Evolutionary Computation at the Edge of Feasibility

Marc Schoenauer[1] and Zbigniew Michalewicz[2]

[1]CMAP – URA CNRS 756
Ecole Polytechnique
Palaiseau 91128, France
marc.schoenauer@polytechnique.fr.

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

Abstract. Numerical optimization problems enjoy a significant popularity in evolutionary computation community; all major evolutionary techniques use such problems for various tests and experiments. However, many of these techniques (as well as other, classical optimization methods) encounter difficulties in solving some real-world problems which include non-trivial constraints. This paper discusses a new development which is based on the observation that very often the global solution lies on the boundary of the feasible region. Thus, for many constrained numerical optimization problems it might be beneficial to limit the search to that boundary, using problem-specific operators. Two test cases illustrate this approach: specific operators are designed from the simple analytical expression of the constraints. Some possible generalizations to larger classes of constraints are discussed as well.

1 Introduction

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

However, many of these techniques have difficulties in solving constrained numerical optimization problems. Several different search operators have been investigated; several different constraint handling techniques have been experimented with. For many test cases, the results of experiments were far from being satisfactory. It seems that one of the main reasons behind this failure was the inability of evolutionary systems to precisely search the boundary area between feasible and infeasible regions of the search space; in the case of optimization problems with active constraints, such ability is essential.

Some other heuristic methods recognized the need for searching areas close to the boundary of the feasible region. For example, one of the most recently developed approach for constrained optimization is strategic oscillation. Strategic oscillation was originally proposed in accompaniment with the strategy of scatter search [3], and more recently has been applied to a variety of problem settings in combinatorial and nonlinear optimization [4].

It seems that the evolutionary computation techniques have a huge potential in incorporating specialized operators which search the boundary of feasible and infeasible regions in an efficient way. In this paper we discuss some possible operators for such a search and illustrate this approach on a few test cases.

The paper is organized as follows. Section 2 briefly surveys constraint-handling techniques for numerical optimization problems, which have emerged in evolutionary computation over the years. Section 3 introduces a new approach for numerical constrained optimization, based on the idea of searching only the boundary of the feasible search space. This approach is illustrated in section 4 on two difficult problems on which it allows significant improvements over best known results. Section 5 proposes different methods to design operators searching a given surface of $\mathbb{R}^n$ in the general case. These methods are illustrated and discussed on both test cases of section 4.

2 Evolutionary Constraint-Handling Methods

Let us consider the following constrained numerical optimization problem:

$$\text{Find } x \in \mathcal{S} \subset \mathbb{R}^n \text{ such that } \begin{cases} f(x) = \min\{f(y);\ y \in \mathcal{S}\}, & (1) \\ g_i(x) \leq 0, for\ \ i = 1, \dots, q, & (2) \\ g_i(x) = 0, for\ \ i = q+1, \dots, m. & (3) \end{cases}$$

where f and g_i are real-valued functions on $\mathcal{S}$. The set of feasible points (i.e., points satisfying the constraints (2) and (3)) is denoted $\mathcal{F}$.

Restricting the search to the feasible region seems an elegant way to treat constrained problems: in [12], the algorithm maintains feasibility of all linear constraints using a set of closed operators, which convert a feasible solution (feasible in terms of linear constraints only) into another feasible solution.

However, for nonlinear constraints this ideal situation is generally out of reach, and during the last few years, several methods have been proposed for handling constraints by evolutionary algorithms. Most of them are based on the concept of penalty functions: Infeasible individuals are penalized by adding to their fitness $f(x)$ a penalty term $penalty(x)$, which is zero if no violation occurs, and is positive otherwise.

The most severe penalty method is death penalty ($penalty(x) = +\infty$) as in [1]. A usual form of penalty functions is $penalty(x) = \sum_0^q \alpha_i \max(0, g_i(x)) + \sum_{q+1}^m \alpha_i |g_i|$, where parameters α_i are either static parameters (see for instance [5] for a method to determine α_i), dynamically adjusted (as in [6]) or adaptive (two methods are proposed in [2, 19]). Many variations on the penalty approach have been implemented: in [16], an additional penalty term is added to ensure that any feasible point has worse fitness than any infeasible one; in [8], two populations are evolved with two different penalty coefficients. The idea of using two separate populations is also used in Genocop III [13]: a set of reference feasible points is maintained and used to repair infeasible points.

