A Decoder-based Evolutionary Algorithm for Constrained Parameter Optimization Problems

Sławomir Kozieł[1] and Zbigniew Michalewicz[2]

[1]Department of Electronics, Telecommunication and Informatics
Technical University of Gdańsk
ul. Narutowicza 11/12, 80-952 Gdańsk, Poland
koziel@ue.eti.pg.gda.pl

[2]Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA
zbyszek@uncc.edu

Abstract. Several methods have been proposed for handling nonlinear constraints by evolutionary algorithms for numerical optimization problems; a survey paper [7] provides an overview of various techniques and some experimental results, as well as proposes a set of eleven test problems. Recently a new, decoder-based approach for solving constrained numerical optimization problems was proposed [2,3]. The proposed method defines a homomorphous mapping between n-dimensional cube and a feasible search space. In [3] we have demonstrated the power of this new approach on several test cases. However, it is possible to enhance the performance of the system even further by introducing additional concepts of (1) nonlinear mappings with an adaptive parameter, and (2) adaptive location of the reference point of the mapping.

1 Introduction

The nonlinear parameter optimization problem is defined as

$$\text{Find } x \in \mathcal{S} \subset R^n \text{ such that } \begin{cases} f(\boldsymbol{x}) & = \min\{f(\boldsymbol{y});\ \boldsymbol{y} \in \mathcal{S}\}, \quad (1) \\ g_j(\boldsymbol{x}) & \leq 0, \text{for } \ j = 1, \ldots, q, \quad (2) \end{cases}$$

where f and g_i are real-valued functions on $\mathcal{S}$; $\mathcal{S}$ is a search space defined as a Cartesian product of domains of variables x_i's ($1 \leq i \leq n$). The set of feasible points (i.e., points satisfying the constraints (2)) is denoted $\mathcal{F}$.[1]

Several methods have been proposed for handling nonlinear constraints by evolutionary algorithms for numerical optimization problems. The recent survey paper [7] classifies them into four categories (preservation of feasibility, penalty functions, searching for feasibility, and other hybrids). However, there is one central issue that all these methods have to address, which is, whether to allow processing of infeasible solutions? This is the most important issue to resolve. Many constraint-handling methods process infeasible solutions (e.g., various penalty-based methods), on the other hand, many other techniques process only feasible solutions (e.g., methods based on feasibility-preserving operators).

[1] Note, that we do not consider equality constraints; if necessary, an equality $h(\boldsymbol{x}) = 0$ can be replaced by a pair of inequalities $-\epsilon \leq h(\boldsymbol{x}) \leq \epsilon$ for some small $\epsilon > 0$.

In general, restricting the search to the feasible region seems a very elegant way to treat constrained problems. For example, in [5], the algorithm maintains feasibility of linear constraints using a set of closed operators which convert a feasible solution into another feasible solution. Similar approach for the nonlinear transportation problem is described in [4], where specialized operators transform a feasible solution matrix (or matrices) into another feasible solution. This is also the case for many evolutionary systems developed for the traveling salesman problem [4], where specialized operators maintain feasibility of permutations, as well as for many other combinatorial optimization problems.

However, for numerical optimization problems only special cases allowed the use of either specialized operators which preserve feasibility of solutions or repair algorithms, which attempt to convert an infeasible solution into feasible one. For example, a possible use of a repair algorithm was described in [6], but in that approach it was necessary to maintain two separate populations with feasible and infeasible solutions: a set of reference feasible points was used to repair infeasible points. Consequently, most evolutionary techniques for numerical optimization problems with constraints are based on penalties. However, highly nonlinear constraints still present difficulties for evolutionary algorithms, as penalty parameters or strategies are then difficult to adjust.

In this paper we investigate some properties of a recently proposed approach [3] for solving constrained numerical optimization problems which is based on a homomorphous mapping between n-dimensional cube $[-1, 1]^n$ and a feasible search space. This approach constitutes an example of decoder-based approach[2] where the mapping allows to process feasible solutions only. The first results [3] indicated a huge potential of this approach; the proposed method does not require any additional parameters, does not require evaluation of infeasible solutions, does not require any specialized operators to maintain feasibility—or to search the boundary of the feasible region [9], [8]. Moreover, any standard evolutionary algorithm (e.g., binary-coded genetic algorithm or evolution strategy) can be used in connection with the mapping. On the top of that, the method *guarantees* a feasible solution, which is not always the case for other methods.

