Multiple crossover algorithm for constrained optimization problems

TOMASZ D. GWIAZDA
School of Management
Warsaw University
02-678 Warsaw, Szturmowa 3
POLAND

Abstract: Studies of constrained optimization problems are usually focused on the development of new evolutionary algorithms or operators. This paper does not propose anything new apart from a more effective use of the already existing operator. It is focused on the crossover operator. Starting from the standard form – the linear combination, we study the effects of multiple crossover of the same pair of parents, and then multiple crossover of a parent with all other individuals from the population. Such an approach, although very costly from the point of view of the number of function evaluations, is effective, which is proved by the results of eleven test cases from the literature.

Key-Words: evolutionary algorithms, optimization technique, constrained optimization, genetic operators, crossover operator

1 Introduction

The general nonlinear parameter optimization problem is defined as:

optimize: $f(x), x = (x_1, ..., x_n) \in F \subseteq S \subseteq R^n$ (1)

such that:

$$\begin{cases} g_i(x) \le 0, & \text{for } i = 1,...,q \quad \text{inequality constaraints} \\ h_j(x) = 0, & \text{for } j = q+1,...,m \quad \text{equality constraints} \end{cases} \quad (2)$$

where f, g_i and h_j are real-valued functions on the search space S. The satisfaction of the set of constraints (g_i, h_j) defines the feasible region F.

There have been many efforts to use evolutionary algorithms for constrained numerical optimization [11][6]. They can be grouped into five basic categories:

- Methods based on preserving the feasibility of solutions.
- Methods based on penalty functions.
- Methods based on a search for feasible solutions
- Methods based on decoders.
- Other hybrid methods.

The method shown in this paper may be regarded as belonging to the third of the aforementioned categories. In this category evolutionary methods emphasize a distinction between feasible and infeasible solutions:

- Behavioral memory method [14] – considers the problem constrains in sequence, once a sufficient number of feasible solutions is found in the presence of one constraint, the next constraint is considered;
- Method of superiority of feasible points [12] – assumes that any feasible solution is better than any infeasible solution;
- Genocop III [9] – repairs infeasible individuals.

The paper is organized as follows. Section 2 presents the proposed method. Section 3 describes the first test performed on the set of 11 test cases from the literature. Section 4 adds an additional mechanism to the proposed method and analyses its effects in the second test. Section 5 concludes the paper.

2 A description of the method

The proposed method is based on greatly simplified analogy with the world of plants, where one can hardly speak of selection of parents for reproduction and crossing-over occurs on a mass scale, on an everyone with everyone basis, e.g. a tree has no influence on the selection of partners and has to generate offspring on a mass scale with every tree of the same kind that grows within its range. The mass character and certain egalitarianism of reproduction is the key to success.

The fact that crossover of parents takes place only once and, as a result, the chances for accumulation of the valuable genetic material of individual parents in one offspring are very small, is a common characteristic of evolutionary algorithms, regardless of the type of problem that is optimized. Therefore, despite the potential of parents or, in a wider sense, of the whole current population, frequently the maximum possible progress in searching is not achieved, because the full crossover potential is not utilized due to complete reliance on randomness of crossover operator.

Consequently the proposed method, which is referred to as multiple-crossover algorithm (denoted here as M-CA) is based on following five assumptions.

- Multiple (instead of single) crossover of the same pair of parents may lead to better utilization of the potential of the parents and, in a wider sense, in the whole population. The idea of multiple crossover is not new (see [3]), but its efficiency has not been tested yet in the context of the problems considered here.
- Secondly, the proposed method assumes that any feasible solution is better than any infeasible solution. This assumption is modeled on the paper by Powell and Skolnick [12]. That paper used a method based on a classic penalty approach, but with one notable exception. Each individual is evaluated by the formula:

$$evaluation(x) = f(x) + r\sum_{j=1}^{m} f_j(x) + q(t,x) \qquad (3)$$

- where r is a constant, the original component $q(t,x)$ is an additional iteration- dependent function that influences the evaluations of infeasible solutions. The point is that the method distinguishes between feasible and infeasible individuals by adopting an additional heuristic. For any feasible individual x and any infeasible individual *y*: *evaluation(x)* < *evaluation(y)*[1], i.e. any feasible solution is better than any infeasible one. This can be achieved in many ways. Powell and Skolnick achieved this by mapping evaluations of feasible solutions into interval *(-¥,1)* and infeasible solutions into interval *(1,¥)*. In the method described in this paper the selection operator is not used (which is described below), so the only thing needed here is a method of comparing individuals in order to find the best individual in the population. It is necessary to select the best individual, because in accordance with the principles of the elitist strategy applied here it will automatically become a member of the offspring population. Individuals are compared by the following method: If we compare two individuals, the one with a smaller deviation from the constraints is better. If the deviation is the same for both individuals, the one with lower (higher for maximum problems) evaluation function value is better. In the case of test function analyzed here, the quotient of the total of absolute distances from the respective constraints and the number of constraints is the measure of deviation from constraints.
- The effectiveness of multiple crossover should be independent on the number of individuals in the population

