Model Optimization with Evolutionary Algorithms

Thomas Bartz-Beielstein[1], Mike Preuss[2],
Karlheinz Schmitt[2], and Hans–Paul Schwefel[2]

[1] Cologne University of Applied Sciences, Cologne, Germany,
thomas.bartz-beielstein@fh-koeln.de,
[2] University of Dortmund, Dortmund, Germany,
{mike.preuss,karlheinz.schmitt,hans-paul.schwefel}@cs.uni-dortmund.de

Does one need more than one optimization method? Or, stated differently, is there an optimal optimization method? Following from the No Free Lunch theorem (NFL, Wolpert and Macready [1]), in the general case—without clearly specified task—there is not. For every single task, creating a specialized method would be advantageous. Unfortunately, this requires (i) a lot of effort, and (ii) extensive knowledge about the treated problem, and is thus not practiced. Alternatively, two strategies are usually followed when tackling a 'new' optimization problem:

- Adapt an existing algorithm to the problem in its current form, and/or
- model/formulate the problem appropriately for an existing algorithm.

The first strategy modifies the algorithm design, whereas the second strategy modifies the problem design. These designs will be discussed in detail in the remainder of this article. Whereas 'traditional' mathematical optimization approaches mostly favor the second approach, it may provoke unwanted side-effects: One has to make sure that the most important features of the original problem are taken over into the model. E.g., matching the problem to an existing algorithm may obscure its real global or good local optimizers so that they become unreachable for the optimization algorithm. Besides, many existing algorithms require the problem to fulfill properties it obviously or possibly does not, e.g. continuity and differentiability. Particularly, in cases where computing the quality value of a solution candidate requires running a complex simulation software, one seldomly knows in advance which properties the underlying (unknown) objective function possesses.

When nothing more than quality determining response values for any set of input variables are known for a problem, we speak of *black box* optimization. In the single-objective case, the common notion of an objective function and its global optimum/global optimizers—as given in eqn. 1 for unconstrained problems—is still useful. However, global optimizers, the set of input vectors $\mathbf{x}$ for which $f(\mathbf{x})$ is optimal, cannot be determined analytically. An empirical trial and error method is the only way to find them.

$$f^{*\,G} = \min\{f(\mathbf{x}) | \mathbf{x} \in X\} \tag{1}$$

The black box concept immediately leads to *direct search* methods—such a method only utilizes objective function responses and "does not 'in its heart' develop an approximate gradient", as Wright [2] puts it. As far back as in the 1960s, many direct search methods have been invented, e.g. the famous *Nelder-Mead simplex algorithm* [3]. At the same time, the first steps into the world of *evolutionary computation* (EC) were taken, presenting very simple versions of what is now subsumed under the unified denotation *evolutionary algorithms* (EA). These do not only use bio-inspired heuristics, they also employ randomness. However, the extensive use of random numbers and the fragmentary theory supporting EAs may be considered a drawback. Nevertheless, these optimization methods have demonstrated their problem solving capability in numerous real-world applications.

Interestingly, in recent years, the mathematical optimization community has again shown increased interest in direct search methods, e.g. Kolda et al. [4]. This may have to do with (i) the fact that these techniques simply did not go extinct on the practitioners side, and (ii) improved theoretical analysis methods that now help tackling heuristic algorithms. In computer science, the growing field of *randomized algorithms* is exclusively dealing with algorithms employing random numbers — not only in optimization. Motwani and Raghavan [5] give an overview.

This section targets at introducing the main EA concepts and specialized techniques for three important application areas: Multiobjective optimization, optimization under uncertainty, and multimodal optimization. These are relevant to the topic of this book as they are closely interrelated and often encountered conjoined in real-world applications.

Historical roots Although there have been precursors in proposing the utilization of evolutionary concepts for optimization tasks, as e.g. Bremermann [6] (also see Fogel's fossil record [7]), invention and development of the first evolutionary algorithms is nowadays attributed to a handful of pioneers who independently suggested three different approaches.

- Fogel, Owens, and Walsh introduced evolutionary programming (EP) [8], at first focused at evolving finite automata, later on modified into a numerical optimization method.
- Genetic algorithms (GAs), as laid out by Holland [9], mainly dealed with combinatorial problems and consequentially started with binary strings, inspired by the genetic code found in natural life.
- Evolution strategies (ESs) as brought up by Rechenberg [10] and Schwefel [11] began with solving experimental engineering problems by hand using discrete/integer parameters, but turning to real-valued representations when numerical problems had to be solved.

In the early 1990s, a fourth branch of evolutionary algorithms emerged, explicitly performing optimization of programs: Genetic programming (GP), suggested by Koza [12]. Since about the same time, these four techniques are collec-

tively referred to as evolutionary algorithms, building the core of the evolutionary computation (EC) field.

What is an evolutionary algorithm? Today, there is little doubt about components and general structure of an EA. It is understood as population based direct search algorithm with stochastic elements that in some sense mimics the organic evolution.

Besides initialization and termination as necessary constituents of every algorithm, EAs consist of three important factors: A number of search operators, an imposed control flow (fig. 1), and a representation that maps adequate variables to implementable solution candidates.

Although different EAs may put different emphasis on the search operators mutation and recombination, their general effects are not in question. Mutation means neighborhood based movement in search space that includes the exploration of the 'outer space' currently not covered by a population, whereas recombination rearranges existing information and so focuses on the 'inner space.' Selection is meant to introduce a bias towards better fitness values; GAs do so by regulating the crossover via mating selection, ESs utilize the environmental selection.

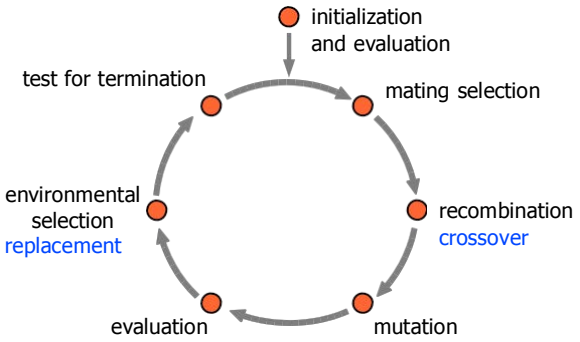

Fig. 1. The evolutionary cycle, basic working scheme of all EAs. Terms common for describing evolution strategies are used, alternative (GA) terms are added below.

A concrete EA may contain specific mutation, recombination, or selection operators, or call them only with a certain probability, but the control flow is usually left unchanged. Each of the consecutive cycles is termed a *generation*. Concerning the representation, it should be noted that most empiric studies are based on canonical forms as binary strings or real-valued vectors, whereas many real-world applications require specialized, problem dependent ones.