The paper is organized as follows. The following section presents the new method, whereas section 3 discusses the research issues of this paper. Section 4 presents some experimental results and section 5 concludes the paper.

2 The homomorphous mapping

The idea behind this technique is to develop a homomorphous mapping φ, which transforms the n-dimensional cube $[-1, 1]^n$ into the feasible region $\mathcal{F}$ of the problem [3]. Note, that $\mathcal{F}$ need not be convex; it might be concave or even can consist of disjoint (non-convex) regions.

The search space $\mathcal{S}$ is defined as a Cartesian product of domains of variables of the problem, $l(i) \leq x_i \leq u(i)$, for $1 \leq i \leq n$, whereas a feasible part $\mathcal{F}$ of

[2] Actually, this is the first approach of this type; until recently, mappings (or decoders) were applied only to discrete optimization problems.

the search space is defined by problem specific constraints: inequalities (2) from the previous section. Assume, a solution $\boldsymbol{r_0}$ is feasible (i.e., $\boldsymbol{r_0} \in \mathcal{F}$). Then any boundary point $\boldsymbol{s}$ of the search space $\mathcal{S}$ defines a line segment L between $\boldsymbol{r_0}$ and $\boldsymbol{s}$ (figure 1 illustrates the case). Note that such a line segment may intersect a boundary of the feasible search space $\mathcal{F}$ in more than just one point.

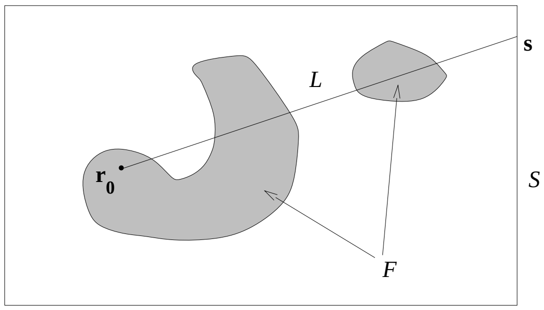

Fig. 1. A line segment in a non-convex space $\mathcal{F}$ (two-dimensional case)

Let us define an additional one-to-one mapping g between the cube $[-1,1]^n$ and the search space $\mathcal{S}$. Then the mapping $g : [-1,1]^n \to S$ can be defined as

$$g(\boldsymbol{y}) = \boldsymbol{x}, \text{ where } x_i = y_i \frac{u(i)-l(i)}{2} + \frac{u(i)+l(i)}{2}, \text{ for } i = 1, \ldots, n.$$

Indeed, for $y_i = -1$ the corresponding $x_i = l(i)$, and for $y_i = 1$, $x_i = u(i)$.

A line segment L between any reference point $\boldsymbol{r_0} \in \mathcal{F}$ and a point $\boldsymbol{s}$ at the boundary of the search space $\mathcal{S}$, is defined as

$$L(\boldsymbol{r_0}, \boldsymbol{s}) = \boldsymbol{r_0} + t \cdot (\boldsymbol{s} - \boldsymbol{r_0}), \text{ for } 0 \le t \le 1.$$

Clearly, if the feasible search space $\mathcal{F}$ is convex,[3] then the above line segment intersects the boundary of $\mathcal{F}$ in precisely one point, for some $t_0 \in [0,1]$. Consequently, for convex feasible search spaces $\mathcal{F}$, it is possible to establish a one-to-one mapping $\varphi : [-1,1]^n \to \mathcal{F}$ as follows:

$$\varphi(\boldsymbol{y}) = \begin{cases} \boldsymbol{r_0} + y_{max} \cdot t_0 \cdot (g(\boldsymbol{y}/y_{max}) - \boldsymbol{r_0}) & \text{if } \boldsymbol{y} \neq \boldsymbol{0} \\ \boldsymbol{r_0} & \text{if } \boldsymbol{y} = \boldsymbol{0} \end{cases}$$

where $r_0 \in \mathcal{F}$ is a reference point, and $y_{max} = \max_{i=1}^{n} |y_i|$. Figure 2 illustrates the transformation φ.

