Vol. 45 No. SIG 2(TOM 10) IPSJ Transactions on Mathematical Modeling and Its Applications Feb. 2004

Invited Paper

An Ideal Evolutionary Multi-Objective Optimization Procedure

Kalyanmoy Deb†

1. Introduction

Many real-world search and optimization problems are naturally posed as non-linear programming problems having multiple objectives. Due to lack of suitable solution techniques, such problems are artificially converted into a single-objective problem and solved. The difficulty arises because such problems give rise to a set of Pareto-optimal solutions, instead of a single optimum solution. It then becomes important to find not just one Pareto-optimal solution but as many of them as possible. Classical methods are not quite efficient in solving these problems because they require repetitive applications to find multiple Pareto-optimal solutions and in some occasions repetitive applications do not guarantee finding distinct Pareto-optimal solutions. The population approach of evolutionary algorithms (EAs) allows an efficient way to find multiple Pareto-optimal solutions simultaneously in a single simulation run.

In this paper, we discuss one implementation of an EA which uses the non-domination concept[17]. Simulation results on a number of test problems show the efficacy of the method. It is clear from the discussions that evolutionary search methods have a niche in solving multi-objective optimization problems compared to classical approaches. This is why multi-objective optimization using EAs is getting a growing attention in the recent years. This paper also suggests two other purposes of studying with evolutionary multi-objective optimization. The motivated readers may explore current research issues and other important studies from various texts[2),5),7)], conference proceedings[12),19)] and numerous research papers[4)].

† Kanpur Genetic Algorithms Laboratory (KanGAL), Department of Mechanical Engineering, Indian Institute of Technology Kanpur

2. Multi-Objective Optimization Problem

A multi-objective optimization problem involves a number of objective functions which are to be either minimized or maximized. As in the single-objective optimization problem, the multi-objective optimization problem usually has a number of constraints which any feasible solution (including the optimal solution) must satisfy. In the following, we state the multi-objective optimization problem (MOOP) in its general form:

$$\left.\begin{array}{rl} \text{Minimize/Maximize} & \\ f_m(\mathbf{x}), & m = 1, 2, \ldots, M; \\ \text{subject to} & \\ g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}\right\} \quad (1)$$

A solution $\mathbf{x}$ is a vector of n decision variables: $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. The last set of constraints are called variable bounds, restricting each decision variable x_i to take a value within a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound. The solutions satisfying the constraints and variable bounds constitute a *feasible decision variable space* $\mathcal{S}$, or simply the decision space. One of the striking differences between single-objective and multi-objective optimization is that in multi-objective optimization the objective functions constitute a multi-dimensional space, in addition to the usual decision variable space. This additional space is called the *objective space*, $\mathcal{Z}$. For each solution $\mathbf{x}$ in the decision variable space, there exists a point in the objective space, denoted by $\mathbf{f}(\mathbf{x}) = \mathbf{z} = (z_1, z_2, \ldots, z_M)^T$. The mapping takes place between an n-dimensional solution vector and an M-dimensional objective vector. **Figure 1** illustrates these two spaces and a mapping between them.

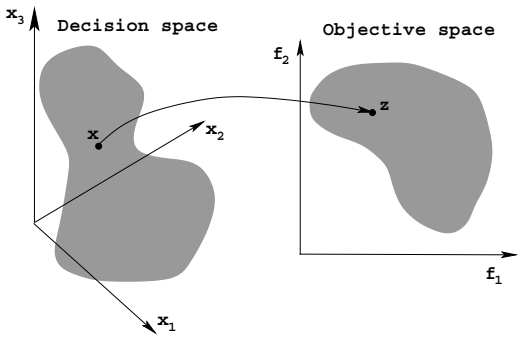

Fig. 1 Representation of the decision variable space and the corresponding objective space.

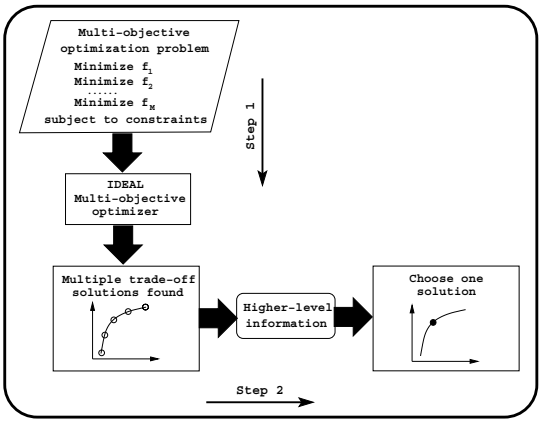

Fig. 2 Schematic of an ideal multi-objective optimization procedure.

3. Principles of Multi-Objective Optimization

Most real-world search and optimization problems naturally involve multiple objectives. The extremist principle of finding the optimum solution cannot be applied to one objective alone, when the rest of the objectives are also important. Different solutions may produce trade-offs (conflicting scenarios) among different objectives. A solution that is extreme (in a better sense) with respect to one objective requires a compromise in other objectives. This prohibits one to choose a solution which is optimal with respect to only one objective. Clearly, there are two goals of multi-objective optimization:

(1) Find a set of solutions close to the optimal solutions, and

(2) Find a set of solutions which are diverse enough to represent the spread of the true optimal solutions.

