Chapter 4
Derivative-Free Optimization

Oliver Kramer, David Echeverría Ciaurri, and Slawomir Koziel

Abstract. In many engineering applications it is common to find optimization problems where the cost function and/or constraints require complex simulations. Though it is often, but not always, theoretically possible in these cases to extract derivative information efficiently, the associated implementation procedures are typically non-trivial and time-consuming (e.g., adjoint-based methodologies). Derivative-free (non-invasive, black-box) optimization has lately received considerable attention within the optimization community, including the establishment of solid mathematical foundations for many of the methods considered in practice. In this chapter we will describe some of the most conspicuous derivative-free optimization techniques. Our depiction will concentrate first on local optimization such as pattern search techniques, and other methods based on interpolation/approximation. Then, we will survey a number of global search methodologies, and finally give guidelines on constraint handling approaches.

4.1 Introduction

Efficient optimization very often hinges on the use of derivative information of the cost function and/or constraints with respect to the design variables. In the last

Oliver Kramer
UC Berkeley, Berkeley, CA 94704, USA
e-mail: okramer@icsi.berkeley.edu

David Echeverría Ciaurri
Department of Energy Resources Engineering,
Stanford University, Stanford, CA 94305-2220, USA
e-mail: echeverr@stanford.edu

Slawomir Koziel
Engineering Optimization & Modeling Center,
School of Science and Engineering, Reykjavik University, Menntavegur 1,
101 Reykjavik, Iceland
e-mail: koziel@ru.is

S. Koziel & X.-S. Yang (Eds.): Comput. Optimization, Methods and Algorithms, SCI 356, pp. 61–83.
springerlink.com © Springer-Verlag Berlin Heidelberg 2011

decades, the computational models used in design have increased in sophistication to such an extent that it is common to find situations where (reliable) derivative information is not available. Although in simulation-based design there are methodologies that allow one to extract derivatives with a modest amount of additional computation, these approaches are in general invasive with respect to the simulator (e.g., adjoint-based techniques [1]), and thus, require precise knowledge of the simulation code and access to it. Moreover, obtaining derivatives in this intrusive way often implies significant coding (not only at the code development stage, but also subsequently, when maintaining or upgrading the software), and consequently, many simulators simply yield, as output, the data needed for the cost function and/or constraint values. Furthermore, optimal design has currently a clear multidisciplinary nature, so it is reasonable to expect that some components of the overall simulation do not include derivatives. This situation is even more likely when commercial software is used, since then the source code is typically simply inaccessible.

In this chapter we review a number of techniques that can be applied to generally constrained continuous optimization problems for which the cost function and constraint computation can be considered as a black box system. We wish to clearly distinguish between methods that aim at providing just a solution (local optimization; see Section 4.3), and approaches that try to avoid being trapped in local optima (global optimization; see Section 4.4). Local optimization is much easier to handle than global optimization, since, in general, there is no algorithmically suitable characterization of global optima. As a consequence, there are more theoretical results of practical relevance for local than for global optimizers (e.g., convergence conditions and rate). For more details on theoretical aspects of derivative-free optimization we strongly recommend both the review [2] and the book [3]. The techniques are described for continuous variables, but it is possible to apply, with care, extensions of many of them to mixed-integer scenarios. However, since mixed-integer nonlinear programming is still an emergent area (especially in simulated-based optimization), we prefer not to include recommendations in this case.

In some situations, numerical derivatives can be computed fairly efficiently (e.g., via a computer cluster), and still yield results that can be acceptable in practice. However, if the function/constraint evaluations are even moderately noisy, numerical derivatives are usually not useful. Though methods that rely on approximate derivatives are not derivative-free techniques per se, for example, in the absence of noise, they can address optimization in a black box approach. We should note that in addition to their inherent additional computational costs, numerical derivatives very often imply the tuning of the derivative approximation together with the simulation tolerances, and this is not always easy to do. Implicit filtering [4, 5] may somehow alleviate some of these issues. This approach is essentially a gradient-based procedure where the derivative approximation is improved as the optimization progresses. Implicit filtering has been recommended for problems with multiple local optima (e.g., noisy cost functions). For more details on gradient-based methodologies the reader is encouraged to regard nonlinear optimization references (for example, [6, 7]).

Many derivative-free methods are easy to implement, and this feature makes them attractive when approximate solutions are required in a short time frame. An obvious statement that is often neglected is that the computational cost of an iteration of an algorithm is not always a good estimate of the time needed within a project (measured from its inception) to obtain results that are satisfactory. However, one important drawback of derivative-free techniques (when compared, for example, with adjoint-based approaches) is the limitation on the number of optimization variables that can be handled. For example, in [3] and [2] the limit given is a few hundred variables. However, this limit in the problem size can be overcome, at least to some extent, if one is not restricted to a single sequential environment. For some of the algorithms though, adequately exploiting parallelism may be difficult or even impossible. When distributed computing resources are scarce or not available, and for simulation-based designs with significantly more than a hundred optimization variables, some form of parameter reduction is mandatory. In these cases, surrogates or reduced order models [8] for the cost function and constraints are desirable approaches. Fortunately, suitable parameter and model order reduction techniques can often be found in many engineering applications, although they may give rise to inaccurate models. We should add that even in theory, as long as a problem with nonsmooth/noisy cost functions/constraints can be reasonably approximated by a smooth function (see [9], Section 10.6), some derivative-free optimization algorithms perform well with nonsmooth/noisy cost functions, as has been observed in practice [2, 3].

In the last decade, there has been a renaissance of gradient-free optimization methodologies, and they have been successfully applied in a number of areas. Examples of this are ubiquitous; to name a few, derivative-free techniques have been used within molecular geometry [10], aircraft design [11, 12], hydrodynamics [13, 14], medicine [15, 16] and earth sciences [17, 18, 19, 20]. These references include generally constrained cases with derivative-free objective functions and constraints, continuous and integer optimization variables, and local and global approaches. In spite of all this apparent abundance of results, we should not disregard the general recommendation (see [3, 2]) of strongly preferring gradient-based methods if accurate derivative information can be computed reasonably efficiently and globally.

This chapter is structured as follows. In Section 4.2 we introduce the general problem formulation and notation. A number of derivative-free methodologies for unconstrained continuous optimization are presented in the next two sections. Section 4.3 refers to local optimization, and Section 4.4 is devoted to global optimization. Guidelines for extending all these algorithms to generally constrained optimization are given in Section 4.5. We bring the chapter to an end with some conclusions and recommendations.

4.2 Derivative-Free Optimization

A general single-objective optimization problem can be formally stated as:

$$\min_{\mathbf{x}\in\Omega\subset\mathbb{R}^n} f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{g}(\mathbf{x}) \leq 0, \tag{4.1}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x} \in \mathbb{R}^n$ is the vector of control variables, and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ represents the nonlinear constraints in the problem. Bound and linear constraints are included in the set $\Omega \subset \mathbb{R}^n$. For many approaches it is natural to treat any constraints for which derivatives are available separately. In particular, bounds and linear constraints, and any other structure than can be exploited, should be. So for example, nonlinear least-squares problems should exploit that inherent structure whenever possible (see e.g. [21]). We are interested in applications for which the objective function and constraint variables are computed using the output from a simulator, rendering function evaluations expensive and derivatives unavailable.