[1] Assuming the problem considered here is a minimum problem, otherwise: *evaluation(x)>evaluation(y)*

- Parent selection operator can be replaced by mass reproduction.
- The idea of mass reproduction should the answer to the problem of constraints satisfaction

On the basis of the assumptions discussed above, a method presented in Figure 1 was developed. First, population *P(0)* is initialized. In accordance with the third assumption the number of individuals (denoted as *N*) in the population is small (here *N = 10*). Then that population is evaluated (as described in the second assumption) and the best individual is determined. The rest of the method is described in the form of a loop, which ends when the previously prescribed number of iterations has been performed.

The main loop of the method starts with placing the best individual from *P(t-1)* in the offspring population *P(t)*. Then the remaining *N-1* offspring are generated in the following way: a parent *X Î P(t-1)* is selected by pure random selection, each of *X Î P(t-1)* may be selected only once; an evolutionary operator is selected at random and then an offspring is generated with the use of the randomly selected operator. A selected parent is the argument of the operator (Fig.2 r.1; Fig.3 r.1), for multiple crossover operator the other parent (different than *X*) is selected at random in the body of the operator (Fig.2 r.3). The loop ends with the evaluation of the newly generated population.

A selection operation in the classic sense is not used here – it is a consequence of fourth assumption. Moreover, it can be seen that as a result of the application of the elitist strategy one individual from *P(t-1)* will not be selected for further processing – that individual may die.

Random selection of an evolutionary operator depends on the probability of mutation. The value of this parameter is relatively high (see description of the experiments) in order to ensure the maximum intensity of exploration of search space, which is meant to supplement the intensive utilization of potential of the current population, realized using multiple-crossover operator.

Figure 1. Multiple-crossover algorithm

```
1.  Procedure M-CA
2.  set starting generation number $t=0$
3.  randomly initialize population $P(t)$
4.  evaluate $P(t)$
5.  while (not termination-condition) do
6.  $t=t+1$
7.  insert best individual from $P(t-1)$ to $P(t)$
8.    while ($P(t)$ is not full) do
9.    select parent X Î P(t-1) which was not selected
      before in this loop
10.   set $b$ = random (real) number form $<0,1>$
11.     if $b \le$ probability of mutation then
```

```
12.     S = mutation operator
13.     else
14.     S = multiple crossover operator
15.     end if
16.   generate offspring X^I Î P(t) from parent X Î P(t-1)
      by operator S
17.   end while
18. evaluate P(t)
19. end while
20. end procedure
```

The crossover and mutation operators used here are modelled on the most frequently used operators for the analysed class of problems. The multiple crossover operator (Figure 2) is modelled on the arithmetic-crossover operator [7] and the difference, resulting from the first assumption, is the multiple crossover of the same pair of parents and then the selection of one (instead of a pair) best offspring from among all the generated offspring.

Depending on the test number, mutation is performed with the use of one or both of the following operators: the first one, MUT1, involves a simple replacement of a parent with a completely new randomly generated individual; the second one, MUT2 (see Figure 3), is based on non-uniform mutation [7], with the reservation that in MUT2 each gene from solution vector has a chance for mutation (Fig.3 r.2) with probability equal to 0.5

Figure 2. Multiple crossover operator (version 1)

```
1.  Procedure Multiple-crossover-1 (X Î P(t-1))
2.  BEST = X
3.  select by random parent Y Î P(t-1) different than X
4.  for k-times generate offspring V
5.  set a = random (real) number form <0,1>
6.  V = aX + (1 - a)Y
7.    if V is better that BEST then
8.    BEST = V
9.    end if
10. end for
11. return(BEST)
12. end procedure
```