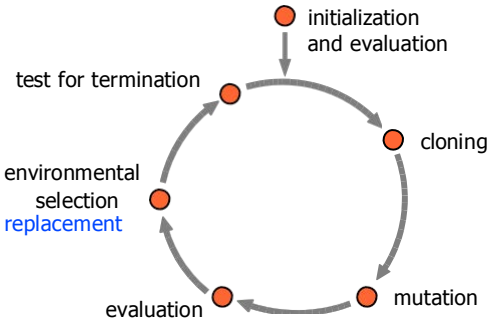

Fig. 2. The evolutionary cycle of a two-membered (1+1) evolution strategy.

For an in-depth coverage on the defining components of an EA and their connection to natural evolution, see Eiben and Schoenauer [13], Eiben and Smith [14], and Bäck, Fogel, and Michalewicz [15].

Evolution strategies In the following, we introduce the most important canonical ES variants for single objective optimization, which serve as basis for more specialized algorithms later on.

The (1 + 1)-*ES* The first ES, the so-called (1 + 1)-ES or *t*wo membered evolution strategy, uses one parent and one offspring only. Two rules have been applied to these candidate solutions:

1. Apply small, random changes to all variables simultaneously.
2. If the offspring solution is not worse (in terms of its function value) than the parent, take it as the new parent, otherwise retain the parent.

Schwefel [16] describes this algorithm as "the minimal concept for an imitation of organic evolution." The (1 + 1)-ES (fig. 2) is applied by many optimization practitioners to their optimization problem and included in this article for three reasons: (i) It is easy to implement, (ii) it requires only few exogenous parameters, and (iii) it defines a standard for comparisons.

The first (1 + 1)-ES used binomially distributed mutations for integer variables (Schwefel [17]). These have been replaced by Gaussian mutations for continuous variables. Rechenberg [18] already proposed a simple rule to control the mutation strength, the so-called 1/5 success rule. This simple ES requires the specification of at four parameters (factors), namely the adaptation interval, the required success rate, the step size adjustment factor[3], and the step size starting value.

[3] This is a constant factor c with $1 \leq c \leq 0.85$, the lower bound being theoretically near-optimal for simple model problems like the sphere model.

Population Based ESs Population based ESs use μ parents and λ offspring. Rechenberg introduced the first multimembered ES, the so-called $(\mu + 1)$-ES. It uses μ parents and one offspring and is referred to as the *steady-state* ES. Schwefel introduced the $(\mu + \lambda)$-ES, in which $\lambda \geq 1$ candidate solutions are created each generation, and the best μ out of all $\mu + \lambda$ individuals survive, and the (μ, λ)-ES, in which the parents are forgotten and only the best μ out of λ candidate solutions survive. These selection schemes will be discussed later in this section (p. 6).

A birth surplus is necessary for the (μ, λ)-ES, that is $\lambda > \mu$. Schwefel et al. [19] and Beyer and Schwefel [20] provide a comprehensive introduction to evolution strategies.

Note that whereas GAs rely upon a start population uniformly scattered in a closed search region, ESs—even if population based—may be started around any start vector like standard optimization algorithms, without lower and upper bounds for the variables.

Variation in ESs The use of populations enables an extension of the rather simple 1/5 success rule to control the mutation strength (Schwefel [11]). Beyer and Schwefel [20] propose some guidelines derived from the philosophy of Darwinian evolution to design these variation operators.

1. A *state* comprises a set of object and strategy parameter values $(x^{(t)}, s^{(t)})$. *Reachability* demands that any state can be reached within a finite number of iterations. This feature is necessary to prove (theoretically) global convergence.
2. Variation operators (mutation and recombination) should not introduce any bias, e.g. by considering only good candidate solutions. Variation operators are designed to *explore* the search space in contrast to selection operators that exploit the gathered information. Recombination works, according to Beyer [21], mainly as gene repair operator, not only as building block collection mechanism.
3. *Scalability* is the third criterion that should be fulfilled by variation operators: Small changes of the representation should cause small changes in the function values.

The standard ES recombination operators produce one offspring from a family of ρ parent individuals (usually $\rho = 2$). Consider a set of μ parental vectors of length N, representing either object or strategy parameters:

$$\{(x_{11}, \ldots, x_{1N}), (x_{21}, \ldots, x_{2N}), \ldots, (x_{\mu 1}, \ldots, x_{\mu N})\}. \tag{2}$$

Two recombination schemes are commonly used in ESs. Both use a set $R = \{r_1, r_2, \ldots, r_\rho\}$, that represents the indices of the mating partners. It is constructed by randomly (uniformly) choosing ρ numbers (with replacement or not) from the set $\{1, 2, \ldots, \mu\}$. *Discrete recombination* selects the entries of the offspring randomly from R, whereas *intermediary recombination* averages the ρ corresponding values of all mating pool members in each component of the newly generated vector.

Mutation is applied to the recombined intermediate solution. Mutation in multimembered ESs is a self-adaptive process that relies on the individual coupling of endogenous strategy parameters with object parameters. After being varied as described above, the strategy parameters (standard deviations, also called mean step sizes or mutation strengths) are applied to mutate the object parameters. To illustrate this procedure, algorithms with one common σ are considered first. To prevent negative standard deviations, mutation of this σ should be done multiplicatively. Beyer and Schwefel [20] discuss an additional argument for a multiplicative mutation of the mutation strength on the sphere model. It can be shown, that in expectation σ should be changed by a factor that only depends on N. Therefore, the mutation operator can be implemented as

$$\sigma^{(t+1)} = \sigma^{(t)} \cdot \exp(\tau z), \tag{3}$$

where z is a realization of an $N(0, 1)$ distributed random variable. The parameter τ is the so-called *learning rate*. The object variables are mutated next:

$$x^{(t+1)} = x^{(t)} + w, \tag{4}$$

where w is a realization of an $N(0, \sigma^{(t+1)})$ distributed random variable. The multiplicative mutation scheme for one σ can be extended to several strategy parameters $\sigma = (\sigma_1, \ldots, \sigma_N)$. Schwefel [22] proposes the following extended log-normal rule:

$$\sigma^{(t+1)} = \left(\sigma_1^{(t)} \exp(\tau z_1), \ldots, \sigma_d^{(t)} \exp(\tau z_N) \right), \tag{5}$$

where z_i are realizations of N standard normally distributed random variables, $1 \le i \le N$. This mutation scheme employs a single learning rate τ for all strategy parameters. An alternative procedure that utilizes a global and a local learning parameter τ_0 and τ, respectively, is suggested by Bäck and Schwefel [23]. Self-adaptive correlated mutations have already been introduced in 1974, see Schwefel [24] and Schwefel [25].