On the other hand, if the feasible search space $\mathcal{F}$ is not convex, then the line segment L may intersect the boundary of $\mathcal{F}$ in many points (see figure 1).

[3] Note, that the convexity of the feasible search space $\mathcal{F}$ is not necessary; it is sufficient if we assume the existence of the reference point $\boldsymbol{r_0}$, such that every line segment originating in $\boldsymbol{r_0}$ intersects the boundary of $\mathcal{F}$ in precisely one point. This requirement is satisfied, of course, for any convex set $\mathcal{F}$.

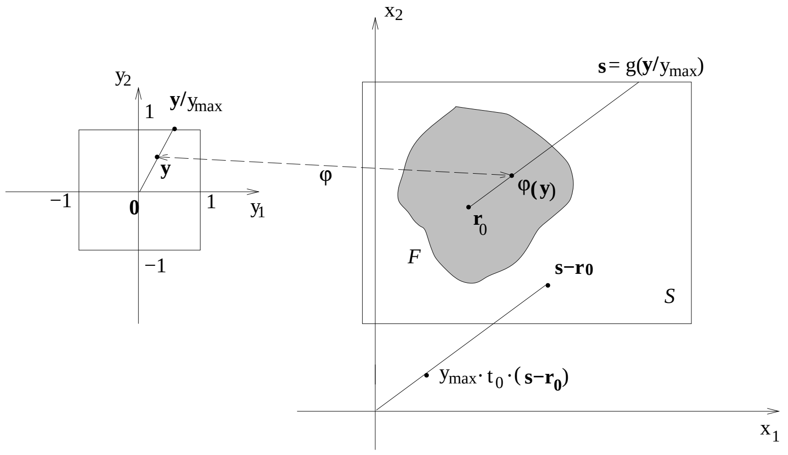

Fig. 2. A mapping φ from the cube $[-1,1]^n$ into the space $\mathcal{F}$ (two-dimensional case), with particular steps of the transformation

Let us consider an arbitrary point $\boldsymbol{y} \in [-1,1]^n$ and a reference point $\boldsymbol{r_0} \in \mathcal{F}$. A line segment L between the reference point $\boldsymbol{r_0}$ and the point $\boldsymbol{s} = g(\boldsymbol{y}/y_{max})$ at the boundary of the search space $\mathcal{S}$, is defined as before, however, instead of a single interval of feasibility $[0, t_0]$ for convex search spaces, we may have several intervals of feasibility: $[t_1, t_2], \ldots, [t_{2k-1}, t_{2k}]$. Assume there are altogether k subintervals of feasibility for a such line segment and t_i's mark their limits. Clearly, $t_1 = 0$, $t_i < t_{i+1}$ for $i = 1, \ldots, 2k-1$, and $t_{2k} \leq 1$. Thus, it is necessary to introduce an additional mapping γ, which transforms interval $[0,1]$ into sum of intervals $[t_{2i-1}, t_{2i}]$. However, we define such a mapping γ between $(0,1]$ and the sum of intervals $(t_{2i-1}, t_{2i}]$:

$$\gamma : (0,1] \rightarrow \bigcup_{i=1}^{k} (t_{2i-1}, t_{2i}].$$

Note, that due to this change, one boundary point (from each interval $1 \leq i \leq k$) is lost. However, this is not a serious problem, since we can approach the lost points with arbitrary precision. On the other hand, the benefits are clear: it is possible to "glue together" intervals which are open at one end and closed at another; additionally, such a mapping is one-to-one. There are many possibilities for defining such a mapping; we have used the following. First, let us define a reverse mapping δ:

$$\delta : \bigcup_{i=1}^{k} (t_{2i-1}, t_{2i}] \rightarrow (0,1] \text{ as follows: } \delta(t) = (t - t_{2i-1} + \textstyle\sum_{j=1}^{i-1} d_j)/d,$$

where $d_j = t_{2j} - t_{2j-1}$, $d = \sum_{j=1}^{k} d_j$, and $t_{2i-1} < t \leq t_{2i}$. Clearly, the mapping γ is reverse of δ:

$$\gamma(a) = t_{2j-1} + d_j \frac{a - \delta(t_{2j-1})}{\delta(t_{2j}) - \delta(t_{2j-1})},$$

where j is the smallest index such that $a \le \delta(t_{2j})$.

