Structural Optimization 10, 94–99 © Springer-Verlag 1995

A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm

A. Osyczka and S. Kundu

Department of Precision Engineering, Tokyo Metropolitan University, 1-1 Minami Ohsawa, Hachioji Shi, Tokyo 192 - 03, Japan

Abstract Genetic algorithms (GAs), which are directed stochastic hill climbing algorithms, are a commonly used optimization technique and are generally applied to single criterion optimization problems with fairly complex solution landscapes. There has been some attempts to apply GA to multicriteria optimization problems. The GA selection mechanism is typically dependent on a single-valued objective function and so no general methods to solve multicriteria optimization problems have been developed so far. In this paper, a new method of transformation of the multiple criteria problem into a single-criterion problem is presented. The problem of transformation brings about the need for the introduction of the *Pareto* set estimation method to perform the multicriteria optimization using GAs. From a given solution set, which is the population of a certain generation of the GA, the Pareto set is found. The fitness of population members in the next GA generation is calculated by a distance metric with a reference to the Pareto set of the previous generation. As we are unable to combine the objectives in some way, we resort to this distance metric in the positive Pareto space of the previous solutions, as the fitness of the current solutions. This new GA-based multicriteria optimization method is proposed here, and it is capable of handling any generally formulated multicriteria optimization problem. The main idea of the method is described in detail in this paper along with a detailed numerical example. Preliminary computer generated results show that our approach produces better, and far more Pareto solutions, than plain stochastic optimization methods.

1 Introduction

In recent years, genetic algorithms (GAs) (Holland 1975) have emerged as one of the most effective and robust optimization techniques. GAs have been commonly used for directed stochastic hill climbing in single-criterion optimization problems. However, many real world problems are multicriteria in nature. All research efforts at developing a GA-based multicriteria optimization method have gone either into the design of a more general *Pareto set* based selection mechanism capable of handling multiple criteria, or to the development of combination techniques to combine the multiple criterion somehow, to produce a scalar quantity for the simple GA selection scheme to work. These techniques have used the *niched Pareto* (Horn *et al.* 1993), *Pareto tournament selections* (Louis 1993) or *Pareto nondomination based ranking* (Belegundu *et al.* 1994) approaches, which fail to give very robust methods to solve the generalized multicriteria optimization problems with GAs. In some approaches, the GA uses some ad-hoc weighted function of the multiple criterion to produce a scalar *fitness function*, which is then used to implement the GA's proportionate selection scheme which selects offspring based on the ratio of the solution's fitness value to the population's total fitness value. The GA selection scheme models nature's survival-of-the-fittest policy, to insure that future generations of any particular species will have members with an increased average fitness compared to that of the present generation. Thus when selection is implemented, better (more-fit) solutions survive and worse (less-fit) solutions perish. In the simple genetic algorithm, a more fit solution receives a proportionately higher number of copies to be included in the population of the next generation, thus awarding the population a higher probability of survival through subsequent generations.

By the word *fitness* here we mean a certain measure of the performance of a member of the population, in its given reaction environment (Holland 1975). The fitness is a single number (value) in a given scale and within certain boundaries. This fitness value is an absolute measure, which the GA uses to cull the population in a certain generation and thereby let better fit members remain and produce offspring. This is the equivalent of the selection process described above. However, in theoretical optimization jargon, the notion fitness would simply mean a single value of the objective function for a certain solution. Thus either in GA terms or in optimization terms, incorporating a single fitness value to a population member (string) would be quite straightforward for a single criterion problem, but would be a nondescript and ambiguous quantity for a multicriteria optimization problem. The idea of a single-value fitness, which is the impetus for the success of the GA, would fall short in applicability in the multicriteria optimization problem.

The most common method to combine multiple criteria into a single objective function is to use weights for a weighted linear combination of different objective values into a single scalar objective value. In these methods there is an implicit assumption that, for a given problem, the weights of the different criteria can be established and also there is a common metric between the different criteria, which makes the weighted combination possible.

