A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems

Zbigniew Michalewicz
Department of Computer Science
University of North Carolina
Charlotte, NC 28223
zbyszek@uncc.edu

Girish Nazhiyath
Department of Computer Science
University of North Carolina
Charlotte, NC 28223
gnazhiya@uncc.edu

Maciej Michalewicz
Institute of Computer Science
Polish Academy of Sciences
ul. Ordona 21
01-237 Warsaw, Poland
michalew@ipipan.waw.pl

Abstract

Numerical optimization problems enjoy a significant popularity in evolutionary computation community; all major evolutionary techniques (genetic algorithm, evolution strategies, evolutionary programming) have been applied to these problems.

However, many of these techniques (as well as other, classical optimization methods) have difficulties in solving some real-world problems which include non-trivial constraints. For such problems, very often the global solution lies on the boundary of the feasible region. Thus it is important to investigate some problem-specific operators, which search this boundary in an efficient way.

In this study we discuss a new experimental evidence on usefulness of so-called *geometrical crossover*, which might be used for a boundary search for particular problems. This operator enhances also the effectiveness of evolutionary algorithms (based on floating point representation) in a significant way.

1 Introduction

For many years, most evolutionary techniques were evaluated and compared with each other in the domain of function optimization. It seems also that the domain of function optimization will remain the primary test-bed for many new comparisons and new features of various algorithms. In particular, numerical optimization problems enjoy a significant popularity in evolutionary computation community; all major evolutionary techniques (genetic algorithms, evolution strategies, evolutionary programming) can be applied to these problems.

However, in this study we depart from differences between various evolutionary techniques for numerical optimization problems, and discuss rather some experimental evidence on usefulness of a new operator, so-called *geometrical crossover*. This operator seems to enhance the effectiveness of evolutionary algorithms (based on floating point representation) in a significant way; later in the paper we compare it with a better known *arithmetical crossover* on a few test cases.

The geometrical crossover emerged as a problem-specific operator for one particular constrained problem. In a similar way, we can construct (for other constrained problems) various crossover operators (e.g., *sphere crossover*; see Conclusions) and analyse their performance. Thus this study indicates another possible way for constraint handling in evolutionary methods, based on a search for a global solution on the boundary of the feasible region. Some other heuristic techniques (e.g., tabu search) recognized recently the importance of such search by investigating *strategic oscillation* (see Kelly et al. 1993, Glover and Kochenberger, 1995).

The paper is organized as follows. The following two sections survey briefly several operators and a few constraint-handling techniques for numerical optimization problems, which have emerged in evolutionary computation techniques (floating point representation) over the years. Section 4 presents a test case of a constrained optimization problem, which proved to be extremely difficult for most optimization methods. Section 5 discusses a special, problem-specific evolutionary system, which was "tuned" towards this particular problem; the system incorporates a new operator, which we call *geometrical crossover*. There is some experimental evidence on a general usefulness of this operator, which outperforms a better known *arithmetical crossover*; this is presented in Section 6. We conclude the paper by providing an example of another problem-specific crossover, *sphere crossover*, and indicate its usefulness on another test case.

2 Numerical Optimization and Operators

Most evolutionary algorithms use vectors of floating point numbers for their chromosomal representations. For such representation, many operators have been proposed during the last 30 years. We discuss them briefly in turn.

The most popular mutation operator is *Gaussian mutation*, which modified all components of the solution vector $\vec{x} = \langle x_1, \ldots, x_n\rangle$ by adding a random noise:

$$\vec{x}^{t+1} = \vec{x}^t + N(0, \vec{\sigma}),$$

$N(0, \vec{\sigma})$ is a vector of independent random Gaussian numbers with a mean of zero and standard deviations $\vec{\sigma}$. Such a mutation is used in evolution strategies (Bäck et al. 1991) and evolutionary programming (Fogel 1992). (One of the differences between these techniques lies in adjusting vector of standard deviations $\vec{\sigma}$.)

Other types of mutations include *non-uniform mutation*, where

$$x_k^{t+1} = \begin{cases} x_k^t + \triangle(t, right(k) - x_k) & \text{if } r \text{ is } 0 \\ x_k^t - \triangle(t, x_k - left(k)) & \text{if } r \text{ is } 1 \end{cases}$$

for $k = 1, \ldots, n$ (r is a random binary digit). The function $\triangle(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\triangle(t, y)$ being close to 0 increases as t increases (t is the generation number). This property causes this operator to search the space uniformly initially (when t is small), and very locally at later stages. In experiments reported in Michalewicz et al. (1994), the following function was used:

$$\triangle(t, y) = y \cdot r \cdot (1 - \frac{t}{T})^b,$$

where r is a random number from $[0..1]$, T is the maximal generation number, and b is a system parameter determining the degree of non–uniformity.