Now we are ready to define the mapping φ, which is the essence of the method of transformation of constrained optimization problem to the unconstrained one for every feasible set $\mathcal{F}$. The mapping φ is given by the following formula:

$$\varphi(\boldsymbol{y}) = \begin{cases} \boldsymbol{r_0} + t_0 \cdot (g(\boldsymbol{y}/y_{max}) - \boldsymbol{r_0}) & \text{if } \boldsymbol{y} \neq \boldsymbol{0}, \\ \boldsymbol{r_0} & \text{if } \boldsymbol{y} = \boldsymbol{0}, \end{cases}$$

where $r_0 \in \mathcal{F}$ is a reference point, $y_{max} = \max_{i=1}^{n} |y_i|$, and $t_0 = \gamma(|y_{max}|)$.

Finally, it is necessary to consider a method of finding points of intersections t_i. Let us consider any boundary point $\boldsymbol{s}$ of $\mathcal{S}$ and the line segment L determined by this point and a reference point $\boldsymbol{r_0} \in \mathcal{F}$. There are m constraints $g_i(\boldsymbol{x}) \le 0$ and each of them can be represented as a function β_i of one independent variable t (for fixed reference point $\boldsymbol{r_0} \in \mathcal{F}$ and the boundary point $\boldsymbol{s}$ of $\mathcal{S}$):

$$\beta_i(t) = g_i(L(\boldsymbol{r_0}, \boldsymbol{s})) = g_i(\boldsymbol{r_0} + t \cdot (\boldsymbol{s} - \boldsymbol{r_0})), \text{ for } 0 \le t \le 1 \text{ and } i = 1, \ldots, m.$$

As stated earlier, the feasible region need not be convex, so it may have more than one point of intersection of the segment L and the boundary of the set $\mathcal{F}$. Therefore, let us partition the interval $[0,1]$ into v subintervals $[v_{j-1}, v_j]$, where $v_j - v_{j-1} = 1/v$ $(1 \le j \le v)$, so that equations $\beta_i(t) = 0$ have at most one solution in every subinterval.[4] In that case the points of intersection can be determined by a binary search.

Once the intersection points between a line segment L and all constraints $g_i(\boldsymbol{x}) \le 0$ are known, it is quite easy to determine intersection points between this line segment L and the boundary of the feasible set $\mathcal{F}$.

3 Adaptation issues

In [3] we reported on experimental results of the system based on the mapping described in the previous section. The system was based on Gray coding with 25 bits per variable, and incorporated proportional selection (no elitism), function scaling, and standard operators (flip mutation and 1-point crossover). All parameters were fixed: pop_size = 70, generation gap = 100%, and $p_c = 0.9$. The only non-standard feature incorporated into the system (to increase fine tuning capabilities of the system [1]) was a variable probability of mutation.[5] In all experiments, $p_m(0) = 0.005$, $r = 4$, and $p_m(T) = 0.00005$.

The system provided very good results [3], which were better than for any other constraint handling method reported in literature. Yet there were some additional possibilities for a further improvement and unresolved issues which we address in this paper.

First of all, it is important to investigate the role of the reference point $\boldsymbol{r_0}$. Note that instead of keeping this point static during the evolutionary process,

[4] Density of the partition is determined by parameter v, which is adjusted experimentally (in all experiments reported in section 4, $v = 20$).

[5] $p_m(t) = p_m(0) - (p_m(0) - p_m(T)) \cdot (t/T)^r$, where t and T are the current and maximal generation numbers, respectively.

it can change its location. In particular, it can "follow" the best solution found so far. In that way, the reference point can "adapt" itself to the current state of the search. One of the aims of this paper is to compare the proposed method with static versus dynamic reference point. In the latter case, the quotient of the total number of generations and the number of changes of the reference point during the run, gives the number of generations between each change; the new reference point is the best individual of the current generation.