We will begin by discussing some general issues with respect to optimization with derivatives since they have important relevancy to the derivative-free case. Essentially all approaches to the former are somewhere between steepest descent and Newton's method, or equivalently use something that is between a linear and a quadratic model. This is reinforced by the realization that almost all practical computation is linear at its core, and (unconstrained) minima are characterized by the gradient being zero, and quadratic models give rise to linear gradients. In fact, theoretically at least, steepest descent is robust but slow (and in fact sometimes so slow that in practice it is not robust) whereas Newton's method is fast but may have a very small radius of convergence. That is, one needs to start close to the solution. It is also computationally more demanding. Thus in a sense, most practical unconstrained algorithms are intelligent compromises between these two extremes. Although, somewhat oversimplified, one can say that the constrained case is dealt with by being feasible, determining which constraints are tight, linearizing these constraints and then solving the reduced problem determined by these linearizations. Therefore, some reliable first-order model is essential, and for faster convergence, something more like a second-order model is desirable. In the unconstrained case with derivatives these are typically provided by a truncated Taylor series model (in the first-order case) and some approximation to a truncated second-order Taylor series model. A critical property of such models is that as the step sizes become small the models become more accurate. In the case where derivatives, or good approximations to them, are not available, clearly, one cannot use truncated Taylor series models. It thus transpires that, if for example, one uses interpolation or regression models, that depend only on function values, one can no longer guarantee that as the step sizes become small the models become more accurate. Thus one has to have some explicit way to make this guarantee, at least approximately. It turns out that this is usually done by considering the geometry of the points at which the function is evaluated, at least, before attempting to decrease the effective maximum step size. In pattern search methods, this is done by explicitly using a pattern with good geometry, for example, a regular mesh that one only scales while maintaining the a priori good geometry.

In the derivative case the usual stopping criteria relates to the first-order optimality conditions. In the derivative-free case, one does not explicitly have these, since they require (approximations to) the derivatives. At this stage we just remark that any criteria used should relate to the derivative case conditions, so, for example one needs something like a reasonable first-order model, at least asymptotically.

4.3 Local Optimization

The kernel of many optimizers are local methods. This is not surprising, since, as we already mentioned, there is no suitable algorithmic characterization of global optima unless one considers special situations such as where all local optima are global, as for example in convex minimization problems. In this section we concentrate on local search methods based on pattern search and also on interpolation and approximation models. Some constraint handling procedures are described in Section 4.5.

4.3.1 Pattern Search Methods

Pattern search methods are optimization procedures that evaluate the cost function in a stencil-based fashion determined by a set of directions with intrinsic properties meant to be desirable from a geometric/algebraic point of view. This stencil is sequentially modified as iterations proceed. The recent popularity of these schemes is due in part to the development of a mathematically sound convergence theory [2, 3]. Moreover, they are attractive because they can relatively easily leverage the widespread availability of parallel computing resources. However, most published computational results are not parallel exploiting.

4.3.1.1 Generalized Pattern Search

Generalized pattern search (GPS; [22, 23]) refers to a whole family of optimization methods. GPS relies on polling (local exploration of the cost function on the pattern) but may be enhanced by additional searches, see [23]. At any particular iteration a stencil (pattern) is centered at the current solution. The stencil comprises a set of directions such that at least one direction is a descent direction. This is also called a generating set (see e.g. [2]). If any of the points in the stencil represent an improvement in the cost function, the stencil is moved to one of them. Otherwise, the stencil size is decreased. The optimization progresses until some stopping criterion is satisfied (typically, a minimum stencil size). Generalized pattern search can be further generalized by polling in an asymptotically dense set of directions (this set varies with the iterations). The resulting algorithm is the mesh adaptive direct search (MADS; [24]). In particular, some generalization of a simple fixed pattern is essential for constrained problems. The GPS method parallelizes naturally since, at a particular iteration, the objective function evaluations at the polling points can be accomplished in a distributed fashion. The method typically requires on the order of n function evaluations per iteration (where n is the number of optimization variables).

4.3.1.2 Hooke-Jeeves Direct Search

The Hooke-Jeeves direct search (HJDS; [25]) is another pattern search method and was the first to use the term ‘direct search’ method and take advantage of the idea

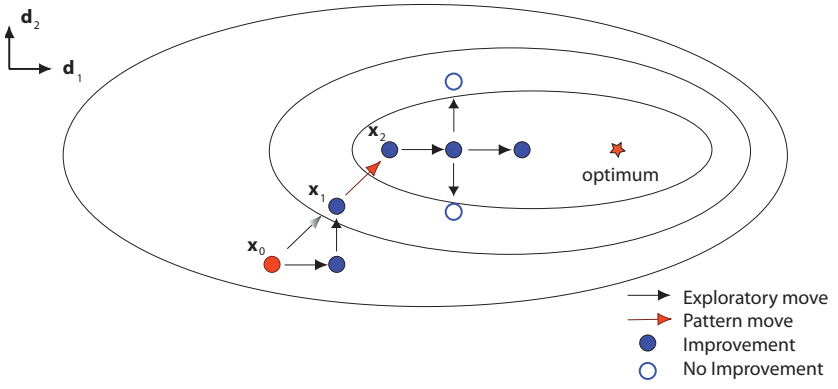

Fig. 4.1 Illustration of exploratory and pattern moves in Hooke-Jeeves direct search (modified from [19]). The star represents the optimum.

of a pattern. HJDS is based on two types of moves: exploratory and pattern. These moves are illustrated in Figure 4.1 for some optimization iterations in $\mathbb{R}^2$.

The iteration starts with a base point $\mathbf{x}_0$ and a given step size. During the exploratory move, the objective function is evaluated at successive changes of the base point in the search (for example coordinate) directions. All the directions are polled sequentially and in an opportunistic way. This means that if $\mathbf{d}_1 \in \mathbb{R}^n$ is the first search direction, the first function evaluation is at $\mathbf{x}_0 + \mathbf{d}_1$. If this represents an improvement in the cost function, the next point polled will be, assuming $n > 1$, $\mathbf{x}_0 + \mathbf{d}_1 + \mathbf{d}_2$, where $\mathbf{d}_2$ is the second search direction. Otherwise the point $\mathbf{x}_0 - \mathbf{d}_1$ is polled. Upon success at this last point, the search proceeds with $\mathbf{x}_0 - \mathbf{d}_1 + \mathbf{d}_2$, and alternatively with $\mathbf{x}_0 + \mathbf{d}_2$. The exploration continues until all search directions have been considered. If after the exploratory step no improvement in the cost function is found, the step size is reduced. Otherwise, a new point $\mathbf{x}_1$ is obtained, but instead of centering another exploratory move at $\mathbf{x}_1$, the algorithm performs the pattern move, which is a more aggressive step that moves further in the underlying successful direction. After the pattern move, the next polling center $\mathbf{x}_2$ is set at $\mathbf{x}_0 + 2(\mathbf{x}_1 - \mathbf{x}_0)$. If the exploratory move at $\mathbf{x}_2$ fails to improve upon $\mathbf{x}_1$, a new polling is performed around $\mathbf{x}_1$. If this again yields no cost function decrease, the step size is reduced, keeping the polling center at $\mathbf{x}_1$.