4. The Ideal Multi-Objective Optimization

Although the fundamental difference between single and multiple objective optimization lies in the cardinality in the optimal set, from a practical standpoint a user needs only one solution, no matter whether the associated optimization problem is single-objective or multi-objective. In the case of multi-objective optimization, the user is now in a dilemma. Which of these optimal solutions must one choose? This is not an easy question to answer. It involves many higher-level information which are often non-technical, qualitative and experience-driven. However, if a set of many trade-off solutions are already worked out or available, one can evaluate the pros and cons of each of these solutions based on all such non-technical and qualitative, yet still important, considerations and compare them to make a choice. Thus, in a multi-objective optimization, ideally the effort must be made in finding the set of trade-off optimal solutions by considering all objectives to be important. After a set of such trade-off solutions are found, a user can then use higher-level qualitative considerations to make a choice. In view of these discussions, we suggest the following principle for an *ideal multi-objective optimization procedure*:

Step 1 Find multiple trade-off optimal solutions with a wide range of values for objectives.

Step 2 Choose one of the obtained solutions using higher-level information.

Figure 2 shows schematically the principles in an ideal multi-objective optimization procedure. In Step 1 (vertically downwards), multiple trade-off solutions are found. Thereafter, in Step 2 (horizontally, towards the right), higher-level information is used to choose one of the trade-off solutions. With this procedure in mind, it is easy to realize that single-objective optimization is a degenerate case of multi-objective optimization. In the case of single-objective optimization with only one global optimal solution, Step 1 will find only one solution, thereby not requiring us to proceed to Step 2. In the case of single-objective optimization with multiple global optima, both steps are necessary to first find all or many of the global optima and then to choose one from them by using the higher-level information about the problem.

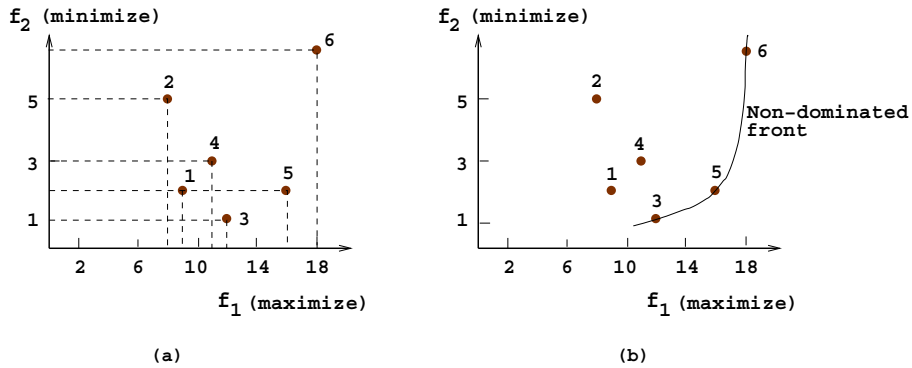

Fig. 3 A set of solutions and the best non-dominated front.

In the next section, we present an evolutionary multi-objective optimization algorithm to find a set of Pareto-optimal solutions, instead of just one of them.

5. Elitist Non-Dominated Sorting GA (NSGA-II)

The NSGA-II procedure[10] for finding multiple Pareto-optimal solutions in a multi-objective optimization problem has the following three features:

(1) It uses an elitist principle,
(2) It uses an explicit diversity preserving mechanism, and
(3) It emphasizes the non-dominated solutions.

In NSGA-II, the offspring population Q_t is first created by using the parent population P_t and the usual genetic operators[13),16)]. Thereafter, the two populations are combined together to form R_t of size $2N$. Then, a non-dominated sorting is used to classify the entire population R_t. The *domination* between two solutions is defined as follows[7),17)]:

Definition 1 *A solution* $\mathbf{x}^{(1)}$ *is said to dominate the other solution* $\mathbf{x}^{(2)}$*, if both the following conditions are true:*

(1) The solution $\mathbf{x}^{(1)}$ *is no worse than* $\mathbf{x}^{(2)}$ *in all objectives.*
(2) The solution $\mathbf{x}^{(1)}$ *is strictly better than* $\mathbf{x}^{(2)}$ *in at least one objective.*

For a given set of solutions (for example, those shown in **Fig. 3** (a)), a pair-wise comparison can be made using the above definition and whether one solution dominates the other can be established. All solutions which are not dominated by any other members of the set are called the non-dominated solutions of level one.

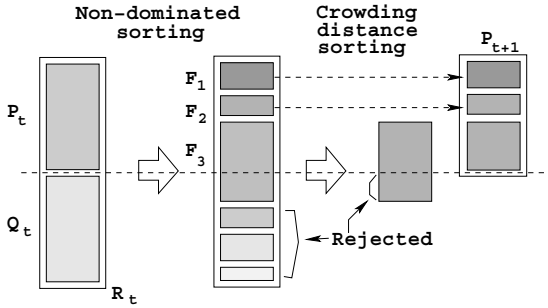

Fig. 4 Schematic of the NSGA-II procedure.