Selection in ESs Selection should direct the evolutionary search toward promising regions. In ESs, only candidate solutions with good function values are allowed to reproduce. The replacement (environmental selection) process is deterministic in contrast to the random processes used in GAs. This selection scheme is known as *truncation* or *breeding selection* in biology. The κ-selection scheme takes the age of candidate solutions into account: Only candidate solutions that are younger than κ generations may survive, regardless of their fitness. For $\kappa = 1$ this selection method is referred to as *comma-selection*: only offspring individuals can reproduce. The κ-selection is referred to as *plus-selection* for $\kappa = \infty$: Both the offspring and the parents belong to the mating pool. The plus-selection is an elitist selection scheme, because it guarantees the survival of the best individual found so far.

Table 1 summarizes important ES parameters [26]. These parameters build an algorithm design. In addition to algorithm designs optimization practitioners have to cope with problem designs which will be discussed next.

Table 1. Algorithm design of ES

Symbol	Parameter	Range
μ	Number of parent individuals	N
$\nu = \lambda/\mu$	Offspring-parent ratio	R_+
$\sigma_i^{(0)}$	Initial standard deviations	R_+
n_σ	Number of standard deviations. N denotes the problem dimension	$\{1, N\}$
τ_0, τ	Multiplier for mutation parameters	R_+
ρ	Mixing number	$\{1, \mu\}$
r_x	Recombination operator for object variables	{intermediary, discrete}
r_σ	Recombination operator for strategy variables	{intermediary, discrete}
κ	Maximum age	R_+

Ways to Cope with Uncertainty In the following, we will distinguish three types of parameters that influence experimental results [27]. The first type of parameter to be mentioned is a *control* parameter. Control parameters can be set by an experimenter to "control" the experiment.

The second type of parameter, so–called *environmental* parameter depends on the environment at the time the experiment is performed. Some authors refer to environmental parameters as "noise" parameters. Note, that environmental parameters include measurement errors such as falsely calibrated measurement instruments, inexact scales, scale reading errors, etc. Data preprocessing techniques were developed to reduce this source of error, which occurs in nearly every field setting. In some situations, environmental parameters can be treated as having a given distribution that is characteristic for the given experimental setup.

The third type of parameter, so–called *model* parameter describes the uncertainty of the mathematical modeling. First, we have to take into account that computer simulations require a model which simplifies the underlying real-world scenario. Therefore, simulation results are only approximations of the corresponding real-world data. Next, if stochastic (and not deterministic) simulations are considered, the measurements may be exact (because there is no environmental noise), but some of the models' parameters are random parameters. In some cases, there is a known (subjective) distribution which describes this uncertainty.

As an example, we consider a sequence of traffic signals along a certain route or elevators' movements in high-rise buildings. *Optimization via simulation* subsumes all problems in which the performance of the system is determined by running a computer simulation. If the result of a simulation run is a random variable, we cannot optimize the actual value of the simulation output, or a singular performance of the system. One goal of optimization via simulation may be to optimize the expected performance. In addition, consider a field study which was performed to validate the results from the computer simulation. This field study includes environmental parameters.

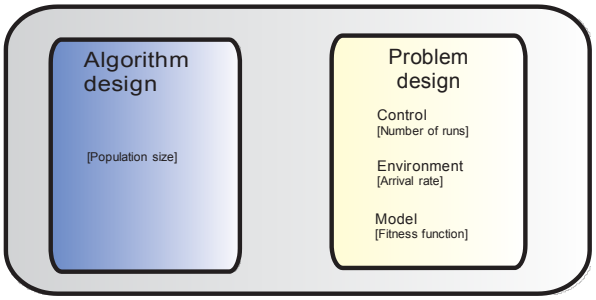

Fig. 3. Before an EA can be started, the optimization practitioner has to specify several parameters. Examples are shown in brackets. Environmental and model parameters can be affected by noise.

Summarizing, there are two fundamental sources of uncertainty (or noise) that can be described by environmental and model parameters. Figure 3 illustrates these parameters in the context of algorithm and problem designs.

The efficiency of the evaluation and selection method is a crucial point, since averaging over repeated runs reduces the efficiency of the optimization process.

The Impact of Noise on EAs Noise makes it difficult to compare different solutions and select the better ones. Noise affects the selection process in evolutionary algorithms: In every iteration, the best μ out of λ candidate solutions have to be determined.

Wrong decisions can cause *stagnation* of the search process: Over-valuated candidates—solutions that are only seemingly better—build a barrier around the optimum and prevent convergence. Or, even worse, the search process can be *misguided:* The selection of seemingly good candidates moves the search away from the optimum. This phenomenon occurs if the noise level is high and the probability of a correct selection is very small.

One may attempt to reduce the effect of noise explicitly (*explicit* averaging). The simplest way to do so is to sample a solution's function value n times, and use the average as estimate for the true expected function value. This reduces the standard deviation of the noise by a factor of $\sqrt{n}$, while increasing the running time by a factor of n.

In contrast to explicit averaging, some authors proposed *implicit* averaging, i.e., increasing the population size to cope with uncertainty in evolutionary optimization. Theoretical results lead to contradictory recommendations: In [28] the authors conclude that it is better to increase the population size whereas [29] shows that increasing the sample size is advantageous.

Further means used by evolutionary algorithms to cope with noise are averaging techniques based on statistical tests, local regression methods for function value estimation, or methods to vary the population size [30–36]. Because uncer-

tainties complicate the selection process for direct search methods, some authors suggested modified selection operators.

A Taxonomy of Selection Methods As introduced above, noise affects selection. Following Bechhofer, Santner, and Goldsman [37] and Bartz-Beielstein [38], we present a taxonomy of elementary selection methods. Depending on a priori knowledge, selection schemes can be classified according to the following criteria:

Threshold: subset selection – indifference zone.
Termination: single stage – multi stage (sequential).
Sample size: open procedures – closed procedures.
Variances: known – unknown, equal – unequal.

The goal of subset selection is the identification of a subset containing the best candidate. It is related to screening procedures. *Subset selection* is used when analyzing results, whereas the *indifference zone* (IZ) approach is used when designing experiments. The sample size is known in subset selection approaches, it is determined prior to the experiments in the indifference zone approaches. *Single stage* procedures can be distinguished from *multi stage* procedures. The terms "multi stage" and "sequential" will be used synonymously. The latter can use *elimination*: If inferior solutions are detected, they are eliminated immediately. Selection procedures are *closed*, if prior to experimentation an upper bound is placed on the number of observations to be taken from each candidate. Otherwise, they are *open*. Furthermore, it is important to know whether the variance is common or known. Bartz-Beielstein [38] discussed similarities and differences of these approaches. He also analyzed threshold-based procedures, which were successfully applied to noisy, dynamic functions, e.g., in elevator group control. Threshold rejection increases the chance of rejecting a worse candidate at the expense of accepting a good candidate. It might be adequate if there is a very small probability of generating a good candidate.

How can the experimenter cope with this multitude of selection methods? Surely, there is no general rule for the determination of the best selection method. Many theoretical results consider simplified sources of uncertainty, e.g. they regard environmental parameters as random with a distribution that is known. Performing experiments in a systematic manner might be useful. Modern approaches such as racing or sequential parameter optimization (SPO) can be recommended in this context [39, 40]. A typical result from an SPO analysis is shown in Figure 4.