Notice the clear serial nature of the algorithm. This makes HJDS a reasonable pattern search option when distributed computing resources are not available. Because of the pattern move, HJDS may also be beneficial in situations where an optimum is far from the initial guess. One could argue that initially pattern search techniques should use a relatively large stencil size on the hope that this feature enables them to avoid some local minima and, perhaps, some robustness against noisy cost functions.

4.3.2 Derivative-Free Optimization with Interpolation and Approximation Models

The other major approach to derivative-free optimization is based on building models that are meant to approximate the functions and then make use of derivative methods on the models. The advantage is that one is trying to take account of the shape of the function rather than naively just using the function evaluations alone. As our introductory remarks in Section 4.2 suggest we can expect our models to be at least first-order models or better still, second-order.

A major drawback of this approach is that, since the models are not based upon an a priori pattern, as with just polling, the geometry of the sample points used requires special attention. Additionally, one pays for the extra sophistication of these methods in that they are not obviously parallelizable. Some of the better known algorithms in this class include DFO [3], NEWUOA [26] and BOOSTERS [27]. The basic ideas will be given here but it is recommended that the diligent reader consult Chapters 3-6 of [3].

First of all, what does good geometry mean? Essentially, for example, if one wants to consider interpolation by a polynomial of degree d, where $d = 1$, that is linear interpolation, one needs $n+1$ points and good geometry means they do not lie on or close to a linear surface. Similarly, if one wants to consider interpolation by a polynomial of degree d, where $d = 2$, that is quadratic interpolation, one needs $(n+1)(n+2)/2$ points and good geometry means they do not lie on or close to a quadratic or linear surface. The extension to higher degree is clear. One can also see why the problem goes away if one works with a suitable pattern, as in a pattern search method.

Now, all three methods mentioned above are trust-region based. For an introduction to trust-region techniques the readers are referred to [7], or [9] for a monographic volume. In the case with derivatives the essential ingredients are the following. Starting at a given point $\mathbf{x}_0$ one has a region about that point, coined the trust region and denoted by Δ_0. The trust region is typically a sphere in the Euclidean or in the infinity norm. One then requires a model $m(\mathbf{x})$ for the true objective function that is relatively easy to minimize within the trust region (e.g., a truncated first-order Taylor series or an approximation to a truncated second-order Taylor series, about the current point). A search direction from the current point is determined based upon the model and one (approximately) minimizes the model within the trust region.

The trust region can be updated in the following manner. Suppose $\mathbf{y}_1$ is the approximate minimizer of the model within the trust region Δ_0. We then compare the predicted reduction to truth in the sense that we consider

$$\rho = \frac{f(\mathbf{x}_0) - f(\mathbf{y}_1)}{m(\mathbf{x}_0) - m(\mathbf{y}_1)}.$$

Then typically one assigns some updating strategy to the trust-region radius Δ_0 like

$$\Delta_1 = \begin{cases} 2 \cdot \Delta_0, & \text{if } \rho > 0.9, \\ \Delta_0, & \text{if } 0.1 \leq \rho \leq 0.9, \\ 0.5 \cdot \Delta_0 & \text{if } \rho < 0.1, \end{cases}$$

where Δ_1 denotes the updated radius. In the first two cases $\mathbf{x}_1 = \mathbf{y}_1$ and in the third case $\mathbf{x}_1 = \mathbf{x}_0$.

Thus, although oversimplified, if we are using Taylor series approximations for our models, within the trust management scheme one can ensure convergence to a solution satisfying first-order optimality conditions [9]. Perhaps the most important difference once derivatives are not available is that we cannot take Taylor series models and so, in general, optimality can no longer be guaranteed. In fact, we have to be sure that when we reduce the trust-region radius it is because of the problem and not just a consequence of having a bad model as a result of poor geometry of the sampling points. So it is here that one has to consider the geometry. Fortunately, it can be shown that one can constructively ensure good geometry, and with that, support the whole derivative-free approach with convergence to solutions that satisfy first-order optimality conditions. For details see [3], Chapter 6.

4.4 Global Optimization

In the previous section we have concentrated on local search methods. Unfortunately, most real-world problems are multimodal, and global optima are generally extremely difficult to obtain. Local search methods find local optima that are not guaranteed to be global. Here we will give a short survey of global optimization methods. However, the reader should take note of the following. In practice, often good local optima suffice. If one is considering even a modest number of variables, say fifty, it is generally very difficult, if not impossible, to ensure convergence to a provable global solution, in a reasonable length of time, even if derivatives are available, not to mention in the derivative-free case. Almost all algorithms designed to determine local optima are significantly more efficient than global methods.

Many successful methods in global optimization are based on stochastic components, as they allow to escape from local optima and overcome premature stagnation. Famous classes of families of stochastic global optimization methods are evolutionary algorithms, estimation of distribution algorithms, particle swarm optimization, and differential evolution. Further heuristics known in literature are simulated annealing [28, 29], tabu search [30, 31], ant colony optimization [32, 33], and artificial immune systems [34, 35]. In this section, we concentrate on the first four classes of methods that have been successful in a number of practical applications.

4.4.1 Evolutionary Algorithms

A history of more than forty years of active research on evolutionary computation indicates that stochastic optimization algorithms are an important class of

```
1   Start
2     Initialize solutions x_i of population P
3     Evaluate objective function for the solutions x_i in P
4     Repeat
5       For i = 0 To λ
6         Select ρ parents from P
7         Create new x_i by recombination
8         Mutate x_i
9         Evaluate objective function for x_i
10        Add x_i to P'
11      Next
12      Select μ parents from P' and form new P
13    Until termination condition
14  End
```

Fig. 4.2 Pseudocode of a generic evolutionary algorithm.

derivative-free search methodologies. The separate development of evolutionary algorithms (EAs) in the United States and Europe led to different kinds of algorithmic variants. Genetic algorithms were developed by John Holland in the United States at the beginning of the seventies. Holland's intention was to exploit adaptive behavior. In his book *Adaptation in Natural and Artificial Systems* [36] he describes the development of genetic algorithms (GAs). His original algorithm is today known as simple GA. Evolutionary programming by Fogel, Owens and Walsh [37] was originally designed for optimization of the evolvement of deterministic finite automata, but has today been extended to numerical optimization. Evolution strategies (ES) were developed by Rechenberg and Schwefel in the middle of the sixties in Germany [38, 39, 40]. In the following, we introduce the idea of evolutionary optimization, that is closely related to evolution strategies.