These solutions (3, 5, and 6 in the figure) are also said to lie on the best non-dominated front. By temporarily discounting these solutions, the above principle can be followed again and the next non-dominated front (solutions 1 and 4 in the figure) can be identified. Figure 3 (b) shows how the given set of six solutions are classified into three non-dominated fronts. Once the non-dominated sorting is over, the new population is filled by solutions of different non-dominated fronts, one at a time. The filling starts with the best non-dominated front 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_t is $2N$, not all fronts may be accommodated in N slots available in the new population. All fronts which could not be accommodated are simply deleted. When the last allowed front is being considered, there may exist more solutions in the last front than the remaining slots in the new population. This scenario is illustrated in **Fig. 4**. Instead of arbitrarily discarding some members from the last front, the solutions which will make the diversity of the selected solutions the highest are chosen.

The crowding-sorting of the solutions of front i (the last front which could not be accommo-

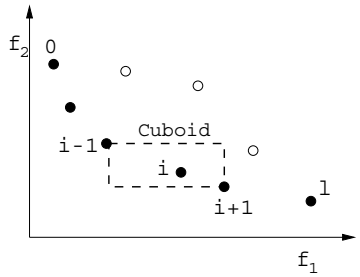

Fig. 5 The crowding distance calculation.

dated fully) is performed by using a *crowding distance metric*, which we will describe a little later. The population is arranged in descending order of magnitude of the crowding distance values. In Step 4, a crowding tournament selection operator, which also uses the crowding distance, is used.

5.1 Crowded Tournament Selection Operator

The crowded comparison operator ($<_c$) compares two solutions and returns the winner of the tournament. It assumes that every solution i has two attributes:

(1) A non-domination rank r_i in the population.

(2) A local crowding distance (d_i).

The crowding distance d_i of a solution i is a measure of the search space around i which is not occupied by any other solution in the population. Here, we simply calculate this quantity d_i by estimating the perimeter of the cuboid formed by using the nearest neighbors as the vertices (we call this the *crowding distance*). In **Fig. 5**, the crowding distance of the i-th solution in its front (marked with filled circles) is the average side-length of the cuboid (shown by a dashed box). Based on these two attributes, we can define the crowded tournament selection operator as follows.

Definition 2 *Crowded Tournament Selection Operator: A solution i wins a tournament with another solution j if any of the following conditions are true:*

(1) If solution i has a better rank, that is, $r_i < r_j$.

(2) If they have the same rank but solution i has a better crowding distance than solution j, that is, $r_i = r_j$ and $d_i > d_j$.

The first condition makes sure that chosen solution lies on a better non-dominated front. The second condition resolves the tie of both solutions being on the same non-dominated front by deciding on their crowded distance. The one residing in a less crowded area (with a larger crowding distance d_i) wins.

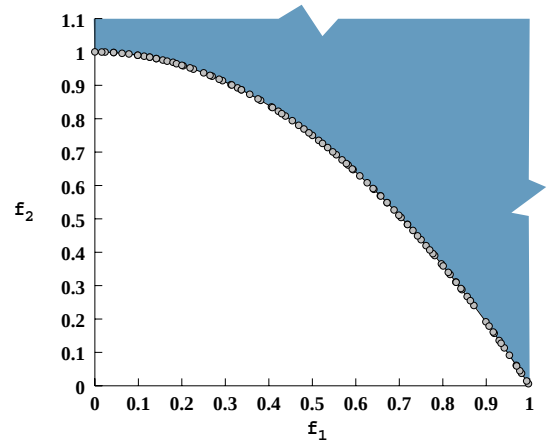

Fig. 6 NSGA-II on ZDT2.

5.2 Sample Simulation Results

In this subsection, we show the simulation results of NSGA-II on two test problems. The first problem (SCH1) is simple two-objective problem with a convex Pareto-optimal front.

$$\mathsf{ZDT2}: \begin{cases} \text{Minimize} \\ f_1(x) = x_1, \\ \text{Minimize} \\ f_2(x) = g\left[1 - (f_1/g)^2\right], \\ \text{where} \\ g(\mathbf{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \\ \\ -10^3 \le x \le 10^3. \end{cases} \quad (2)$$

The second problem (KUR) has a disjointed set of Pareto-optimal fronts:

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

NSGA-II is run with a population size of 100 and for 250 generations. **Figures 6** and **7** show that NSGA-II converges on the Pareto-optimal front and maintains a good spread of solutions on both test problems.

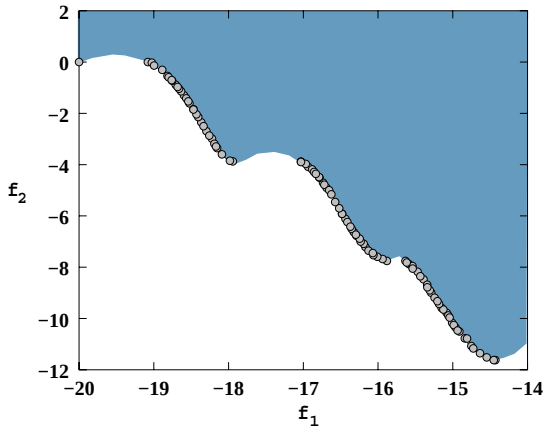

Fig. 7 NSGA-II on KUR.