Regarding the classification from fig. 3, there two starting points to cope with noise: (i) varying the algorithm design, e.g., choosing a modified selection operator or (ii) modifying the problem design, e.g., refining the fitness function. Evolutionary optimization itself can be considered as an evolutionary process. Based on results from previous optimization runs, the experimenter may gain insight into the behavior of the evolutionary algorithm and into the structure of the problem as well. He is able to modify (improve) algorithm and problem designs—black box situations turn into gray box situations. Combinations of

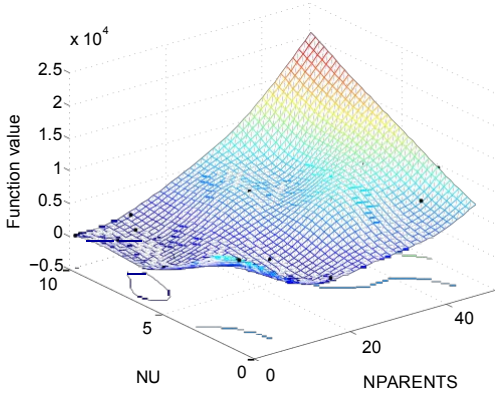

Fig. 4. SPO combines classical and modern statistical tools for the analysis of algorithms. Modifying population size (NPARENTS) and selective pressure (NU) can improve algorithm's performance significantly. Evolution strategies with small population sizes and moderate selective pressure perform best in this setting.

classical and evolutionary methods (meta heuristics) may be useful in these situations.

Multiple Objectives For many problems of high practical interest in science and engineering, several possibly contradicting objectives shall be pursued simultaneously. In daily life we are confronted with many examples. E.g. in chemical process engineering, where the productivity of chemical reactors is in contrast to their loss during the start up and shut down phases. In the textile industry, a similar conflict arises for the production of fabrics. Figure 5 shows a simple discrete example. Total elongation (F_1) and extensibility (F_2) of the fabric shall be improved, by means of maximizing F_1 and minimizing F_2. All objectives are sufficiently defined and in this case pointwise quantifiable. Their values are determined by three adjustable control factors (decision variables): Number of knitting skewers (x_1), number of knitting rows (x_2) and number of weft threads (x_3) per inch. The challenge for a multi-objective optimization algorithm consists of finding decision variable value sets that fulfill all objectives as well as possible.

In this context, the Pareto [41] concept of optimality proved as suitable. During the beginning of an optimization run, it is often not hard to find solutions that simultaneously improve both objectives. However, if an objective can be improved further only by worsening an other objective, a solution is called *Pareto-optimal*. Due to different possible preferences concerning the single objectives, this leads to a set of Pareto-optimal solutions, each of them representing a valid optimal solution for the multi-objective problem (MOP). Figure 5 shows

six solutions in the decision variable space (a) and the objective space (b) for the fabric improvement example. In this example, the decision variable space is discrete and constrained as indicated by the surrounding solid line. Consequently, there is only a finite number of possible objective value combinations. Direct comparison of solutions 5 and 6 shows that the former improves on F_1 without changing F_2. According to the Pareto dominance concept, solution 5 *dominates* solution 6. However, pairwise comparison of solutions 1 to 5 does not result in recognizing any such domination as improvement in one objective always comes along with worsening in the other. These solutions are therefore indifferent to each other, hence incomparable or *non-dominated*. If due to problem-specific constraints no further improvements can be obtained (solutions 1-5 are on the border of the feasible region) the set of all non-dominated solutions represents the *Pareto Set* in the decision space and the *Pareto Front* in the objective space. Since in each case only one solution can be realized, preference information of a decision maker (DM) must be used next to select the final solution of the MOP.

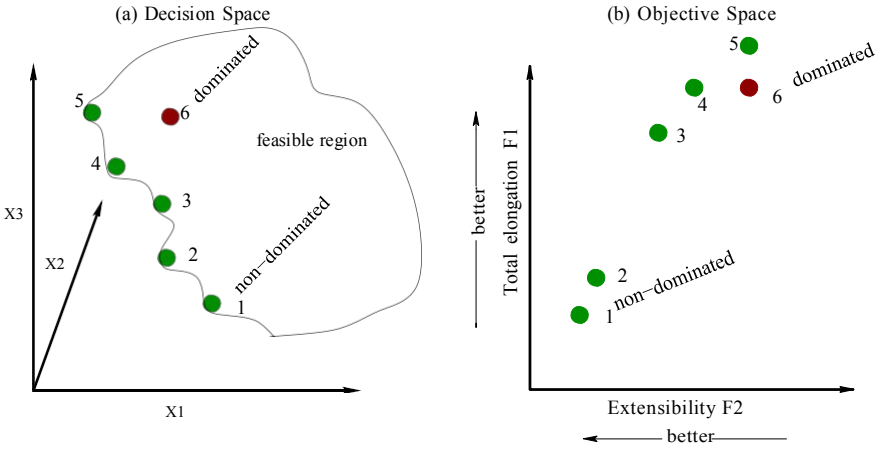

Fig. 5. The Pareto-dominance concept. (a) Decision space, (b) objective space

Why Use Evolutionary Algorithms? Problems with several conflicting criteria have been treated for many years, e.g. with a considerable variety of techniques developed in Operational Research. Concise overviews of existing approaches can be found in Achilles et al. [42] and Miettinen [43]. Usually one tries to reduce the MOP into a single-objective problem, so that it can be solved by means of methods from single-objective optimization. One possible approach consists of choosing a single criterion as main objective, and transform the other objectives to constraints with lower or upper bounds. Without specific knowledge of the problem, the choice of concrete upper and lower bounds suffers from arbitrariness. Alternatively, one may try aggregation-based approaches. These combine all criteria into a single, parametrized one. The aggregation can be accomplished

by any combination of arithmetical operations (i.e. a weighted sum), according to some understanding of the problem. However, these techniques have several limitations. Some of them are e.g. susceptible to the shape (convex/concave) of the Pareto front, others to its continuity (connected/disconnected). In addition, most of the 'conventional' approaches are only able to compute one single non-dominated solution per run. Searching for a representative set of non-dominated solutions requires a restart with different external parameter settings and different starting points for each run.

Evolutionary algorithms are robust search methods, whose success and failure is by far less susceptible to the shape or the continuity of the Pareto front. Their greatest advantage is that they are able to provide a point-wise approximation of the whole Pareto front in one go by employing cooperative search of a whole population.