It is also possible to experiment with a *uniform mutation*, which changes a single component of the solution vector; e.g., if $\vec{x}^t = (x_1, \ldots, x_k, \ldots, x_q)$, then $\vec{x}^{t+1} = (x_1, \ldots, x'_k, \ldots, x_q)$, where x'_k is a random value (uniform probability distribution) from the domain of variable x_k. A special case of uniform mutation is *boundary mutation*, where x'_k is either left of right boundary of the domain of x_k.

There are a few interesting crossover operators. The first one, *uniform crossover* (known also as discrete crossover in evolution strategies; Schwefel 1981), works as follows. Two parents,

$$\vec{x}^1 = \langle x_1^1, \ldots, x_n^1\rangle \text{ and } \vec{x}^2 = \langle x_1^2, \ldots, x_n^2\rangle,$$

produce an offspring,

$$\vec{x} = \langle x_1^{q_1}, \ldots, x_n^{q_n}\rangle,$$

where $q_i = 1$ or $q_i = 2$ with equal probability for all $i = 1, \ldots, n$ (the second offspring is created by setting $q_i := 3 - q_i$). Of course, the uniform crossover is a generalization of *1-point*, *2-point*, and *multi-point crossovers*.

Another possibility is *arithmetical crossover*. Here two vectors, $\vec{x}^1$ and $\vec{x}^2$ produce two offspring, $\vec{x}^3$ and $\vec{x}^4$, which are a linear combination of their parents, i.e.,

$$\vec{x}^3 = a \cdot \vec{x}^1 + (1 - a) \cdot \vec{x}^2 \text{ and}$$
$$\vec{x}^4 = (1 - a) \cdot \vec{x}^1 + a \cdot \vec{x}^2.$$

Such a crossover was called also a *guaranteed average crossover* (Davis, 1989, when $a = 1/2$), and an *intermediate crossover* (Schwefel, 1981).

There is also an interesting *heuristic crossover* proposed by Wright (1991); it is a unique crossover for the following reasons: (1) it uses values of the objective function in determining the direction of the search, (2) it produces only one offspring, and (3) it may produce no offspring at all. The operator generates a single offspring $\vec{x}^3$ from two parents, $\vec{x}^1$ and $\vec{x}^2$ according to the following rule:

$$\vec{x}^3 = r \cdot (\vec{x}^2 - \vec{x}^1) + \vec{x}^2,$$

where r is a random number between 0 and 1, and the parent $\vec{x}^2$ is not worse than $\vec{x}^1$, i.e., $f(\vec{x}^2) \geq f(\vec{x}^1)$ for maximization problems and $f(\vec{x}^2) \leq f(\vec{x}^1)$ for minimization problems.

Mühlenbein and Voigt (1995) investigated the properties of a recombination operator, called *gene pool recombination* (default recombination mechanism in evolution strategies, see Schwefel, 1981), where the genes are randomly picked from the gene pool defined by the selected parents. An interesting aspect of this operator is that it allows so-called *orgies*: several parents in producing an offspring. Such a multi-parent crossover was also investigated also by Eiben et al. (1994) in the context of combinatorial optimization. Renders and Bersini (1994) experimented with *simplex crossover* for numerical optimization problems; this crossover involves computing centroid of group of parents and moving from the worst individual beyond the centroid point. Also, Glover's (1977) scatter search techniques propose the use of multiple parents.

3 Constraint-Handling Methods

During the last two years several methods were proposed for handling constraints by genetic algorithms for numerical optimization problems. Most of them are based on the concept of penalty functions, which penalize unfeasible solutions, i.e.,

$$eval(\overline{X}) = \begin{cases} f(\overline{X}), & if\ \overline{X} \in \mathcal{F} \\ f(\overline{X}) + penalty(\overline{X}), & if\ \overline{X} \in \mathcal{S} - \mathcal{F}, \end{cases}$$

where the set $\mathcal{S} \subseteq R^n$ defines the search space and the set $\mathcal{F} \subseteq \mathcal{S}$ defines a *feasible* search space, $penalty(\overline{X})$ is zero, if no violation occurs, and is positive, otherwise (in the rest of this section we assume minimization problems). In most methods a set of functions f_j $(1 \leq j \leq m)$ is used to construct the penalty; the function f_j measures the violation of the j-th constraint in the following way:

$$f_j(\overline{X}) = \begin{cases} \max\{0, g_j(\overline{X})\}, & if\ 1 \leq j \leq q \\ |h_j(\overline{X})|, & if\ q+1 \leq j \leq m. \end{cases}$$

However, these methods differ in many important details, how the penalty function is designed and applied to unfeasible solutions.