2 Genetic algorithms

2.1 Introduction to genetic algorithms

The motivational idea behind GAs is natural selection implemented through selection and recombination operators (Goldberg 1989). A population of "organisms" (usually represented as bit strings) is modified by the probabilistic ap-

```
Genetic algorithm (the basic algorithm in program form)
Procedure GA
begin
    t = 0
    initialize at random P(t)
    evaluate P(t)
    while termination is not valid;
        begin
            forall P(t)
                selection P(t);
                crossover P(t);
                mutation P(t);
                evaluate P(t);
            endforall;
            t = t + 1
        end;
    endwhile;
end;
```

Fig. 1. The simple genetic algorithm

plication of the genetic operators from one generation to the next. The basic algorithm where $P(t)$ is the population of strings at generation t is given in Fig. 1.

Evaluation of each string that corresponds to a point in a search space is based on a fitness function that is problem dependent. This corresponds to the environmental determination of survivability in natural selection. Selection is done on the basis of relative fitness and it probabilistically culls from the population those points that have relatively low fitness. Recombination, which consists of mutation and crossover, imitates sexual reproduction. Mutation, as in natural systems, is a very low probability operator and just flips a specific bit. Crossover, in contrast, is applied with high probability. It is a structured, yet stochastic operator that allows information exchange between points. Simple crossover is implemented by choosing a random point in the selected pair of strings and exchanging the substrings defined by that point. Figure 2 shows how crossover mixes information from two parent strings, producing offspring made up of parts from both parents. We note that this operator, which does no table lookups or backtracking, is very efficient because of its simplicity.

2.2 Genetic algorithm encodings

Understanding how to represent a problem, and any domain knowledge of the problem, in a genetic algorithm requires (i) knowing which properties of a search space GAs use to guide their exploration and (ii) the need to produce viable offspring during crossover. A smidgeon of GA theory clears the way (Gero *et al.* 1994).

Crossover causes genotypes (an encoded individual) to be cut and spliced. This means that instead of considering the fate of individual strings in analysing a genetic algorithm, we must consider the substrings created and manipulated by crossover. These substrings define regions of the search space and are called *schemas*. More formally, a schema is a template that identifies a subset of strings with similarities at certain string positions. Holland's *schema theorem* (Holland 1975) is fundamental to the theory of genetic algorithms. For example, consider binary strings of length 6. The schema

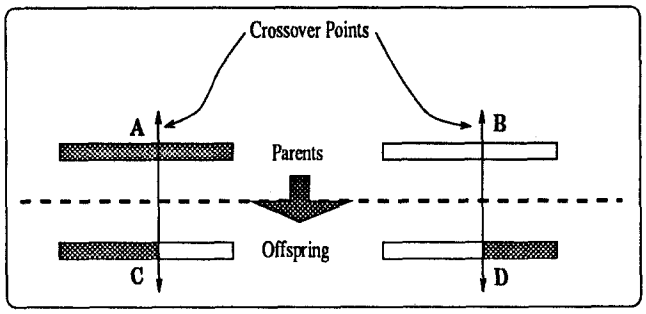

Fig. 2. Crossover of the two parents A and B produces the two children C and D. Each child consists of parts from both parents, which leads to information exchange

1**0*1 describes the set of all strings of length 6 with 1*s* at positions 1 and 6 and a 0 at position 4. The "*" denotes a "don't care" symbol, which means that positions 2, 3 and 5 can be either a 1 or a 0. Although we only consider a binary alphabet, this notation can be easily extended to nonbinary alphabets. The *order* of a schema is defined as the number of fixed positions in the template, while the *defining length* is the distance between the first and last specific positions. The order of 1**0*1 is 3 and its defining length is 5. The *fitness* of a schema is the average fitness of all strings matching the schema.