Algorithm Design If one regards the development of the *evolutionary multi-objective (EMO) algorithms* within the last two decades, then the rise of suggested approaches is impressing. The largest well-known collection of existing approaches was arranged by Coello Coello and contains over 1900 entries [44]. A common classification of all EMO-algorithms comes from Masud [45]. Depending on the time at which the preference information from the DM is used, four classes can be differentiated: *(i)* Non-preference, *(ii)* a-priori, *(iii)* interactive, and *(iv)* a-posteriori. In the following, this classification is not discussed in detail as most EMO-algorithms can be assigned to the last category. The optimization process takes place before any preference information is incorporated. This entails a clear task definition: Find a representative set of non-dominated solutions as close (convergence) as possible to the *Pareto optimal set/front*. Additionally, the resulting approximation has to exhibit a good distribution of solutions in terms of both spread and uniformity - usually described by the term of *diversity*. The aim of this section is to give an overview of the main methods that have been developed in order to achieve these goals.

Fitness Assignment When moving from single-objective to multi-objective optimization while applying EAs, the most important changes to be made concern the selection operator and especially the fitness assignment. In EAs, the fittest individuals have better chances to survive and reproduce. For single-objective optimization, only one scalar fitness value exists. However, in the multi-objective case we have to deal with a fitness vector. Since EAs need a scalar to work on, generally two design decisions must be made: On the one hand, this vector must be scaled to enable for EA selection, and on the other hand the two conflicting tasks of convergence and diversity shall be respected. But how to assign the fitness of an individual in order to express suitability towards both goals? We can roughly divide the existing answers into two categories:

Combined Fitness Assignment: Fitness is assigned such that the fitness value represents convergence and diversity at the same time.

Single Fitness Assignment: Fitness assignment respects only one goal. Usually, this is convergence, as in the single-objective case.

Aggregation-, performance-, and Pareto-based approaches belong to the first category. Aggregation-based approaches are the most traditional as well as simplest possibility. Recently, performance-based fitness assignment strategies are successfully used to evaluate the fitness of a new individual in relation to the entire population. For example, the S-metric selection (SMS)-EMOA utilizes the well-known S-metric (hypervolume) to calculate the fitness of an individual. This measure is commonly used to evaluate the performance of an EMOA. It respects proximity to the Pareto front as well as diversity of the solution set.

Pareto-based approaches use the Pareto dominance concept itself for fitness assignment. Differences between these approaches arise in the methods employed to exploit the partial order. According to Zitzler et al. [46], this kind of information can be divided into: *(i) Dominance rank:* The number of solutions in the population that dominate the solution under consideration, *(ii) dominance count:* The number of solutions in the population that are dominated by the solution under consideration, and *(iii) dominance depth:* The rank of the solution in the non-dominated sorted population. The latter approach is utilized by many successful algorithms, e.g. the *Non-dominated Sorting Genetic Algorithm (NSGA)-II* by Deb and others [47]. Dominance rank was first employed by Fonseca and Fleming in their *Pareto envelope-based algorithm (PESA)* [48]. Today, a multiplicity of methods are based on this principle, see for example Bosman and Thierens [49]. Dominance depth and dominance rank are successfully combined in the *Strength Pareto Evolutionary Algorithm 2 (SPEA2)* approach by Zitzler and others [50].

However, most of these algorithms apply a *secondary fitness assignment strategy* that serves the goal of diversity. In most cases they try to incorporate density information into the selection process (mating/environmental), according to the rule: The smaller the density of individuals within a neighborhood, the larger the chance of an individual to reproduce. Figure 6 shows the three most frequently used methods: *Kernel-based*, *grid-based* and *nearest-neighborhood* measures. Fitness sharing, as e.g. used in NSGA, is a kernel-based strategy. The distance of an individual to all other individuals in the population is calculated and summed up. These values are then used to deflect the evolutionary search out of densely populated regions. Grid-based techniques as e.g. utilized by the *Pareto Archived Evolution strategy (PAES)* of Knowles and Corne [51], employ hypergrids to define neighborhoods within the objective space. The more individuals in a box, the heavier they are penalized (see fig. 6). Nearest neighborhood techniques as used in SPEA2 and its variants calculate the distance between an individual and its nearest neighbor in order to estimate the neighborhood density.

Criterion-based approaches represent the second category of fitness assignment strategies. They all share the same basic idea: The fitness value of an individual is determined by only one of the criteria according to the goal of convergence. However, the choice of a single criterion for any individual shall be reconsidered repeatedly (in each generation). As thereby parts of the population are selected according to different criteria, it is hoped that the goal of diversity can be achieved indirectly (see Schaffer [52] and Laumanns and others [53]).

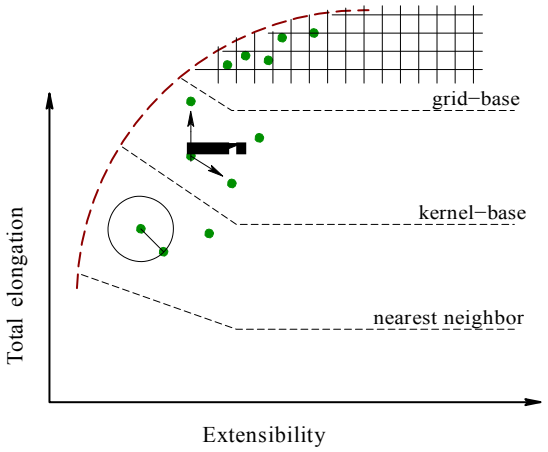

Fig. 6. Most common diversity preservation strategies in EMOA.

Representations and Variation Operators Design and analysis of representations and corresponding genetic operators is prevalent in the field of evolutionary computation. Often, an adept combination of all components determines the system's success or failure. This insight is ubiquitous in the case of single-objective optimization. However, in multi-objective optimization, the conceptual approaches are still mainly concerned with the selection operator. Research focusing on variation operators or representations remains rare. Some recent approaches are: Rudolph [54] and Hanne [55] who investigate control mechanisms for the mutation strength in the multi-objective case. Grimme and Schmitt [56] focus on recombination operators that produce diverse offspring in each generation.

Elitism Elitism preserves previously attained good solutions from one generation to the next. The prime example of an elitist algorithm in the single-objective case is the 'plus'-selection ES. In the multi-objective case two types of elitism are used: Maintaining elitism in the current population, as is already done in the single-objective case, or doing so in an archive (secondary population) that stores non-dominated solutions externally. Archive contents may or may not be integrated again into the optimization process (Zitzler and others [46]). Of vital importance is the criterion used to control replacement of archive members, the most commonly used of which is the dominance criterion. It leads to an archive of non-dominated solutions, relative to all solutions generated during a run.

Future Perspectives As has been hinted to in the previous paragraphs, a lot of work remains to be done on EMOAs. We briefly discuss the currently most promising paths:

Investigating representations and variation operators: Büche and others [57] show that the interaction between selection and search operators is often not co-ordinated well, and that approximation of the Pareto front

cannot be done with arbitrary precision. Further on, there is the dilemma of stagnation with good diversity of the solution set on the one hand, or arbitrarily exact approximation of a few points on the Pareto front. We conjecture that this trade-off between convergence and diversity can be attributed to the fact that variation operators cannot simply be taken over from the single-objective case and that changing only the selection operator is not sufficient to meet the requirements of multi-objective optimization.