5.3 Constraint Handling

The constraint handling method modifies the binary tournament selection, where two solutions are picked from the population and the better solution is chosen. In the presence of constraints, each solution can be either feasible or infeasible. Thus, there may be 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 (we call it the *constrain-domination* condition for any two solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$):

Definition 3 *A solution $\mathbf{x}^{(i)}$ is said to 'constrain-dominate' a solution $\mathbf{x}^{(j)}$ (or $\mathbf{x}^{(i)} \preceq_c \mathbf{x}^{(j)}$), 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 (Definition 1).

The above change in the definition allows a minimal change in the NSGA-II procedure described earlier. **Figure 8** shows the non-dominated fronts on a six-membered population due to the introduction of two constraints (the problem is described as Constr-Ex in the next subsection). In the absence of the constraints, the non-dominated fronts (shown by dashed lines) would have been `((1,3,5), (2,4,6))`, but in their presence, the new fronts are `((4,5), (6), (2), (1), (3))`. The first non-dominated front is constituted with the best feasible solutions in the population and any feasible solution lies on a better non-dominated front than an feasible solution.

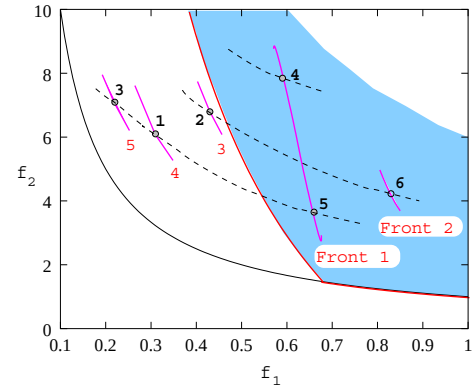

Fig. 8 Non-constrain-dominated fronts.

5.3.1 Sample Simulation Results

In the following, we show simulation results of NSGA-II applied with the above constraint handling mechanism to two test problems (CONSTR and TNK):

$$\begin{aligned}
\text{Min.} \quad & f_1(\mathbf{x}) = x_1, \\
\text{Min.} \quad & f_2(\mathbf{x}) = \frac{1+x_2}{x_1}, \\
\text{s.t.} \quad & g_1(\mathbf{x}) \equiv x_2 + 9x_1 \geq 6, \\
& g_2(\mathbf{x}) \equiv -x_2 + 9x_1 \geq 1, \\
& 0.1 \leq x_1 \leq 1, \\
& 0 \leq x_2 \leq 5.
\end{aligned}$$

$$\begin{aligned}
\text{Min.} \quad & f_1(\mathbf{x}) = x_1, \\
\text{Min.} \quad & f_2(\mathbf{x}) = x_2, \\
\text{s.t.} \quad & x_1^2 + x_2^2 - 1 - 0.1\cos\left(16\tan^{-1}\frac{x_1}{x_2}\right) \\
& \geq 0, (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5, \\
& 0 \leq x_1 \leq \pi, \\
& 0 \leq x_2 \leq \pi.
\end{aligned}$$

With identical parameter settings as before, NSGA-II finds a good distribution of solutions on the Pareto-optimal front in both problems (**Figs. 9** and **10**, respectively).

6. How are Evolutionary Multi-Objective Optimization Useful?

It is amply demonstrated above and in the EMO literature[4] that there exist a number of evolutionary multi-objective optimization (EMO) algorithms for finding a well-converged and a well-distributed set of solutions near the true Pareto-optimal front. In this section, we stress three different purposes for using EMO algorithms in practice:

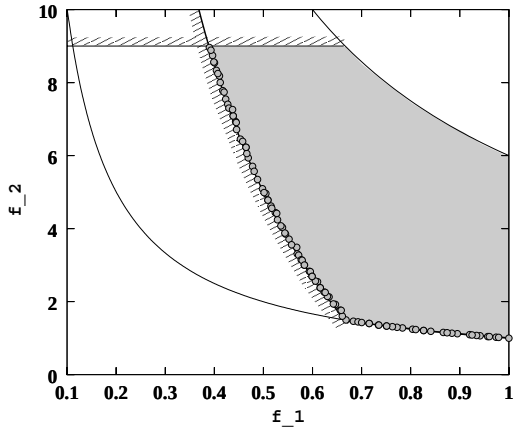

Fig. 9 Obtained non-dominated solutions with NSGA-II on the constrained problem CONSTR.

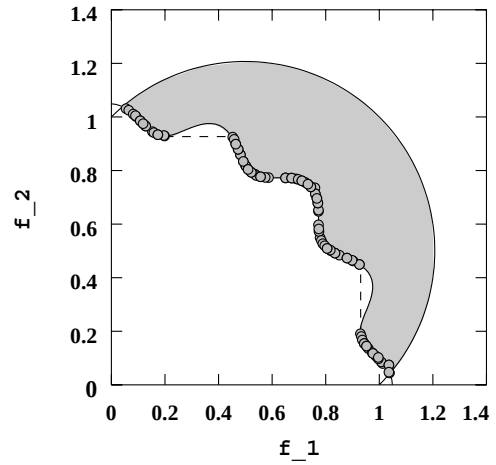

Fig. 10 Obtained non-dominated solutions with NSGA-II on the constrained problem TNK.