The genetic algorithm therefore implicitly processes schemas. Because of the bias introduced by selection and crossover, certain types of schemas rapidly increase their proportions in a population. Not only does the rate of increase depend on their fitness, it also depends on the syntactic properties of order and defining length. This is of great importance when incorporating domain knowledge into the bit strings that a GA manipulates. The schema theorem proves that relatively short, low-order, above average schemas obtain an exponentially increasing number of copies in subsequent generations. This leads to the building block hypothesis, which states that genetic algorithms work near-optimally by combining short, low-order, high fitness schemas called *building blocks* (Goldberg 1989). Thus when encoding domain knowledge, we should choose an encoding that reflects a GA's bias toward short, low-order building blocks. This bias is critically affected by the crossover operator. The plethora of crossover operators, each with its own bias, implies analysing and carefully matching an operator's bias in the encoded representation of domain knowledge. In practice, many encodings work well because of the large number of schemas and a GA's reliance on fitness information.

Representation of the problem space in terms of the binary string is thus a major issue in determining the GAs success. In our problem we concatenated the binary coding of decision variables to form the GA binary string. The number of bits (binary digits) used to code each decision variable were the same and the evaluation program of the GA decoded these bits into its decimal equivalent, which was then flexibly scaled to retrieve each decision variable.

3 Problem formulation

The general nonlinear multicriteria optimization problem is formulated as follows:

$$\min_{x \in \mathbb{R}^n} \mathbf{f}(\mathbf{x}), \tag{1}$$

under inequality constraints

$$g_k(\mathbf{x}) \geq 0, \quad k = 1,2,\ldots,K, \tag{2}$$

and equality constraints

$$h_m(\mathbf{x}) = 0, \quad m = 1,2,\ldots,M, \tag{3}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ is a vector of decision variables, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_I(\mathbf{x})]^T$ is a vector of objective functions. The solution of the problem is to find the set of Pareto optimal solutions (nondominated solution). Let X be a set of feasible solutions, i.e. a set of solutions which satisfies constraints given in (2) and (3). The Pareto optimum is defined as follows. A solution $\mathbf{x}^* \in X$ is Pareto optimal if and only if there exists no $\mathbf{x} \in X$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for $i = 1,2,\ldots,I$ with $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for at least one i.

This definition is based upon the intuitive conviction that the point $\mathbf{x}^*$ is chosen as optimal if no criterion can be improved without worsening at least one other criterion.

The proposed method generates a solution and compares the objective functions considering Pareto optimality conditions. Thus the constrained optimization problem is transformed into an unconstrained problem using an exterior penalty function method. In this case the objective functions are evaluated using the formula

$$\phi_i(\mathbf{x}) = f_i(\mathbf{x}) + r \sum_{m=1}^{M} [h_m(\mathbf{x})]^2 + r \sum_{k=1}^{K} G_k [g_k(\mathbf{x})]^2 ,$$

$$\text{for} \quad i = 1,2,\ldots,I, \tag{4}$$

where G_k is the Heaveside operator such that $G_k = 0$ for $g_k(\mathbf{x}) \geq 0$ and $G_k = 1$ for $g_k(\mathbf{x}) < 0$, and r is a positive multiplier which controls the magnitude of the penalty terms.

As in other applications of the penalty method, the value of penalty multiplier r might have influence on the optimization process. This problem will be discussed later.

Stochastic methods generate solutions within the assumed bounds. Thus for the method presented in this paper, all solutions are generated within the given bounds using the formula

$$x_n^\ell \leq x_n \leq x_n^u, \quad \text{for} \quad n = 1,2,\ldots,N, \tag{5}$$

where x_n^ℓ and x_n^u are estimated lower and upper bounds of the n-th decision variable.

4 The new algorithm

4.1 Detailed description of the method

The general idea of the method is contained in evaluating the fitness for each solution that the GA generates and this fitness has a greater value if the solution is farther away from the existing Pareto set.

Each Pareto solution has a value that will be called the *distance* value and will be denoted by d_ℓ for $\ell = 1,2,\ldots,\ell_p$, where ℓ_p is the number of existing Pareto solutions. Let $\mathbf{f}_\ell = [f_{1\ell}, f_{2\ell}, \ldots, f_{I\ell}]^T$ be a vector of objective functions for the ℓ-th Pareto solution. For each new solution $\mathbf{x}$, that is generated by the GA, we can easily calculate relative distances from all Pareto solutions using the following formula:

$$z_\ell(\mathbf{x}) = \sqrt{\sum_{i=1}^{I} \left(\frac{f_{i\ell}^p - \phi_i(\mathbf{x})}{f_{i\ell}^p} \right)^2}, \quad \text{for } \ell = 1,2,\ldots,\ell_p. \tag{6}$$

The minimum distance solution is taken into consideration while calculating the fitness value.

