Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization

Sławomir Kozieł* and Zbigniew Michalewicz†

Abstract

During the last five years, several methods have been proposed for handling nonlinear constraints by evolutionary algorithms (EAs) for numerical optimization problems. Recent survey papers classify them into four categories (preservation of feasibility, penalty functions, searching for feasibility, and other hybrids).

In this paper we investigate a new approach for solving constrained numerical optimization problems which incorporates a homomorphous mapping between n-dimensional cube and a feasible search space. This approach constitutes an example of the fifth, decoder-based category of constraint handling techniques. We demonstrate the power of this new approach on several test cases and discuss its further potential.

Keywords: *evolutionary computation, optimization technique, nonlinear programming, constrained optimization, decoder, homomorphous mapping.*

1 Introduction

The general nonlinear programming (NLP) problem is to find $\vec{x}$ so as to

$$\text{optimize } f(\vec{x}),\ \vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n,$$

where $\vec{x} \in \mathcal{F} \subseteq \mathcal{S}$. The *objective function* f is defined on the *search space* $\mathcal{S} \subseteq \mathbb{R}^n$ and the set $\mathcal{F} \subseteq \mathcal{S}$ defines the *feasible region*. Usually, the search space $\mathcal{S}$ is defined as a n-dimensional rectangle in $\mathbb{R}^n$ (domains of variables defined by their lower and upper bounds):

$$l(i) \leq x_i \leq u(i), \quad 1 \leq i \leq n,$$

whereas the feasible region $\mathcal{F} \subseteq \mathcal{S}$ is defined by a set of m additional constraints ($m \geq 0$):

$$g_j(\vec{x}) \leq 0, \text{ for } j = 1, \ldots, q, \text{ and } h_j(\vec{x}) = 0, \text{ for } j = q+1, \ldots, m.$$

*Department of Electronics, Telecommunication and Informatics, Technical University of Gdańsk, ul. Narutowicza 11/12, 80-952 Gdańsk, Poland; e-mail: *koziel@ue.eti.pg.gda.pl*

†Department of Computer Science, University of North Carolina, Charlotte, NC 28223, USA *and* at the Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, e-mail: *zbyszek@uncc.edu*

It is a common practice to replace the equations $h_j(\vec{x}) = 0$ by a set of inequalities $h_j(\vec{x}) \leq \delta$ and $h_j(\vec{x}) \geq -\delta$ for some small $\delta > 0$. In the rest of this paper we assume this is the case; consequently, the set of constraints consists of m inequalities $g_j(\vec{x}) \leq 0$, for $j = 1, \ldots, m$.[1] At any point $\vec{x} \in \mathcal{F}$, the constraints g_j that satisfy $g_j(\vec{x}) = 0$ are called the *active* constraints at $\vec{x}$.

The NLP problem, in general, is intractable: it is impossible to develop a deterministic method for the NLP in the global optimization category, which would be better than the exhaustive search (Gregory, 1995). This makes a room for evolutionary algorithms, which aim at complex objective functions (e.g., non differentiable or discontinuous) extended by some constraint-handling methods. Indeed, during the last few years, several evolutionary algorithms have been proposed for the NLP; a recent survey paper (Michalewicz and Schoenauer 1996) provides an overview of these algorithms.

In this paper we propose an alternative approach to the NLP: the evolutionary algorithm uses a decoder, which is based on the transformation of constrained problem at hand to the unconstrained one via a homomorphous mapping. The method, based on earlier work of the first author (Kozieł, 1997), has several advantages over methods proposed earlier (no additional parameters, no need to evaluate—or penalize—infeasible solutions, easiness of approaching a solution located on the edge of the feasible region, no need for special operators, etc). We demonstrate the power of this new approach on a few test cases and discuss its further potential.

The paper is organized as follows. The following section surveys briefly several constraint-handling techniques for numerical optimization problems which have emerged in evolutionary computation techniques over the last years. Sections 3 and 4 discuss the new method (for convex and non-convex feasible regions $\mathcal{F}$, respectively), whereas section 5 presents some experimental results. Section 6 concludes the paper and indicates some directions for future research.

2 Constraint-handling methods

During the last few years several methods were proposed for handling constraints by genetic algorithms for parameter optimization problems. These methods were grouped (Michalewicz and Schoenauer 1996) into four categories: (1) methods based on preserving feasibility of solutions, (2) methods based on penalty functions, (3) methods which make a clear distinction between feasible and infeasible solutions, and (4) other hybrid methods. We discuss them briefly in turn.

2.1 Methods based on preserving feasibility of solutions

The best example of this approach is Genocop (for GEnetic algorithm for Numerical Optimization of COnstrained Problems) system (Michalewicz and Janikow, 1991; Michalewicz et al., 1994). The idea behind the system is based on specialized operators which transform feasible individuals into feasible individuals, i.e., operators, which are closed on the feasible part $\mathcal{F}$ of the search space. The method assumes linear constraints only and a feasible starting point (or feasible initial population). Linear equations are used to eliminate some variables; they are replaced as a linear combination of remaining variables. Linear inequalities are updated accordingly. A closed set of operators maintains feasibility of solutions. For example, when a particular component x_i of a solution vector $\vec{x}$ is mutated, the system determines its current domain $dom(x_i)$ (which is a function of linear constraints and remaining values of the solution vector $\vec{x}$) and the new value of x_i is taken from

[1] After replacement of equations $h_j(\vec{x}) = 0$ ($j = q+1, \ldots, m$) by pairs of inequalities, the total number of inequality constraints is $q + 2 \cdot (m-q) = 2m - q$. However, to simplify the notation, we assume there are m inequality constraints.

this domain (either with flat probability distribution for uniform mutation, or other probability distributions for non-uniform and boundary mutations). In any case the offspring solution vector is always feasible. Similarly, arithmetic crossover, $a\vec{x} + (1-a)\vec{y}$, of two feasible solution vectors $\vec{x}$ and $\vec{y}$ yields always a feasible solution (for $0 \leq a \leq 1$) in convex search spaces (the system assumes linear constraints only which imply convexity of the feasible search space $\mathcal{F}$).

Recent work (Michalewicz et al., 1996; Schoenauer and Michalewicz, 1996; Schoenauer and Michalewicz, 1997) on systems which search only the boundary area between feasible and infeasible regions of the search space, constitutes another example of the approach based on preserving feasibility of solutions. These systems are based on specialized boundary operators (e.g., sphere crossover, geometrical crossover, etc.): it is a common situation for many constrained optimization problems that some constraints are active at the target global optimum, thus the optimum lies on the boundary of the feasible space.

2.2 Methods based on penalty functions

Many evolutionary algorithms incorporate a constraint-handling method based on the concept of (exterior) penalty functions, which penalize infeasible solutions. Usually, the penalty function is based on the distance of a solution from the feasible region $\mathcal{F}$, or on the effort to "repair" the solution, i.e., to force it into $\mathcal{F}$. The former case is the most popular one; in many methods a set of functions f_j $(1 \leq j \leq m)$ is used to construct the penalty, where the function f_j measures the violation of the j-th constraint in the following way:

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

However, these methods differ in many important details, how the penalty function is designed and applied to infeasible solutions. For example, a method of static penalties was proposed (Homaifar et al., 1994); it assumes that for every constraint we establish a family of intervals which determine appropriate penalty coefficient. The method of dynamic penalties was examined (Joines and Houck, 1994), where individuals are evaluated (at the iteration t) by the following formula:

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

where C, α and β are constants. Another approach (Genocop II), also based on dynamic penalties, was described (Michalewicz and Attia, 1994). In that algorithm, at every iteration active constraints only are considered, and the pressure on infeasible solutions is increased due to the decreasing values of temperature τ. In (?) a method for solving constraint satisfaction problems that changes the evaluation function based on the performance of a EA run was described: the penalties (weights) of those constraints which are violated by the best individual after termination are raised, and the new weights are used in the next run. A method based on adaptive penalty functions was was developed in (Bean and Hadj-Alouane, 1992; Hadj-Alouane and Bean, 1992): one component of the penalty function takes a feedback from the search process. Each individual is evaluated by the formula:

$$eval(\vec{x}) = f(\vec{x}) + \lambda(t) \sum_{j=1}^{m} f_j^2(\vec{x}),$$

where $\lambda(t)$ is updated every generation t with respect to the current state of the search (based on last k generations). The adaptive penalty function was also used in (Smith and Tate, 1993), where both the search length and constraint severity feedback was incorporated. It involves the estimation

of a near-feasible threshold q_j for each constraint $1 \leq j \leq m$); such thresholds indicate distances from the feasible region $\mathcal{F}$ which are "reasonable" (or, in other words, which determine "interesting" infeasible solutions, i.e., solutions relatively close to the feasible region). Additional method (so-called segregated genetic algorithm) was proposed in (Leriche et al., 1995) as yet another way to handle the problem of the robustness of the penalty level: two different penalized fitness functions with static penalty terms p_1 and p_2 were designed (smaller and larger, respectively). The main idea is that such an approach will result roughly in maintaining two subpopulations: the individuals selected on the basis of f_1 will more likely lie in the infeasible region while the ones selected on the basis of f_2 will probably stay in the feasible region; the overall process is thus allowed to reach the feasible optimum from both sides of the boundary of the feasible region.