Figure 3. Mutation operator MUT2. Where: $D(t,y)=yr(1-t/T)^b$; r – random (real) number from $<0,1>$; t – current generation number; T – maximal generation number; b – parameter determining the degree of non-uniformity (here b is set to 6 in order to achieve fine local tuning in the final phase of the algorithm run); *left(i)* and *right(i)* indicate the bottom and top limit of the domain of i-th component.

```
1.  Procedure MUT2 (X Î P(t-1))
2.  for every i-th component of vector X
3.  set a = random (real) number from <0,1>
4.    if a < 0.5 then
5.    set b = random (real) number from <0,1>
6.      if b < 0.5 then
7.      $v_i=x_i+D(t,right(i)-x_i)$
8.      else
9.      $v_i=x_i-D(t,x_i-left(i))$
10.     end if
11.   else
12.   $v_i=x_i$
13.   end if
14. end for
15. return(V)
16. end procedure
```

2 The first test

The known set of eleven test functions referred to as G1-G11 was used to test M-CA; the first five functions were published in [5] and the current content of G1-G11 set was published in [11]. Table 1 summarises the best results for the G1-G11 set published to date. It groups the results of a number of methods, because so far none of them has given equally good results for all test functions when used on its own. As we can see, the optimum results have been found for only 7 out of the 11 test functions. Moreover, the obtained results are stable (i.e. the same in all runs) for 5 functions only. This last feature is of course characteristic of evolutionary methods, but the difference between the worst and the best result is often too big (especially for the function G10).

Table 1. Summary of the results obtained so far by the best of the published evolutionary algorithms. The first column shows problem Ids. The second column shows the global solution, if known, if not – it shows in parentheses the best solution obtained so far. The third, fourth and fifth columns show the best, average and worst solutions obtained by specific EA (GEN stands for Genocop [8], S.O. for Strategic Oscillation [10], H.M. form Homomorphous Mappings [4], D.P. for Dynamic Penalty [2], S.R. for Stochastic Ranking [13], A.A. for Adaptive Algorithm [1]).

Problem ID	Global solution	Best	Average	Worst	Specific method
G1 (min)	-15	**-15**	**-15**	**-15**	GenocopII & S.R.
G2 (max)	Unknown (0.803619)	0.803553	n.a.	0.802964	S.O.
G3 (max)	1.0	**1**	0.999844	n.a.	A.A.
G4 (min)	-30665,5	**-30665.5**	**-30655.5**	**-30655.5**	S.R.
G5	Unknown	4707.52	n.a.	n.a.	D.P.

(min)	(4221.956)				
G6 (min)	-6961.81	**-6961.81**	**-6961.81**	**-6961.81**	A.A.
G7 (min)	24.3062	24.307	24.37	24.642	S.R.
G8 (min)	Unknown (0.095825)	**0.095825**	**0.095825**	**0.095825**	A.A. & S.R.
G9 (min)	680.63	**680.63**	680.648	n.a.	A.A.
G10 (min)	7049.33	7054.316	7559.192	8835.655	S.R.
G11 (min)	0.75	**0.75**	**0.75**	**0.75**	H.M. & A.A. & S.R.

The first test has been divided into two stages - Test 1-A and Test 1-B. During each stage, 20 runs of the M-CA algorithm were performed for each of the G1-G11 functions. In each run, the number of iterations was equal to 20000, the number of repetitions *k* = 50 (see Fig2. r.4), probability of mutation = 0.2. In Test 1-A both mutation operators – MUT1 i MUT2 were used. Each time, one of them was selected by random selection with equal probability. In Test 1-B only the MUT1 operator was used.

Systems defining constraints for the functions G3, G5 and G11 contain equations. In such cases, it is an accepted practice either to eliminate them [8] or transform into inequalities [1]. In all tests performed here the equations of functions G3 and G11 were transformed into inequalities (with the accuracy *e* = 0.0001) and the equations of function G5 were eliminated – their transformation into inequalities brought poor results.

Tables 2 and 3 present the results of the performed tests. As shown, the obtained results are equal to the global solution for 8 functions in Test 1-A and 7 functions in Test 1-B. Furthermore, 6 results in Test 1-A and 7 in Test 1-B were completely stable – identical in all test runs.

It should be noted, however, that the basic disadvantage of the M-CA method is its cost measured in terms of the number of evaluations of the tested functions. The estimated number of such evaluations is 8*50 = 400 evaluations in each run, which considerably exceeds the number of function evaluations in other evolutionary algorithms for G1-G11 published.