For an experimental comparison of some of these methods and some others on a few test cases, see [10, 15]. However, highly nonlinear constraints still present difficulties for evolutionary algorithms, as penalty parameters or strategies are then difficult to adjust.

3 Searching the boundary of the feasible region

It is a common situation for many constrained optimization problems that some constraints are active at the target global optimum. This optimum thus lies on the boundary of the feasible space. One the other hand, it is commonly acknowledged that restricting the size of the search space in evolutionary algorithms (as in most search algorithms) is generally beneficial. Hence, it seems natural in the context of constrained optimization to restrict the search of the solution to the boundary of the feasible part of the space.

We suppose in the rest of the paper that we are searching on a Riemanian surface S of dimension $n-1$ in the space $\mathbb{R}^n$. This surface is supposed to be regular, i.e., the gradient vector, orthogonal to the surface, is defined almost everywhere. The Euclidean measure of $\mathbb{R}^n$ thus classically induces a distance on that surface.

The basic components of an evolutionary search are the following.

An initialization procedure: This procedure must sample surface S as uniformly as possible according to the distance at hand. Though conceptually simple, this procedure can be quite hard to design: in many real-world engineering problems, the main difficulty can be to find just *one* feasible point.

Evolution operators on the surface: The first necessary conditions for the evolution operators is to be *closed*, i.e., transforming point(s) of the surface into point(s) of the surface. Moreover, these operators should (as much as possible) respect some empirically derived properties [17, 11]:

– recombination should be able to generate all points "between" the parents;
– mutation should be *ergodic*, having non-zero probability to reach any point within a finite number of application, and should respect the principle of *strong causality* [18], i.e., small mutations must result in small changes in the fitness function.

These ideas will be validated on two test cases in next section: specific evolutionary algorithms searching the boundary of the feasible regions will be designed.

4 Two test cases

4.1 Problem on an hyperboloid

An interesting constrained numerical optimization test case emerged recently; the problem [7] is to maximize $f(\mathbf{x}) = \left|\frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}}\right|$, with constraints $\prod_{i=1}^{n} x_i \geq 0.75$, $\sum_{i=1}^{n} x_i \leq 7.5n$, and $0 \leq x_i \leq 10$ for $1 \leq i \leq n$.

Function f is nonlinear and its global maximum is unknown, lying somewhere near the origin. The problem has one nonlinear constraint and one linear constraint; the latter one is inactive around the origin and will be forgotten in the following

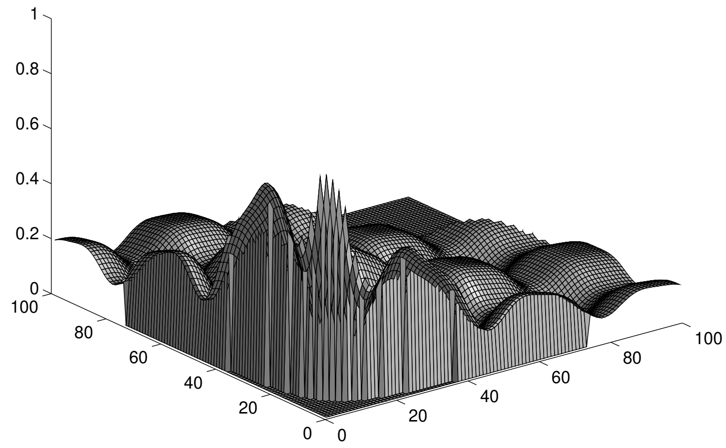

Fig. 1. The graph of function f for $n = 2$. Infeasible solutions were assigned value zero

Some potential difficulties of solving this test case are illustrated on Figure 1: infeasible points were assigned a value of zero. The F-surface is defined by the equation $\Pi x_i = 0.75$. It is a difficult problem, on which no standard method (be it deterministic or evolutionary) gave satisfactory results. In [7], a parallel GA with 12bit binary encoding using a modified Fiacco-McCormick constraint penalty function gets values like 0.76 after 20,000 evaluations (for $n = 20$).