2.3 Methods based on a search for feasible solutions

There are a few methods which emphasize the distinction between feasible and infeasible solutions in the search space $\mathcal{S}$. One method, proposed in (Schoenauer and Xanthakis, 1993) (called a "behavioral memory" approach) considers the problem constraints in a sequence; a switch from one constraint to another is made upon arrival of a sufficient number of feasible individuals in the population.

The second method, developed in (Powell and Skolnick, 1993) is based on a classical penalty approach with one notable exception. Each individual is evaluated by the formula:

$$eval(\vec{x}) = f(\vec{x}) + r \sum_{j=1}^{m} f_j(\vec{x}) + \theta(t, \vec{x}),$$

where r is a constant; however, the original component $\theta(t, \vec{x})$ is an additional iteration dependent function which influences the evaluations of infeasible solutions. The point is that the method distinguishes between feasible and infeasible individuals by adopting an additional heuristic rule (suggested earlier in (Richardson et al., 1989)): for any feasible individual $\vec{x}$ and any infeasible individual $\vec{y}$: $eval(\vec{x}) < eval(\vec{y})$, i.e., any feasible solution is better than any infeasible one.[2]

The third method (Genocop III), proposed in (Michalewicz and Nazhiyath, 1995) is based on the idea of repairing infeasible individuals. Genocop III incorporates the original Genocop system, 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 P_s consists of so-called search points from $\mathcal{F}_l$ which satisfy linear constraints of the problem. The feasibility (in the sense of linear constraints) of these points is maintained by specialized operators. The second population P_r consists of so-called reference points from $\mathcal{F}$; these points are fully feasible, i.e., they satisfy *all* constraints. Reference points $\vec{r}$ from P_r, being feasible, are evaluated directly by the objective function (i.e., $eval(\vec{r}) = f(\vec{r})$). On the other hand, search points from P_s are "repaired" for evaluation.

2.4 Hybrid methods

It is relatively easy to develop hybrid methods which combine evolutionary computation techniques with deterministic procedures for numerical optimization problems. In (Waagen et al. 1992) a combined evolutionary algorithm with the direction set method of Hooke-Jeeves is described; this hybrid method was tested on three (unconstrained) test functions. In (Myung et al., 1995) the

[2]For minimization problems.

authors considered a similar approach, but they experimented with constrained problems. Again, they combined evolutionary algorithm with some other method—developed in (Maa and Shanblatt, 1992). However, while the method of (Waagen et al. 1992) incorporated the direction set algorithm as a problem-specific operator of his evolutionary technique, in (Myung et al., 1995) the whole optimization process was divided into two separate phases.

Several other constraint handling methods deserve also some attention. For example, some methods use of the values of objective function f and penalties f_j ($j = 1, \ldots, m$) as elements of a vector and apply multi-objective techniques to minimize all components of the vector. For example, in (Schaffer, 1985), Vector Evaluated Genetic Algorithm (VEGA) selects $1/(m+1)$ of the population based on each of the objectives. Such an approach was incorporated by Parmee and Purchase (Parmee and Purchase, 1994) in the development of techniques for constrained design spaces. On the other hand, in the approach by (Surry et al., 1995), all members of the population are ranked on the basis of constraint violation. Such rank r, together with the value of the objective function f, leads to the two-objective optimization problem. This approach gave a good performance on optimization of gas supply networks.

Also, an interesting approach was reported in (Paredis, 1994). The method (described in the context of constraint satisfaction problems) is based on a co-evolutionary model, where a population of potential solutions co-evolves with a population of constraints: fitter solutions satisfy more constraints, whereas fitter constraints are violated by more solutions. There is some development connected with generalizing the concept of "ant colonies" (Colorni et al., 1996) (which were originally proposed for order-based problems) to numerical domains (Bilchev and Parmee, 1995); first experiments on some test problems gave very good results (Wodrich and Bilchev, 1997). It is also possible to incorporate the knowledge of the constraints of the problem into the belief space of cultural algorithms (Reynolds, 1994); such algorithms provide a possibility of conducting an efficient search of the feasible search space (Reynolds et al., 1995).

3 The homomorphous mapping: convex search spaces

Decoders offer an interesting option for all practitioners of evolutionary techniques. In these techniques a chromosome "gives instructions" on how to build a feasible solution. For example, a sequence of items for the knapsack problem can be interpreted as: "take an item if possible"—such interpretation would lead always to a feasible solution.

However, it is important to point out that several factors should be taken into account while using decoders. Each decoder imposes a mapping T between a feasible solution and decoded solution. It is important that several conditions are satisfied: (1) for each solution $s \in \mathcal{F}$ there is an encoded solution d, (2) each encoded solution d corresponds to a feasible solution s, and (3) all solutions in $\mathcal{F}$ should be represented by the same number of encodings d.[3] Additionally, it is reasonable to request that (4) the transformation T is computationally fast and (5) it has locality feature in the sense that small changes in the coded solution result in small changes in the solution itself. An

[3]However, as observed by Davis (1997), the requirement that all solutions in $\mathcal{F}$ should be represented by the same number of decodings seems overly strong: there are cases in which this requirement might be suboptimal. For example, suppose we have a decoding and encoding procedure which makes it impossible to represent suboptimal solutions, and which encodes the optimal one: this might be a good thing. (An example would be a graph coloring order-based chromosome, with a decoding procedure that gives each node its first legal color. This representation could not encode solutions where some nodes that could be colored were not colored, but this is a good thing!)

interesting study on coding trees in genetic algorithm was reported in (Palmer and Kershenbaum, 1994), where the above conditions were formulated.

However, the use of decoders for continuous domains has not been investigated. Actually, this might be a quite promising direction for approaching nonlinear programming problems by evolutionary methods. For example, it is relatively easy to establish a one-to-one mapping between arbitrary convex feasible search space $\mathcal{F}$ and the n-dimensional cube $[-1,1]^n$ (see figure 1).

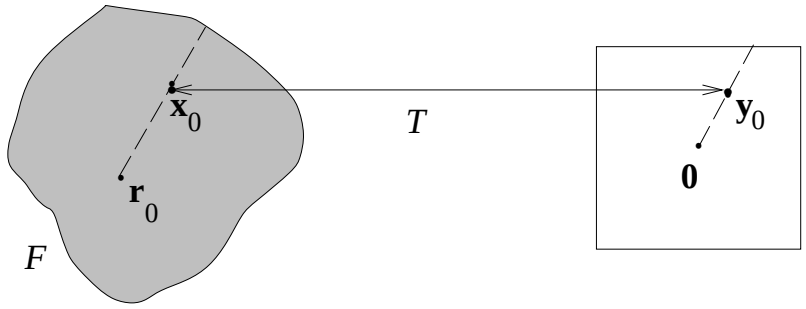

Figure 1: A mapping T from a space $\mathcal{F}$ into a cube $[-1,1]^n$ (two-dimensional case)

Note that an arbitrary (different than $\vec{0}$) point $\vec{y_0} = (y_{0,1}, \ldots, y_{0,n}) \in [-1,1]^n$ defines a line segment from the $\vec{0}$ to the boundary of the cube; this segment is described by:

$y_i = y_{0,i} \cdot t$, for $i = 1, \ldots, n$, where

t varies from 0 to $t_{max} = 1/\max\{|y_{0,1}|, \ldots, |y_{0,n}|\}$. Clearly, for $t = 0$, $\vec{y} = \vec{0}$, and for $t = t_{max}$, $\vec{y} = (y_{0,1}t_{max}, \ldots, y_{0,n}t_{max})$—a boundary point of the $[-1,1]^n$ cube.