(1) Aid in choosing a compromised solution,
(2) Aid in understanding the optimality properties of solutions better, and
(3) Aid in solving other optimization problems.

We discuss each of the above issues in the following subsections.

6.1 In Choosing a Compromised Solution

There is no doubt that the availability of a number of compromised solutions is always useful to a decision-maker in choosing a particular solution. Although specific systematic procedures must be outlined to help choose a solution, the task of concentrating in a particular region on the Pareto-optimal front based on some pre-defined weight information or other means and without a prior exploration of various trade-off options may not be a desirable procedure. Although a number of subjective considerations can occur in certain applications, here, we show a couple of generic techniques can be adopted to concentrate near a particular region on the Pareto frontier. We illustrate these techniques through a mechanical component shape design problem.

A common approach for shape optimization using evolutionary algorithms is to divide a region into many small elements and treat the presence or absence of these elements as binary decision variables[14]. This way different shapes can be evolved by making elements on or off. The task of an evolutionary algorithm is to find which elements should be kept and which should be sacrificed so that the resulting shape is optimal with respect to one or more objectives. Two conflicting objectives are chosen here: the weight of the resulting component and the maximum deflection anywhere in it. The maximum stress and deflection values are restricted to lie within specified limits of the design by using them as constraints. For details on the representation technique and the evaluation procedure, readers may refer to the original study[11].

A rectangular plate (1.2×2 m^2) is fixed at one end and a 100 kN load is applied to the center element of the opposite end. NSGA-II is applied for 100 generations with a population size of 54 and crossover probability of 0.95. In order to increase the quality of the obtained solutions, we use an incremental grid-tuning technique. The NSGA-II and the first local search procedure are run with a coarse grid structure (6×10 or 60 elements). After the first local search procedure, each grid is divided into four equal-sized grids, thereby having a 12×20 or 240 elements. The new smaller elements inherit its parent's status of being present or absent. After the second local search is over, the elements are divided again, thereby making 24×40 or 960 elements. In all cases, an automatic mesh-generating finite element method is used to analyze the developed structure.

Figure 11 shows the obtained front with eight solutions. The trade-off between the weight and deflection is clear from the figure. **Figure 13** shows the shape of these eight solutions. The solutions are arranged according to increasing weight from left to right and top

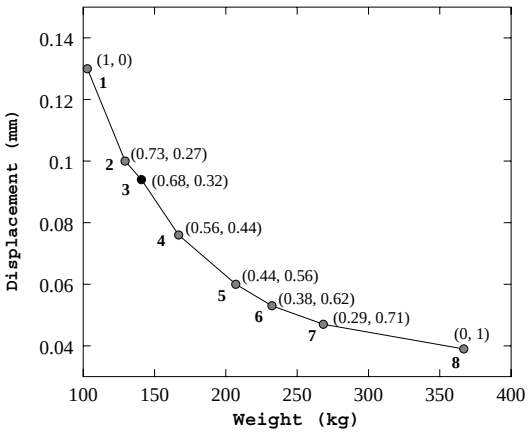

Fig. 11 Obtained front with eight clustered solutions are shown for the cantilever plate design problem. The pseudo-weight for each solution is also shown.

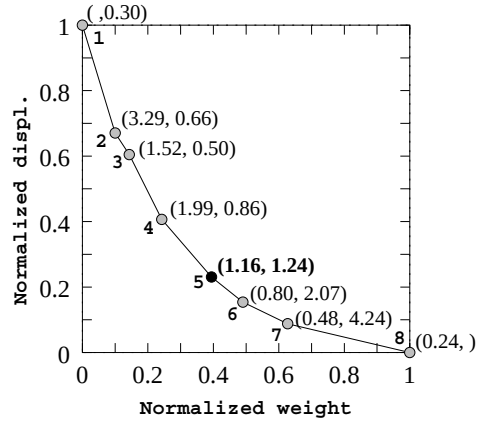

Fig. 12 The marginal gain to loss ratio is shown from each solution.

to bottom. Thus, the minimum-weight solutions the top-left solution and the minimum-deflection solution is the right-bottom solution. The transition from a simple thin two-armed cantilever plate having a minimum-weight solution to a complete plate with edges rounded off having a minimum-deflection solution proceeds by discovering a vertical stiffener connecting the two arms and then by widening the arms and then by gradually thickening the stiffener. Such a transitional trade-off in optimal designs is useful to a decision-maker in arriving at a solution. We show two procedures which are generic and can help a decision-maker to choose a region of importance.

