators of the NSGA-II procedure are also applied only on one half of the structure. Figure 27 shows the procedure of creating symmetric shapes. The connecting plate is designed on the basis of tensile and bending stresses. Figure 26 schematically shows the boundary conditions as well as geometric conditions. In the analysis, the properties of the material and NSGA-II parameters are kept the same as those in the previous cases. Geometric parameters used for the analysis are shown in the Figure 26.

Figure 28 plots the non-dominated front after the NSGA-II run (Step 1) and the initial population. In Figure 29, plots of non-dominated front after both local-searches (Step 5) are presented.

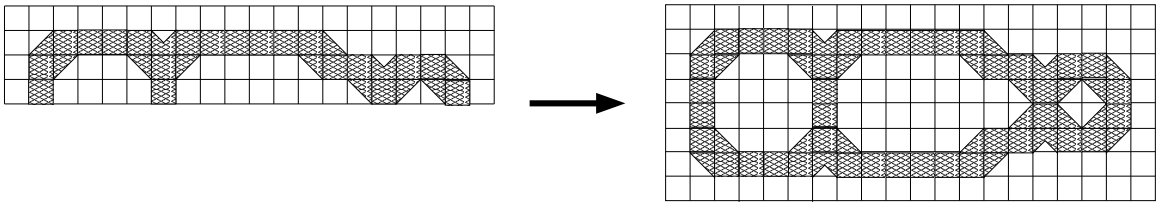

Figure 27: Consideration of the half of the connecting plate for optimization.

The extreme right solution (that is, the minimum-deflection solution) becomes dominated after

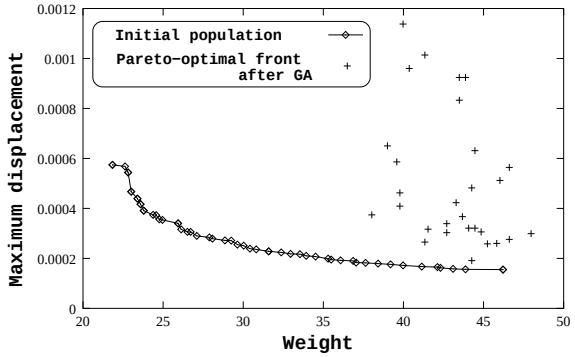

Figure 28: Non-dominated front after the NSGA-II run.

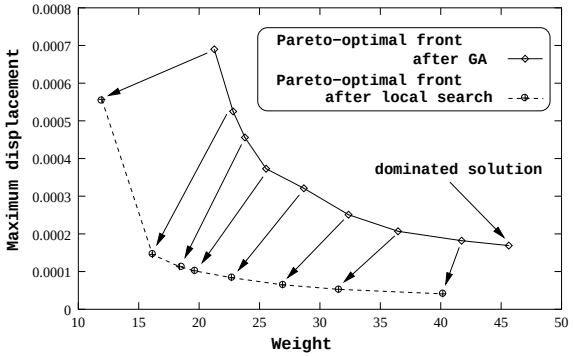

Figure 29: Non-dominated front after the final local-search.

the local-search. Due to the self-weight of the structure, the deflection of this solution is much more and hence the solution gets dominated by another solution. All eight non-dominated shapes are presented in Figure 30. The minimum-weight solution simply connects the two holes with thin plates to provide just enough strength. Although the complete rectangular region was available, the minimum-weight solution uses a smaller region to save weight. Thereafter, the near-support region gets thicker to provide adequate stiffness to the structure. Then on, the thickness of the arms get thicker and thicker and finally almost the whole plate becomes the minimum-deflection solution. It is interesting to note that the thickness of the connectors increases near the hole region (which acts as a support) to give more strength. These well-engineered phenomena evolve automatically without any prior information.

The overall function evaluations needed for the design of the connecting plate are 15,650, of which the NSGA-II procedure took 10,800 evaluations. Eight non-dominated solutions took, on an average, 606 evaluations each for the local-search procedure. The total processing time was seven hours on the 18-node PC cluster.