Note that using (6) the distances do not depend on the units in which the objective functions are expressed.

The steps of the method are as follows.

1. Substitute $t = 1$ and $j = 1$, where t is the index of generation, and j is the index of the generated solutions within the t generation.
2. Generate at random a solution using the GA. This first solution is taken as a Pareto optimal solution with fitness F equal to d_1, where d_1 is an arbitrarily chosen value called the *starting* distance.
3. Substitute $j = j + 1$ and check if $j \leq J$, where J is the assumed number of solutions generated within the t generation. If this is true go to step 4, and if not, go to step 8.
4. If this is the first generation, the GA generates at random the solution $\mathbf{x}$. If not, the GA generates the solution $\mathbf{x}$ on the basis of the fitness F calculated in step 7.
5. For the solution $\mathbf{x}$ generated by the GA calculate the relative distances from all existing Pareto solutions using the formula (6), i.e. calculate $z_\ell(\mathbf{x})$ for $\ell = 1,2,\ldots,\ell_p$.
6. Find the minimum value from the set $\{z_\ell(\mathbf{x})\}$, and the index ℓ^* for which this minimum was found. In other words, find
$$z_{\ell^*}(\mathbf{x}) = \min_{\ell = 1,2,\ldots,\ell_p} \{z_\ell(\mathbf{x})\}. \tag{7}$$
The index ℓ^* indicates which of the existing Pareto optimal solutions is the nearest to the newly generated solution.
7. Check if the newly generated solution $\mathbf{x}$ is a new Pareto solution.
If it is, calculate the fitness value for this new Pareto solution using the formula
$$F = d_{\ell^*} + z_{\ell^*}(\mathbf{x}). \tag{8}$$
Update the set of existing Pareto solutions, i.e. eliminate all solutions that the new Pareto solution dominates or if not, gets added to the existing set. Set the distance value for the new Pareto solution equal to this fitness value F, and go to step 3.
If the generated solution is not a new Pareto solution, calculate the fitness value for this solution using the formula
$$F = d_{\ell^*} - z_{\ell^*}(\mathbf{x}). \tag{9}$$
If $F \geq 0$ go to step 3. If not, then substitute $F = 0$ and then go to step 3. This substitution is dictated by the fact that the simple GA operates only on positive fitness values.
8. Find the maximum value of the distances from all existing Pareto solutions, i.e. find
$$d_{\max} = \max_{\ell = 1,2,\ldots,\ell_p} \{d_\ell\}, \tag{10}$$
where ℓ_p is the number of existing Pareto solutions.

9. Substitute $d_{\max}$ for all existing Pareto solutions

$$d_\ell = d_{\max} \quad \text{for } \ell = 1,2,\ldots,\ell_p . \tag{11}$$

10. Substitute $t = t + 1$. If $t > T$, then terminate calculations where T is the preassigned number of generations that we want to consider. If $t \leq T$, we check if there is an improvement through t^* number of generations, where t^* is also a user preassigned number. If there is any improvement, substitute $j = 0$ and go to step 3. Otherwise terminate the GA runs and calculations. Such a situation means that the GA is not able to find any new Pareto optimal solution through last t^* generations.

Here we use two types of termination criterion. The termination criterion, which indicates that there is absolutely no improvement in the fitness after running t^* numbers of generations, is the more important criterion. We can alternatively stop the calculations just after running the preassigned number of generations. This second criterion refers to the computational time for generating the Pareto solutions.

For selection of the Pareto optimal solutions and for updating the Pareto sets we use the algorithm presented by Osyczka (1984). Part of the program code for this algorithm is taken from CAMOS (Osyczka 1992). Both the algorithms and the code were suitably modified for our purposes here.