Consequently, the corresponding (to $\vec{y_0} \in [-1,1]^n$) feasible point $\vec{x_0} \in \mathcal{F}$ (with respect to some reference point[4] $\vec{r_0}$) is defined as

$$\vec{x_0} = \vec{r_0} + \vec{y_0} \cdot \tau,$$

where $\tau = \tau_{max}/t_{max}$, and τ_{max} is determined (with arbitrary precision) by a binary search procedure such that

$$\vec{r_0} + \vec{y_0} \cdot \tau_{max}$$

is a boundary point of the feasible search space $\mathcal{F}$.

The above homomorphous mapping satisfies all requirements of a "good" decoder: apart from being one-to-one, the transformation is fast and has locality feature.

Additionally, there are a few other features which makes the proposed method very interesting; these include:

- as opposed to most of other constraint-handling methods, there is no need for any additional parameters (e.g., frequency of seven operators in Genocop (Michalewicz, 1996), penalty coefficients, etc): only basic parameters of an evolutionary algorithm (e.g., population size, probability of crossover, etc.) are required.

[4]A reference point $\vec{r_0}$ is an arbitrary internal point of the convex set $\mathcal{F}$. 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 $\vec{r_0}$, such that every line segment originating in $\vec{r_0}$ intersects the boundary of $\mathcal{F}$ in precisely one point. This requirement is satisfied, of course, for any convex set $\mathcal{F}$.

- as opposed to some other constraint-handling methods, there is no need for any specialized operators to maintain the feasibility of solutions (e.g., operators of Genocop (Michalewicz, 1996) to maintain linear constraints, specialized boundary operators to search the boundary between feasible and infeasible parts of the search space, etc.); any evolutionary algorithm can be used together with the proposed mapping.

- as opposed to most of other constraint-handling methods (i.e., all methods which do not reject infeasible solutions), there is no need to evaluate infeasible solutions (in particular, no need to penalize them, tuning penalty coefficients, no need to repair it, etc).

- as opposed to some other constraint-handling methods (e.g., methods based on penalty functions, hybrid methods), the proposed method always returns a feasible solution.

The proposed approach can also be extended by an additional method of iterative solution improvement, which is based on the relationship between the location of the reference point and the efficiency of the proposed approach. It is clear, that location of the reference point $\vec{r_0}$ has an influence on "deformation" of the domain of optimized function: evolutionary algorithm does not optimize the objective function, but rather some other function which is topologically equivalent to the original one. For example, consider the case, when the reference point is located near by the edge of the feasible region $\mathcal{F}$—it is easy to notice a strong irregularity of transformation T: the part of the cube $[-1, 1]^2$, which is on the left side of the vertical line, is transformed into much smaller part of the set $\mathcal{F}$ than the part on the right side of this line (see figure 2).

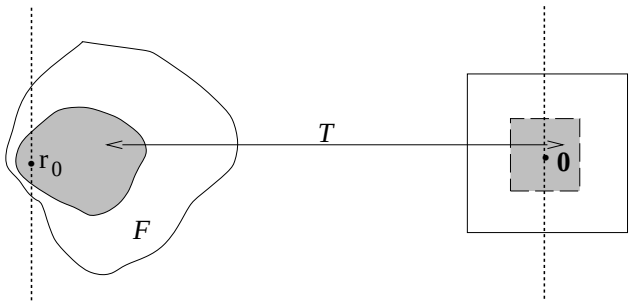

Figure 2: The influence of location of the reference point on the transformation T

According to these considerations, it seems intuitively profitable to localize the reference point in the neighborhood of expected optimum, if this optimum is close to the edge of the set $\mathcal{F}$: in such case the area between the edge of $\mathcal{F}$ and the reference point $\vec{r_0}$ is explored more precisely.

In the case of lack of information about approximate localization of solution, the reference point should be placed close to the geometrical center of the set $\mathcal{F}$. This can be done easily by sampling set $\mathcal{F}$ and setting

$$\vec{r_0} = 1/k \sum_{i=1}^{k} \vec{x_i},$$

where $\vec{x_i}$ are samples from $\mathcal{F}$. It is also possible to take advantage of the mentioned effect for the purpose of iterative improvement of the best found solution. To obtain this effect it is necessary to repeat optimization process with a new reference point $\vec{r_0'}$, which is located on a line segment between the current reference point $\vec{r_0}$ and the best solution $\vec{b}$ found so far:

$$\vec{r_0'} = t \cdot \vec{r_0} + (1-t) \cdot \vec{b},$$

where $t \in (0, 1]$ should be close to zero. This change of the location of the reference point causes that in the next iteration the neighborhood of the found optimum is explored more precisely in comparison with the remaining part of the feasible region. Our experiments show (see section 5) that such a method usually gives good results for problems with optimal solutions localized on the edge of the feasible region.

The proposed approach can be also extended to handle non-convex search spaces; we discuss this generalization in the next section.

4 The homomorphous mapping: non-convex search spaces

In this section we present a generalization of the method described in the previous section to handle *arbitrary* constraints for numerical optimization problems. The task 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. Note, that $\mathcal{F}$ need not be convex; it might be concave or even can consist of disjoint (non-convex) regions.

This homomorphous mapping φ, clearly, would be more complex than T of the previous section. Note that any line segment L which originates at a reference point $\vec{r_0} \in \mathcal{F}$ may intersect a boundary of the feasible search space $\mathcal{F}$ in more than just one point (see figure 3).

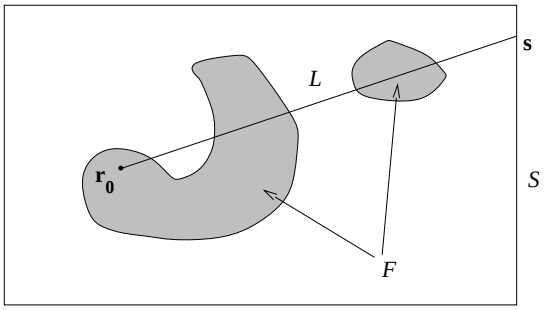

Figure 3: A line segment in a non-convex space $\mathcal{F}$ (two-dimensional case)

Because of that, it is necessary to take into account the domains of the variables and to re-design the original mapping T.

First, let us define an additional one-to-one mapping g between the cube $[-1, 1]^n$ and the search space $\mathcal{S}$ (note, that the search space $\mathcal{S}$ is defined as a Cartesian product of domains of all problem variables; see Introduction). Then the mapping $g : [-1, 1]^n \to S$ can be defined as

$$g(\vec{y}) = \vec{x},$$

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 $\vec{r_0} \in \mathcal{F}$ and a point $\vec{s}$ at the boundary of the search space $\mathcal{S}$, is defined as

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

Clearly, if the feasible search space $\mathcal{F}$ is convex, 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:

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

where $r_0 \in \mathcal{F}$ is a reference point, and $y_{max} = \max_{i=1}^{n} |y_i|$. Figure 4 illustrates the transformation φ.