The most severe penalty is a death penalty; some method rejects unfeasible individuals. Such a method has been used by by many techniques, e.g., evolution strategies (Bäck et al. 1991) and simulated annealing.

Some methods use static penalties; for example, a method proposed by Homaifar, Lai, and Qi (1994) assumes that for every constraint we establish a family of intervals which determine appropriate penalty coefficient. It works by creating (for each constraint) several (ℓ) levels of violation; creating a penalty coefficient R_{ij} (for each level of violation and for each constraint; higher levels of violation require larger values of this coefficient; starting with a random population of individuals (feasible or unfeasible); and evolving the population, where the evaluation function used is

$$eval(\overline{X}) = f(\overline{X}) + \textstyle\sum_{j=1}^{m} R_{ij} f_j^2(\overline{X}).$$

Some other methods apply dynamic penalties. For example, in the method proposed by Joines and Houck (1994) individuals are evaluated (at the iteration t) by the following formula:

$$eval(\overline{X}) = f(\overline{X}) + (C \times t)^{\alpha} \textstyle\sum_{j=1}^{m} f_j^{\beta}(\overline{X}),$$

where C, α and β are constants. A reasonable choice for these parameters is $C = 0.5$, $\alpha = \beta = 2$.

Also, Michalewicz and Attia (1994) experimented with the procedure, where the algorithm maintains feasibility of all linear constraints using a set of closed operators, which convert a feasible solution (feasible in terms of linear constraints only) into another feasible solution. At every iteration the algorithm considers active constraints only, the pressure on unfeasible solutions is increased due to the decreasing values of temperature τ.

It is also possible to incorporate adaptive penalties; Bean and Hadj-Alouane (1992) and Smith and Tate (1993) experimented with them. Bean and Hadj-Alouane (1992) change the penalty coefficients on the basis of the number of feasible and infeasible individuals in the last k generations, whereas in the Smith and Tate (1993) approach penalty measure depends on the number of violated constraints, the best feasible objective function found, and the best objective function value found.

Yet another approach was proposed recently by Le Riche et al. (1995). The authors designed a (segregated) genetic algorithm which uses two values of penalty parameters (for each constraint) instead of one; these two values aim at achieving a balance between heavy and moderate penalties by maintaining two subpopulations of individuals. The population is split into two cooperating groups, where individuals in each group are evaluated using either one of the two penalty parameters.

Additional method was proposed by Schoenauer and Xanthakis (1993) The method requires a linear order of all constraints which are processed in turn. At iteration j, solutions that do not satisfy one of the 1st, 2nd, ..., or $(j-1)$-th constraint are eliminated from the population and the stop criterion is the satisfaction of the j-th constraint by the flip threshold percentage ϕ of the population. In the last step the algorithm optimizes the objective function, rejecting unfeasible individuals.

Another method was developed by Powell and Skolnick (1993). The method is a classical penalty method with one notable exception. Each individual is evaluated by the formula:

$$eval(\overline{X}) = f(\overline{X}) + r \textstyle\sum_{j=1}^{m} f_j(\overline{X}) + \lambda(t, \overline{X}),$$

where r is a constant; however, there is also a component $\lambda(t, \overline{X})$. This is an additional iteration dependent function which influences the evaluations of unfeasible solutions. The point is that the method distinguishes between feasible and unfeasible individuals by adopting an additional heuristic rule (suggested earlier by Richardson et al. 1989): for any feasible individual $\overline{X}$ and any unfeasible individual $\overline{Y}$: $eval(\overline{X}) < eval(\overline{Y})$, i.e., any feasible solution is better than any unfeasible one.

One of the most recent methods (Genocop III; see Michalewicz and Nazhiyath 1995) incorporates the original Genocop system (Michalewicz et al. 1994), but also extends it by maintaining two separate populations, where a development in one population influences evaluations of individuals in the other population. The first population consists of so-called search points from $\mathcal{S}$ which satisfy linear constraints of the problem (as in the original Genocop system). The feasibility (in the sense of linear constraints) of these points is maintained, as before, by specialized operators. The second population consists of so-called reference points from $\mathcal{F}$; these points are fully feasible, i.e., they satisfy *all* constraints. Reference points $\overline{R}$, being feasible, are evaluated directly by the objective function (i.e., $eval(\overline{R}) = f(\overline{R})$). On the other hand, unfeasible search points are "repaired" for evaluation.