Focusing on the region of interest (ROI): In the last years, most EMO researchers focus on algorithms that are able to find the whole Pareto front. However, in practice, the decision maker is only interested in a specific region of the Pareto-front. Focusing on a region derived from user preferences may help to increase convergence speed and/or quality and also simplify solution selection by the DM later on.

Parallelism: Considering the suitability of EAs working in a parallel manner, one should expect that the development of parallel approaches stands only at the beginning. Apart from first successful attempts to convert the state-of-the-art algorithms into a parallel version [58], an increasing number of parallel approaches has been published only recently [59, 60].

Parameter tuning: Attaining good parameter settings for a given problem-algorithm combination currently is one of the hot topics in single-objective optimization [38]. It is necessary to adapt those techniques for the multi-objective case in order to avoid the commonly used manual parameter tuning and provide important insight into parameter interactions.

Multimodal Problems Although, during the last decades, many empirical and most of the theoretical studies in EC have been devoted to simple test problems with only one extremal point, the great majority of practical applications requires optimization in far more complex fitness landscapes. Multimodality—the presence of more than one locally optimal point—requires a shift from a hill-climbing oriented towards a global perspective. At the top of the hill, the need arises to somehow 'escape' the associated local optimum. This may be done in two different ways. Either, one tries to save as much positional and learned (step sizes/mutation strengths) information as possible and, preserving this information, attempts to jump over the neighboring valleys. Or, one completely gives up the current search space location and performs random initialization again. For mutation strenghts getting larger and larger, the former scenario more and more resembles the latter.

However, if the treated optimization problem is not available in a closed algebraic form, detecting the arrival at a local optimum may not be trivial, depending on the employed variable representation. Combinatorial and binary encoded optimization problems come with a natural minimal step definition which enables enumeration of the neighborhood. For real-valued representations, eqn. 6 specifies a necessary and sufficient condition for a local optimum, with $\mathbf{x}^* \in$ meaning its search space location, $d(\mathbf{x}, \mathbf{y})$ a distance metric, and ε the maximal distance to tested neighboring search points. Nevertheless, the bounded but still

infinite neighborhood cannot be completely explored efficiently and one has to rely on the strong causality assumption (Rechenberg[61]: similar causes entail similar effects) to identify local optima at least in probability.

$$\mathbf{x}^{*\,L} \text{ is local minimizer iff } \exists \varepsilon : \forall \mathbf{x} \in X : d(\mathbf{x}, \mathbf{x}^{*\,L}) < \varepsilon \Rightarrow f(\mathbf{x}^{*\,L}) \leq f(\mathbf{x}) \quad (6)$$

Strongly related to the notion of local optima is the one of basins of attraction; these emcompass the search space portion leading to an optimum if the steepest descent is followed. For this local search process, efficient approximation methods are known, e.g. quasi-Newton algorithms. However, identification of different basins is even more difficult than local optimum detection if no further information regarding size and/or location of the basins is available. The key property of multimodal optimization methods is thus how efficient they are in finding the different search space regions that contain the best local optima.

Canonical population based EAs perform global and local search at the same time, gradually narrowing their focus to the most promising regions, and more sooner than later to a single basin of attraction (e.g. Preuss, Schönemann and Emmerich [62]). From the discussion above, it becomes clear that the ability to explore multiple promising regions—either concurrently or sequentially—is decisive for obtaining well performing EA variants. But for a given limit of available computational time, these always have to face the global vs. local search tradeoff like any other global optimization algorithm.

One possible way to speedup local optimization, so that more effort can be diverted to search space exploration, is to hybridize EAs with existing local search methods. These approaches are subsumed under the term *memetic algorithms* (MA) that was introduced by Moscato [63]. A recent overview is given by Krasnogor and Smith [64], together with a suggested taxonomy.

Most other specialized EAs strive for enhanced global search capabilities by means of at least one of the following three techniques:

Restarts are utilized to enhance the chance of reaching the/a basin of attraction of the global optimum. As an example, an efficient restart CMA-ES for multimodal problems has been suggested by Auger and Hansen [65]. Multistart methods obtain potential solutions consecutively, and every new instantiation may be provided with search results of completed previous runs. They avoid the problem of jumping into a neighboring good region by giving up the current search space location completely.

Diversity maintenance aims for a uniform distribution of individuals over the whole search space. Comparing relative or absolute distances of solution candidates and applying clustering methods are common means to prevent overlapping search paths and promote good search space coverage. Diversity may be held up explicitly or implicitly. Following Eiben and Smith [14], explicit means that active measures are taken to model the distribution of search points in the desired way, whereas implicit stands for deliberately slowing down information exchange by restricting recombination or selec-

tion/replacement. Classical island models provide implicit diversity maintenance by building relatively independent subpopulations. Spatially structured EAs [66] do so by restricting the effect of recombination and selection operators to the local neighborhood. *Shifting balance* GAs by Oppacher and Wineberg [67] exemplify explicit diversity maintenance as they prevent subpopulation overlap which is measured by absolute population distances.

Niching methods also strive for a suitable spread of search points, only on the level of basins of attraction. As Mahfoud [68] points out, it is the aim of *niching* algorithms to detect separate basins and keep them in focus of the search. Unfortunately, *basin identification* within an EA is not easy and prone to error, so that endogenously retrieved basin information is highly unreliable and nonexistent when the optimization starts. Crowding by De Jong [69] and fitness sharing by Goldberg and Richardson [70] are regarded as the classical niching methods. The former employ relative, the latter absolute distances. These have been carried further e.g. by Li et al. [71], Streichert et al. [72], and Shir [73], but still the radii employed for detecting search points located together in a basin remain problematic. Only few approaches integrate fitness topology information into the basin identification process, e.g. the *universal evolutionary global optimizer* (UEGO) by Jelasity [74], Ursem's multinational GA [75], and the sample-based crowding method proposed by Ando et al. [76].

It shall be noted that solving multimodal problems is related to tackling constrained or multiobjective ones. Removing constraints from a problem by transforming it by means of (metric) penalty functions (see e.g. Michalewicz and Schoenauer [77] and Coello Coello [78]) as commonly done in EC most often leads to multimodal problems even if the original problem was unimodal.

In multi-objective optimization, the focus has been mainly on the objective space for a long time. Today, it becomes increasingly clear that population movement in the decision (search) space heavily depends on the multimodal search properties of the applied optimization algorithms (Preuss, Naujoks and Rudolph [79]).