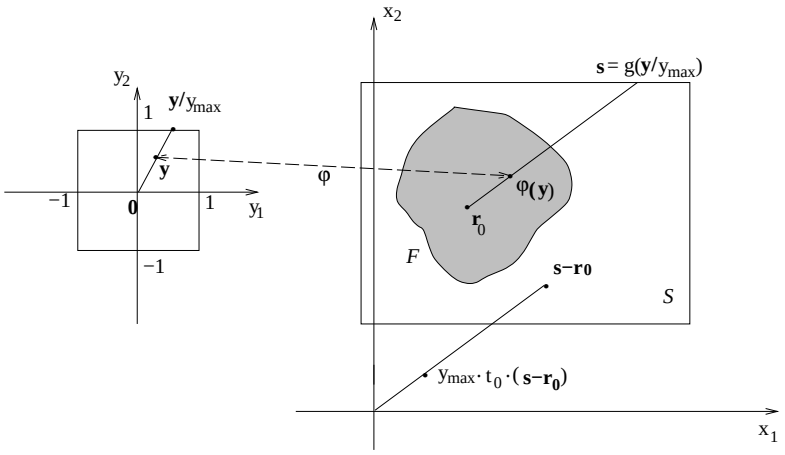

Figure 4: A mapping φ from the cube $[-1, 1]^n$ into the convex space $\mathcal{F}$ (two-dimensional case), with particular steps of the transformation

Now we are ready to return to the general case of arbitrary constraints (i.e., non-convex feasible search spaces $\mathcal{F}$). Let us consider an arbitrary point $\vec{y} \in [-1, 1]^n$ and a reference point $\vec{r_0} \in \mathcal{F}$. A line segment L between the reference point $\vec{r_0}$ and the point $\vec{s} = g(\vec{y}/y_{max})$ at the boundary of the search space $\mathcal{S}$, is defined as before:

$$L(\vec{r_0}, \vec{s}) = \vec{r_0} + t \cdot (\vec{s} - \vec{r_0}), \text{ for } 0 \leq t \leq 1,$$

however, it may intersect the boundary of $\mathcal{F}$ in many points (see figure 3). In other words, 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 sub-intervals 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 $t_{2k} \leq 1$ (see figure 5).

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 rather between $(0, 1]$ and the sum of intervals $(t_{2i-1}, t_{2i}]$:

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

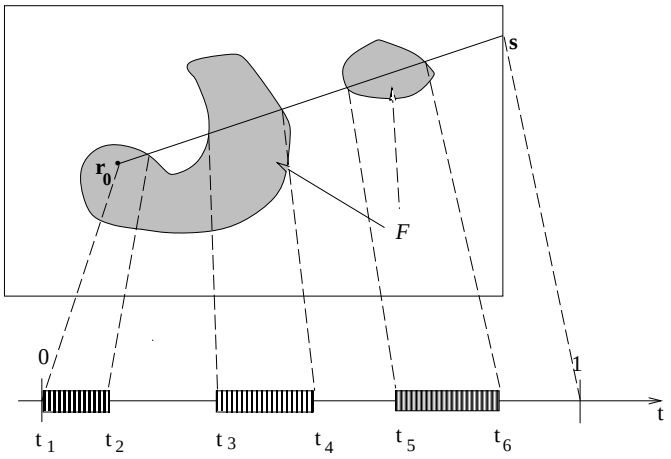

Figure 5: A line segment in a non-convex space $\mathcal{F}$ and corresponding sub-intervals (two-dimensional case)

Note, that due to this change, left boundary point (from each interval $1 \leq i \leq k$) is lost. 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:

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

as follows:

$$\delta(t) = (t - t_{2i-1} + \sum_{j=1}^{i-1} d_j)/d,$$

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 $a \leq \delta(t_{2j})$.

Now we are ready to define the general mapping φ, which is the essence of our 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:

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

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

It is interesting to note, that the definition of the mapping φ is almost identical to the previous one (for convex feasible search spaces); the only difference is in the use of additional mapping γ (due to the fact that there may be several intersection points between a line segment and the boundary of $\mathcal{F}$).

Finally, it is necessary to consider a method of finding such points of intersections t_i (see figure 5). This was relatively easy for convex sets, since there was only one point of intersection. Now the problem is more complex. In our implementation we have used the following approach. Let us

consider any boundary point $\vec{s}$ of $\mathcal{S}$ and the line segment L determined by this point and a reference point $\vec{r_0} \in \mathcal{F}$. There are m constraints $g_i(\vec{x}) \leq 0$ and each of them can be represented as a function β_i of one independent variable t (for fixed reference point $\vec{r_0} \in \mathcal{F}$ and the boundary point $\vec{s}$ of $\mathcal{S}$):

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

As stated earlier, the feasible region need not be convex, so it may have more than one point of intersection of the segment L with the boundaries of the set $\mathcal{F}$. Therefore, let us partition the interval $[0, 1]$ into v subintervals $[v_{j-1}, v_j]$, where $v_j - v_{j-1} = 1/v$ $(1 \leq j \leq v)$, so that equations $\beta_i(t) = 0$ have at most one solution in every subinterval.[5] 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 $g_i(\vec{x}) \leq 0$ are known, it is quite easy to determine intersection points between this line segment L and the boundary of the feasible set $\mathcal{F}$.

5 Experimental study

In (Michalewicz and Schoenauer 1996) eleven test cases for constrained numerical optimization problems were proposed (they are listed in the Appendix in this paper). These test cases include objective functions of various types (linear, quadratic, cubic, polynomial, nonlinear) with various number of variables and different types (linear inequalities, nonlinear equations and inequalities) and numbers of constraints. The ratio between the size of the feasible search space $\mathcal{F}$ and the size of the whole search space $\mathcal{S}$ for these test cases vary from 0% to almost 100%; the topologies of feasible search spaces are also quite different. These test cases are summarized in table 1. For each test case we list number n of variables, type of the function f, the relative size of the feasible region in the search space given by the ratio ρ, the number of constraints of each category (linear inequalities *LI*, nonlinear equations *NE* and inequalities *NI*), and the number a of active constraints at the optimum (including equality constraints).

To test the proposed method an evolutionary algorithm based on Gray coding was implemented (25 bits represented every variable). The algorithm (ALG_g) incorporated proportional selection (no elitism), function scaling, and standard operators (flip mutation and 1-point crossover). All parameters were fixed:

$$\text{pop\_size} = 70, \text{ generation gap} = 100\%, \text{ and } p_c = 0.9.$$

The only non-standard feature incorporated into the system was a variable probability of mutation:

$$p_m(t) = p_m(0) - (p_m(0) - p_m(T)) \cdot (t/T)^r,$$

where t and T are the current and maximum generation numbers, respectively. In all experiments, $p_m(0) = 0.005$, $r = 4$, and $p_m(T) = 0.00005$ (Bäck and Schütz, 1996).

Three types of experiments were performed for each test case:

Experiment #1: 20 runs were executed. For each run the maximum number of generations was set to $T = 5000$, and for each run a random reference point $\vec{r_0}$ was selected (i.e., the first randomly generated feasible point was accepted as a reference point).

[5] Density v of the partition is adjusted experimentally. In all experiments reported in section 5, $v = 20$.

Function	n	Type of f	ρ	LI	NE	NI	a
$G1$	13	quadratic	0.0111%	9	0	0	6
$G2$	k	nonlinear	99.8474%	0	0	2	1
$G3$	k	polynomial	0.0000%	0	1	0	1
$G4$	5	quadratic	52.1230%	0	0	6	2
$G5$	4	cubic	0.0000%	2	3	0	3
$G6$	2	cubic	0.0066%	0	0	2	2
$G7$	10	quadratic	0.0003%	3	0	5	6
$G8$	2	nonlinear	0.8560%	0	0	2	0
$G9$	7	polynomial	0.5121%	0	0	4	2
$G10$	8	linear	0.0010%	3	0	3	6
$G11$	2	quadratic	0.0000%	0	1	0	1

Table 1: Summary of eleven test cases. The ratio $\rho = |\mathcal{F}|/|\mathcal{S}|$ was determined experimentally by generating 1,000,000 random points from $\mathcal{S}$ and checking whether they belong to $\mathcal{F}$ (for $G2$ and $G3$ we assumed $k = 50$). *LI, NE*, and *NI* represent the number of linear inequalities, and nonlinear equations and inequalities, respectively

Experiment #2: everything was the same as in experiment #1 except that the maximum number of generations was increased to $T = 20000$.

Experiment #3: everything was the same as in experiment #1 except that the selected reference point $\vec{r_0}$ was the best solution obtained from experiment #1. Also, only 10 runs were executed.