This test case was the first one on which the idea of searching only the boundary was used [14]: due to the simple analytical formulation of the constraint, *ad hoc* specific initialization procedure and operators could be designed.

- **Initialization**: Randomly choose a positive variable for x_i, and use its inverse as a variable for x_{i+1}. The last variable is either 0.75 (when n is odd), or is multiplied by 0.75 (if n is even), so that the point lies on the F-surface.
- **Crossover**: The *geometrical crossover* is defined by

  $$(x_i)(y_i) \rightarrow (x_i^{\alpha} y_i^{1-\alpha}), \text{ with } \alpha \text{ randomly chosen in } [0, 1]$$

  Figure 2 in section 5.2 illustrates the possible offspring from two parents on the F-surface for all values of α.
- **Mutation**: Pick two variables randomly, multiply one by a random factor q and the other by $\frac{1}{q}$ (restrict q to respect the bounds on the variables).

The simple evolutionary algorithm described above gave outstanding results in terms of quality of the solution. For the case $n = 20$ the system reached the value of 0.80 in less than 4,000 generations (with population size of 30, probability of crossover $p_c = 1.0$, and probability of mutation $p_m = 0.06$) in all runs. The best value found (namely 0.803553) was better than the best values of any method discussed earlier, whereas the worst value found was 0.802964. Similarly, for $n = 50$, all results (in 30,000 generations) were better than 0.83 (with the best of 0.8331937).

It was interesting to note the importance of geometrical crossover. With fixed population size (kept constant at 30), the higher values of probability of crossover p_c, the better results of the system were observed. Similarly, the best mutation rates were relatively low ($p_m \approx 0.06$).

4.2 Constrained problem on the sphere

The main interest of the sphere as a surface in $\mathbb{R}^n$ come from both its simple analytical expression and its nice symmetrical properties. Hence different methods to design evolution operators on the sphere can be used (see sections 5.3 and 5.2). However, we shall first present results of an *ad hoc* evolutionary algorithm that was designed in [14], based on the simple analytical formulation of the constraint.

The test problem for this case is to maximize $f(\mathbf{x}) = (\sqrt{n})^n \cdot \prod_{i=1}^n x_i$, with constraints $\sum_{i=1}^n x_i = 1$ and $0 \le x_i \le 1$ for $1 \le i \le n$.
The function 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. The evolutionary algorithm uses the following components:

- **Initialization**: Randomly generate n variables y_i, calculate $s = \sum_{i=1}^n y_i^2$, and initialize an individual (x_i) by $x = y_i/s$ for $i \in [1, n]$.
- **Crossover**: The *sphere crossover* produces one offspring (z_i) from two parents (x_i) and (y_i) by:

$$z_i = \sqrt{\alpha x_i^2 + (1-\alpha) y_i^2})\ i \in [1, n], \text{ with } \alpha \text{ randomly chosen in } [0, 1]$$

- **Mutation**: Similarly, the problem-specific mutation transforms (x_i) by selecting two indices $i \neq j$ and a random number p in $\langle 0, 1 \rangle$, and setting:

$$x_i \rightarrow p \cdot x_i \text{ and } x_j \rightarrow q \cdot x_j, \text{where } q = \sqrt{(\frac{x_i}{x_j})^2 (1 - p^2) + 1}.$$

The simple evolutionary algorithm described above gave very good results. For the case $n = 20$ the system reached the value of 0.99 in less than 6,000 generations (with population size of 30, probability of crossover $p_c = 1.0$, and probability of mutation $p_m = 0.06$) in all runs. The best value found in 10,000 generations was 0.999866.

5 Surface-searching evolutionary operators

In this section, we propose possible approaches to the design of evolutionary operators on a general surface of dimension $n-1$ given in analytical form.