The GA maximizes the fitness which should have a positive value. Thus the values of d_1 and r should be properly chosen. If the generated solution is out of the feasible region, then the calculated value of $z_\ell(\mathbf{x})$ will be greater, the bigger value of r, and (9) will produce a value of F which would be less than 0. In such a case, the finally returned fitness is equal to zero (see step 7 of the algorithm). If the returned fitness has a value of zero for too many solutions, then the GA might have some difficulties to allocate the selective pressure properly while implementing the proportionate selection scheme. Equation (9) might also give F less than zero in an event where the initial distance value d_1 is too small. On the other hand, if the initial distance value d_1 is too large, then the difference between fitnesses may become too small, and thereby again the convergence in the GA will slow down as the selective pressure is less or is very evenly distributed.

The method described here was coded in Fortran computer programming language and tested for use with a simple GA which was coded in *C* programming language. Part of the SGA (Goldberg 1989) code was used for the GA engine. The program is able to solve any nonlinear multicriteria optimization problem formulated as (1), (2) and (3), and is coded in the same way as in CAMOS (Osyczka 1992).

4.2 Handworked example

We shall consider a simple example to show in detail, via illustrations, the implementation of the major steps of our algorithm. The problem is to minimize the following two objective functions:

$$f_1(\mathbf{x}) = x_1 + x_2^2, \quad f_2(\mathbf{x}) = x_1^2 + x_2 ,$$

under the inequality constraints

$$g_1(\mathbf{x}) \equiv 12 - x_1 - x_2 \geq 0 ,$$

$$g_2(\mathbf{x}) \equiv -x_1^2 + 10x_1 - x_2^2 + 16x_2 - 80 \geq 0 .$$

We assume that upper and lower bounds on the decision variable are

Table 1. Table for worked out numerical example - generation number 1

Solution No.	x_1	x_2	$\phi_1(\mathbf{x})$	$\phi_2(\mathbf{x})$	$g_1(\mathbf{x})$	$g_2(\mathbf{x})$	ℓ^*	$z_{\ell^*}(\mathbf{x})$	F	ℓ_p
1	2.542	9.882	152.36	68.50	-0.424	-0.583	-	-	20	1
2	3.089	9.301	104.89	34.13	-0.391	3.658	1	0.590	20.590	1
3	3.612	8.807	98.85	39.52	-0.420	6.424	1	0.168	20.758	2
4	4.189	8.269	93.68	46.93	-0.459	8.270	2	0.195	20.953	3
5	4.712	7.712	82.27	48.00	-0.425	8.834	3	0.124	21.077	4
6	5.269	7.189	78.06	56.06	-0.459	8.270	4	0.176	21.253	5
7	5.763	6.612	63.65	53.99	-0.376	6.493	5	0.188	21.441	5
8	6.340	6.075	60.50	63.53	-0.415	3.499	4	0.183	21.624	6
9	6.882	5.537	92.15	107.51	-0.419	-0.608	6	0.867	20.757	6
10	5.714	5.435	35.25	38.09	0.850	1.910	5	0.535	21.976	2

Table 2. Table for worked out numerical example - generation number 2

Solution No.	x_1	x_2	$\phi_1(\mathbf{x})$	$\phi_2(\mathbf{x})$	$g_1(\mathbf{x})$	$g_2(\mathbf{x})$	ℓ^*	$z_{\ell^*}(\mathbf{x})$	F	ℓ_p
1	2.601	7.228	54.85	13.99	0.217	2.650	1	0.758	22.734	2
2	3.143	7.771	63.53	17.65	1.086	5.433	1	0.305	22.429	2
3	2.166	8.856	80.60	13.54	0.977	0.236	1	0.470	23.205	3
4	3.686	5.601	35.05	19.18	2.712	0.151	2	0.496	22.472	3
5	6.403	6.686	169.90	166.48	-1.089	5.305	2	8.585	13.886	3
6	2.381	9.203	87.08	14.87	0.416	0.693	3	0.126	23.078	3
7	3.153	6.668	47.85	16.63	2.160	3.864	1	0.227	22.962	4
8	6.281	5.171	74.64	86.25	0.548	-0.644	2	3.673	18.799	4
9	2.379	8.196	69.55	13.82	1.432	8.196	3	0.138	23.343	5
10	3.886	5.361	32.63	20.46	2.751	0.799	2	0.096	22.568	6