4.4.1.1 Algorithmic Framework

The basis of evolutionary search is a population $\mathscr{P} := \{\mathbf{x}_1, \ldots, \mathbf{x}_\lambda\}$ of candidate solutions, also called individuals. Figure 4.2 shows the pseudocode of a general evolutionary algorithm. The optimization process takes three steps. In the first step the recombination operator (see Section 4.4.1.2) selects ρ parents and combines them to obtain new solutions. In the second step the mutation operator (see Section 4.4.1.3) adds random noise to the preliminary candidate solution. The objective function $f(\mathbf{x})$ is interpreted in terms of the quality of the individuals, and in EA lexicon is called fitness. The fitness of the new offspring solution is evaluated. All individuals of a generation form the new population $\mathscr{P}'$. In the third step, when λ solutions have been produced, μ individuals, with $\mu < \lambda$, are selected (see Section 4.4.1.4), and form the new parental population of the following generation. The process starts again until a termination condition is reached. Typical termination conditions are the accomplishment of a certain solution quality, or an upper bound

on the number of generations. We now concentrate on the stochastic operators that are often used in evolutionary computation.

4.4.1.2 Recombination

In biological systems recombination, also known as crossover, mixes the genetic material of two parents. Most EAs also make use of a recombination operator and combine the information of two or more individuals into a new offspring solution. Hence, the offspring carries parts of the genetic material of its parents. The use of recombination is discussed controversially within the building block hypothesis by Goldberg [41, 42], and the genetic repair effect by Beyer [43].

Typical recombination operators for continuous representations are dominant and intermediary recombination. Dominant recombination randomly combines the genes of all parents. If we consider parents of the form $\mathbf{x} = (x_1, \ldots x_n)$, dominant recombination with ρ parents $\mathbf{x}^1, \ldots, \mathbf{x}^\rho$ creates the offspring vector $\mathbf{x}' = (x'_1, \ldots, x'_n)$ by random choice of the i-th component x'_i:

$$x'_i := x_i^k, \quad k \in \text{ random } \{1, \ldots, \rho\}. \tag{4.2}$$

Intermediate recombination is appropriate for integer and real-valued solution spaces. Given ρ parents $\mathbf{x}^1, \ldots, \mathbf{x}^\rho$ each component of the offspring vector $\mathbf{x}'$ is the arithmetic mean of the components of all ρ parents. Thus, the characteristics of descendant solutions lie between their parents:

$$x'_i := \frac{1}{\rho} \sum_{k=1}^{\rho} x_i^k. \tag{4.3}$$

Integer representations may require rounding procedures to produce intermediate integer solutions.

4.4.1.3 Mutation

Mutation is the second main source for evolutionary changes. According to Beyer and Schwefel [38], a mutation operator is supposed to fulfill three conditions. First, from each point in the solution space each other point must be reachable. Second, in unconstrained solution spaces a bias is disadvantageous, because the direction to the optimum is not known. And third, the mutation strength should be adjustable in order to adapt to solution space conditions. In the following, we concentrate on the well-known Gaussian mutation operator. We assume that solutions are vectors of real values. Random numbers based on the Gaussian distribution $\mathcal{N}(0,1)$ satisfy these conditions in continuous domains. The Gaussian distribution can be used to describe many natural and artificial processes. By isotropic Gaussian mutation each component of $\mathbf{x}$ is perturbed independently with a random number from a Gaussian distribution with zero mean and standard deviation σ.

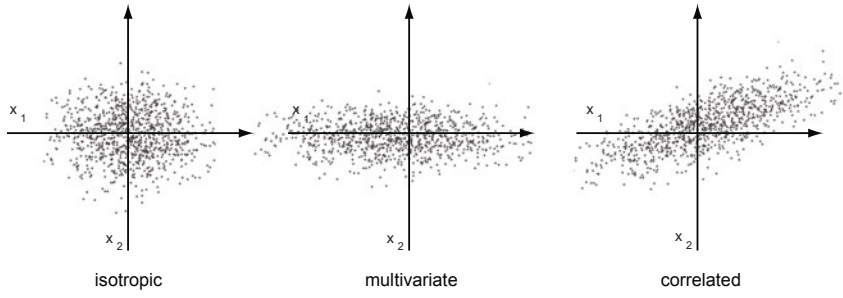

Fig. 4.3 Gaussian mutation: isotropic Gaussian mutation (left) uses one step size σ for each dimension, multivariate Gaussian mutation (middle) allows independent step sizes for each dimension, and correlated mutation (right) introduces an additional rotation of the coordinate system

The standard deviation σ plays the role of the mutation strength, and is also known as step size. The step size σ can be kept constant, but convergence can be improved by adapting σ according to the local solution space characteristics. In case of high success rates, i.e., a high number of offspring solutions being better than their parents, large step sizes are advantageous in order to promote the exploration of the search space. This is often the case at the beginning of the search. Small step sizes are appropriate for low success rates. This is frequently adequate in later phases of the search, when the optimization history can be exploited while the optimum is approximated. An example for an adaptive control of step sizes is the $1/5$-th success rule by Rechenberg [39] that increases the step size if the success rate is over $1/5$-th, and decreases it, if the success rate is lower.

The isotropic Gaussian mutation can be extended to the multivariate Gaussian mutation by introducing a step size vector σ with independent step sizes σ_i. Figure 4.3 illustrates the differences between isotropic Gaussian mutation (left) and the multivariate Gaussian mutation (middle). The multivariate variant considers a mutation ellipsoid that adapts flexibly to local solution space characteristics.

Even more flexibility can be obtained through the correlated mutation proposed by Schwefel [44] that aligns the coordinate system to the solution space characteristics. The mutation ellipsoid is rotated by means of an orthogonal matrix, and this rotation can be modified along iterations. The rotated ellipsoid is also shown in Figure 4.3 (right). The covariance matrix adaptation evolution strategies (CMA-ES) and derivates [45, 46] are self-adapting control strategies based on an automatic alignment of the coordinate system.

4.4.1.4 Selection

The counterpart of the variation operators mutation and recombination is selection. Selection gives the evolutionary search a direction. Based on the fitness, a subset of the population is selected, while the rest is rejected. In EAs the selection operator

can be utilized at two points. Mating selection selects individuals for recombination. Another popular selection operator is survivor selection, corresponding to the Darwinian principle of *survival of the fittest*. Only the individuals selected by survivor selection are allowed to confer genetic material to the following generation. The elitist strategies plus and comma selection choose the μ best solutions and are usually applied for survivor selection. Plus selection selects the μ best solutions from the union $\mathscr{P} \cup \mathscr{P}'$ of the last parental population $\mathscr{P}$ and the current offspring population $\mathscr{P}'$, and is denoted by $(\mu + \lambda)$-EA. In contrast to plus selection, comma selection, which is denoted by (μ, λ)-EA, selects exclusively from the offspring population, neglecting the parental population – even if individuals have superior fitness. Though disregarding these apparently promising solutions may seem to be disadvantageous, this strategy that prefers the new population to the old population can be useful to avoid being trapped in unfavorable local optima.

The deterministic selection scheme described in the previous paragraph is a characteristic feature of ES. Most evolutionary algorithms use selection schemes containing random components. An example is fitness proportionate selection (also called roulette-wheel selection) popular in the early days of genetic algorithms [41]. Another example is tournament selection, a widely used selection scheme for EAs. Here, the candidate with the highest fitness out of a randomly chosen subset of the population is selected to the new population. The stochastic-based selection schemes permit survival of not-so-fit individuals and thus helps with preventing premature convergence and preserving the genetic material that may come in handy at later stages of the optimization process.

