Problem-Independent Handling of Constraints by Use of Metric Penalty Functions

Frank Hoffmeister
EUnet Deutschland GmbH
Emil-Figge-Straße 80
D-44227 Dortmund

Joachim Sprave
University of Dortmund
Department of Computer Science
D-44221 Dortmund

Abstract

Unlike scientific test functions, real world problems are in general *constrained* optimization problems. Furthermore, due to undetermined first and second order derivatives only direct optimization methods can be used to attack these problems. Then, in most cases, constraints are dealt with by use of so-called penalty functions. An alternate approach as used by Rechenberg [1] and Schwefel [2] simply discards infeasible test points while generating a new one.

This paper presents a synthesis of both approaches avoiding the design of problem-dependent surrogate functions in the former case as well as the excessive waste of computational resources and implicit information contained in infeasible test points in the latter case. It is applicable to *any* direct optimization method with just minor modifications. Its feasibility is demonstrated by its integration into an Evolution Strategy (ES), a special kind of Evolutionary Algorithm (EA), which — due to its internal model of the problem (internal strategy parameters) — has the capacity to profit from the implicit information provided by infeasible test points.

Introduction

So-called *real world* optimization problems often not only have a high complexity with respect to the number of variables (dimension of the problem) but also have a possibly large number of constraints to be satisfied for a feasible solution. In many cases, satisfaction of the constraints is a bigger problem than finding an optimal value for the objective function.

Definition 1 (Constrained Optimization)
Minimize $f(\vec{x})$ subject to:

$$g_i(\vec{x}) \geq 0 \quad ; \forall i \in \{1, \ldots, m\} \tag{1}$$

$$h_j(\vec{x}) = 0 \quad ; \forall j \in \{1, \ldots, l\} \tag{2}$$

$$\begin{array}{llllll} \vec{x} & = & (x_1, \ldots, x_n) & \in & \mathbb{R}^n & \\ f & : & D & \rightarrow & \mathbb{R} & ; \quad D \subseteq \mathbb{R}^n \\ g_i & : & \mathbb{R}^n & \rightarrow & \mathbb{R} & \\ h_j & : & \mathbb{R}^n & \rightarrow & \mathbb{R} & \end{array}$$

where $g_i(\vec{x})$ is called an *unequality constraint* and $h_j(\vec{x})$ is called an *equality constraint*. A test point $\vec{x}$ satisfying (1) and (2) is called *feasible*, if not *infeasible*. The set $M \subseteq D$ satisfying (1) and (2) is called the *feasible domain.*

In general, due to undetermined derivatives direct optimization methods are the only ones which can be applied to these problems. The basic principle of a direct optimization method follows the working scheme sketched below:

$$\vec{x}'_t = \gamma(\vec{x}_t) = \vec{x}_t + \Delta\vec{x}_t \tag{3}$$

$$\vec{x}_{t+1} = \begin{cases} \vec{x}'_t & ; \text{if } F(\vec{x}'_t) \trianglelefteq F(\vec{x}_t) \\ \vec{x}_t & ; else \end{cases} \tag{4}$$

where $\trianglelefteq$ denotes the compare operator used to valuate different test points, and $F : \mathbb{R}^n \rightarrow \mathbb{R}$ the valuation function. For a simple hill climbing strategy, $\trianglelefteq$ is the less-or-equal operator, and $F = f$. In the following $F(\vec{x})$ is used to show how different ways of handling constraints are incorporated into direct optimization methods.

Most direct optimization methods do *not* care for constraints and domains $D \subset \mathbb{R}^n$. Special heuristics for dealing explicitly with constraints like reflecting an infeasible test point along the border to the feasible domain are in use [3], but their effectiveness highly depends on the characteristics of the given problem. Hence, they require careful and sometimes tedious analysis of the problem prior to their application.