$$2 \leq x_1 \leq 7, \quad 5 \leq x_2 \leq 10 .$$

We make an assumption here that for evaluating the values of the $\phi_i(\mathbf{x})$ function, the penalty parameter r is 100. We shall present the results assuming values of $J = 10$, $d_1 = 20$ and $T = 2$.

The results of calculations of $z_{\ell^*}(\mathbf{x})$ and F for all generated points are presented in Tables 1 and 2. For each solution generated in the second generation, the value of F is taken as the fitness of the solution for the GA to perform its selection scheme.

The first solution generated is a Pareto optimal solution with an initial distance value of 20. The second solution eliminates the first one and since the relative distance is 0.590 this solution obtains its distance value equal to 20.590. The third solution generated is a new Pareto solution, thus it is added to the previous one, and since the relative distance is equal to 0.168 this solution obtains the distance value equal to 20.758. The same situation occurs for solutions 4, 5 and 6. Solution 7 is also a new Pareto solution, but eliminates the fifth solution from the existing set. This is illustrated in Fig. 3a. Solution 8 is a new Pareto solution and so it is added to the existing set. Solution 9 is not a Pareto solution, with a relative distance value 0.867 to the sixth Pareto solution which is the nearest one to it. Thus this solution obtains a fitness value $21.624 - 0.867 = 20.757$. Solution 10 is a new Pareto solution nearest to the fifth Pareto solution. Thus it obtains the distance value 21.976 and since it eliminates four Pareto solutions, it eventually becomes the second Pareto solution in the updated Pareto set. This is illustrated in Fig. 3b. The initial generation ended with two Pareto solutions. The maximum value of the distances among these two solutions, i.e. the value of 21.976 is substituted for both of these Pareto