Note that a change of the reference point $\boldsymbol{r_0}$ changes the phenotypes of the genotypes in the population. Thus it might be worthwhile to consider an additional option: after each change of the reference point, all genotypes in the population are modified accordingly to yield the same phenotype as before the change. For example, if a genotype (0101101...0111) corresponded to the phenotype $(-2.46610039, 1.09535518)$ just before the change of the reference point $\boldsymbol{r_0}$, then, after the change, the genotype is changed in such a way, that its phenotype is still $(-2.46610039, 1.09535518)$ for a new reference point.

Also, in the proposed method it is important to investigate a non-uniform distribution of values of vectors $\boldsymbol{y} \in [-1, 1]^n$; this can be achieved, for example, by introducing an additional mapping $\omega : [-1, 1]^n \to [-1, 1]^n$:

$$\omega(\boldsymbol{y}) = \boldsymbol{y}', \text{ where } y'_i = a \cdot y_i,$$

where a is a parameter of the mapping, and $0 < a \leq 1/y_{max}$. Such exploration of non-uniform distribution of $\boldsymbol{y}$ provides additional possibilities for tuning the search:

- an increase in value of parameter a would result in selecting new vectors $\boldsymbol{y}'$ closer to a boundary of the feasible part of the search space. Thus, it is possible to use this approach to search the boundary of the feasible search space (e.g., instead of using specialized boundary operators [9]).[6]
- a decrease in value of parameter a would result in selecting new vectors $\boldsymbol{y}'$ closer to zero (i.e., the corresponding new search point would be closer to the reference point). This may work very well with the mechanism of adaptive change of the reference point: the system explores points closer to the reference point which, in turn, "follows" the best solution found so far.

Of course, there are many mappings which introduce a non-uniform distribution of values of vectors $\boldsymbol{y}$; in this paper we experimented with the following mapping:

$$\omega(\boldsymbol{y}) = \boldsymbol{y}', \text{ where } y'_i = y_i \cdot y_{max}^{k-1},$$

where $k > 1$ is a parameter. Clearly, larger k would move new search points closer to the reference point (this corresponds to a decrease in value of parameter a, of course). However, such a mapping concentrates the search around the reference point, hence is not helpful in cases where the optimum solution is located on the boundary of the feasible part of the search space. Thus an additional option

[6] Note, however, that in general (e.g., non-convex feasible search spaces) only a part of the boundary will be explored.

(direction of change) was considered: if a vector $\boldsymbol{c}$ represents the normalized direction vector of the last change of the reference point, then the constant parameter k is replaced by a variable k' calculated (for every vector $\boldsymbol{y}$) as follows:

$$k' = \begin{cases} 1 + (k-1) \cdot (1 - \cos^2(\boldsymbol{c}, \boldsymbol{y})) & \text{if } \cos(\boldsymbol{c}, \boldsymbol{y}) > 0 \\ k & \text{if } \cos(\boldsymbol{c}, \boldsymbol{y}) \leq 0, \end{cases}$$

Note that if the angle between $\boldsymbol{c}$ and $\boldsymbol{y}$ is close to zero, then $\cos(\boldsymbol{c}, \boldsymbol{y})$ is close to one, and, consequently, the value of parameter k is close to one.

4 Experimental results

Ten versions of an evolutionary system were considered (see Table 1).

Version number	Number of changes of r_0 during run	Change of genotype	Value of k	Option: direction of change
1	0	N/A	1.0	N/A
2	3	N	1.0	N
3	3	N	3.0	N
4	3	N	3.0	Y
5	20	N	1.0	N
6	20	N	3.0	N
7	20	N	3.0	Y
8	3	Y	1.0	N
9	20	Y	3.0	N
10	20	Y	3.0	Y

Table 1. Ten versions of the evolutionary system. For each version we report the number of changes of the reference point during the run (0 corresponds to the case where is no change, thus some other options are not applicable N/A), whether an option of re-coding the genotype was used (Yes or No), the value of scaling parameter k, and whether the option (direction of change) was used (Y) or not (N).