The first experiment is a standard one (as a standard Gray-coded GA is used). The motivation for making extended runs (experiment #2) was to examine the speed of convergence of the algorithm towards the global optimum. On the other hand, the results of experiment #3 indicate the importance of the selection process of the reference point.

5.1 Results

In the following subsection we report on the results of all experiments; these results are summarized in tables 2 and 3.

The experimental results provided interesting data. First of all, it is important to note that the proposed approach gave satisfactory results for all test cases, except for the test case $G5$, which involves (apart from two inequalities) three equations:

$$h_1(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0,$$
$$h_2(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0,$$
$$h_3(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0.$$

In this test case, the replacement of equations by inequalities $h_j(\vec{x}) \leq \delta$ and $h_j(\vec{x}) \geq -\delta$ $(j = 1, 2, 3)$ did not provide quality results. Note, however, that the test cases $G3$ and $G11$ have one equality constraint each, and the replacement of these equations by a pair of inequalities was successful

Function	Optimum value	Experiment #1			Experiment #2		
		worst	best	avg.	worst	best	avg.
$G1$	-15	-14.0566	-14.7207	-14.4609	-14.6154	-14.7864	-14.7082
$G2$	0.803553	0.78427	0.79506	0.79176	0.79119	0.79953	0.79671
$G3$	1.0	0.9917	0.9983	0.9965	0.9978	0.9997	0.9989
$G4$	-30665.5	-30617.0	$-30662.5.3$	-30643.8	-30645.9	-30664.5	-30655.3
$G5$	5126.4981	—	—	—	—	—	—
$G6$	-6961.8	-4236.7	-6901.5	-6191.2	-5473.9	-6952.1	-6342.6
$G7$	24.306	38.682	25.132	26.619	25.069	24.620	24.826
$G8$	0.095825	0.0291434	0.0958250	0.0871551	0.0291438	0.0958250	0.0891568
$G9$	680.63	682.88	681.43	682.18	683.18	680.91	681.16
$G10$	7049.33	11894.5	7215.8	9141.7	9659.3	7147.9	8163.6
$G11$	0.75	0.75	0.75	0.75	0.75	0.75	0.75

Table 2: Summary of results of the homomorphous mapping method on eleven test cases; experiments #1 and #2. The test case $G2$ was run with $k = 20$ variables, whereas $G3$—with $k = 10$ variables

Function	Optimum value	Experiment #3		
		worst	best	avg.
$G1$	-15	-14.5732	-14.7184	-14.6478
$G2$	0.803553	0.78279	0.79486	0.78722
$G3$	1.0	0.9960	0.9978	0.9970
$G4$	-30665.5	-30645.6	-30661.5	-30653.1
$G5$	5126.4981	—	—	—
$G6$	-6961.8	-6390.6	-6944.4	-6720.4
$G7$	24.306	26.182	25.090	25.545
$G8$	0.095825	0.0958246	0.0958250	0.0958248
$G9$	680.63	683.58	681.72	682.56
$G10$	7049.33	7685.8	7321.2	7498.6
$G11$	0.75	0.75	0.75	0.75

Table 3: Summary of results of the homomorphous mapping method on eleven test cases; experiment#3. The test case $G2$ was run with $k = 20$ variables, whereas $G3$—with $k = 10$ variables

despite the fact that the proposed method of homomorphous mapping was proposed primarily for inequality constraints only.

It is a well-known fact that different constraint-handling techniques provide results of different quality on different test cases (see, for example, (Michalewicz, 1995)). However, the proposed technique was very consistent in locating the area of global optimum; in many cases the difference between the values of the objective function G at the global solution and at the best solution found was due to the shape of the landscape: mimimal changes in $\vec{x}$ result in large changes of evaluation function $G(\vec{x})$. For example, one solution found for $G1$ (experiment #1) was:

$\vec{x}$ =(0.999, 0.999, 0.993, 0.997, 0.978, 0.987, 0.996, 0.999, 0.992, 2.971, 2.920, 2.951, 0.977),

and the Euclidean distance between $\vec{x}$ and the global optimum $\vec{x}^*$ was 0.1 (and $G1(\vec{x}) = -14.72$). Experiment #2 confirmed further this observation as larger number of generations moved the best individual closer to the global solution (for all considered test cases—see table 2).

Moreover, due to the new approach, we were able to correct an error published in (Homaifar et al., 1994) and repeated in (Michalewicz and Schoenauer 1996). The test problem $G4$ was formulated as follows:

$$\text{minimize } G4(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,$$

subject to three double inequalities:

$$\begin{aligned} 0 &\leq 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5 \leq 92, \\ 90 &\leq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leq 110, \\ 20 &\leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25, \end{aligned}$$

and bounds:

$$78 \leq x_1 \leq 102,\ 33 \leq x_2 \leq 45,\ 27 \leq x_i \leq 45 \text{ for } i = 3, 4, 5.$$

The best solution obtained in 10 runs reported in (Homaifar et al., 1994) was

$$\vec{x} = (80.49, 35.07, 32.05, 40.33, 33.34)$$

with $G4(\vec{x}) = -30005.7$, whereas the optimum solution (Himmelblau, 1992) is

$$\vec{x}^* = (78.0, 33.0, 29.995, 45.0, 36.776),$$

with $G4(\vec{x}^*) = -30665.5$. Running experiments with ALG_g we have noticed that the results are *better* than the global optimum, e.g.,

$$\vec{x} = (78.0270, 33.0119, 27.0821, 44.9645, 44.9533),$$

where $G4(\vec{x}) = -31021.3$. By comparing the definition of the problem with the original source (Himmelblau, 1992) we discovered the error; the first double inequality should be

$$0 \leq 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leq 92,$$

(note a small difference in the coefficient of x_1x_4 component).[6] Note also that the algorithm ALG_g, after the problem was corrected, provided with an excellent value of -30664.5—by far the best value reported by any evolutionary system for this test case!

By comparing the results of experiments #1 and #3, it is quite obvious that quality of selected reference point is of importance; values obtained in experiment #3 for some test cases were better (average scores) than these of experiment #1 (note the same number of generations for experiments #1 and #3). There were exceptions to this rule, however, e.g., test cases $G2$ and $G9$, where the selection of the best reference point did not help.[7] Besides, the reference point for experiment #3

[6] The reader is encouraged to make appropriate correction in (Michalewicz and Schoenauer 1996).

[7] $G2$ was also the only test case for which the number of generations was enlarged: from 5,000 to 10,000 for experiments #1 and #3, and from 20,000 to 30,000 for experiment #2.

was always selected as the best solution from the experiment #1; clearly, this criterion of selection need not be optimal.

We have experimented also with another algorithm, ALG_b, based on binary coding. All other components of the first algorithm were left unchanged (i.e., proportional selection, no elitism, standard operators: flip mutation and 1-point crossover, values of all parameters); the only difference between these two algorithms (apart from representation) was that a variable mutation probability p_m was replaced by a variable *distribution* of mutation probability for bits: while the probability of bit mutation remains constant (and equal to 0.005), the probability of mutating more significant bits decreases with generation number (hence the probability of mutating less significant bits grows with generation number). This feature is responsible for fine-tuning capabilities of the system.

For some test problems, ALG_b provided with yet better results. This was the case of problem $G2$ and $G6$. For example, for the latter case, the comparison between ALG_g and ALG_b is illustrated in table 4.

Experiment	ALG_g			ALG_b		
	worst	best	avg.	worst	best	avg.
#1	−4236.7	−6901.5	−6191.2	−5476.3	−6955.8	−6600.4
#2	−5473.9	−6952.1	−6342.6	−5975.8	−6949.3	−6641.6
#3	−6390.6	−6944.4	−6720.4	-6909.1	−6961.1	−6948.1

Table 4: Comparison of results of ALG_g and ALG_b on test case $G6$; experiments #1, #2, and #3. Optimum value equals to −6961.8

It seems that ALG_g is better in locating the area of global optima, but slower in converging to the global solution. On the other hand, ALG_b was often trapped in a local optimum (for most test cases the results were inferior to those reported in tables 2 and 3), but the convergence was much quicker. This suggests another possibility of designing an algorithm which changes the representation from Gray into binary at some stage of the run (e.g., when some 'population diversity coefficient' reaches some threshold). Some other possibilities are discussed in the next section.