Table 2. Results obtained by M-CA in Test 1-A

Problem ID	Global solution	Best	Average	Worst
G1 (min)	-15	**-15**	**-15**	**-15**
G2 (max)	unknown (0.803619)	**0.803619**	0.796229	0.768789
G3 (max)	1.0	**1.0**	**1.0**	**1.0**
G4 (min)	-30665,5	**-30665.5**	-30665.3	-30664.5
G5 (min)	unknown (4221.956)	4342.451	4636.73	4931.009
G6 (min)	-6961.81	**-6961.81**	**-6961.81**	**-6961.81**
G7 (min)	24.3062	24.8324	26.0106	27.3438
G8 (min)	unknown (0.095825)	**0.095825**	**0.095825**	**0.095825**
G9 (min)	680.63	**680.63**	**680.63**	**680.63**
G10 (min)	7049.33	7065.24	7181.425	7307.19
G11 (min)	0.75	**0.75**	**0.75**	**0.75**

Table 3. Results obtained by M-CA in Test 1-B

Problem ID	Global solution	Best	Average	Worst
G1 (min)	-15	**-15**	**-15**	**-15**
G2 (max)	Unknown (0.803619)	0.798181	0.793198	0.785724
G3 (max)	1.0	**1.0**	**1.0**	**1.0**
G4 (min)	-30665,5	**-30665.5**	**-30665.5**	**-30665.5**
G5 (min)	unknown (4221.956)	4397.345	4399.395	4401.445
G6 (min)	-6961.81	**-6961.81**	**-6961.81**	**-6961.81**
G7 (min)	24.3062	24.3472	24.6717	24.8842
G8 (min)	unknown (0.095825)	**0.095825**	**0.095825**	**0.095825**
G9 (min)	680.63	**680.63**	**680.63**	**680.63**
G10 (min)	7049.33	7200.79	7310.85	7481.91
G11 (min)	0.75	**0.75**	**0.75**	**0.75**

3 The second test

The quality of the results obtained in the first test encouraged development of the M-CA method. In the second test, the following changes were introduced to the M-CA:

- The parent randomly selected for crossover is still crossed over (Figure 4) many times (*k* = 50), but this time with all other individuals from the current population in turn (Fig.4 r.3).
- The crossover process itself was also changed, this time (Fig.4 rows 7-12) the linear combination is realised for the randomly selected components of solution vector, while other components remain unchanged.
- The mutation operator probability was increased to *x/2*, where *x* is random (real) number from *<0,1>*.
- 40 M-CA test runs were performed, 20 with mutation realised with the use of MUT1 operator and 20 with MUT2.

The remaining M-CA parameters remained the same as in Test 1.

Figure 4. Multiple crossover operator (version 2)

```
Procedure Multiple-crossover-2 ($X$ Î $P(t-1)$)
BEST = $X$
for every $Y$ Î $P(t-1)$ different than $X$
  for k-times generate offspring $V$
    for every (i-th) component
    set $b$ = random (real) number from $<0,1>$
      if $b < 0.5$ then
      set $a$ = random (real) number from $<0,1>$
      $v_i = ax_i + (1 - a)y_i$
      else
      $v_i = x_i$
      end if
      end repeat
      if $V$ is better that BEST then
      BEST = $V$
      end if
    end for
  end for
  return(BEST)
  end procedure
```

The results of Test 2 are presented in Table 4. For the G7 function the presented results were obtained in 20 runs with the use of MUT2; for the function G10 the presented results were obtained in 20 runs with the use of MUT1. For the remaining functions, the results of runs with the use of MUT1 and MUT2 were identical.

For functions G1-G6, G8, G9, G11 the obtained results are equal to or better than the results obtained with the use of other revolutionary methods. Moreover, they are stable – identical in all runs of the algorithm. For function G7 the best result is worse than the one published in [13], but the difference between the worst and the best result is smaller and the average result is better. However, it should be stressed again that the cost of M-CA measured in terms of the number of function evaluations is very high – in Test 2 it reaches the level of $\cong 5*9*50=1500$ function evaluations in each iteration.

For function G10 the obtained result, although better than the results published to date, did not come close to the global solution value and, moreover, the observed difference between the worst and the best solution is too big. For this reason, Test 2 was repeated for G10 with an increased number of individuals in the population and the parameter k defining number of repetitions.

Test 2 was repeated for G10 with the following parameters:

- The number of individuals in the population = 250
- *k* = 250
- The number of iterations = 250
- The number of runs = 20
- Mutation crossover probability = *x/5*, where *x* is random (real) number from *<0,1>*.