The experiments were made for four functions: $G6$, $G7$, $G9$, and $G10$ from [7].[7] All results are given in Tables 2–3. For each function 10 runs were performed; the tables report the best solution found in all runs, the average value, and the worst one. For $G6$, all runs had 500 generations, whereas all runs for remaining functions had 5,000 generations.

It was interesting to see that:

[7] These functions have 2, 10, 7, and 8 variables, respectively, which a number (between 2 and 8) of (mainly) nonlinear constraints. Most constraints are active at the optimum.

	*G*6			*G*7		
Version number	Minimum value	Average value	Maximum value	Minimum value	Average value	Maximum value
1	−6961.806423	−6403.744816	−5658.854943	26.156504	34.014132	62.015826
2	−6961.813810	−6949.220321	−6880.366641	24.823462	29.702066	37.593063
3	−6961.813769	−6961.616254	−6959.862901	25.667881	31.635635	41.275908
4	−6961.811700	−6961.119165	−6955.609490	24.456143	27.501678	34.224130
5	−6961.813810	−6959.199162	−6936.007217	24.923346	29.034924	36.600579
6	−6962.041796	−6954.089593	−6887.142350	24.493854	27.846996	37.850277
7	−6961.813805	−6961.814303	−6961.813735	25.604691	27.765957	33.025607
8	−6961.813754	−6926.097556	−6605.883742	24.449495	27.451748	34.651248
9	−6961.689228	−6960.275484	−6953.448863	24.987889	27.657595	31.823738
10	−6961.813247	−6960.794588	−6958.289256	26.119342	27.744277	29.447646

Table 2. Results for *G*6 and *G*7. These are minimization problems and the optimum values of these functions are −6961.81381 and 24.3062091, respectively.

	*G*9			*G*10		
Version number	Minimum value	Average value	Maximum value	Minimum value	Average value	Maximum value
1	680.630511	680.660238	680.729387	7160.262971	8592.392352	11511.072437
2	680.630542	680.636662	680.647153	7059.228161	7464.926353	8229.071491
3	680.630181	680.636573	680.664618	7086.430306	7591.768786	9225.975846
4	680.630392	680.637875	680.661187	7197.628211	7819.787329	8827.143414
5	680.631795	680.633758	680.636254	7058.405010	7492.697550	8995.685583
6	680.631554	680.644804	680.703939	7081.945176	7888.418244	9656.438311
7	680.630826	680.634730	680.643466	7078.900133	7607.775554	8695.147494
8	680.631036	680.677782	680.965273	7089.686242	7994.728714	9734.441891
9	680.632734	680.635818	680.639192	7230.799808	7695.850259	8813.595674
10	680.630492	680.638832	680.668193	7063.878216	7597.675949	8637.628629

Table 3. Results for *G*9 and *G*10. These are minimization problems and the optimum values of these functions are 680.6300573 and 7049.330923, respectively.

– for the test case *G*6 the best results were obtained for versions 3, 4, 7, 9, and 10. In all these five versions, the value of the parameter k was set to 3.0; it seems that this factor had a major influence on the quality of the results. Note also, that these five versions include all three versions where the option of "changing directions" was used (versions 4, 7, and 10). Also, the difference between versions 6 and 7 was only in the use of the above option: note the average scores of these two versions. In this case this option proved its usefulness. Similarly, the only difference between versions 3 and 6 was in the number of changes of the reference point made during the run. For this particular test case, a higher value of this parameter was better in combination with the option of changing the genotype, and a lower value

was better without this option (see versions 9 and 10 for the former case, and versions 3 and 4, for the latter).
- it is difficult to evaluate the performance of these versions for the test case $G7$. Note, that a few versions reported good *best* results (out of ten runs), however, the *average* values were less impressive. It seems that slightly better results were obtained for versions 4, 6, 8, and 9, but no interesting patterns emerged. For two of these versions, the number of changes of the reference point during the run was 3, whereas for the other two, it was 20. Two of these versions used the option of changing the genotype, and two others did not. Three versions used a higher value of parameter $k = 3$ and one version used $k = 1$. One version used the "direction of change" option.
- for the test case $G9$ all versions gave very good results, so it was possible to judge the performance of these version only on the basis of precision. The best versions (i.e., versions whose the best result was smaller than 680.631 and the average result smaller than 680.64) were versions 2, 3, 4, 7, and 10. This is consistent with our observations made in connection with the test case $G6$, where almost the same subset was selected.
- for the (hardest) test case $G10$, the best versions were selected on the following basis: the best solution was smaller than 7100 and the average solution was smaller than 7700. Only 5 versions satisfied these criterion; these were versions 2, 3, 5, 7, and 10. Again, as for test cases $G6$ and $G9$, versions 3, 7, and 10 are among the best.