6.1.1 Pseudo-Weight Method

The location of the obtained solutions in the objective space is used to derive a pseudo-weight vector for each solution. The following procedure can be used for this purpose for minimization problems:

$$\bar{w}_j^{\mathbf{x}} = \frac{(f_j^{\max} - f_j(\mathbf{x}))/(f_j^{\max} - f_j^{\min})}{\sum_{k=1}^{M}(f_k^{\max} - f_k(\mathbf{x}))/(f_k^{\max} - f_k^{\min})}, \tag{4}$$

where $f_j^{\min}$ and $f_j^{\max}$ are the minimum and maximum values of the k-th objective in the set. Figure 11 also marks the pseudo-weight vector for each of the eight obtained solutions. In a sense, these weight vectors indicate a relative importance of each objective associated with a solution. If the decision-maker had a pre-conceived notion of arriving at a solution having a specific importance vector (say w_d), the solution having the closest pseudo-weight vector can be chosen as a representative solution. For example, in this example case, if the decision-maker is interested in arriving at a solution providing the weight objective (f_1) twice as important than the deflection objective (f_2), the desired weight vector is $w_d = (0.67, 0.33)$. Clearly, the solution 3 produces a similar trade-off among all the other solutions. The decision-maker can now explore the solution and its neighborhood for a more detail and desired solution.

6.1.2 Gain-to-Loss Ratio Method

In this approach, for each obtained solution a gain-to-loss ratio is calculated. We calculate this ratio here by first comparing a solution with its neighboring solutions and calculating the ratio between the gain achieved in one objective for a loss in another objective by remaining to the current solution (and not selecting the neighboring solution). Since in a two-objective problem, there will be two-neighboring solutions, except the extreme solutions, we compute both such ratios and work with the minimum of the two ratios. It is needless to write that before such calculations are performed, the objective values can be normalized using the extreme solutions. The solution having the maximum value of this ratio will indicate it as providing a good compromise between the objectives on the Pareto-optimal front.

Figure 12 shows the two-sided ratios for each solution (except for the extreme solutions, where there is only one ratio). Interestingly, the solution 5 makes two-sided ratios each of which is greater than one. In a sense, this solu-

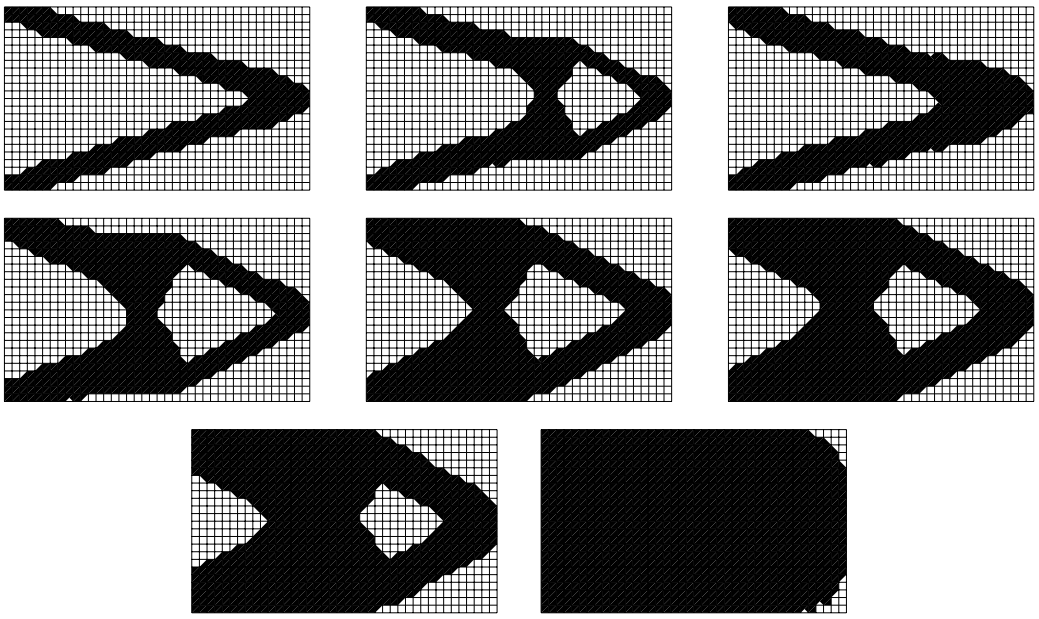

Fig. 13 Eight trade-off solutions of the cantilever plate design problem.

tion indicates that for a unit loss in one objective there is a larger than unity gain in another objective in either direction on the normalized objective space. Without the desire of any specific relative importance among the objectives as shown in the previous method, this method can be used a generic procedure for choosing well-compromised regions on the Pareto frontier.

6.2 In Capturing Insights about the Problem

Through a number of multi-objective optimization problems, a recent study[9] has discovered the following two interesting properties of the Pareto-optimal solutions:

- There exist some *common* properties among all Pareto-optimal solutions, and
- There exist some other properties which cause the trade-off between the Pareto-optimal solutions.

Although the above observations are purely based on simulation results derived using the NSGA-II algorithm on a number of case studies, there exist a number of mathematical optimality criteria[7),17)] which every Pareto-optimal solution must satisfy. Putting it otherwise, since all Pareto-optimal solutions have to satisfy some criteria, it may not be surprising that the computer simulations using NSGA-II have discovered such commonality properties among the obtained optimal solutions. What we stress here is that if there exist commonality properties among optimal solutions, they would provide useful information about the problem at hand. Since the mathematical conditions involve gradient computations and involves solving a number of non-linear equations, they may not be of much use in practice. The only way to discover such useful information is to first apply an EMO to find a number of solutions close to the true Pareto-optimal solutions and then analyze them for discovering any such properties. In the following, we describe the procedure to a simple truss-structure optimization problem.