As a workaround, a so-called *penalty function* $p(\vec{x})$ is used to provide a valuation that handicaps infeasible test points against feasible ones by defining a surrogate function F which is evaluated instead of the objective function on $\mathbb{R}^n$. Indirect methods normally require the surrogate function to preserve properties like smoothness and differentiability. Furthermore it must satisfy

$$f(\vec{x}) \leq f(\vec{y}) \quad \Rightarrow \quad F(\vec{x}) \leq F(\vec{y}) \tag{5}$$

to guarantee that an optimum of the objective function is an optimum of the surrogate function, as well.

Some direct search algorithms also integrate *partial penalty functions* into their evaluation procedure, e.g. the Rosenbrock strategy[4]. Partial penalty functions take effect only near the borders of the feasible region:

$$F(\vec{x}) \;=\; \begin{cases} f(\vec{x}) & ;\text{if } g_i(\vec{x}) \geq \epsilon \\ p(\vec{x}) & ;\text{else} \end{cases} \tag{6}$$

where ϵ is a rather crucial parameter: a large value can hide an optimum near the border from the algorithm, a very low value may not penalize infeasible points sufficiently.

Ideally, ϵ would be set to zero, and the surrogate function F would be defined with respect to

$$F(\vec{x}) < F(\vec{x}') \quad ; \forall \vec{x} \in M, \vec{x}' \notin M \tag{7}$$

A surrogate function like this normally causes the loss of smoothness at the transition from the feasible to the infeasible region which even some direct methods require for their step size adaption. Nevertheless, if a search strategy rests only on comparisons of test points an ideal partial penalty function can be defined as follows:

$$p(\vec{x}) \;=\; m_c(\vec{x}) + d_c(\vec{x}) \tag{8}$$

$$d_c(\vec{x}) \;=\; ||\vec{x} - \vec{c}|| \tag{9}$$

where $d_c(\vec{x})$ is a measure for the distance to the feasible domain with $\vec{c}$ representing the closest feasible point near $\vec{x}$ defined by an arbitrary norm $||\cdot||$. $d_c(\vec{x})$ is required to guide the search towards the feasible domain M while sampling the infeasible domain $\mathbb{R}^n \setminus M$, $m_c(\vec{x})$ being an adjusting value required to link $f(\vec{x})$ and $p(\vec{x})$. A penalty function according to (8) is very handy if the design of a feasible starting point is difficult to accomplish.

With $m_c(\vec{x}) = \text{const}$, e.g.

$$m_c(\vec{x}) \;=\; \max_{\vec{x}^i \in M} \{f(\vec{x}^i)\} \tag{10}$$

and $d_c(\vec{x}) = 0$ the above penalty function degradates to a *barrier function*, which has the serious disadvantage of *not* guiding the search towards the feasible domain, which might result in premature stagnation. Hence, with this kind of penalty function a feasible starting point is always recommended if not required.

In any case, objective function $f(\vec{x})$ and penalty function $p(\vec{x})$ need to be linked by $m_c(\vec{x})$ (8) in order to guide the search and to support the generation of the next test point, if $\Delta\vec{x}$ turns out to be a function based on the history of the search, e.g.

$$\Delta\vec{x} \;=\; z\left(f(\vec{x}_t), f(\vec{x}_{t-1})\right) \tag{11}$$

$$\;=\; ||f(\vec{x}_t) - f(\vec{x}_{t-1})||\; G(0,1) \tag{12}$$

with $G(0,1)$ being a normalized Gaussian random number. For a complicated objective function $f(\vec{x})$ with unknown characteristics defining $m_c(\vec{x})$ can be a tedious task, especially, if a case like (12) has to be considered.