For an experimental comparison of some of these methods on a few test cases, see Michalewicz (1995).

4 The Test Case

An interesting constrained numerical optimization test case emerged recently; the problem (Keane, 1994) is to maximize a function:

$$f(\vec{x}) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|,$$

where

$\prod_{i=1}^{n} x_i \geq 0.75,$ (1)
$\sum_{i=1}^{n} x_i \leq 7.5n,$ (2)
and $0 \leq x_i \leq 10$ for $1 \leq i \leq n$.

The problem has two constraints; the function f is nonlinear and its global maximum is unknown.

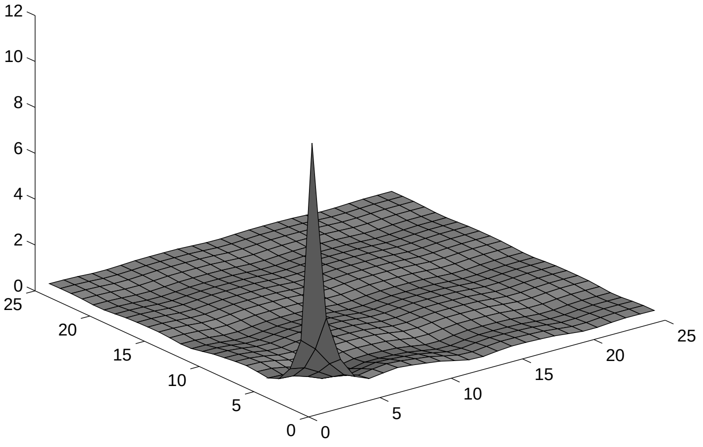

Figure 1: The graph of function f for $n = 2$

To illustrate some potential difficulties of solving this test case, a few graphs (for case of $n = 2$) are displayed in Figures 1 – 4. Figure 1 gives a general overview of the objective function f: the whole landscape seems to be relatively flat except for a sharp peek around the point $(0, 0)$. Figures 2 and 3 incorporate the active constraint: infeasible solutions were assigned a value of zero. In that case the objective function f takes values from the range $\langle 0, 1\rangle$; because of the scaling, the landscape is more visible. Figure 3 displays only the area of interest (i.e., the area of global optimum). Figure 4 presents a side view of the landscape along one of the axis. Again, the only feasible part of the landscape is displayed.

All methods (described in section 3) were tried on this test case with quite poor results. As Keane (1994) noted:

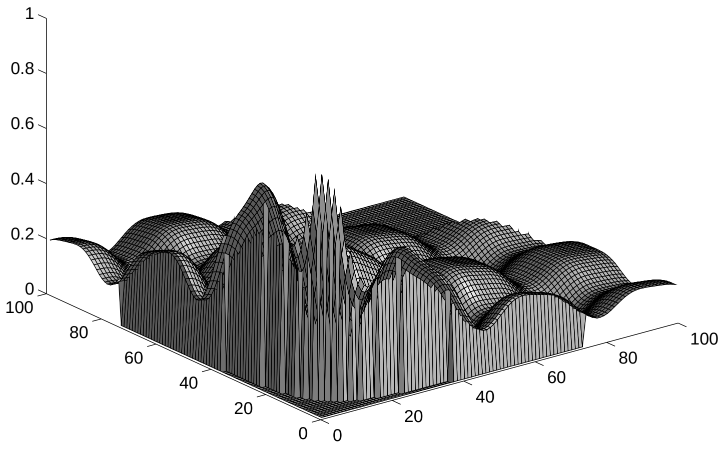

Figure 2: The graph of function f for $n = 2$. Infeasible solutions were assigned value zero

"I am currently using a parallel GA with 12bit binary encoding, crossover, inversion, mutation, niche forming and a modified Fiacco-McCormick constraint penalty function to tackle this. For $n =$ 20 I get values like 0.76 after 20,000 evaluations."

We obtained the best results from Genocop III; however, the results varied from run to run (between 0.75 and 0.80, case of $n = 20$).