The objective in a truss-structure design (with all three design considerations – sizing, configuration and topology) is to find which optional nodes are necessary in a truss, what are the coordinates of these optional nodes, which members must be present and what are their cross-sectional areas, so that certain objectives (usually the weight of the truss and the maximum deflection in the truss) are minimized by satisfying certain constraints (often the stresses in the members and the displacements of nodes). The problem details can be found in the original study[7].

Figure 14 shows the three-bar truss with three loading cases considered in the study. The NSGA-II is run for 150 generations and the 50 obtained population members are plotted in **Fig. 15**. The following important observations

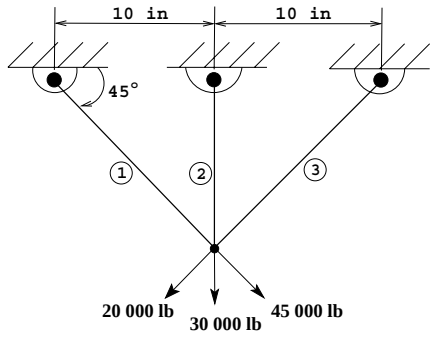

Fig. 14 The three-bar truss.

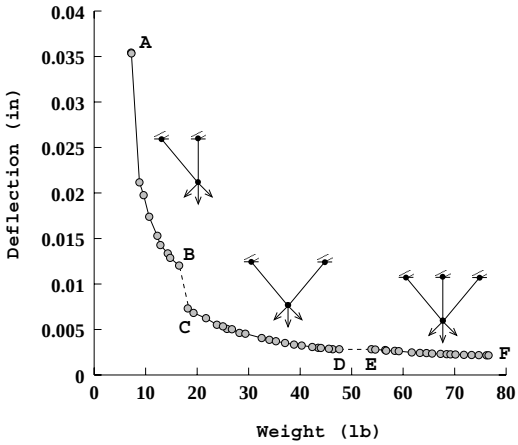

Fig. 15 Obtained non-dominated solutions using the NSGA-II.

can be made from the solutions:

(1) First, out of all possible topologies, the three topologies shown in the figure are found to be Pareto-optimal.

(2) Secondly, each Pareto-optimal topology has a particular *niche* on the obtained front. The solutions on a region of the Pareto-optimal front has all have the *same topology*. Neighboring solutions under each topology vary only in their cross-sectional sizes. This property among neighboring solutions was certainly not intuitively known beforehand. An investigation of the obtained solution reveals this property.

(3) For a smaller trade-off in solutions, a cross-sectional change in an appropriate manner is adequate, whereas for a larger trade-off among objectives, both topology and cross-sectional sizes must be changed.

A similar observation is obtained from another

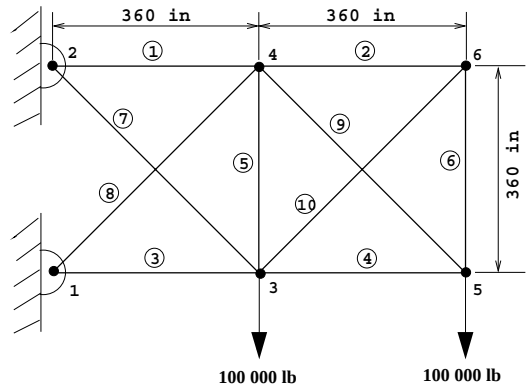

Fig. 16 The ten-bar truss.

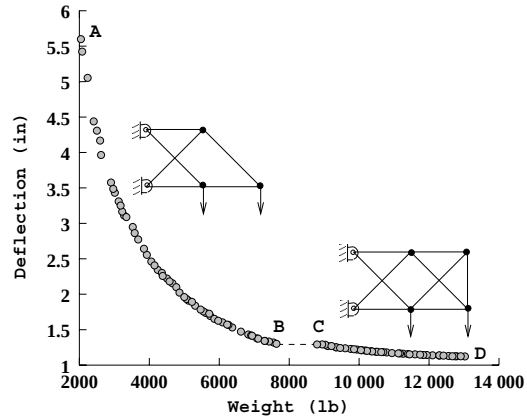

Fig. 17 Obtained non-dominated solutions for the 10-bar truss.

10-bar truss-structure design (**Fig. 16**) with results shown in **Fig. 17**.

7. In Solving Other Optimization Problems

The principle of evolutionary multi-objective optimization is also used to solve a number of other kinds of optimization problems, such as single-objective constrained optimization[18], diversity preservation in a GA[1),15)], reducing bloating in a GP application[3)], goal programming problems[8)]. The presence of an additional objective allows an EA to have a more flexible search. For example, in the usual penalty function approach of handling constraints, the choice of specific penalty parameters to handle constraint violations distorts the search space in a specific way[6)] and restricts a particular way a solution can become feasible and reach near the optimum. On the other hand, by keep-

ing the constraint violation as the second objective, solutions having a small constraint violation and solutions having a better objective values can coexist in a population. A recombination of these solutions may with in a GA may lead more flexible ways to reach near the true constrained optimum. For details, readers may refer to the above references.