Conclusions May it be (or not) that one day there is no more need to invent new optimization tools because we have got the best tailored ones already for every possible real-world problem. May it be (or not) that then the dream of hardliners has come true that all of these best tailored methods can abstain from using pseudo random numbers for deciding upon the next iteration in the search for the solution. But, contemporary tools are still well advised not to rely on deterministic algorithms alone. That is, why an idea from the early days of digital computers is still alive, i.e., the idea to mimic procedures found in nature that obviously have led to remarkably effective systems or subsystems. One may think that nature had enough time to achieve a good solution by means of pure chance, but time has always been scarce when there are competitors, and the way nature finds its way is much more sophisticated.

Anyway, it is a matter of fact that evolutionary algorithms have become widely used in practice since their invention in the 1960s and even found their way into articles in the field of theoretical computer science. Their domain of application are 'black box' situations, where the analysis of the situation at hand does not help or is too costly or dangerous, i.e., in case of experimental design and even computer simulation of nonlinear dynamic systems and processes. However, situations may occur where the black box situations turn into gray or even white box situations. EAs can be combined with classical methods which leads to *m* eta heuristics, and the optimization practitioner can get the best from both worlds.

References

1. D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
2. M.H. Wright. Direct search methods: Once scorned, now respectable. In *Proc. 1995 Dundee Biennial Conf. in Numerical Analysis)*, volume 344 of *Pitman Res. Notes Math. Ser.*, pages 191–208. CRC Press, Boca Raton, FL, 1995.
3. J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, 1965.
4. T.G. Kolda, R.M. Lewis, and V.J. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.
5. R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, New York, 1995.
6. H.J. Bremermann. Optimization through evolution and recombination. In M.C. Yovits, G.T. Jacobi, and G.D. Goldstein, editors, *Self-Organizing Systems*. Spartan Books, Washington DC, 1962.
7. D.B. Fogel. *Evolutionary Computation: The Fossil Record*. Wiley–IEEE Press, New York, 1998.
8. L.J. Fogel, A.J. Owens, and M.J. Walsh. Artificial intelligence through a simulation of evolution. In A. Callahan, M. Maxfield, and L.J. Fogel, editors, *Biophysics and Cybernetic Systems*. Spartan Books, Washington DC, 1965.
9. J.H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal of Computing*, 2(2):88–105, 1973.
10. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.
11. H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Department of Process Engineering, Technical University of Berlin, Germany, 1975.
12. J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press. Cambridge, MA, 1992.
13. A.E. Eiben and M. Schoenauer. Evolutionary computing. *Information Processing Letters*, 82(1):1–6, 2002.
14. A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.
15. Th. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Oxford University Press, New York, and Institute of Physics Publ., Bristol, 1997.
16. H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley Interscience, New York, 1995.

17. H.-P. Schwefel. Kybernetische Evolution als Strategie der exprimentellen Forschung in der Strömungstechnik. Master's thesis, Technical University of Berlin, Germany, 1965.
18. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Department of Process Engineering, Technical University of Berlin, Germany, 1971.
19. H.-P. Schwefel, G. Rudolph, and Th. Bäck. Contemporary evolution strategies. In F. Morán, A. Moreno, J.J. Merelo, and P. Chacón, editors, *Advances in Artificial Life – Proc. Third European Conf. Artificial Life (ECAL'95)*, pages 893–907. Springer, Berlin, 1995.
20. H.-G. Beyer and H.-P. Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.
21. H.-G. Beyer. Toward a theory of evolution strategies: On the benefit of sex – the $(\mu/\mu, \lambda)$-theory. *Evolutionary Computation*, 3(1):81–111, 1995.
22. H.-P. Schwefel. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research*. Birkhäuser, Basle, Switzerland, 1977.
23. Thomas Bäck and Hans-Paul Schwefel. Evolutionary algorithms: Some very old strategies for optimization and adaptation. In D. Perret-Gallix, editor, *New Computing Techniques in Physics Research II*, pages 247–254. World Scientific, Singapore, 1992.
24. Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.
25. Hans-Paul Schwefel. Collective phenomena in evolutionary systems. In P. Checkland and I. Kiss, editors, *Problems of Constancy and Change – The Complementarity of Systems Approaches to Complexity, Proc. 31st Annual Meeting*, volume 2, pages 1025–1033. Int'l Soc. for General System Research, 1987.
26. Th. Bartz-Beielstein. Experimental analysis of evolution strategies— Overview and comprehensive introduction. Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence CI–157/03, Universität Dortmund, Germany, 2003.
27. T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*. Springer, Berlin, 2003.
28. Hans-Georg Beyer. Towards a theory of evolution strategies: Some asymptotical results from the $(1 +, \lambda)$-theory. *Evolutionary Computation*, 1(2):165–188, 1993.
29. J.M. Fitzpatrick and J.J. Grevenstette. Genetic algorithms in noisy environments. *Machine learning*, 3:101–120, 1988.
30. P. Stagge. Averaging efficiently in the presence of noise. In A. Eiben, editor, *Parallel Problem Solving from Nature, PPSN V*, pages 188–197. Springer, Berlin, 1998.
31. H.-G. Beyer. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *CMAME (Computer methods in applied mechanics and engineering)*, 186:239–267, 2000.
32. Y. Sano and H. Kita. Optimization of Noisy Fitness Functions by Means of Genetic Algorithms using History of Search. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN VI)*, volume 1917 of *LNCS*, pages 571–580. Springer, Berlin, 2000.
33. D.V. Arnold. Evolution strategies in noisy environments — A survey of existing work. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing, pages 239–249. Springer, Berlin, 2001.

34. J. Branke, C. Schmidt, and H. Schmeck. Efficient fitness estimation in noisy environments. In L.Spector et al., editor, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'01)*, pages 243–250. Morgan Kaufmann, San Francisco, 2001.
35. Th. Bartz-Beielstein and S. Markon. Tuning search algorithms for real-world applications: A regression tree based approach. In G. W. Greenwood, editor, *Proc. 2004 Congress on Evolutionary Computation (CEC'04), Portland, OR*, volume 1, pages 1111–1118. IEEE Press, Piscataway NJ, 2004.
36. Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–318, JUN 2005.
37. R. E. Bechhofer, T. J. Santner, and D. M. Goldsman. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. Wiley, 1995.
38. Th. Bartz-Beielstein. *Experimental Research in Evolutionary Computation—The New Experimentalism*. Springer, Berlin, 2006.
39. M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editor, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufmann, 2002.
40. Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuß. Sequential parameter optimization. In B. McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.
41. V. Pareto. *Cours d'Economie Politique 1*. Lausanne, Rouge, 1896.
42. A. Achilles, K.H. Elster, and R. Nehse. Bibliographie zur Vektoroptimierung. Math. Op. forsch. Stat., Ser. Optim. 10, (2), 1979.
43. K. Miettinen. *Nonlinear Multiobjective Optimization*. Int. series in operations research and management science. Kluwer Academic Publishers, Boston, 1998.
44. C.A. Coello Coello. The EMOO repository: A resource for doing research in evolutionary multiobjective optimization. *IEEE Computational Intelligence Magazine*, 1(1):37–45, 2006.
45. C.L. Hwang and A.S.M. Masud. *Multiple Objective Descision Making – Methods and Applications: A State-of-the-Art Survey*, volume 186 of *Lecture Notes in Economics and mathematical Systems*. Springer, Berlin, 1979.
46. E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In *Workshop on Multiple Objective Metaheuristics (MOMH 2002)*. Springer, Berlin, 2003.
47. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Proc. of the 6th Int'l Conf. on Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *LNCS*, pages 849–858. Springer, Berlin, 2000.
48. C.M. Fonseca and P.J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In H.-M. Voigt, W.-Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*. Springer, Berlin, 1996.
49. P.A.N. Bosman and D. Thierens. The naive MIDEA: A baseline multi-objective EA. In C.A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Proc. Evolutionary Multi-Criterion Optimization: Third Int'l Conference (EMO 2005)*, volume 3410 of *LNCS*, pages 428–442. Springer, Berlin, 2005.

50. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou, D.T. Tsahalis, J. Periaux, K.D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 1–6. International Center for Numerical Methods in Engineering(CIMNE), Barcelona, 2001.
51. J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proc. Congress on Evolutionary Computation, (CEC'99)*, volume 1, pages 98–105. IEEE Press, Washington DC, 1999.
52. J.D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.
53. M. Laumanns, G. Rudolph, and H.-P. Schwefel. A spatial predator-prey approach to multi-objective optimization: A preliminary study. In A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature — PPSN V*, pages 241–249, Amsterdam, Holland, 1998. Springer, Berlin.
54. G. Rudolph. On a multi–objective evolutionary algorithm and its convergence to the Pareto set. In D.B. Fogel, H.-P. Schwefel, Th. Bäck, and X. Yao, editors, *Proc. Fifth IEEE Conf. Evolutionary Computation (ICEC'98), Anchorage AK*, pages 511–516. IEEE Press, Piscataway NJ, 1998.
55. T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal Of Operational Research*, 117(3):553–564, 1999.
56. C. Grimme and K. Schmitt. Inside a predator-prey model for multi-objective optimization: A second study. In H.-G. Beyer et al., editor, *Proc. Genetic and Evolutionary Computation Conf. (GECCO 2006), Seattle WA*, pages 707–714. ACM Press, New York, 2006.
57. D. Büche, S. Müller, and P. Koumoutsakos. Self-adaptation for multi-objective evolutionray algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization, Second Int.'l Conf., (EMO 2003)*, number 2632 in LNCS, pages 267–281. Springer, Berlin, 2003.
58. T. Okuda, T. Hiroyasu, M. Miki, and S. Watanabe. DCMOGA: Distributed Cooperation model of Multi-Objective Genetic Algorithm. In *MPSN - II, The Second Workshop on Multiobjective Problem Solving from Nature*, Granada, 2002.
59. J.L.A. Coello and C.A. Coello. MRMOGA: Parallel evolutionary multiobjective optimization using multiple resolutions. In D. Corne et al., editor, *Proc. 2005 IEEE Congress on Evolutionary Computation, (CEC 2005)*, volume 3, pages 2294–2301. IEEE Press, 2005.
60. J. Mehnen, Th. Michelitsch, K. Schmitt, and T. Kohlen. pMOHypEA: Parallel evolutionary multiobjective optimization using hypergraphs. Technical Report of the Collaborative Research Centre 531 *Computational Intelligence* CI–189/04, University of Dortmund, 2004.
61. Ingo Rechenberg. Evolution strategy—nature's way of optimization. In H. W. Bergmann, editor, *Optimization: Methods and Applications, Possibilities and Limitations*. Springer, Berlin, 1989.
62. M. Preuss, L. Schönemann, and M. Emmerich. Counteracting genetic drift and disruptive recombination in ($\mu \dot{+} \lambda$)-ea on multimodal fitness landscapes. In H.-G. Beyer, editor, *Proc. 2005 Conf. on Genetic and Evolutionary Eomputation, (GECCO 2005)*, pages 865–872. ACM Press, New York, 2005.

63. P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, CA, 1989.
64. N. Krasnogor and J.E. Smith. A tutorial for competent memetic algorithms: Model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation*, 5(9):474–488, 2005.
65. A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In B. McKay et al., editors, *Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 2, pages 1769–1776. IEEE Press, Piscataway NJ, 2005.
66. M. Tomassini. *Spatially Structured Evolutionary Algorithms Artificial Evolution in Space and Time*. Natural Computing Series. Springer, Berlin, 2005.
67. F. Oppacher and M. Wineberg. The shifting balance genetic algorithm: Improving the GA in a dynamic environment. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proc. Genetic and Evolutionary Computation Conf. (GECCO 1999), Orlando FL*, volume 1, pages 504–510. Morgan Kaufmann, San Francisco, 1999.
68. S.W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana Champaign, 1995.
69. K.A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.
70. D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proc. of the Second Int'l Conf. on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 41–49. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1987.
71. J.-P. Li, M.E. Balazs, G.T. Parks, and P.J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234, 2002.
72. F. Streichert, G. Stein, H. Ulmer, and A. Zell. A clustering based niching method for evolutionary algorithms. In E. Cantú-Paz, editor, *Proc. 2003 Conf. on Genetic and Evolutionary Computation, (GECCO 2003)*, pages 644–645. Springer, Berlin, 2003.
73. O.M. Shir. Niching in evolution strategies. In H.-G. Beyer, editor, *Proc. 2005 Conf. on Genetic and Evolutionary Computation, (GECCO 2005)*, pages 865–872, New York, 2005. ACM Press, New York.
74. M. Jelasity. UEGO, an abstract niching technique for global optimization. In A. E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proc. Parallel Problem Solving from Nature – PPSN V, Amsterdam*, pages 378–387. Springer, Berlin, 1998.
75. R.K. Ursem. Multinational evolutionary algorithms. In P.J. Angeline, editor, *Proc. of the Congress of Evolutionary Computation (CEC-99)*, volume 3, pages 1633–1640. IEEE Press, Piscataway, NJ, 1999.
76. S. Ando, E. Suzuki, and S. Kobayashi. Sample-based crowding method for multimodal optimization in continuous domain. In B. McKay et al., editor, *Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 2, pages 1867–1874. IEEE Press, Piscataway NJ, 2005.
77. Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.

78. C.A. Coello Coello. 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.
79. M. Preuss, B. Naujoks, and G. Rudolph. Pareto set and EMOA behavior for simple multimodal multiobjective functions. In Th.Ph. Runarsson et al., editor, *Parallel Problem Solving from Nature (PPSN IX)*, volume 4193 of *LNCS*, pages 513–522. Springer, Berlin, 2006.