4.4.2 Estimation of Distribution Algorithms

Related to evolutionary algorithms are estimation of distribution algorithms (EDAs). They also operate with a set of candidate solutions. Similar to ES, a random set of points is initially generated, and the objective function is computed for all these points. The core of EDAs are successive steps where distributions of the best solutions within a population are estimated, and a new population is sampled according to the previous distribution estimation.

The principle has been extended in a number of different manners. Most EDAs make use of parametric distributions, i.e., the parameters of distribution functions are determined in the estimation step. The assumption of a Gaussian distribution is frequent in EDAs. EDAs may suffer from premature convergence. The weighted variance estimator introduced in [47] has been observed to alleviate that convergence issue. Adaptive variance scaling [48], i.e., the variance can be increased if good solutions are found, otherwise it is decreased, has also been suggested to avoid early stagnation. The sampling process can be enhanced by anticipated mean shift (AMS; [49]). In this approach, about two thirds of the population are sampled regularly, and the rest is shifted in the direction of a previously estimated gradient. If this estimate is accurate, all the shifted individuals, together with part of the non-shifted

individuals, may survive, and the variance estimate in the direction of the gradient could be larger than without AMS.

4.4.3 Particle Swarm Optimization

Similar to evolutionary algorithms, particle swarm optimization (PSO) is a population approach with stochastic components. Introduced by Kennedy and Eberhart [50], it is inspired by the movement of natural swarms and flocks. The algorithm utilizes particles with a position $\mathbf{x}$ that corresponds to the optimization variables, and a speed $\mathbf{v}$ which is similar to the mutation strength in evolutionary computation. The principle of particle swarm optimization is based on the idea that the particles move in the solution space, influencing each other with stochastic changes, while previous successful solutions act as attractors.

In each iteration the position of particle $\mathbf{x}$ is updated by adding the current velocity $\mathbf{v}$

$$\mathbf{x}' := \mathbf{x} + \mathbf{v}. \tag{4.4}$$

The velocity is updated as follows

$$\mathbf{v}' := \mathbf{v} + c_1 r_1 (\mathbf{x}_p^* - \mathbf{x}) + c_2 r_2 (\mathbf{x}_s^* - \mathbf{x}), \tag{4.5}$$

where $\mathbf{x}_p^*$ and $\mathbf{x}_s^*$ denote the best previous positions of the particle and of the swarm, respectively. The weights c_1 and c_2 are acceleration coefficients that determine the bias of the particle towards its own or the swarm history. The recommendation given by Kennedy and Eberhart is to set both parameters to one. The stochastic components r_1 and r_2 are uniformly drawn from the interval $[0,1]$, and can be used to promote the global exploration of the search space.

4.4.4 Differential Evolution

Another population-based optimization approach is differential evolution (DE), originally introduced by Storn and Price [51]. As the algorithms in the previous three subsections, DE exploits a set of candidate solutions (agents in DE lexicon). New agents are allocated in the search space by combining the positions of other existing agents. More specifically, an intermediate agent is generated from two agents randomly chosen from the current population. This temporary agent is then mixed with a predetermined target agent. The new agent is accepted for the next generation if and only if it yields reduction in objective function.

The basic DE algorithm uses a random initialization. A new agent $\mathbf{y} = [y_1, \ldots, y_n]$ is created from the existing one $\mathbf{x} = [x_1, \ldots, x_n]$ as indicated below.

1. Three agents $\mathbf{a} = [a_1, \ldots, a_n]$, $\mathbf{b} = [b_1, \ldots, b_n]$ and $\mathbf{c} = [c_1, \ldots, c_n]$ are randomly extracted from the population (all distinct from each other and from $\mathbf{x}$).
2. A position index $p \in \{1, \ldots, N\}$ is determined randomly.

3. The position of the new agent $\mathbf{y}$ is computed by means of the following iteration over $i \in \{1,\ldots,n\}$:
   i) select a random number $r_i \in (0,1)$ with uniform probability distribution;
   ii) if $i = p$ or $r_i < CR$ let $y_i = a_i + F(b_i - c_i)$, otherwise let $y_i = x_i$; here, $F \in [0,2]$ is the differential weight and $CR \in [0,1]$ is the crossover probability, both defined by the user;
   iii) if $f(\mathbf{y}) < f(\mathbf{x})$ then replace $\mathbf{x}$ by $\mathbf{y}$; otherwise reject $\mathbf{y}$ and keep $\mathbf{x}$.

Although DE resembles some other stochastic optimization techniques, unlike traditional EAs, DE perturbs the solutions in the current generation vectors with scaled differences of two randomly selected agents. As a consequence, no separate probability distribution has to be used, and thus the scheme presents some degree of self-organization. Additionally, DE is simple to implement, uses very few control parameters, and has been observed to perform satisfactorily in a number of multi-modal optimization problems [52].

4.5 Guidelines for Generally Constrained Optimization

We now describe nonlinear constraint handling techniques that can be combined with the optimization methods presented in Sections 4.3 and 4.4.

4.5.1 Penalty Functions

The penalty function method (cf. [7]) for general optimization constraints involves modifying the objective function with a penalty term that depends on the constraint violation $h : \mathbb{R}^n \to \mathbb{R}$. The original optimization problem in (4.1) is thus modified as follows:

$$\min_{\mathbf{x}\in\Omega\subset\mathbb{R}^n} f(\mathbf{x}) + \rho\, h(\mathbf{x}), \tag{4.6}$$

where $\rho > 0$ is a penalty parameter. The modified optimization problem may still have constraints that are straightforward to handle.

If the penalty parameter is iteratively increased (tending to infinity), the solution of (4.6) converges to that of the original problem in (4.1). However, in certain cases, a finite (and fixed) value of the penalty parameter ρ also yields the correct solution (this is the so-called *exact* penalty; see [7]). For exact penalties, the modified cost function is not smooth around the solution [7], and thus the corresponding optimization problem can be significantly more involved than that in (4.6). However, one can argue that in the derivative-free case exact penalty functions may in some cases be attractive. Common definitions of $h(\mathbf{x})$, where I and J denote the indices that refer to inequality and equality constraints, respectively, are

$$h(\mathbf{x}) = \tfrac{1}{2}\left(\sum_{i\in I} \max(0, g_i(\mathbf{x}))^2 + \sum_{j\in J} g_i^2(\mathbf{x})\right)$$

the quadratic penalty and

$$h(\mathbf{x}) = \sum_{i \in I} \max(0, g_i(\mathbf{x})) + \sum_{j \in J} |g_i(\mathbf{x})|$$

an exact penalty. It should be noticed that by these penalties, the search considers both feasible and infeasible points. Those optimization methodologies where the optimum can be approached from outside the feasible region are known as exterior methods.

The log-barrier penalty (for inequality constraints)

$$h(\mathbf{x}) = -\sum_{i \in I} \log(-g_i(\mathbf{x}))$$

has to be used with a decreasing penalty parameter (tending to zero). This type of penalty methods (also known as barrier methods) confines the optimization to the feasible region of the search space. Interior methods aim at reaching the optimum from inside the feasible region.