5.2 Disjoint components

Note, that all test cases discussed earlier in section 5.1 had a small number (usually one) of disjoined components of feasible part of the search space. It is interesting to investigate the behavior of the algorithm in cases where such a number is much larger.

Let us consider the following problem. Maximize

$$G12(\vec{x}) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100,$$

subject to the constraints[8]:

$$(x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 \leq 0.25,$$

[8]Note that in this test problem, as opposed to all other test problems $G1 - G11$, the feasible search space $\mathcal{F}$ is defined as a union of all constraints (spheres), i.e., a point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that $(x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 \leq 0.25$.

for $p, q, r = 1, 3, 5, 7, 9$ and bounds:

$$0 \leq x_i \leq 10 \ (1 \leq i \leq 3).$$

Note that the feasible region of the search space consists of $5^3 = 125$ disjoint spheres (all of them have a radius of 0.5). The function $G12$ has a global maximum at $\vec{x}^* = (5, 5, 5)$, where $G12(\vec{x}^*) = 1$.

10 runs of the algorithm ALG_g were executed (with the same parameters as given earlier in this section). For each run the maximum number of generations was set to $T = 500$, and, as before, for each run a random reference point $\vec{r_0}$ was selected (i.e., the first randomly generated feasible point was accepted as a reference point). The algorithm did not have any difficulties in locating the global optimum. The best, average, and worst values of $G12$ (out of 10 runs) were:

best = 1.000000000,
average = 0.999934768,
worst = 0.999694591.

It is possible to modify the above test case to increase the number of disjoint components (by increasing the number of spheres–constraints and decreasing their radius). For example, if we change the constraints of the above problem:

$$(x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 \leq 0.0625,$$

for $p, q, r = 1, \ldots, 9$, the feasible region of the search space would consist of $9^3 = 729$ disjoint spheres (all of them with a radius of 0.25). The function $G12$ still has a global maximum at $\vec{x}^* = (5, 5, 5)$, and $G12(\vec{x}^*) = 1$. Again, the algorithm performed very well; the best, average, and worst values of $G12$ (out of 10 runs) were:

best = 0.999999857,
average = 0.999134613,
worst = 0.991950498.

These limited experiments demonstrate the ability of the method to deal with multiple disjoint regions of the feasible search space. However, further analysis and experiments are necessary for the full evaluation of this constraint-handling method.

6 Conclusions

The paper presents a new approach for constrained numerical optimization, based on homomorphous mapping between the cube $[-1, 1]^n$ and the feasible part of the search space. This is the first approach of this type; until now, mappings (or decoders) were applied only to discrete optimization problems. As indicated in the Introduction, two main constraint-handling methods were based on penalty functions and preservation of feasible solutions by specialized operators or repair algorithms. Thus the proposed method is the first one in a new category of methods based on decoders (for parameter optimization problems).

The reported results indicate its huge potential; the proposed method does not require 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 (Schoenauer and Michalewicz, 1997), (Schoenauer and Michalewicz, 1996). 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. However, it should be also pointed out that this method

- introduces an additional, problem-dependent parameter, v (usually determined experimentally before the run of the algorithm) to partition the interval $[0, 1]$ into v subintervals of equal length such that equations $\beta_i(t) = 0$ (see section 4) have at most one solution in every subinterval.

- loses the locality feature of the mapping for non-convex feasible search spaces: a small changes in the coded solution may result in huge changes in the solution itself (e.g., when solution "moves" from one disjoint region to the other);

- requires additional computational effort (binary search) for finding all intersection points t_i for a line segment with the boundaries of the feasible region.

As described in the previous section, the proposed method of homomorphous mapping provided with quality results for *all* test cases which involved inequalities only (and on two additional test cases which involved single equations). The results were much better than for any earlier reported method. We have already discussed (previous section) the case of $G4$, where the performance of the system allowed a discovery of error in the problem formulation. Additionally, (Michalewicz, 1995) reports experimental results of several constraint handling techniques (static and dynamic penalties, behavioural memory, death penalty, promoting feasible solutions, etc.) on five test cases (four of which are included in the test suite $G1$–$G11$). By comparing the results of the other constraints handling techniques with the results reported in the previous section (on common test cases: $G1$, $G7$, $G9$, and $G10$), the superiority of the proposed method is quite clear.

Moreover, it should be relatively easy to enhance further the performance of the algorithm. If proportional selection is replaced by a tournament selection with elitism, the algorithm provides with much better results. For example, for the test case $G2$, the results of all runs were around 0.8035 for 5,000 generations (experiment #1), whereas the best result was 0.8036. For $G3$ the value of 0.99999 was reached in as little as 1,000 generations, whereas for $G9$ the best result was 680.634. (Compare these values with those reported in Table 2). Also, for the test case of $G6$, the global optimum was found in almost all runs (value -6961.8), and for $G4$, the results stayed within a range from -30665.5 to -30664, with the best solution of -30665.5 within less than 5,000 generations (again, refer to the Table 2, experiment #1, for comparisons). Similar (or better) results are also expected for a floating-point representation of solutions. Note that none of the other constraint-handling method, reported results of such quality for the test cases $G1$–$G11$.

Let us also point out, however, that it is quite difficult to compare different constraint-handling algorithms due to different computational effort they require. Note, that the concept of "evaluating a solution" is not defined clearly. For example, assume we are solving a parameter optimization problem defined by the objective function f and m constraints. Then a single evaluation of individual requires:

- for a penalty method: $m + 1$ function calculations (one for the objective function and m for constraints),

- for a repair algorithm: $1 + \mu \cdot m$, where μ represents an average number of iterations required to 'repair' an individual.

In the proposed method of homomorphous mapping (due to a binary search involved to establish intersection points of a line segment and the boundaries of the feasible part of the search space $\mathcal{F}$), the number of function calculations per single evaluation is even higher than in a simple 'repair' approach.

To evaluate computational overhead introduced by the proposed method, an additional set of experiments was performed. The execution times of the algorithm ALG_g were recorded for all test problems $G1$–$G12$ in two scenarios: (1) all constraints were considered and (2) all constraints (apart from the bounds of the variables) were ignored. Table 5 reports the increase of the CPU time of the algorithm after inclusion of constraints (i.e., it provides, for each of the test cases, the ratio t_2/t_1, where t_2 and t_1 denote CPU time of the algorithm for a test case with and without constraints, respectively).

Function	$G1$	$G2$	$G3$	$G4$	$G6$	$G7$	$G8$	$G9$	$G10$	$G11$
t_2/t_1	2.53	1.61	1.59	2.33	2.43	2.77	1.63	2.10	2.36	1.73

Table 5: The ratio between computational times t_2/t_1 of the algorithm ALG_g for all test problems with (t_2) and without constraints (t_1)

The inclusion of constraints in test cases $G1$–$G11$ resulted in an average two-fold increase of computational time of the algorithm. Note, however, that all the test cases $G1$–$G11$ have a small number of disjoint components of the feasible part of the search space (usually one). Thus the experiment with two versions of $G12$ (with 125 and 729 disjoint components) was more meaningful, resulting in ratios of 21.5 and 103, respectively. Clearly, larger number of disjoint components implies larger number of intersections of a line segment with the boundaries of the feasible region, hence a significant increase in computation time.

There are some additional issues which might be investigated in connection with this type of mapping. For example, by setting $\tau = \tau_{max}$, only part of the boundary of the feasible region $\mathcal{F}$ will be explored[9] (thus, it is possible to use this approach to search the boundary of the feasible search space). Moreover, it is interesting to investigate non-uniform distributions of values of t (between 0 and t_{max}): by increasing the probability of selecting t close to t_{max}, the system explores points closer to the boundary of $\mathcal{F}$. On the other hand, by increasing the probability of selecting t close to zero, the system explores points closer to the reference point. Thus exploration of non-uniform distribution of t provides with additional possibilities for tuning the search.