5.1 Curve-based operators

The first method to design suitable operators is based on *curves* drawn on the surface.

Crossover operators:
From curves joining two different points, one can derive a crossover operator by choosing as offspring one (two) point(s) on that curve. Minimal-length curves (termed *geodesical curves*) seem a priori a good choice: their existence is guaranteed, locally on any regular surface from standard Cauchy-Lipschitz theorem, and globally (i.e. joining any pair of points of the surface) if the surface is *geodesically complete*[1] (Hopf-Rinov theorem) [9]. Moreover, in the linear case, the geodesical curve between two points is the line segment between them, and the curve-based operator is nothing but the standard linear recombination operator [11]. But recent experiments suggest that other paths between both parents can be used successfully [14]: the minimal length requirement does not seem mandatory.

Mutation operators:
From a beam of curves starting from one point, one can derive a mutation operator by first choosing randomly one curve in the beam, then choosing a point on the chosen curve. A desirable property of the beam of curves related to the ergodicity of the resulting mutation operator is that a large enough neighborhood of the starting point is covered by such set of curves: the local geodesical curves defined from the parent point and one tangent direction are such sets, defined almost everywhere on regular surfaces [9]. On the sphere, for instance, the geodesical curves from a pole are the meridians, which in that case cover not only a whole neighborhood of the parent point, but the whole surface. Furthermore, a tight control of the distance between parents and offspring allows for a simple implementation of adaptive mutation respecting the strong causality principle.

Unfortunately, in the general case, even with analytical definitions of the constraints, the derivation of the geodesical curves (or the exponential curves beam) is not tractable: it involves heavy symbolic computations, plus the numerical solution of many local second-order systems of differential equations. Moreover, unavoidable numerical errors would probably move the solutions off the desired curves.

However, in some situations, the ideas presented above can be implemented, and evolutionary search on the boundary can be successfully performed without the need for exact geodesical curves. The geometrical and spherical crossovers

[1] A surface is geodesically complete if no geodesical curve encounters a dead-end.

presented in section 3 are particular examples of such curves. Next subsections will also present particular cases where such curves are available.

5.2 Parametric representation of the surface

From a parametric representation of a surface, it is easy to design a complete evolutionary algorithm on that surface. Suppose the surface S (of dimension $n-1$) is defined by $x_i = s_i(t_1, \dots t_{n-1}), \ \ t_i \in [a_i, b_i]$, for $i = 1, \dots, n$, where the functions s_i are regular functions from R^{n-1} into R. In the following, we denote this relation $(x_i) = S[(t_i)]$. The following elements can be designed:

- **Initialization**: A random choice of a point on S amounts to the choice of the $n-1$ values of the parameters $t_1, \dots t_{n-1}$, uniformly on $\Pi[a_i, b_i]$.
- **Crossover**: The crossover can be defined by:
$$S[(t_i)], S[(u_i)] \rightarrow S[(\alpha t_i + (1-\alpha)u_i)]$$
for some α randomly chosen in $[0, 1]$.
- **Mutation**: Similarly, the mutation can be given by
$$S[(t_i)] \rightarrow S[(t_i + N(0, \sigma_i))]$$
where $N(0, \sigma)$ denotes the normal distribution with zero mean and σ variance. The parameters σ_i can be either user-supplied (eventually dynamically) or adaptive, i.e., encoded in the individual, as in evolution strategies.

In that context, the choice of a specific parametric representation fully determines the operators. Further subsection presents the common situation in which many different parameterization of surface S are available, for both cases of section 4.

Parametric operators on the hyperboloid. In the case of the hyperboloid (section 4.1), surface S is defined by the equation $\prod x_i = 0.75$. Any $n-1$ variables can be chosen as parameters, the last one is then defined by the equation of the surface itself. There are n different systems of such parametric representation, for the n different choices of $i_0 = 1, \dots n$:
$$x_{i_0} = 0.75/\Pi_{i \neq i_0} x_i$$
Figure 2 presents three possible parametric crossovers (as defined above) in the case $n = 3$, termed C_i when variable i is taken as unknown, together with the geometrical crossover introduced in section 4.1 termed G. It is clear from that figure that the possible locations for the offspring depends on both the chosen parameterization and the position of the parents on the surface.