In [53], non-quadratic penalties have been suggested for pattern search techniques. However, the optimizations presented in that work are somewhat simpler than those found in many practical situations, so the recommendations given might not be generally applicable. In future research, it will be useful to explore further the performance of different penalty functions in the context of simulation-based optimization.

4.5.2 Augmented Lagrangian Method

As mentioned above, in exterior penalty function methods, as $\rho \to \infty$ the local minimum is approached from outside the feasible region. Not surprisingly, there is a way to shift the feasible region so one is able to determine the local solution for a finite penalty parameter. See, for example, [54, 55] for original references, and also [7], Chapter 17.

Augmented Lagrangian methods [56, 57] aim at minimizing, in the equality constraint case, the following extended cost function

$$\min_{\mathbf{x} \in \Omega \subset \mathbb{R}^n} f(\mathbf{x}) + \tfrac{1}{2}\rho \, \|\mathbf{g}(\mathbf{x})\|_2^2 + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) \,, \tag{4.7}$$

where $\rho > 0$ is a penalty parameter, and $\boldsymbol{\lambda} \in \mathbb{R}^m$ are Lagrange multipliers. This cost function can indeed be interpreted as a quadratic penalty with the constraints shifted by some constant term [56]. As in penalty methods, the penalty parameter and the Lagrange multipliers are iteratively updated. It turns out that if one is sufficiently stationary for Equation (4.7), which is exactly when we have good approximations for the Lagrange multipliers, then $\boldsymbol{\lambda}$ can be updated via

$$\boldsymbol{\lambda}^+ = \boldsymbol{\lambda} + \rho \, \mathbf{g}(\mathbf{x}) \,, \tag{4.8}$$

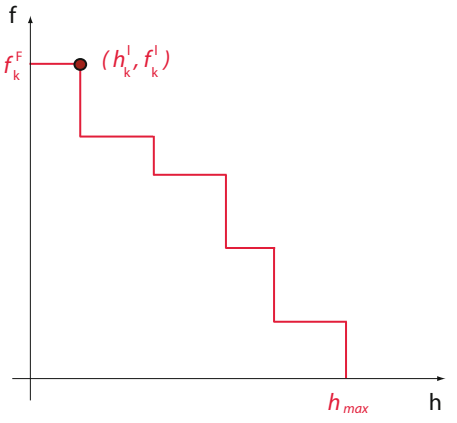

Fig. 4.4 An idealized (pattern search) filter at iteration k (modified from [19])

where $\boldsymbol{\lambda}^+$ denotes the updated Lagrange multipliers. Otherwise one should increase the penalty parameter ρ (say by multiplying it by 10). The Lagrange multipliers are typically initialized to zero. What is significant is that one can prove (see e.g. [56]) that after a finite number of iterations the penalty parameter is never updated, and that the whole scheme eventually converges to a solution of the original optimization problem in (4.1). Inequality constraints can also be incorporated in the augmented Lagrangian framework by introducing slack variables and simple bounds [56]. The augmented Lagrangian approach can be combined with most optimization algorithms. For example, refer to [58] for a nonlinear programming methodology based on generalized pattern search.

4.5.3 Filter Method

A relatively recent approach that avoids using a penalty parameter and has been rather successful is the class of so-called filter methods [59, 7]. Using filters, the original problem (4.1) is typically viewed as a bi-objective optimization problem. Besides minimizing the cost function $f(\mathbf{x})$, one also seeks to reduce the constraint violation $h(\mathbf{x})$. The concept of dominance, crucial in multi-objective optimization, is defined as follows: the point $\mathbf{x}_1 \in \mathbb{R}^n$ dominates $\mathbf{x}_2 \in \mathbb{R}^n$ if and only if either $f(\mathbf{x}_1) \leq f(\mathbf{x}_2)$ and $h(\mathbf{x}_1) < h(\mathbf{x}_2)$, or $f(\mathbf{x}_1) < f(\mathbf{x}_2)$ and $h(\mathbf{x}_1) \leq h(\mathbf{x}_2)$. A filter is a set of pairs $(h(\mathbf{x}), f(\mathbf{x}))$, such that no pair dominates another pair. In practice, a maximum allowable constraint violation $h_{\max}$ is specified. This is accomplished by introducing the pair $(h_{\max}, -\infty)$ in the filter. An idealized filter (at iteration k) is shown in Figure 4.4.

A filter can be understood as essentially an add-on for an optimization procedure. The intermediate solutions proposed by the optimization algorithm at a given

iteration are accepted if they are not dominated by any point in the filter. The filter is updated at each iteration based on all the points evaluated by the optimizer. We reiterate that, as for exterior methods, the optimization search is enriched by considering infeasible points, although the ultimate solution is intended to be feasible (or very nearly so). Filters are often observed to lead to faster convergence than methods that rely only on feasible iterates.

Pattern search optimization techniques have been previously combined with filters [60]. In Hooke-Jeeves direct search, the filter establishes the acceptance criterion for each (unique) new solution. For schemes where, in each iteration, multiple solutions can be accepted by the filter (such as in GPS), the new polling center must be selected from the set of validated points. When the filter is not updated in a particular iteration (and thus the best feasible point is not improved), the pattern size is decreased. As in [60], when we combine GPS with a filter, the polling center at a given iteration will be the feasible point with lowest cost function or, if no feasible points remain, it will be the infeasible point with lowest constraint violation. These two points, $\left(0, f_k^{\mathrm{F}}\right)$ and $\left(h_k^{\mathrm{I}}, f_k^{\mathrm{I}}\right)$, respectively, are shown in Figure 4.4 (it is assumed that both points have just been accepted by the filter, and thus it makes sense to use one of them as the new polling center). Refer to [60] and [61] for more details on pattern search filter methods.

4.5.4 Other Approaches

We will now briefly overview a number of constraint handling methodologies that have been proposed for evolutionary algorithms. Repair algorithms [62, 63] project infeasible solutions back to the feasible space. This projection is in most cases accomplished in an approximate manner, and can be as complex as solving the optimization problem itself. Repair algorithms can be seen as local procedures that aim at reducing constraint violation. In the so-called Baldwinian case, the fitness of the repaired solution replaces the fitness of the original (infeasible) solution. In the Lamarckian case, feasible solutions prevail over infeasible solutions.

Constraint-handling techniques borrowed from multi-objective optimization are based on the idea of dealing with each constraint as an additional objective [64, 65, 66, 67, 68, 69]. Under this assumption, multi-objective optimization methods such as NSGA-II [70] or SPEA [71] can be applied. The output of a multi-objective approach for constrained optimization is an approximation of a Pareto set that involves the objective function and the constraints. The user may then select one or more solutions from the Pareto set. A simpler but related and computationally less expensive procedure is the behavioral memory method presented in [72]. This evolutionary method concentrates on minimizing the constraint violation of each constraint sequentially, and the objective function is addressed separately afterwards. However, treating objective function and constraints independently may yield in many cases infeasible solutions.