5 A New System and the Geometrical Crossover

It seems that majority of constraint-handling methods have serious difficulties in returning a high quality solution for the above problem. It might be possible, however, to build a dedicated system for this particular test case, which would incorporate the problem specific knowledge. This knowledge can emerge from analysis of the objective function f and the constraints; thus we assume that (a) the domain for all variables is $\langle 0.1, 10.0\rangle$, (b) constraint (1) is active at the global optimum, and (c) the constraint (2) is not.

Now it is possible to develop an evolutionary system which would search just the surface defined by the first constraint:

$$\prod_{i=1}^{n} x_i = 0.75.$$

Thus the search space is greatly reduced; we consider only points which satisfy the above equation. Such a system would start from a population of feasible points, i.e., points, for which a product of all coordinates is equal to 0.75. It is relatively easy to develop such initialization routine:

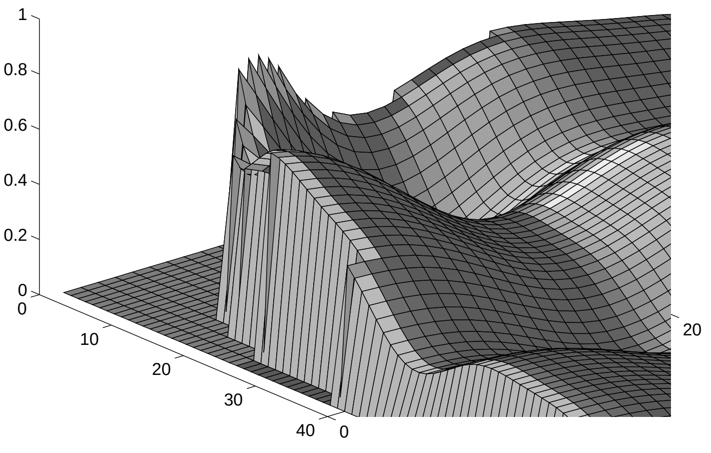

Figure 3: The graph of function f for $n = 2$. Infeasible solutions were assigned value zero; only the corner around the global optimum is shown

```
for $i = 1$ to $n$ do
begin
    if $i$ is even
        $x_{i-1}$ = random (0.1, 10)
        $x_i = 1/x_{i-1}$
end
if $n$ is odd
    $x_n = 0.75$
else
    for some random $i$
    $x_i = 0.75 \cdot x_i$
```

It is important to note, that the above initialization routine is not significant for the performance of any system; it just ensures that all points in the population are at the boundary between feasible and infeasible regions. Many other methods were tried with such initialized populations and gave poor results. Also, the value of the best individual in the population initialized in such a way is around 0.30.

The geometrical crossover takes two parents and produces a single offspring; for parents $\vec{x}^1$ and $\vec{x}^2$ the offspring $\vec{x}^3$ is

$$\vec{x}^3 = \langle \sqrt{x_1^1 \cdot x_1^2}, \ldots, \sqrt{x_n^1 \cdot x_n^2} \rangle. \quad (3)$$

Note, that the offspring $\vec{x}^3$ also lies on the boundary of feasible region:

$$\prod_{i=1}^n x_i^3 = 0.75.$$

Of course, it is an easy task to generalize the above geometrical crossover into

$$\vec{x}^3 = \langle (x_1^1)^\alpha \cdot (x_1^2)^{(1-\alpha)}, \ldots, (x_n^1)^\alpha \cdot (x_n^2)^{(1-\alpha)} \rangle, \quad (4)$$

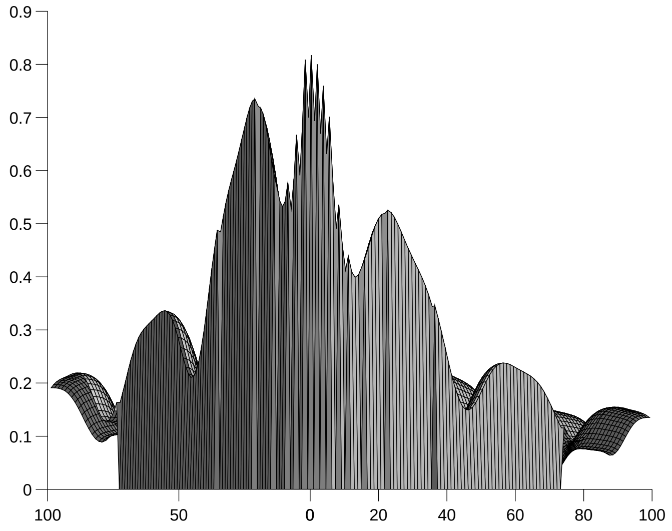

Figure 4: The vertical view of the function f for $n = 2$

for $0 \le \alpha \le 1$. Also it is possible to include several (say, k) parents:

$$\vec{x}^{k+1} = \langle (x_1^1)^{\alpha_1} \cdot (x_1^2)^{\alpha_2} \cdot \ldots \cdot (x_1^k)^{\alpha_k}, \ldots, (x_n^1)^{\alpha_1} \cdot (x_n^2)^{\alpha_2} \cdot \ldots \cdot (x_n^k)^{\alpha_k} \rangle, \quad (5)$$

where $\alpha_1 + \ldots + \alpha_k = 1$. However, in the experiments reported in this paper, we limited ourselves to the simplest geometrical crossover (3).