An alternative to using a surrogate function $F(\vec{x})$ is the implicit handling of constraints as part of the generation of a new test point (3) as it is done in Evolution Strategies according to Schwefel [2], i.e.

$$\vec{x}_t' \;=\; \gamma'(\vec{x}_t) \tag{13}$$

$$\gamma'(\vec{x}_t) \;=\; \begin{cases} \gamma(\vec{x}_t) & ;\text{if } g_i(\vec{x}) \geq 0 \\ & \;\wedge\; h_j(\vec{x}) = 0 \\ \gamma'(\vec{x}_t) & ;\text{else} \end{cases} \tag{14}$$

Equations (3) and (13) provide a recursive definition for a *generating loop*, which guarantees to yield a feasible point. It iterates as long as $\gamma(\vec{x}_t)$ generates an infeasible test point.

Using a generating loop frees from defining a surrogate function $F(\vec{x})$ (6) and from establishing the required link between $p(\vec{x})$ and $f(\vec{x})$ (8). This approach is universal and can be incorporated into any other search methods as EAs, although there is no evidence that it is actually done.

Effectively, (13) is equivalent to a barrier function as part of a surrogate function. If an optimum is located at an active constraint $g_i(\vec{x}) = 0$ a great deal of effort is wasted to generate feasible test points without adjusting the internal strategy parameters to cope with the situation, i.e. implicit information provided by infeasible points is *not* exploited. Hence, there is no guidance towards the feasible domain to speed up the search. When starting the search with an infeasible starting point is *not* handled efficiently this way because the search might be stuck on a plateau. Depending on the characteristics of generating $\Delta\vec{x}$ (e.g. normal or uniform distributed) the optimization method will not have a chance to find a feasible starting point.

As a conclusion, in order to deal effectively with constraints, for any given test point $\vec{x}$ a function $d_c(\vec{x})$ is required (9). It provides some measure of "infeasibility" guiding the search towards the feasible domain, thus increasing the exploitation of *any* generated test point speeding up the overall search.

In two recent studies, Michalewicz [5, 6] compared several approaches to handle constraints in Genetic Algorithms (GAs) using a measure of infeasibility as described above. Except for the lethal offspring method of

ES, all of them define surrogate functions $F : \mathbb{R}^n \rightarrow \mathbb{R}$, so they have to satisfy (5) and (7). Although the ES approach has been shown as inferior to integrated penalty functions, the advantage of generality cannot be denied.

In fact, there is no need to assign a real valued quality to search points to compare them, as we will show in this paper. As reasoned above, only a quality order on test points must be provided, which is also sufficient for EAs with rank based selection. By introducing a generalized compare operator $\trianglelefteq$ we will combine the ease of the generating loop with the measure of infeasibility provided by $d_c(\vec{x})$ in order to avoid the definition of a problem-dependent and/or optimization method-dependent surrogate function $F(\vec{x})$ and show the feasibility of this approach. By changing the quality domain we avoid all problems that arise from (5) and (7).

Transparent Constraints

The basic principle of a direct optimization method was defined in (3). For our purpose $F(\vec{x})$ and $\trianglelefteq$ will be redefined:

Definition 2

$$
\begin{aligned}
F &: \mathbb{R}^n \rightarrow \mathbb{B} \times \mathbb{R} && (15)\\
F(\vec{x}) &= \begin{cases} (0, f(\vec{x})) & ;\text{if } \vec{x} \in M \\ (1, d_c(\vec{x})) & ;\text{else} \end{cases} && (16)\\
\trianglelefteq &: (\mathbb{B}\times\mathbb{R}) \times (\mathbb{B}\times\mathbb{R}) \rightarrow \mathbb{B} && (17)\\
y'_i &= (b_i, y_i) && (18)\\
y'_1 \trianglelefteq y'_2 &= (b_1 < b_2) && (19)\\
&\vee \quad (b_1 = b_2) \wedge (y_1 \leq y_2) && (20)
\end{aligned}
$$

with $\mathbb{B} = \{0, 1\}$. The function d_c will be reffered to as the *metric penalty function* (MPF) in the following. In order to modify an existing (direct) optimization method according to (15) and (17) the following steps have to be accomplished:

- change the domain of variables receiving objective values
- replace any invocation of $f(\vec{x})$ by $F(\vec{x})$
- replace any comparison of objective values $y_1 \leq y_2$ by $y_1 \trianglelefteq y_2$

If for a given set of constraints it is impossible to determine an appropriate measure for the distance to the feasible domain (9), defining $d_c(\vec{x})$ = *number of violated constraints* might be a last resort providing a very "rough" measure (??).