It seems that the three versions which gave the best performance overall are versions 3, 7, and 10. Judging from the characteristics of these versions, we may conclude that generally:

– the higher value of parapeter k ($k = 3$) gives better results,

– small number of changes of the reference point does not require changes of genotypes nor the "direction of change" option,

– if the number of changes of the reference point is larger, it is not important whether genotypes in the population are adjusted (for each change) or not. However, it is important to keep "direction of change" option.

5 Conclusions

The results of these preliminary experiments are not, of course, conclusive. It is necessary to conduct a larger number of runs for a larger set of test cases (e.g., $G1$–$G11$, see [7]) to understand better the interactions among various options available. It is also necessary to extend this preliminary study for a larger set of parameters values (different values of k, different values of a number of changes of the reference point, etc). Further, a connection between the type of the problem (size of the feasible search space, number of active constraints at the optimum, modality of the objective function) and the characteristics of various versions discussed earlier, must be studied carefully.

Results of some further experiments performed for problems $G2$ and $G3$ suggest that the change of the reference point is not always beneficial. For these functions, version #1 (no change of reference point) gave the best results among all versions. It seems that a change of the reference point is beneficial only for some types of functions: thus such a change should be controlled by a feedback from the search process. A preliminary version of a new system with adaptive change of the reference point gave the best performance on all mentioned problems (from $G2$ to $G10$), making appropriate number of changes (e.g., zero changes for $G2$ and $G3$) for different problems. A connection between number of changes and the characteristic of the problem will be studied and reported in the next (full) version of the paper.

Also, currently a new version of the system based on floating point representation is being developed. Note that for such a system there would be no need for adjusting genotypes in the population, as the algorithm operates on phenotypes. A comparison between these systems (i.e., based on binary and floating point representations) should provide additional clues.

Acknowledgments:
This material is based upon work supported by the the grant 8 T11B 049 10 from the Polish State Committee for Scientific Research (KBN) and the grant IRI-9725424 from the National Science Foundation.

References

1. Kozieł, S. (1996). Non-uniform and non-stationary mutation in numerical optimization using genetic algorithms. Electronics and Telecomm. Quarterly, 42 (3), pp. 273–285.
2. Kozieł, S. (1997). Evolutionary algorithms in constrained numerical optimization problems on convex spaces. Electronics and Telecomm. Quarterly, 43 (1), pp. 5–18.
3. Kozieł, S. and Michalewicz, Z. (1997). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. To appear in Evolutionary Computation, 1998.
4. Michalewicz, Z. (1996). *Genetic Algorithms+Data Structures=Evolution Programs.* New-York: Springer Verlag. 3rd edition.
5. Michalewicz, Z. and C. Z. Janikow (1991). Handling constraints in genetic algorithms. In R. K. Belew and L. B. Booker (Eds.), *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pp. 151–157. Morgan Kaufmann.
6. Michalewicz, Z. and G. Nazhiyath (1995). Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In D. B. Fogel (Ed.), *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pp. 647–651. IEEE Press.
7. Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary computation for constrained parameter optimization problems. Evolutionary Computation, Vol.4, No.1, pp.1–32.
8. Schoenauer, M. and Z. Michalewicz (1996). Evolutionary computation at the edge of feasibility. W. Ebeling, and H.-M. Voigt (Eds.), *Proceedings of the* 4^{th} *Conference on Parallel Problems Solving from Nature*, pp.245–254, Springer Verlag.
9. Schoenauer, M. and Z. Michalewicz (1997). Boundary Operators for Constrained Parameter Optimization Problems. Proceedings of the 7th International Conference on Genetic Algorithms, pp.320–329, July 1997.