Parametric operators on the sphere. Similarly, an evolutionary algorithm can be defined as described earlier in this section from any standard spherical parametric representation (which are the straightforward extension of the usual spherical coordinates of $\mathbb{R}^3$). Here, there are $n!$ valid parametric representations for the sphere, corresponding to the permutations among coordinates. Figure 3-a gives an example of three curves defining these operators on the sphere in the case $n = 3$ (denoted P_1, P_2, and P_3), together with the geodesical curve (denoted G) and the curve corresponding to the spherical crossover presented in section 4.2 (denoted Sp).

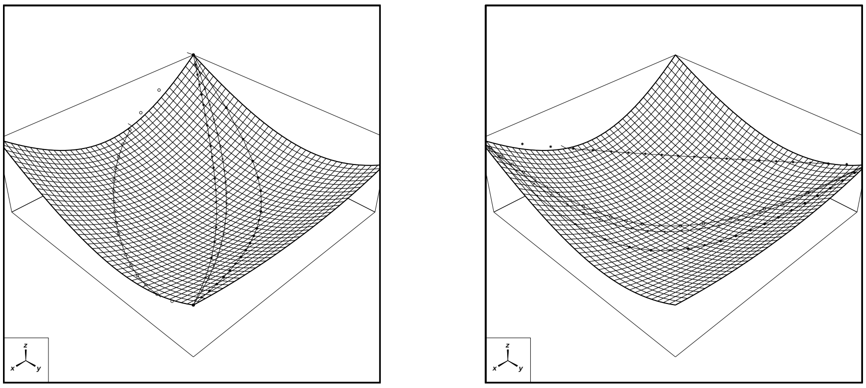

Fig. 2. Parametric crossovers for the hyperboloid problem, for different parametric representations and different parents. (a) From left to right, C_2, C_1, G, C_3 (b) From top to bottom, C_1, C_2, G, C_3

5.3 Plane-based operators

Another general method to design curves, and hence operators to evolve on a surface is to use the intersection of that surface with 2-dimensional planes.

Consider two points A and B belonging to S, and one vector $\mathbf{v}$ which are not collinear to $\mathbf{AB}$ and which is not orthogonal to the gradient vector at point A. Hence the plane defined by $(A, \mathbf{AB}, \mathbf{v})$ intersects the surface around A, defining a curve on S. If this curve is connected, a crossover operator can be designed as described in section 5.1. But this procedure can fail if the intersection is not connected (as for the "horse-shoe sausage" of R^3 with A and B at both end).

Similarly, the mutation operator can be designed by choosing the gradient at point A instead of vector $\mathbf{AB}$ above, with a prescribed distance off point A.

Examples of plane-based operators will now be given in the simple case where S is a sphere.

Plane-based operators on the sphere. In the case of the sphere, the derivation of curves joining two points by intersecting the sphere with a plane is straightforward calculation. Moreover, the geodesical curves are a particular case of such plane-based curves, corresponding to the case where the chosen plane goes through the center of the sphere.

Figure 3-b shows some plane-based curves for different values of the angle between the plane and the gradient of the surface. The angles were such that the resulting curves approximately match the curves of Figure 3-a, obtained by parametric operators (section 5.2) and by the *spherical* crossover of section 4.2. Obviously, in that particular case, plane-based operators are more general operators: particular plane-based crossovers can give almost the same resulting offspring than any other crossover presented here.