The task of designing a feasibility-preserving mutation is relatively simple; if the i-th component is selected for mutation, the following algorithm is executed:

```
determine random $j$, $1 \le j \le n$, $j \ne i$
select $q$ such that:
    $0.1 \le x_i \cdot q \le 10.0$ and
    $0.1 \le x_j / q \le 10.0$
$x_i = x_i \cdot q$
$x_j = x_j / q$
```

A simple evolutionary algorithm (200 lines of code!) with geometrical crossover and problem-specific mutation gave an outstanding results. For the case $n = 20$ the system reached the value of 0.80 in less than 4,000 generations (with population size of 30, probability of crossover $p_c = 1.0$, and probability of mutation $p_m = 0.06$) in all runs. The best value found (namely 0.803553) was better than the best values of any method discussed earlier, whereas the worst value found was 0.802964. Similarly, for $n = 50$, all results (in 30,000 generations) were better than 0.83 (with the best of 0.8331937);

$$\vec{x} = (6.28006029, 3.16155291, 3.15453815, 3.14085174, 3.12882447, 3.11211085, 3.10170507, 3.08703685, 3.07571769, 3.06122732, 3.05010581, 3.03667951, 3.02333045, 3.00721049, 2.99492717, 2.97988462, 2.96637058, 2.95589066, 2.94427204, 2.92796040,$$

0.40970641, 2.90670991, 0.46131119, 0.48193336, 0.46776962, 0.43887550, 0.45181099, 0.44652876, 0.43348753, 0.44577143, 0.42379948, 0.45858049, 0.42931050, 0.42928645, 0.42943302, 0.43294361, 0.42663351, 0.43437257, 0.42542559, 0.41594154, 0.43248957, 0.39134723, 0.42628688, 0.42774364, 0.41886297, 0.42107263, 0.41215360, 0.41809589, 0.41626775, 0.42316407),

It was interesting to note the importance of geometrical crossover. With fixed population size (kept constant at 30), the higher values of probability of crossover p_c, the better results of the system were observed. Similarly, the best mutation rates were relatively low ($p_m \approx 0.06$). We illustrate the average values (out of 10 runs) of the best values found for different probabilities of mutation (with fixed $p_c = 1.0$) in Figure 5.

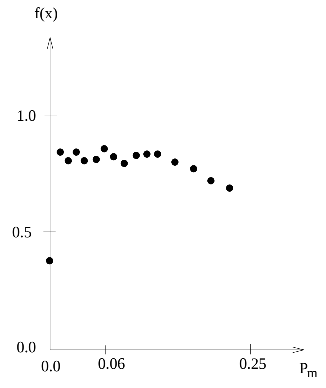

Figure 5: The performance of the system with $p_c = 1.0$ and a variable mutation rate p_m

Clearly, geometrical crossover can be applied only for problems where each variable takes nonnegative values only, so its use is quite restricted in comparison with other types of crossovers (e.g., arithmetical crossover). However, for many engineering problems, all problem variables are positive; moreover, it is always possible to replace variable $x_i \in \langle a_i, b_i \rangle$ which can take negative values (i.e., where $a_i < 0$) with a new variable $y_i = x_i - a_i$.

6 Further Experiments and Results

We compared arithmetical and geometrical crossovers on several test cases. The preliminary experiments indicate that geometrical crossover outperforms arithmetical crossover on majority of unconstrained test problems.

For example, one of the selected test cases was to

$$\begin{aligned} \text{minimize } f(\overline{X}) = {} & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + \\ & 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + \\ & 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + \\ & 19.8(x_2 - 1)(x_4 - 1), \end{aligned}$$

subject to:

$$0.0 \le x_i \le 10.0, \; i = 1, 2, 3, 4.$$

The global solution is $\overline{X}^* = (1, 1, 1, 1)$, and $f(\overline{X}^*) = 0$.