It would be also important to investigate the role of the reference point $\vec{r_0}$, which can "follow" the best solution found so far (the method of iterative solution improvement). In that way, the reference point can adapt itself to the current state of the search. This research direction seems very promising (as are the results of preliminary runs, see see (Kozieł and Michalewicz, 1998)), especially in connection with some ideas presented earlier (e.g., a non-uniform distribution of values of t can concentrate the search around the reference point which follows the best solution).

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); the grants IRI-9322400, IRI-9725424 from the National

[9]If $\mathcal{F}$ is convex, the whole boundary of $\mathcal{F}$ is explored.

Science Foundation, and the ESPRIT Project 20288 Cooperation Research in Information Technology (CRIT-2): "Evolutionary Real-time Optimisation System for Ecological Power Control". The authors wish to thank anonymous reviewers for their useful comments.

References

Bäck, T. and M. Schütz, (1996). Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In Z.W. Ras and M. Michalewicz (Eds), *Proceedings of the 9th International Symposium on Methodologies of Intelligent Systems*, pp.158–167, Springer-Verlag, New York, NY.

Bean, J. C. and A. B. Hadj-Alouane (1992). A dual genetic algorithm for bounded integer programs. Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan.

Bilchev, G. and I. Parmee (1995). Ant colony search vs. genetic algorithms. Technical report, Plymouth Engineering Design Centre, University of Plymouth.

Davis, L. (1997). Private communication.

Dorigo M., V. Maniezzo, and A. Colorni (1996). The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26 (1), pp.29–41.

Eiben, A., P.-E. Raue, and Z. Ruttkay (1994). Genetic algorithms with multi-parent recombination. In Y. Davidor, H.-P. Schwefel, and R. Manner (Eds), *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, Number 866 in LNCS, pp. 78–87. Springer-Verlag, New York, NY.

Gregory, J. (1995). Nonlinear Programming FAQ, Usenet sci.answers. Available at ftp://rtfm.mit.edu/pub/usenet/sci.answers/nonlinear-programming-faq.

Hadj-Alouane, A. B. and J. C. Bean (1992). A genetic algorithm for the multiple-choice integer program. Technical Report TR 92-50, Department of Industrial and Operations Engineering, The University of Michigan.

Himmelblau, D. (1992). *Applied Nonlinear Programming*. McGraw-Hill, New York, NY.

Homaifar, A., S. H.-Y. Lai, and X. Qi (1994). Constrained optimization via genetic algorithms. *Simulation 62*(4), 242–254.

Joines, J. and C. Houck (1994). On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano (Eds), *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pp. 579–584. IEEE Press, Piscataway, NJ.

Kozieł, S. (1997). Evolutionary algorithms in constrained numerical optimization problems on convex spaces. Electronics and Telecommunications Quarterly, 43 (1), pp. 5–18.

Kozieł, S. and Z. Michalewicz, (1998). A decoder-based evolutionary algorithm for constrained parameter optimization problems. In *Proceedings of the* 5^{th} *Conference on Parallel Problems Solving from Nature*, T. Bäck, A.E. Eiben, M. Schoenauer, and H.-P. Schwefel (Eds), Springer Verlag, New York, NY.

Leriche, R. G., C. Knopf-Lenoir, and R. T. Haftka (1995). A segragated genetic algorithm for constrained structural optimization. In L. J. Eshelman (Ed.), *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pp. 558–565, Morgan Kaufmann, San Mateo, CA.

Maa, C. and M. Shanblatt (1992). A two-phase optimization neural network. *IEEE Transactions on Neural Networks 3*(6), 1003–1009.

Michalewicz, Z. (1995). Genetic algorithms, numerical optimization and constraints. In L. J. Eshelman (Ed.), *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pp. 151–158. Morgan Kaufmann, San Mateo, CA.

Michalewicz, Z. (1996). *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, New York, NY, 3rd edition.

Michalewicz, Z. and N. Attia (1994). Evolutionary optimization of constrained problems. In *Proceedings of the* 3^{rd} *Annual Conference on Evolutionary Programming*, A.V. Sebald and L.J. Fogel (Eds), pp. 98–108. World Scientific, River Edge, NJ.

Michalewicz, Z., D. Dasgupta, R.G. Le Riche, and M. Schoenauer (1996). Evolutionary algorithms for constrained engineering problems. Computers & Industrial Engineering Journal, Vol.30, No.4, September 1996, pp.851–870.

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, San Mateo, CA.

Michalewicz, Z., T. Logan, and S. Swaminathan (1994). Evolutionary operators for continuous convex parameter spaces. In *Proceedings of the* 3^{rd} *Annual Conference on Evolutionary Programming*, A.V. Sebald and L.J. Fogel (Eds), pp. 84–97. World Scientific, River Edge, NJ.

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, Piscataway, NJ.

Michalewicz, Z., G. Nazhiyath, and M. Michalewicz (1996). A note on usefulness of geometrical crossover for numerical optimization problems. In L. J. Fogel, P. J. Angeline and T. Bäck (Eds), *Proceedings of the* 5^{th} *Annual Conference on Evolutionary Programming*, MIT Press, Cambridge, MA, 1996, pp.305–312.

Z. Michalewicz and M. Schoenauer (1996). Evolutionary computation for constrained parameter optimization problems. Evolutionary Computation, Vol.4, No.1, pp.1–32.

Myung, H., J.-H. Kim, and D. Fogel (1995). Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel (Eds), *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming*, pp. 449–463. MIT Press, Cambridge, MA.

Palmer, C.C. and A. Kershenbaum (1994). Representing trees in genetic algorithms. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano (Eds), *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pp.379–384,. IEEE Press, Piscataway, NJ.

Paredis, J. (1994). Coevolutionary constraint satisfaction. In Y. Davidor, H.-P. Schwefel, and R. Manner (Eds), *Proceedings of the* 3^{rd} *Conference on Parallel Problems Solving from Nature*, pp. 46–55. Springer Verlag, New York, NY.

Parmee, I. and G. Purchase (1994). The development of directed genetic search technique for heavily constrained design spaces. In *Proceedings of the Conference on Adaptive Computing in Engineering Design and Control*, I. Parmee (Ed.), pp. 97–102, University of Plymouth.

Powell, D. and M. M. Skolnick (1993). Using genetic algorithms in engineering design optimization with non-linear constraints. In S. Forrest (Ed.), *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pp. 424–430. Morgan Kaufmann, San Mateo, CA.

Reynolds, R. (1994). An introduction to cultural algorithms. In *Proceedings of the* 3^{rd} *Annual Conference on Evolutionary Programming*, A.V. Sebald and L.J. Fogel (Eds), pp. 131–139. World Scientific, River Edge, NJ.

Reynolds, R., Z. Michalewicz, and M. Cavaretta (1995). Using cultural algorithms for constraint handling in Genocop. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel (Eds), *Proceedings of the* 4^{th} *Annual Conference on Evolutionary Programming*, pp. 298–305. MIT Press, Cambridge, MA.

Richardson, J. T., M. R. Palmer, G. Liepins, and M. Hilliard (1989). Some guidelines for genetic algorithms with penalty functions. In J. D. Schaffer (Ed.), *Proceedings of the* 3^{rd} *International Conference on Genetic Algorithms*, pp. 191–197. Morgan Kaufmann, San Mateo, CA.

Schaffer, D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms.* Laurence Erlbaum Associates. Hillsdale, NJ, 1985.

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*, Springer Verlag, New York, NY, pp.245–254.

Schoenauer, M. and Z. Michalewicz (1997). Boundary operators for constrained parameter optimization problems. In T. Bäck (Ed.), *Proceedings of the* 7^{th} *International Conference on Genetic Algorithms*, pp.320–329. Morgan Kaufmann, San Mateo, CA.

Schoenauer, M. and S. Xanthakis (1993). Constrained GA optimization. In S. Forrest (Ed.), *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pp. 573–580. Morgan Kaufmann, San Mateo, CA.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.* John Wiley & Sons, New York, NY, 1995 – 2^{nd} edition.