8. Conclusions

The paper objects the usual practice of treating multi-objective optimization problems by scalarizing them into a single objective and optimizing it. The presence of multiple objectives results in a number of Pareto-optimal solutions, instead of a single optimum solution. In this paper, we have shown how an evolutionary algorithm (NSGA-II) can be suitably used to keep the meaning of multi-objective optimization and find a number of Pareto-optimal solutions on a number of problems in a single simulation run. For a C implementation of NSGA-II procedure described in this paper, interested readers may visit http://www.iitk.ac.in/kangal/soft.htm.

Besides finding the multiple Pareto-optimal solutions, the suggested ideal multi-objective optimization procedure has two other unique advantages. Once a set of Pareto-optimal solutions are found, they can be analyzed. The *transition* from the optimum of one objective to that of another optimum can be investigated. Since all such solutions are optimum, the transition should show interesting trade-off information of sacrificing one objective only to get a gain in other objectives. Moreover, the use of auxiliary objectives may help provide a more flexible search in a number of other optimization tasks than the usual way of treating them.

The field of multi-objective evolutionary algorithms is fairly new. There exists a number of interesting and important research topics[7] which must be investigated before their full potential is discovered. Hopefully, this paper has shown some usefulness in this direction to motivate the readers to pay further attention to this growing field of interest.

References

1) Abbass, H. and Deb, K.: Searching under multi-evolutionary pressures, *Proc. Second Evolutionary Multi-Criterion Optimization (EMO-03) Conference*, pp.391–405 (2003).
2) Bagchi, T.: *Multiobjective Scheduling by Genetic Algorithms*, Boston, Kluwer Academic Publishers (1999).
3) Bleuler, S. Brack, M. and Zitzler, E: Multi-objective genetic programming: Reducing bloat using spea2, *Proc. 2001 Congress on Evolutionary Computation*, pp.536–543 (2001).
4) C.A.C. Coello, http://www.lania.mx/˜ccoello/EMOO/ (2003).
5) Coello, C.A.C., VanVeldhuizen, D.A. and Lamont, G.: *Evolutionary Algorithms for Solving Multi-Objective Problems*, Boston, MA, Kluwer Academic Publishers (2002).
6) Deb, K.: *Optimization for Engineering Design: Algorithms and Examples*, New Delhi, Prentice-Hall (1995).
7) Deb, K.: *Multi-objective optimization using evolutionary algorithms*, Chichester, UK, Wiley (2001).
8) Deb, K.: Nonlinear goal programming using multi-objective genetic algorithms, *Journal of the Operational Research Society*, Vol.52, No.3, pp.291–302 (2001).
9) Deb, K.: Unveiling innovative design principles by means of multiple conflicting objectives, *Engineering Optimization*, Vol.35, No.5 (2003).
10) Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Trans. Evolutionary Computation*, Vol.6, No.2, pp.182–197 (2002).
11) Deb, K. and Goel, T.: A hybrid multi-objective evolutionary approach to engineering shape design, *Proc. First International Conference on Evolutionary Multi-Criterion Optimization (EMO-01)*, pp.385–399 (2001).
12) Fonseca, C., Fleming, P., Zitzler, E., Deb, K. and Thiele, L. (Eds).: *Proc. Second Evolutionary Multi-Criterion Optimization (EMO-03) Conference (LNCS 2632)*, Heidelberg, Springer (2003).
13) Goldberg, D.E.: *Genetic Algorithms for Search, Optimization, and Machine Learning*, Reading, MA, Addison-Wesley (1989).
14) Jakiela, M.J., Chapman, C., Duda, J., Adewuya, A. and Saitou, K.: Continuum structural topology design with genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, Vol.186, No.2–4, pp.339–356 (2000).
15) Jensen, M.T.: Guiding single-objective optimization using multi-objective methods, *EvoWorkshops 2003*, pp.268–279 (2003).
16) Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin, Springer-Verlag (1992).
17) Miettinen, K.: *Nonlinear Multiobjective Optimization*, Kluwer, Boston (1999).

18) Surry, P.D., Radcliffe, N.J. and Boyd, I.D.: A multi-objective approach to constrained optimization of gas supply networks: The COMOGA method, *Evolutionary Computing, AISB Workshop*, pp.166–180. Springer-Verlag (1995).
19) Zitzler, E., Deb, K., Thiele, L., Coello, C.A.C. and Corne, D. (Eds): *Proc. First Evolutionary Multi-Criterion Optimization (EMO-01) Conference (LNCS 1993)*, Heidelberg, Springer (2001).

(Received September 30, 2003)
(Accepted September 30, 2003)

Kalyanmoy Deb Professor in the Department of Mechanical Engineering in Indian Institute of Technology Kanpur. 多目的遺伝的アルゴリズムの先駆的な研究で世界的に著名．著書「Multi-Objective Optimization Using Evolutionary Algorithms」（Wiley）をはじめとして関連の研究論文多数.