We experimented with two simple systems (A and B); both of them use two operators, and one of these operators (in both systems) was a non-uniform mutation (described in section 2). The only difference between systems A and B was, that the former system used arithmetical crossover, i.e., offspring $\vec{z}$ of parents $\vec{x}$ and $\vec{y}$ was defined as

$z_i = a \cdot x_i + (1 - a) \cdot y_i$ for all $1 \le i \le n$ and random $0 \le a \le 1$,

whereas the system B used geometrical crossover:

$z_i = x_i^a \cdot y_i^{1-a}$ for all $1 \le i \le n$ and random $0 \le a \le 1$.

Figure 6 illustrates the difference in the performance of systems A and B.

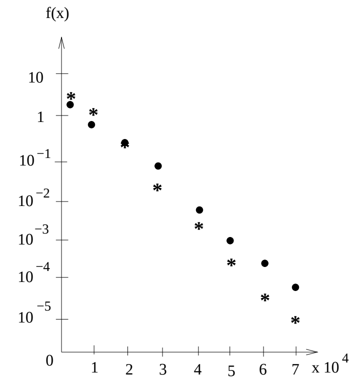

Figure 6: The performance (number of generations in 10,000s versus the average value of the objective function over 20 runs) of the systems A (marked by '•') and B (marked by '*') on one test case

7 Conclusions

It seems that the experiments described in the previous sections can be generalized in the following sense. Initial experiments (using any evolutionary method) may indicate active constraints for a given problem. Then it might be possible to build a specialized system with operators to search a boundary between feasible and infeasible parts of the search space. This was precisely the case of the problem described in section 4; it might be the case for many other optimization problems.

Let us consider, for example, the following optimization problem:

$$\text{maximize } f(\vec{x}) = (\sqrt{n})^n \cdot \prod_{i=1}^n x_i$$

where

$$\sum_{i=1}^{n} x_i^2 \leq 1 \qquad (6)$$
and $0 \leq x_i \leq 1$, for $1 \leq i \leq n$.

Clearly, the objective function reaches its global optimum at

$$\vec{x}^* = \langle 1/\sqrt{n}, \ldots, 1/\sqrt{n} \rangle,$$

and $f(\vec{x}^*) = 1$.

Again, it is relatively easy to initialize the population by boundary points, i.e., points, satisfying

$$\sum_{i=1}^{n} x_i^2 = 1.$$

Then, the sphere crossover produces an offspring $\vec{x}^3$ from two parents $\vec{x}^1$ and $\vec{x}^2$ in the following way:

$$x_i^3 = \sqrt{((x_i^1)^2 + (x_i^2)^2)/2}.$$

As for geometrical crossover, it is easy to generalize sphere crossover:

$$x_i^3 = \sqrt{\alpha (x_i^1)^2 + (1-\alpha)(x_i^2)^2},$$

for $0 \leq \alpha \leq 1$, as well as for multiple parents:

$$x_i^{k+1} = \sqrt{\alpha_1 (x_i^1)^2 + \ldots + \alpha_k (x_i^k)^2},$$

for $\alpha_1 + \ldots + \alpha_k = 1$.

Clearly, $\vec{x}^3$ lies also on the sphere defined by (6). Similarly, a problem-specific mutation (for a parent vector $\vec{x}$) can select two indices, $i \neq j$ and a random number p from the range $\langle 0, 1 \rangle$, and assign:

$$x_i^{t+1} = p \cdot x_i^t, \text{ and}$$
$$x_j^{t+1} = q \cdot x_j^t,$$

where $q = \sqrt{(\frac{x_i}{x_j})^2 (1-p^2) + 1}$. Such a simple evolutionary system finds the global optimum easily (e.g., the case of $n = 20$ requires 10,000 generations with population size equal to 30, $p_c = 1.0$, and $p_m = 0.06$).

It seems that problem-specific operators which search the boundary of a feasible region can enhance the search for a global optimum in a significant way. At least for some numerical optimization problems, it is an interesting option to explore. Resulting evolutionary systems are very easy to construct (less than 200 lines of code) and use. As indicated in section 6, such operators can be useful also for problems without constraints. We plan further study on properties of various crossovers and boundary-search operators for numerical optimization problems.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant IRI-9322400.

References

Bäck, T., F. Hoffmeister and H.-P. Schwefel (1991). A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, Los Altos, CA, Morgan Kaufmann Publishers, 2–9.

Bean, J.C. and A.B. Hadj-Alouane (1992). A Dual Genetic Algorithm for Bounded Integer Programs. Department of Industrial and Operations Engineering, The University of Michigan, TR 92-53.