Smith, A. and D. Tate (1993). Genetic optimization using a penalty function. In S. Forrest (Ed.), *Proceedings of the* 5^{th} *International Conference on Genetic Algorithms*, pp. 499–503. Morgan Kaufmann, San Mateo, CA.

Surry, P., N. Radcliffe, and I. Boyd (1995). A multi-objective approach to constrained optimization of gas supply networks. In T. Fogarty (Ed.), *Proceedings of the AISB-95 Workshop on Evolutionary Computing*, Volume 993, pp. 166–180. Springer Verlag, New York, NY.

Waagen, D., P. Diercks, and J. McDonnell (1992). The stochastic direction set algorithm: A hybrid technique for finding function extrema. In D. B. Fogel and W. Atmar (Eds), *Proceedings of the*

1st Annual Conference on Evolutionary Programming, pp. 35–42. Evolutionary Programming Society, San Diego, CA.

Wodrich, M. and G. Bilchev (1997). Cooperative distributed search: the ant's way. Control & Cybernetics, 26 (3), pp.413–446.

Appendix

The appendix provides the description of 11 test cases, $G1$–$G11$ for constrained parameter optimization problems.

• Minimize

$$G1(\vec{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i,$$

subject to the following constraints:

$$\begin{array}{lll} 2x_1 + 2x_2 + x_{10} + x_{11} \leq 10, & 2x_1 + 2x_3 + x_{10} + x_{12} \leq 10, & 2x_2 + 2x_3 + x_{11} + x_{12} \leq 10, \\ -8x_1 + x_{10} \leq 0, & -8x_2 + x_{11} \leq 0, & -8x_3 + x_{12} \leq 0, \\ -2x_4 - x_5 + x_{10} \leq 0, & -2x_6 - x_7 + x_{11} \leq 0, & -2x_8 - x_9 + x_{12} \leq 0, \end{array}$$

and bounds $0 \leq x_i \leq 1$, $i = 1, \ldots, 9$, $0 \leq x_i \leq 100$, $i = 10, 11, 12$, $0 \leq x_{13} \leq 1$.
The problem has 13 variables and 9 linear constraints; the function $G1$ is quadratic with its global minimum at

$$\vec{x}^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1),$$

where $G1(\vec{x}^*) = -15$. Six (out of nine) constraints are active at the global optimum (all except the following three: $-8x_1 + x_{10} \leq 0$, $-8x_2 + x_{11} \leq 0$, $-8x_3 + x_{12} \leq 0$).

• Maximize

$$G2(\vec{x}) = \left|\frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}\right|,$$

subject to

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

Function $G2$ is nonlinear and its global maximum is unknown. For $n = 20$, solutions $\vec{x}$ for which $G2(\vec{x}) = 0.8036$ were reported.

• Maximize

$$G3(\vec{x}) = (\sqrt{n})^n \cdot \prod_{i=1}^{n} x_i,$$

where

$$\sum_{i=1}^{n} x_i^2 = 1 \quad \text{and} \quad 0 \leq x_i \leq 1 \ \text{ for } \ 1 \leq i \leq n.$$

The function $G3$ has a global solution at $(x_1, \ldots, x_n) = (\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}})$ and the value of the function in this point is 1.

• Minimize

$$G4(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,$$

subject to three double inequalities:

$$\begin{array}{l} 0 \leq 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 \leq 92 \\ 90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leq 110 \\ 20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25, \end{array}$$

and bounds:

$$78 \leq x_1 \leq 102,\ 33 \leq x_2 \leq 45,\ 27 \leq x_i \leq 45 \text{ for } i = 3, 4, 5.$$

The optimum solution is

$$\vec{x}^* = (78.0, 33.0, 29.995, 45.0, 36.776),$$

with $G4(\vec{x}^*) = -30665.5$. Two constraints (upper bound of the first inequality and the lower bound of the third inequality) are active at the optimum.

• Minimize

$$G5(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + 0.000002/3x_2^3$$

subject to

$$x_4 - x_3 + 0.55 \geq 0,\ x_3 - x_4 + 0.55 \geq 0,$$
$$1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$
$$0 \leq x_i \leq 1200,\ i = 1, 2, \text{ and } -0.55 \leq x_i \leq 0.55,\ i = 3, 4.$$

The best known solution is

$$\vec{x}^* = (679.9453, 1026.067, 0.1188764, -0.3962336),$$

and $G5(\vec{x}^*) = 5126.4981$.

• Minimize

$$G6(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3,$$

subject to nonlinear constraints:

$$(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0,$$
$$-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0,$$

and bounds:

$$13 \leq x_1 \leq 100 \text{ and } 0 \leq x_2 \leq 100.$$

The known global solution is $\vec{x}^* = (14.095, 0.84296)$, and $G6(\vec{x}^*) = -6961.81381$. Clearly, both constraints are active at the optimum.

• Minimize

$$G7(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45,$$

subject to the following constraints:

$$105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0, \qquad -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0,$$
$$-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0, \qquad -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0,$$
$$8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0, \qquad -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0,$$
$$3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0, \qquad -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0,$$

and bounds

$$-10.0 \leq x_i \leq 10.0, \quad i = 1, \ldots, 10.$$

The problem has 3 linear and 5 nonlinear constraints; the function $G7$ is quadratic and has its global minimum at

$$\vec{x}^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, \\ 1.430574, 1.321644, 9.828726, 8.280092, 8.375927),$$

where $G7(\vec{x}^*) = 24.3062091$. Six (out of eight) constraints are active at the global optimum (all except the last two).

• Maximize

$$G8(\vec{x}) = \frac{\sin^3(2\pi x_1) \cdot \sin(2\pi x_2)}{x_1^3 \cdot (x_1 + x_2)},$$

subject to the following constraints:

$$x_1^2 - x_2 + 1 \leq 0,$$
$$1 - x_1 + (x_2 - 4)^2 \leq 0$$

and bounds:

$$0 \leq x_1 \leq 10 \text{ and } 0 \leq x_2 \leq 10.$$

Function $G8$ has many local optima, the highest peaks are located along the x axis (e.g., $G8(0.00015, 0.0225) > 1540$). In the feasible region, however, $G8$ has two maxima of almost equal fitness of value of 0.1.

• Minimize

$$G9(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,$$

subject to the following constraint:

$$127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0, \quad 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0,$$
$$196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0, \quad -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0$$

and bounds:

$$-10.0 \leq x_i \leq 10.0, \; i = 1, \ldots, 7.$$

The problem has 4 nonlinear constraints; the function $G9$ is nonlinear and has its global minimum at

$$\vec{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227),$$

where $G9(\vec{x}^*) = 680.6300573$. Two (out of four) constraints are active at the global optimum (the first and the last one).

• Minimize

$$G10(\vec{x}) = x_1 + x_2 + x_3,$$

subject to the following constraints:

$$1 - 0.0025(x_4 + x_6) \geq 0, \quad 1 - 0.0025(x_5 + x_7 - x_4) \geq 0,$$
$$1 - 0.01(x_8 - x_5) \geq 0, \quad x_1 x_6 - 833.33252 x_4 - 100 x_1 + 83333.333 \geq 0,$$
$$x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geq 0, \quad x_3 x_8 - 1250000 - x_3 x_5 + 2500 x_5 \geq 0,$$

and bounds

$$100 \leq x_1 \leq 10000, \quad 1000 \leq x_i \leq 10000, \, i = 2, 3, \quad 10 \leq x_i \leq 1000, \, i = 4, \ldots, 8.$$

The problem has 3 linear and 3 nonlinear constraints; the function $G10$ is linear and has its global minimum at

$$\vec{x}^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979),$$

where $G10(\vec{x}^*) = 7049.330923$. All six constraints are active at the global optimum.

- Minimize

$$G11(\vec{x}) = x_1^2 + (x_2 - 1)^2,$$

subject to a nonlinear constraint:

$$x_2 - x_1^2 = 0,$$

and bounds:

$$-1 \leq x_i \leq 1, \, i = 1, 2.$$

The known global solutions are $\vec{x}^* = (\pm 0.70711, 0.5)$, and $G11(\vec{x}^*) = 0.75000455$.