7 Conclusions

In this chapter, we have suggested the use of two or more conflicting objectives in engineering design activities for two reasons: (i) find a set of trade-off Pareto-optimal solutions and (ii) to decipher salient properties associated with optimal designs. We have suggested a systematic, two-phase procedure for this purpose. The use of meta-models (coarse-to-fine-grained modeling approach), a parallel computer, and efficient genetic operators has also been suggested. On a number of structural shape optimal design of mechanical engineering components, the proposed technique has discovered important properties, such as symmetry in shape, use of stiffeners to

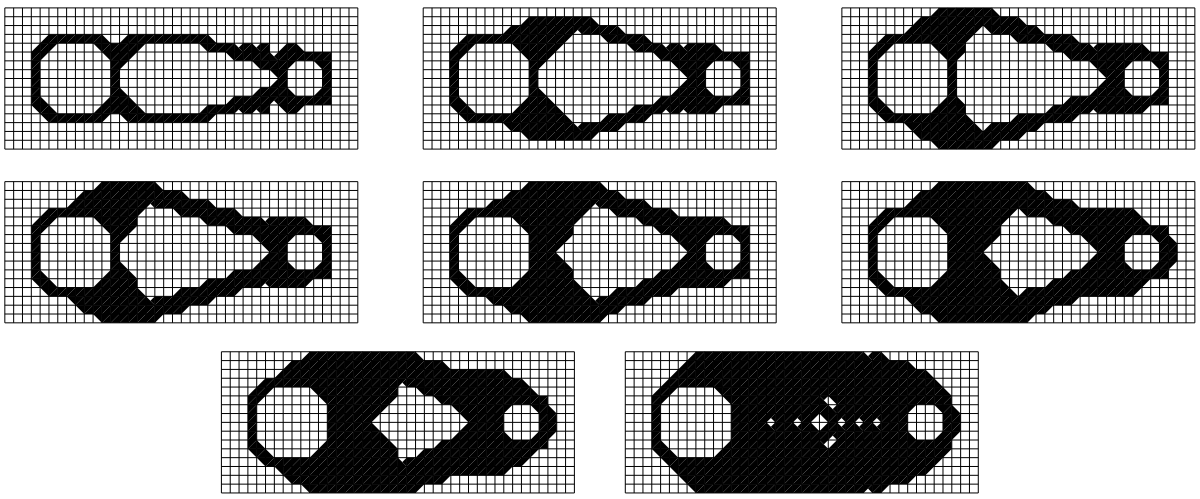

Figure 30: Eight trade-off shapes for the design of the connecting plate.

improve stiffness of designs, creation of internal voids for weight reduction, discovery of truss-like structures from plates, etc., which are found to be common among multiple design solutions corresponding to different trade-offs between two conflicting objectives of design. The procedure suggested here is generic and can be applied to other more complex engineering design problems.

References

[1] D. Corne, J. Knowles, and M. Oates. The Pareto envelope-based selection algorithm for multiobjective optimization. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 839–848, 2000.

[2] K. Deb. *Multi-objective optimization using evolutionary algorithms.* Chichester, UK: Wiley, 2001.

[3] K. Deb. Unveiling innovative design principles by means of multiple conflicting objectives. *Engineering Optimization*, 35(5):445–470, 2003.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[5] K. Deb and T. Goel. A hybrid multi-objective evolutionary approach to engineering shape design. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-01)*, pages 385–399, 2001.

[6] Zitzler E. and Thiele L. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43, Computer Engineering and Networks Laboratory, Switzerland, 1998.

[7] M. Ehrgott. *Multicriteria Optimization.* Berlin: Springer, 2000.

[8] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989.

[9] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and reviews*, 28(3):392–403, 1998.

[10] H. Ishibuchi and T. Murata. Multi-objective genetic local search for minimizing the number of fuzzy rules for pattern classification problems. In *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems*, pages 1100–1105, 1998.

[11] M. J. Jakiela, C. Chapman, J. Duda, A. Adewuya, and K. Saitou. Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):339–356, 2000.

[12] M. T. Jensen. Reducing the run-time complexity of multiobjective EAs. *IEEE Transcations to Evolutionary Computation*, 7(5):503–515, 2003.

[13] K. Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer, Boston, 1999.

[14] S. S. Rao. *Optimization: Theory and Applications.* Wiley, New York, 1984.

[15] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. *Engineering Optimization Methods and Applications.* New York : Wiley, 1983.