Experimental Results

Since ES is the only direct optimization method that already provides a general way to handle constraints, it is consequent to use it for experimental comparisons. While *two-membered ES* can be described simply as (3) where $\Delta\vec{x}_t$ is a vector of Gaussian distributed random variables, recent variants, the *multi-membered ESs*, manage a set of points (*population*) per iteration (*generation*), instead of a single one, and they have more complex acceptance rules. In terms of ES, points in the search space are also called *individuals*, and discarded infeasible points are treated as *lethal mutations*. For a comprehensive introduction to ES the reader is referred to [7].

Some theoretical investigations on ESs were made using a simple constrained function, the so-called *corridor model*([7],p. 134ff.):

$$
\begin{aligned}
f(\vec{x}) &= -x_1 \\
\text{w.r.t.} \quad g_j(\vec{x}) &= b - |x_j| \quad \geq 0
\end{aligned} \tag{21}
$$

This becomes an n-dimensional right-angled corridor along the x_1 axis. Since there is no finite optimum, is it possible to measure the progress after a certain number of evaluations. The corridor model was used for theoretical approximations of the optimum step size (mutation variance) of ESs with lethal mutations[1]. In order to define an MPF for the corridor, the obvious way to valuate the infeasibility of test points is to compare their Euclidean distances to the feasible region:

$$
d_c(\vec{x}) = \sqrt{\sum_{i=0}^{m} H(-g_i(\vec{x}))g_i(\vec{x})^2} \tag{22}
$$

where $H : \mathbb{R} \rightarrow \{0, 1\}$ is the Heavyside function:

$$
H(y) = \begin{cases} 1 & : \quad y > 0 \\ 0 & : \quad y \leq 0 \end{cases} \tag{23}
$$

This is comparable to a partial penalty approach and should lead to a faster progress than lethal mutations. To simulate the case that the constraints are binary, i.e. that they only indicate whether a point is feasible or not, the corridor model was also implemented with

$$
d_c(\vec{x}) = \sum_{i=0}^{m} H(-g(\vec{x})) \tag{24}
$$

The metric used here is the number of violated constraints, which is an information that is always available if more than one restriction function is present.

Two test series were computed on the corridor model with dimension 30. The strategy was a $(15, 100)$-ES. Each