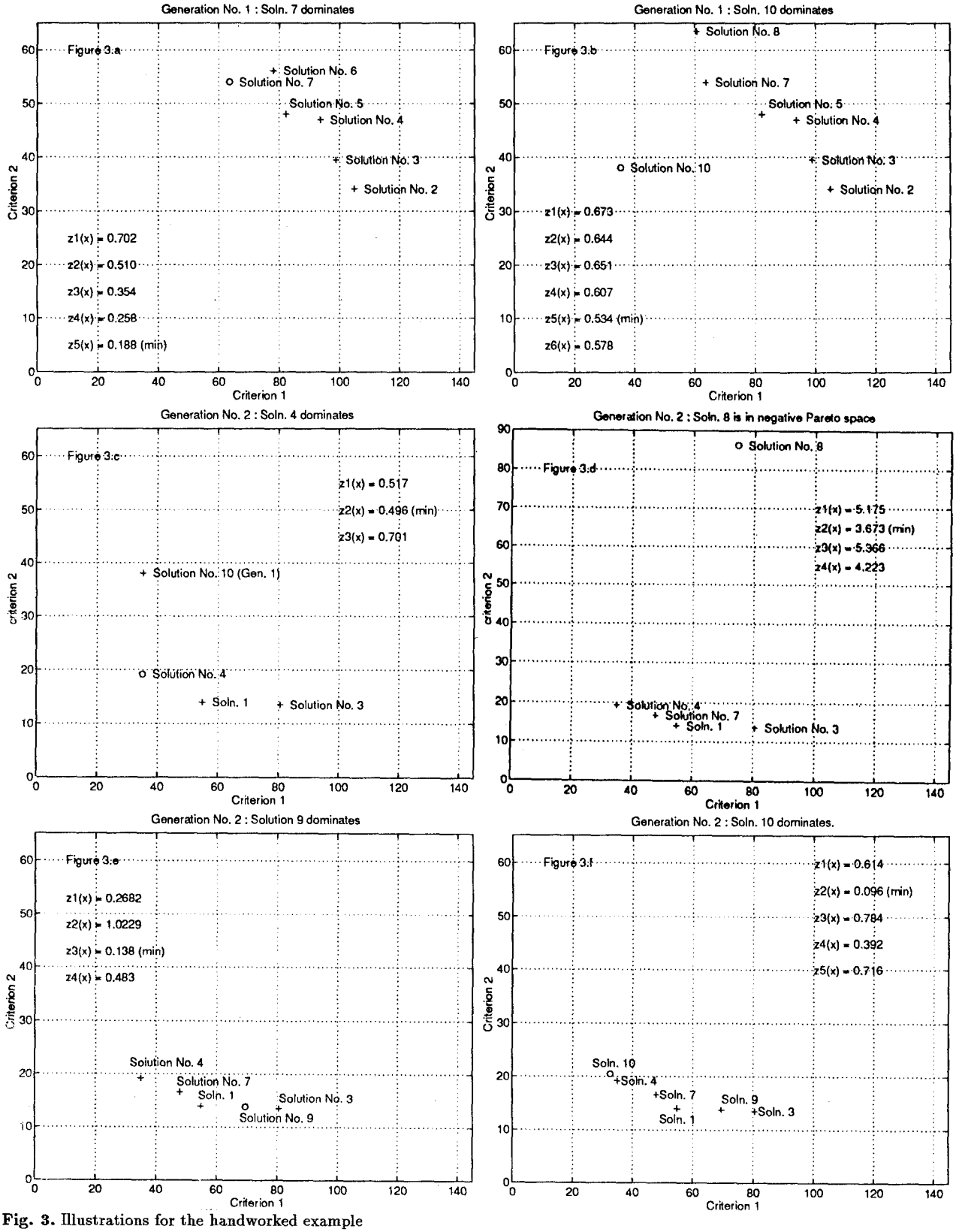

Fig. 3. Illustrations for the handworked example

solutions.

For the second generation solution 1 eliminates the first Pareto solution. Solution 2 is not Pareto optimal, solution 3 is a new Pareto optimal solution added to the previous two

solutions. Solution number 4 is also a new Pareto solution, but it eliminates solution number 2 and thus it becomes the new solution number 2. This is illustrated in Fig. 3c. Solutions 5 and 6 are not Pareto optimal. Solution 7 is a new Pareto solution added to this previous set. Solution 8 is not a Pareto optimal solution with quite a big distance away from the existing Pareto set and this is illustrated in Fig. 3d. Solutions 9 and 10 are new Pareto solutions added to the existing set and are illustrated in Figs. 3e and f. The second generation ended with five Pareto optimal solutions and for the next (third) generation, all of these solutions obtain the distance value equal to 23.343, which is the highest among these.

Note that the first nine solutions from the first generation are not in the feasible region.

4.3 Numerical example worked on the computer

We will now present results for a problem which has been coded and run on the computer. We consider a more complicated example. The problem we deal with here is to minimize

$$f_1(\mathbf{x}) = -(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2),$$

$$f_2(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2,$$

subject to inequality constraints

$$g_1(\mathbf{x}) \equiv x_1 + x_2 - 2 \geq 0,$$
$$g_2(\mathbf{x}) \equiv 6 - x_1 - x_2 \geq 0,$$
$$g_3(\mathbf{x}) \equiv 2 - x_2 + x_1 \geq 0,$$
$$g_4(\mathbf{x}) \equiv 2 - x_1 + 3x_2 \geq 0,$$
$$g_5(\mathbf{x}) \equiv 4 - (x_3 - 3)^2 - x_4 \geq 0,$$
$$g_6(\mathbf{x}) \equiv (x_5 - 3)^2 + x_6 - 4 \geq 0,$$

and side constraints

$$0 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 10,\ 1 \leq x_3 \leq 5,$$
$$0 \leq x_4 \leq 6,\ 1 \leq x_5 \leq 5,\ 0 \leq x_6 \leq 10.$$

In order to compare the results, we perform the calculations for the same number of calls (20000) made for function evaluations using the plain stochastic method and our proposed method. The results are shown in Fig. 4. While using the plain stochastic method we obtained 13 solutions, which are marked by a star (*) in the figure. Using the proposed method we obtained 28 solutions which are marked by plus (+) in Fig. (4). The solutions from the plain stochastic method were obtained while running this method from CAMOS [refer to the method for Pareto set generation in the book by Osyczka (1992)]. Both CAMOS and the presented method use the same Pareto algorithms for selection and updating the set of Pareto optimal solutions. It is clear that our GA-based method generates distinctly better and many more Pareto solutions than the plain stochastic method.