Further constraint handling methods have been proposed in EA literature that do not rely either on repair algorithms or multi-objective approaches. In [73] a technique based on a multi-membered evolution strategy with a feasibility comparison mechanism is introduced. The dynamic multi-swarm particle optimizer studied in [74] makes use of a set of sub-swarms that focus on different constraints, and is coupled with a local search algorithm (sequential quadratic programming).

4.6 Concluding Remarks

In this chapter, we have concentrated on methods for solving optimization problems without derivates. The existence of local optima makes a hard optimization problem even harder. Many methods have been proposed to solve non-convex optimization problems. The approaches range from pattern search for local optimization problems to stochastic bio-inspired search heuristics for multi-modal problems. Deterministic local methods are guaranteed to find local optima, and restart variants can be applied to avoid unsatisfactory solutions. Stochastic methods are not guaranteed to find the global optimum, but in some practical cases they can be beneficial.

The hybridization between local and global optimizers has led to a paradigm sometimes called memetic algorithms or hybrid metaheuristics [75, 76]. A number of hybridizations have been proposed, but they are often tailored to specific problem types and search domains due to their specific operators and methods. In the memetic method introduced in [77] for continuous search spaces, a gradient-based scheme is combined with a deterministic perturbation component. The local optimization procedure for real-valued variables described in [78] is based on variable neighborhood search. It would be very useful if in future research some effort is dedicated to better understand from a theoretical point of view the hybridization of local and global optimization algorithms.

Most problems that can be found in practice present constraints. We have outlined a number of constraint handling techniques that can be incorporated in a derivative-free optimization framework. Though penalty functions are appealing due to their simplicity, some of the other approaches mentioned here may be more efficient and still of a relatively easy implementation.

Multi-objective optimization is an important challenge for derivative-free methodologies. Some of the evolutionary techniques mentioned above have performed successfully in some not especially involved multi-objective test cases. Other areas where derivative-free optimization could potentially be very helpful include dynamic optimization, mixed-integer nonlinear programming, and optimization under uncertainty (stochastic programming).

Acknowledgements. We are grateful to the industry sponsors of the Stanford Smart Fields Consortium for partial funding of this work, and also to J. Smith for his valuable suggestions.

References

[1] Pironneau, O.: On optimum design in fluid mechanics. Journal of Fluid Mechanics 64, 97–110 (1974)

[2] Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Review 45(3), 385–482 (2003)

[3] Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MPS-SIAM Series on Optimization. MPS-SIAM (2009)

[4] Gilmore, P., Kelley, C.T.: An implicit filtering algorithm for optimization of functions with many local minima. SIAM Journal on Optimization 5, 269–285 (1995)

[5] Kelley, C.T.: Iterative Methods for Optimization. In: Frontiers in Applied Mathematics, SIAM, Philadelphia (1999)

[6] Dennis Jr., J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM's Classics in Applied Mathematics Series. SIAM, Philadelphia (1996)

[7] Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Heidelberg (2006)

[8] Schilders, W.H.A., van der Vorst, H.A., Rommes, J.: Model Order Reduction: Theory, Research Aspects and Applications. Mathematics in Industry Series. Springer, Heidelberg (2008)

[9] Conn, A.R., Gould, N.I.M.: Toint, Ph.L.: Trust-Region Methods. MPS-SIAM Series on Optimization. MPS-SIAM (2000)

[10] Meza, J.C., Martinez, M.L.: On the use of direct search methods for the molecular conformation problem. Journal of Computational Chemistry 15, 627–632 (1994)

[11] Booker, A.J., Dennis Jr., J.E., Frank, P.D., Moore, D.W., Serafini, D.B.: Optimization using surrogate objectives on a helicopter test example. In: Borggaard, J.T., Burns, J., Cliff, E., Schreck, S. (eds.) Computational Methods for Optimal Design and Control, pp. 49–58. Birkhäuser, Basel (1998)

[12] Marsden, A.L., Wang, M., Dennis Jr., J.E., Moin, P.: Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation. Journal of Fluid Mechanics 572, 13–36 (2003)

[13] Duvigneau, R., Visonneau, M.: Hydrodynamic design using a derivative-free method. Structural and Multidisciplinary Optimization 28, 195–205 (2004)

[14] Fowler, K.R., Reese, J.P., Kees, C.E., Dennis Jr., J.E., Kelley, C.T., Miller, C.T., Audet, C., Booker, A.J., Couture, G., Darwin, R.W., Farthing, M.W., Finkel, D.E., Gablonsky, J.M., Gray, G., Kolda, T.G.: Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. Advances in Water Resources 31(5), 743–757 (2008)

[15] Oeuvray, R., Bierlaire, M.: A new derivative-free algorithm for the medical image registration problem. International Journal of Modelling and Simulation 27, 115–124 (2007)

[16] Marsden, A.L., Feinstein, J.A., Taylor, C.A.: A computational framework for derivative-free optimization of cardiovascular geometries. Computational Methods in Applied Mechanics and Engineering 197, 1890–1905 (2008)

[17] Artus, V., Durlofsky, L.J., Onwunalu, J.E., Aziz, K.: Optimization of nonconventional wells under uncertainty using statistical proxies. Computational Geosciences 10, 389–404 (2006)

[18] Dadashpour, M., Echeverría Ciaurri, D., Mukerji, T., Kleppe, J., Landrø, M.: A derivative-free approach for the estimation of porosity and permeability using time-lapse seismic and production data. Journal of Geophysics and Engineering 7, 351–368 (2010)
[19] Echeverría Ciaurri, D., Isebor, O.J., Durlofsky, L.J.: Application of derivativefree methodologies for generally constrained oil production optimization problems. International Journal of Mathematical Modelling and Numerical Optimisation 2(2), 134–161 (2011)
[20] Onwunalu, J.E., Durlofsky, L.J.: Application of a particle swarm optimization algorithm for determining optimum well location and type. Computational Geosciences 14, 183–198 (2010)
[21] Zhang, H., Conn, A.R., Scheinberg, K.: A derivative-free algorithm for leastsquares minimization. SIAM Journal on Optimization 20(6), 3555–3576 (2010)
[22] Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization 7(1), 1–25 (1997)
[23] Audet, C., Dennis Jr., J.E.: Analysis of generalized pattern searches. SIAM Journal on Optimization 13(3), 889–903 (2002)
[24] Audet, C., Dennis Jr., J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on Optimization 17(1), 188–217 (2006)
[25] Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the ACM 8(2), 212–229 (1961)
[26] Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. Technical report DAMTP 2004/NA5, Dept. of Applied Mathematics and Theoretical Physics, University of Cambridge (2004)
[27] Oeuvray, R., Bierlaire, M.: BOOSTERS: a derivative-free algorithm based on radial basis functions. International Journal of Modelling and Simulatio 29(1), 26–36 (2009)
[28] Metropolis, N., Rosenbluth, A., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. Chemical Physics 21(6), 1087–1092 (1953)
[29] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.: Optimization by simulated annealing. Science 220(4498), 671–680 (1983)
[30] Glover, F.: Tabu search – part I. ORSA Journal on Computing 1(3), 190–206 (1990)
[31] Glover, F.: Tabu search – part II. ORSA Journal on Computing 2(1), 4–32 (1990)
[32] Dorigo, M.: Optimization, Learning and Natural Algorithms. PhD thesis, Dept. of Electronics, Politecnico di Milano (1992)
[33] Dorigo, M., Stützle, T.: Ant Colony Optimization. Prentice-Hall, Englewood Cliffs (2004)
[34] Farmer, J., Packard, N., Perelson, A.: The immune system, adaptation and machine learning. Physica 2, 187–204 (1986)
[35] Castro, L.N.D., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence. Springer, Heidelberg (2002)
[36] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
[37] Fogel, D.B.: Artificial Intelligence through Simulated Evolution. Wiley, Chichester (1966)
[38] Beyer, H.-G., Schwefel, H.-P.: Evolution strategies - a comprehensive introduction. Natural Computing 1, 3–52 (2002)
[39] Rechenberg, I.: Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog (1973)