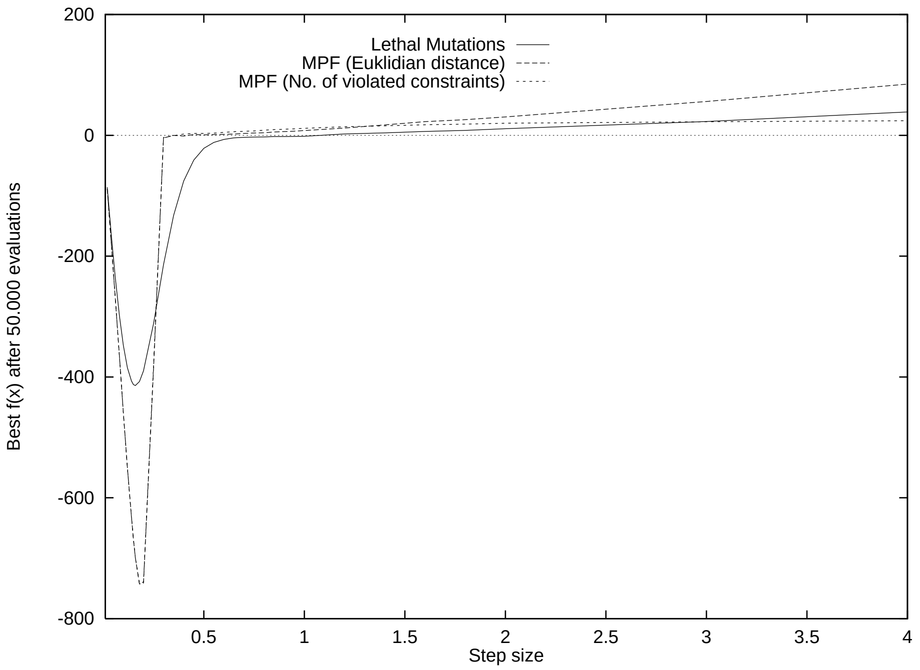

Figure 1: Corridor with fixed mutation variances

run was stopped after 50.000 evaluations, and the best value achieved so far was taken as a measure of progress. Each evaluation of the constraints was counted as an additional objective function call, assuming that restrictions and objective function evaluations take about the same computational time.

The first test series was computed with fixed step sizes in order to find out whether the theoretical optimum variance is also valid for MPF. According to [1], the maximum progress should be achieved with

$$\sigma_{\mathrm{opt}} = b\frac{\sqrt{2\pi}}{n} \tag{25}$$

where b is the corridor width from (21). For $b = 1$ and $n = 30$ we obtain a value of about 0.08. Figure 1 shows the relationship between step sizes and the best objective function value after 50.000 evaluations. It can be seen that both the lethal mutation method and the MPF gain the maximum progress at 0.18, slightly above the theoretical estimation[1], but at different levels: MPF goes nearly twice as far, but is more sensitive to missetting of the step size. The most surprising result from these experiments is that there is no notable difference between the two metrics used as MPF. This might be a property of the ES comma selection type: only the 15 best point are used to generate the next generation, hence it does not matter whether the remaining points are infeasible or just bad. With a step size larger than half the corridor all strategies tend to gain no progress.

In a second test series, the ES self-learning of step sizes was enabled. One common step size for all variables of an individual (test point) was used, and all individuals got the same step size initially. It was of special interest to observe how sensitive the different strategies react on inadequate settings of the initial step sizes.

The results are shown in figure 2. With lethal mutations, no progress could be achieved with initial step sizes above twice the corridor width, and below it is outperformed by both MPF variants.

The Euclidean MPF method entered the feasible region very quickly and adapts suitable step sizes almost independent of the initial step size, as expected. Again, the MPF variant that simply counts the number of violated constraints performs considerable well: although it has only slightly more information about the infeasibilty of points, it succeeds to adapt suitable step sizes with very bad initial settings.

Furthermore, a parallel mixed-integer EA with MPF handling of constraints has been applied to a structural optimization problem with great success[8]. The optimization of multi-layer optical coatings is a mixed-integer and variable dimensional problem. In specific problems, the qualitity of filter designs is known to rise monotonic with the so-called optical thickness. Since this

[1] Since the optimum step size had been determined for an $(1+1)$-ES, it is not surprising that the higher selection pressure of a $(15, 100)$ allows a more risky sampling of test points.