Davis, L. (1989). Adapting Operator Probabilities in Genetic Algorithms. In *Proceeding of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp.61–69.

Eiben, A.E., P.-E. Raue, and Zs. Ruttkay (1994). Genetic Algorithms with Multi-parent Recombination. In *Proceedings of the Third International Conference on Parallel Problem Solving from Nature (PPSN)*, Springer-Verlag, New York, pp.78–87.

Fogel, D.B. (1992). Evolving Artificial Intelligence. PhD Thesis, University of California, San Diego.

Glover, F. (1977). Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences*, Vol.8, No.1, pp.156–166.

Glover, F. and G. Kochenberger (1995). Critical Event Tabu Search for Multidimensional Knapsack Problems. In *Proceedings of the International Conference on Metaheuristics for Optimization*, Kluwer Publishing, pp.113–133.

Hadj-Alouane, A.B. and J.C. Bean (1992). A Genetic Algorithm for the Multiple-Choice Integer Program. Department of Industrial and Operations Engineering, The University of Michigan, TR 92-50.

Homaifar, A., S. H.-Y. Lai and X. Qi (1994). Constrained Optimization via Genetic Algorithms. *Simulation*, **62**: 242–254.

Joines, J.A. and C.R. Houck (1994). On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems With GAs. In *Proceedings of the Evolutionary Computation Conference—Poster Sessions*, part of the IEEE World Congress on Computational Intelligence, Orlando, 26–29 June 1994, 579–584.

Keane, A., (1994). Genetic Algorithms Digest, Thursday, May 19, 1994, Volume 8, Issue 16.

Kelly, J.P., B.L. Golden, and A.A. Assad (1993). Large Scale Controlled Rounding Using Tabu Search with Strategic Oscillation. *Annals of Operations Research*, Vol.41, pp.69–84.

Le Riche, R., C. Knopf-Lenoir, and R.T. Haftka (1995). A Segregated Genetic Algorithm for Constrained Optimization in Structural Mechanics. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, Los Altos, CA, Morgan Kaufmann Publishers, pp.558–565.

Michalewicz, Z., (1995). Genetic Algorithms, Numerical Optimization, and Constraints. In *Proceeding of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp.151–158.

Michalewicz, Z. and N. Attia (1994). In Evolutionary Optimization of Constrained Problems. *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, eds. A.V. Sebald and L.J. Fogel, River Edge, NJ, World Scientific Publishing, 98–108.

Michalewicz, Z., T.D. Logan and S. Swaminathan (1994). Evolutionary Operators for Continuous Convex Parameter Spaces. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, eds. A.V. Sebald and L.J. Fogel, River Edge, NJ, World Scientific Publishing, 84–97.

Michalewicz, Z. and G. Nazhiyath, G. (1995). Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints. In *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, Perth, 29 November – 1 December 1995.

Mühlenbein, H. and H.-M. Voigt (1995). Gene Poool Recombination for the Breeder Genetic Algorithm. In *Proceedings of the Metaheuristics International Conference*, Breckenridge, Colorado, July 22–26, pp.19–25.

Powell, D. and M.M. Skolnick (1993). Using Genetic Algorithms in Engineering Design Optimization with Nonlinear Constraints. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Los Altos, CA, Morgan Kaufmann Publishers, 424–430.

Renders, J.-M., and H. Bersini (1994). Hybridizing Genetic Algorithms with Hill-climbing Methods for Global Optimization: Two Possible Ways. In *Proceedings of the First IEEE ICEC*, pp.312–317.

Richardson, J.T., M.R. Palmer, G. Liepins and M. Hilliard (1989). Some Guidelines for Genetic Algorithms with Penalty Functions. In *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, Morgan Kaufmann Publishers, 191–197.

Schoenauer, M., and S. Xanthakis (1993). Constrained GA Optimization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Los Altos, CA, Morgan Kaufmann Publishers, 573–580.

Schwefel, H.-P. (1981). *Numerical Optimization for Computer Models*. Chichester, UK, Wiley.

Smith, A. and D. Tate (1993). Genetic Optimization Using A Penalty Function. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Los Altos, CA, Morgan Kaufmann Publishers, pp.499–503.

Wright, A.H. (1991). Genetic Algorithms for Real Parameter Optimization. In *Foundations of Genetic Algorithms*, ed. G. Rawlins, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, Los Altos, CA, Morgan Kaufmann Publishers, 205–218.