5 Conclusions

Using any of the methods based on weighting of criteria, we can obtain only one Pareto optimal solution while using the GA. In the same amount of computation time, we usually obtain a reasonable large set of Pareto solutions, using the method we have proposed here. The computational economy of any optimization method not only depends on the final optimal solution found, but also on the number of calls made to the complex evaluation routines. Therefore the comparison of the results obtained by our method and the plain stochastic search methods was made on the basis of the same number of calls made to the evaluation function. We noticed that our GA-based multicriterion optimization method gave distinctly better and many more Pareto solutions than the plain stochastic multicriteria optimization method.

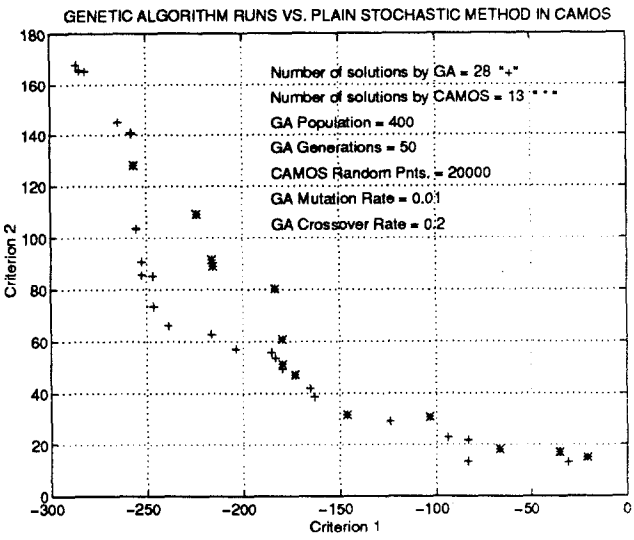

Fig. 4. Computer generated results

References

Belegundu, A.D.; Murthy, D.V.; Salagame, R.R.; Constans, E.W. 1994: Multi-objective optimization of laminated ceramic composites using genetic algorithms. *Proc. 5th AIAA/NASA/USAF/ISSMO Symp. on Multidisciplinary Analysis and Optimization* (held in Panama City, FL), pp. 1015–1022. Washington D.C.: AIAA

Gero, J.S.; Louis, S.; Kundu, S. 1994: Evolutionary learning of novel grammars for design improvement. *Artificial Intelligence in Engineering Design, Analysis and Manufacture (AIEDAM)* 8, 83–94

Goldberg, D.E. 1989: *Genetic algorithms in search, optimization, and machine learning.* Reading, Massachusetts: Addison-Wesley

Holland, J.H. 1975: *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press

Horn, J.; Nafpliotis, N.; Goldberg, D.E. 1993: Multiobjective optimization using the niched Pareto genetic algorithm. *Tech. Report IlliGAL Report No 93005,* IlliGAL, Dept. of General Engineering, University of Illinois at Urbana Champaign

Louis, S.J.; Rawlins, G.E. 1993: Pareto optimality, GA-easiness and deception. *Proc. Fifth Int. Conf. on Genetic Algorithms.* San Mateo: Morgan-Kaufmann

Osyczka, A. 1984: *Multicriterion optimization in engineering with Fortran programs.* Chichester, New York: Ellis Horwood, Wiley

Osyczka, A. 1992: *Computer aided multicriterion optimization system (CAMOS) - software package in Fortran.* Cracow: International Software Publishers

Received Dec. 20, 1994
Revised manuscript received April 10, 1995