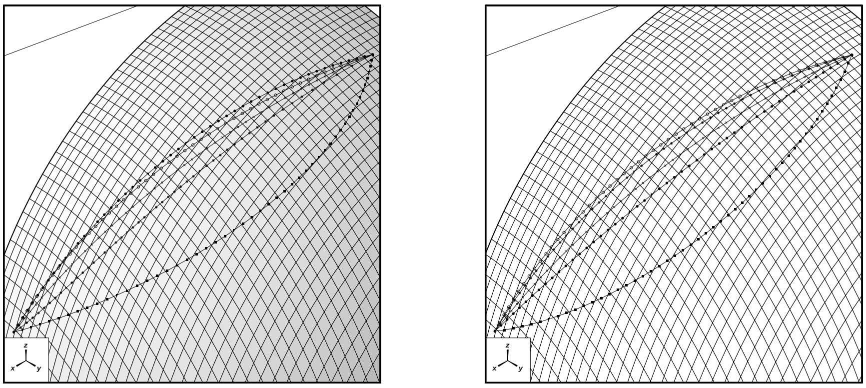

Fig. 3. Crossover operators on the sphere. (a) From top to bottom, P_1, P_2, G, Sp, P_3. (b) From top to bottom, angle between the plane and the gradient is $0, 15, 20, -15, -57$ (in degrees).

6 Conclusions

Using the geodesical curves constitute a theoretical method for constructing specific operators on a surface, but raises technical difficulties. The parametric operators represent an easy way when available with the inconvenience that these operators do depend a lot on the chosen parameterization (see Figure 2).

The plane operators provide another possibility, but they require more complex calculations, and possibly they may lead to pathological solutions (e.g., non-connected parts of curves). The case of the sphere illustrates all above-mentioned types of operators: plane-based operators end up in more general operators, sweeping a large region around the parents, thus avoiding the bias due to the choice of a single curve. Further comparative experiments are needed, on both test cases presented here as well as on other more complex surfaces.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant IRI-9322400. Many thanks to C. Margerin (CMA-Ecole Polytechnique) for his helpful views on Riemanian geometry

References

1. T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* 4^{th} *International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1991.
2. J. C. Bean and A. B. Hadj-Alouane. A dual genetic algorithm for bounded integer programs. Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

3. F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.
4. F. Glover and G. Kochenberger. Critical event tabu search for multidimensional knapsack problems. In *Proceedings of the International Conference on Metaheuristics for Optimization*, pages 113–133. Kluwer Publishing, 1995.
5. A. Homaifar, S. H.-Y. Lai, and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.
6. J.A. Joines and C.R. Houck. 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, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 579–584. IEEE Press, 1994.
7. A. Keane. Genetic Algorithms Digest, May 19 1994. V8n16.
8. R. G. Leriche, C. Knopf-Lenoir, and R. T. Haftka. A segragated genetic algorithm for constrained structural optimization. In L. J. Eshelman, editor, *Proceedings of the 6^{th} International Conference on Genetic Algorithms*, pages 558–565, 1995.
9. C. Margerin. Une introduction à la géométrie. Course at ENSTA – Palaiseau – France, 1990.
10. Z. Michalewicz. Genetic algorithms, numerical optimization and constraints. In L. J. Eshelman, editor, *Proceedings of the 6^{th} International Conference on Genetic Algorithms*, pages 151–158. Morgan Kaufmann, 1995.
11. Z. Michalewicz. *Genetic Algorithms+Data Structures=Evolution Programs*. Springer Verlag, New-York, 1996. 3rd edition.
12. Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. In *Proceedings of the 3^{rd} Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, 1994.
13. Z. Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In D. B. Fogel, editor, *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pages 647–651. IEEE Press, 1995.
14. Z. Michalewicz, G. Nazhiyath, and M. Michalewicz. A note on usefulness of geometrical crossover for numerical optimization problems. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Proceedings of the 5^{th} Annual Conference on Evolutionary Programming*, 1996.
15. Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1), Spring 1996.
16. D. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In S. Forrest, editor, *Proceedings of the 5^{th} International Conference on Genetic Algorithms*, pages 424–430. Morgan Kaufmann, 1993.
17. N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–20, 1991.
18. I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, 1973.
19. A. Smith and D. Tate. Genetic optimization using a penalty function. In S. Forrest, editor, *Proceedings of the 5^{th} International Conference on Genetic Algorithms*, pages 499–503. Morgan Kaufmann, 1993.

This article was processed using the LaTeX macro package with LLNCS style