Table 4. Results obtained by M-CA in Test 2

Problem ID	Global solution	Best	Average	Worst
G1 (min)	-15	**-15**	**-15**	**-15**
G2 (max)	unknown (0.803619)	**0.803619**	**0.803619**	**0.803619**
G3 (max)	1.0	**1.0**	**1.0**	**1.0**
G4 (min)	-30665,5	**-30665.5**	**-30665.5**	**-30665.5**
G5 (min)	unknown (4221.956)	**4221.956**	**4221.956**	**4221.956**
G6 (min)	-6961.81	**-6961.81**	**-6961.81**	**-6961.81**
G7 (min)	24.3062	24.323	24.356	24.399
G8 (min)	unknown (0.095825)	**0.095825**	**0.095825**	**0.095825**
G9 (min)	680.63	**680.63**	**680.63**	**680.63**
G10 (min)	7049.33	7052.24	7087.55	7133.95
G11 (min)	0.75	**0.75**	**0.75**	**0.75**

Table 5 presents the results of the repeated Test 2 for G10. As shown, this time M-CA gave a result equal to global solution. Moreover, the difference between the worst and the best result is small.

Table 5. Results obtained by M-CA repeated for G10 with: *k* = 250, number of individuals = 250, number of iterations = 250

Problem ID	Global solution	Best	Average	Worst
G10 (min)	7049.33	**7049.33**	7052.36	7055.29

4 Conclusion

This paper presents the effects of fuller utilisation of crossover operator for constrained optimization problems. Starting from the standard form – linear combination, the effects of multiple crossover of the same pair of parents and then the effects of multiple crossover of a parent with all other individuals from the population are presented. The proposed method was implemented in the form of the M-CA algorithm, which gave very good results (better or equal than results published to date) on the set of eleven test functions. In first test the obtained results were equal to the global solution for 8 out of 11 tested functions, furthermore 6 results were completely stable – equal to global optimum and identical in all test runs. In the second test 9 out of 11 results were stable and the remaining 2 was better than results obtained by other evolutionary methods in terms of stability or/and value achieved. Of course all results obtained by M-CA were feasible which is not a common feature of other evolutionary methods.

At the same time, however, the cost in terms of the number of function evaluations was very high – in extreme situation equal to 1500 function evaluations in each iteration. That level of cost might be not acceptable in real life problem solving situation.

However, these tests did not examine the effect of the values of a number of arbitrarily adopted M-CA parameters on the quality of the results. Therefore, further study of their effect on the effectiveness of the method is needed.

References:

[1] S. Ben Hamida and M. Schoenauer. An Adaptive Algorithm for constrained optimization problems, Proceedings of *Parallel Problem Solving from Nature VI*, pages 529-538, LNCS 1917, Springer Verlag, 2000

[2] J. Joines and C. Houck. On the Use of Non-stationary Penalty Functions to Solve Nonlinear Optimization Problems with Gas, *Proceedings of the 1994 IEEE conference on Evolutionary Computation*, pages 579-584, IEEE Press, Piscataway, NJ, 1994

[3] L. Kallel and M. Schoenauer. A priori comparison of binary crossover operators: No universal statistical measure, but a set of hints, *Proceedings of the Third European Conference Artificial Evolution 1997*, pages 343-355, 1997

[4] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings and constrained parameter optimization, *Evolutionary Computation*, 7(1):19-44

[5] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1992

[6] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*, Springer, 2000

[7] Z. Michalewicz and C. Janikow. Handling Constraints in Genetic Algorithms, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 151-157, Morgan Kaufmann, San Mateo, CA, 1991

[8] Z. Michalewicz and C. Janikow. GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints, *Communication of the ACM*, 1992

[9] Z. Michalewicz and G. Nazhiyath. GENOCOP III: A Coevolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints, *Proceedings of the 1995 IEEE Conference on Evolutionary Computation*, pages 647-651, IEEE Press, Piscataway, NJ, 1995

[10] Z. Michalewicz, G. Nazhiyath and M. Michalewicz, A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems, *Proceedings of the 5th Annual Conference on Evolutionary Programming*, pages 305-312, MIT Press, Cambridge, MA 1996

[11] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems, *Evolutionary Computation*, 4(1):1-32, 1996

[12] D. Powell and M.M. Skolnick. Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424-430, Morgan Kauffman, San Mateo, CA, 1993

[13] G. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization, IEEE *Transactions on Evolutionary Computation*, 4(3):284-294

[14] M. Schoenauer and S. Xanthakis. Constrained GA Optimization, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 573-580, Morgan Kauffman, San Mateo, CA, 1993