[40] Schwefel, H.-P.: Numerische Optimierung von Computer-Modellen mittel der Evolutionsstrategie. Birkhäuser, Basel (1977)
[41] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
[42] Holland, J.H.: Hidden Order: How Adaptation Builds Complexity. Addison- Wesley, London (1995)
[43] Beyer, H.-G.: An alternative explanation for the manner in which genetic algorithms operate. BioSystems 41(1), 1–15 (1997)
[44] Schwefel, H.-P.: Adaptive mechanismen in der biologischen evolution und ihr einfluss auf die evolutionsgeschwindigkeit. In: Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik, TU Berlin (1974)
[45] Beyer, H.-G., Sendhoff, B.: Covariance matrix adaptation revisited – the CMSA evolution strategy –. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 123–132. Springer, Heidelberg (2008)
[46] Ostermeier, A., Gawelczyk, A., Hansen, N.: A derandomized approach to selfadaptation of evolution strategies. Evolutionary Computation 2(4), 369–380 (1994)
[47] Teytaud, F., Teytaud, O.: Why one must use reweighting in estimation of distributionalgorithms. In: Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 453–460 (2009)
[48] Grahl, J., Bosman, P.A.N., Rothlauf, F.: The correlation-triggered adaptive variance scaling idea. In: Proceedings of the 8th Annual conference on Genetic and Evolutionary Computation (GECCO 2006), pp. 397–404 (2006)
[49] Bosman, P.A.N., Grahl, J., Thierens, D.: Enhancing the performance of maximum–likelihood gaussian eDAs using anticipated mean shift. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 133–143. Springer, Heidelberg (2008)
[50] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
[51] Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
[52] Chakraborty, U.: Advances in Differential Evolution. SCI. Springer, Heidelberg (2008)
[53] Griffin, J.D., Kolda, T.G.: Nonlinearly-constrained optimization using asynchronous parallel generating set search. Technical report SAND2007-3257, Sandia National Laboratories (2007)
[54] Hestenes, M.R.: Multiplier and gradients methods. Journal of Optimization Theory and Applications 4(5), 303–320 (1969)
[55] Powell, M.J.D.: A method for nonlinear constraints in minimization problems. In: Fletcher, R. (ed.) Optimization, pp. 283–298. Academic Press, London (1969)
[56] Conn, A.R., Gould, N.I.M., Toint, P.L.: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM Journal on Numerical Analysis 28(2), 545–572 (1991)
[57] Conn, A.R., Gould, N.I.M., Toint, P.L.: LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A). Computational Mathematics. Springer, Heidelberg (1992)
[58] Lewis, R.M., Torczon, V.: A direct search approach to nonlinear programming problems using an augmented Lagrangian method with explicit treatment of the linear constraints. Technical report WM-CS-2010-01, Dept. of Computer Science, College of William & Mary (2010)

[59] Fletcher, R., Leyffer, S., Toint, P.L.: A brief history of filter methods. Technical report ANL/MCS/JA-58300, Argonne National Laboratory (2006)
[60] Audet, C., Dennis Jr., J.E.: A pattern search filter method for nonlinear programming without derivatives. SIAM Journal on Optimization 14(4), 980–1010 (2004)
[61] Abramson, M.A.: NOMADm version 4.6 User's Guide. Dept. of Mathematics and Statistics, Air Force Institute of Technology (2007)
[62] Belur, S.V.: CORE: constrained optimization by random evolution. In: Koza, J.R. (ed.) Late Breaking Papers at the Genetic Programming 1997 Conference, pp. 280–286 (1997)
[63] Coello Coello, C.A.: Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
[64] Parmee, I.C., Purchase, G.: The development of a directed genetic search technique for heavily constrained design spaces. In: Parmee, I.C. (ed.) Proceedings of the Conference on Adaptive Computing in Engineering Design and Control, pp. 97–102. University of Plymouth (1994)
[65] Surry, P.D., Radcliffe, N.J., Boyd, I.D.: A multi-objective approach to constrained optimisation of gas supply networks: the COMOGA method. In: Fogarty, T.C. (ed.) AISB-WS 1995. LNCS, vol. 993, pp. 166–180. Springer, Heidelberg (1995)
[66] Coello Coello, C.A.: Treating constraints as objectives for single-objective evolutionary optimization. Engineering Optimization 32(3), 275–308 (2000)
[67] Coello Coello, C.A.: Constraint handling through a multiobjective optimization technique. In: Proceedings of the 8th Annual conference on Genetic and Evolutionary Computation (GECCO 1999), pp. 117–118 (1999)
[68] Jimenez, F., Verdegay, J.L.: Evolutionary techniques for constrained optimization problems. In: Zimmermann, H.J. (ed.) 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT 1999). Springer, Heidelberg (1999)
[69] Mezura-Montes, E., Coello Coello, C.A.: Constrained optimization via multiobjective evolutionary algorithms. In: Knowles, J., Corne, D., Deb, K., Deva, R. (eds.) Multiobjective Problem Solving from Nature. Natural Computing Series, pp. 53–75. Springer, Heidelberg (2008)
[70] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
[71] Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100 (2002)
[72] Schoenauer, M., Xanthakis, S.: Constrained GA optimization. In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms (ICGA 1993), pp. 573–580. Morgan Kaufmann, San Francisco (1993)
[73] Montes, E.M., Coello Coello, C.A.: A simple multi-membered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation 9(1), 1–17 (2005)
[74] Liang, J., Suganthan, P.: Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In: Yen, G.G., Lucas, S.M., Fogel, G., Kendall, G., Salomon, R., Zhang, B.-T., Coello Coello, C.A., Runarsson, T.P. (eds.) Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC 2006), pp. 9–16. IEEE Press, Los Alamitos (2006)

[75] Raidl, G.R.: A unified view on hybrid metaheuristics. In: Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez Pérez, M., Roli, A., Sampels, M. (eds.) HM 2006. LNCS, vol. 4030, pp. 1–12. Springer, Heidelberg (2006)

[76] Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)

[77] Griewank, A.: Generalized descent for global optimization. Journal of Optimization Theory and Applications 34, 11–39 (1981)

[78] Duran Toksari, M., Güner, E.: Solving the unconstrained optimization problem by a variable neighborhood search. Journal of Mathematical Analysis and Applications 328(2), 1178–1187 (2007)