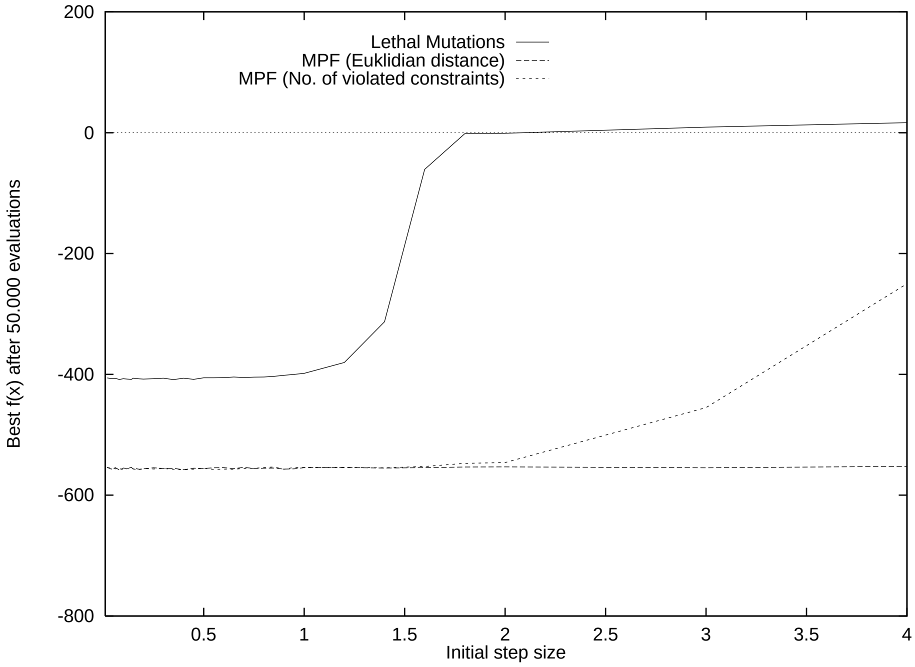

Figure 2: Corridor with variable mutation variances

value is restricted by the production process, the mathematical model must be constrained in order to obtain useful results. In [8], the relation between optical thicknesses and the corresponding optimum filter qualtity has been estimated numerically and compared to theoretical estimations [9, 10] on a certain design problem. Due to the MPF, designs of arbitrary optical thicknesses could be evolved. Over the whole range of reasonable optical thicknesses, the results from [8] are at least very close to solutions found by others, even better in some ranges. Furthermore, the results lead to conjectures about a more accurate estimation of the relation mentioned above. To achieve the same results with traditional penalty functions, one had to analyze the problem in order to obtain a lower bound for the quality of a design depending on its thickness. But this, in turn, is just the problem itself.

Conclusions

An alternative approach to handle constraints in direct search methods is presented in this paper, introducing a compromise between the most general way to handle constraints by just discarding infeasible points, and the definition of problem-specific penalty functions. The method of metric penalty functions described here bases on the premise that infeasible points must be always valuated worse than feasible ones. There are of course problems where this assumption does not hold, as reasoned in [6]. The major advantage of MPFs is that any information available about violated constraints can be used to guide the search towards the feasible region, without the construction of an explicit surrogate function or biasing partial penalty functions. No external parameter must be added to an algorithm to profit from MPFs.

The experimental results demonstrate the capability of MPFs to exploit the information gained from the restriction functions, even if they only indicate whether a point is feasible or not. MPFs are especially useful for Evolutionary Algorithms because they extent their robustness against non-optimum external parameter settings to the constraint handling, as well.

References

[1] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann–Holzboog, Stuttgart, 1973.

[2] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research.* Birkhäuser, Basel, 1977.

[3] S.M. Roberts and H.I. Lyvers. The gradient method in process control. *Ind. Eng.*, 1961.

[4] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[5] Z. Michalewicz. Genetic algorithms, nonlinear optimization, and constraints. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the 6th International Conference*, pages 151–158. Morgan Kaufmann Publishers, San Francisco, CA, 1995.

[6] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 135–155. The MIT Press, Cambridge, MA, 1995.

[7] H.-P. Schwefel. *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[8] Martin Schütz and Joachim Sprave. Application of parallel mixed-integer evolution strategies with mutation rate pooling. In P. Angeline and T. Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming.* The MIT Press, Cambridge, MA, 1996.

[9] R. R. Willey. Predicting achievable design performance of broadband antireflection coatings. *Applied Optics*, 32:5447–5451, 1993.

[10] A. V. Tikhonravov, M. K. Trubetskov, J. A. Dobrowolski, and B. T. Sullivan. Optimum Solutions to Single-Band Normal Incidence Antireflection Coating Problems. In *Optical Interference Coatings, Vol. 17, OSA Technical Digest Series*, pages 49–51, Washington DC, 1995. Optical Society of America.