Evolutionary Algorithms for Parameter Optimization—Thirty Years Later

Thomas H. W. Bäck t.h.w.baeck@liacs.leidenuniv.nl
Anna V. Kononova a.kononova@liacs.leidenuniv.nl
Bas van Stein b.van.stein@liacs.leidenuniv.nl
Hao Wang h.wang@liacs.leidenuniv.nl
Kirill A. Antonov k.antonov@liacs.leidenuniv.nl
Roman T. Kalkreuth r.t.kalkreuth@liacs.leidenuniv.nl
Jacob de Nobel j.p.de.nobel@liacs.leidenuniv.nl
Diederick Vermetten d.l.vermetten@liacs.leidenuniv.nl
Roy de Winter r.de.winter@liacs.leidenuniv.nl
Furong Ye f.ye@liacs.leidenuniv.nl
Leiden Institute of Advanced Computer Science, Leiden University, Netherlands

https://doi.org/10.1162/evco_a_00325

Abstract

Thirty years, 1993–2023, is a huge time frame in science. We address some major developments in the field of evolutionary algorithms, with applications in parameter optimization, over these 30 years. These include the covariance matrix adaptation evolution strategy and some fast-growing fields such as multimodal optimization, surrogate-assisted optimization, multiobjective optimization, and automated algorithm design. Moreover, we also discuss particle swarm optimization and differential evolution, which did not exist 30 years ago, either. One of the key arguments made in the paper is that we need fewer algorithms, not more, which, however, is the current trend through continuously claiming paradigms from nature that are suggested to be useful as new optimization algorithms. Moreover, we argue that we need proper benchmarking procedures to sort out whether a newly proposed algorithm is useful or not. We also briefly discuss automated algorithm design approaches, including configurable algorithm design frameworks, as the proposed next step toward designing optimization algorithms automatically, rather than by hand.

Keywords

Evolutionary computation, evolutionary algorithms, natural computing, parameter optimization.

1 Introduction

When being asked to write a paper 30 years after the original work was published, namely our paper, "An Overview of Evolutionary Algorithms for Parameter Optimization" (Bäck and Schwefel, 1993), I (Thomas Bäck) immediately thought this would be too big a challenge. In the early 1990s, the "big unification" was achieved in the field that formerly consisted of relatively strictly separated algorithmic branches: Genetic Algorithms, Genetic Programming, Evolutionary Strategies, and Evolutionary Programming, which were then united into what we call today *Evolutionary Computation*. Due to Evolutionary Computation's success in a huge range of application domains and its

Manuscript received: 22 February 2023; accepted: 22 February 2023.

importance as a foundation for many subfields of computer science, the development of the field over the past 30 years has been tremendous, going far beyond what we can report in this paper. In the late 1990s, the *Handbook of Evolutionary Computation* (Bäck et al., 1997) represented an attempt towards summarizing the state of the art based on a unified perspective of the field, and the editors were taking up this challenge since there was at least some hope to make it happen.

For today's paper, 30 years later, we can face this challenge only by taking a personal perspective of the field, making the corresponding (necessarily subjective) choices, and—fortunately—inviting co-authors who I (Thomas Bäck) have the pleasure to work with at the Leiden Institute of Advanced Computer Science. Like in the original paper, we will also mostly restrict this paper to *continuous parameter optimization* problems of the form:

$$\begin{aligned} &\text{minimize} \quad f : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}^k \text{ where } k \geq 1 \\ &\text{subject to} \quad g_i(\mathbf{x}) \leq 0 \; \forall i \in \{1, \ldots, m\}; \; \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

(i.e., we explicitly include multiobjective optimization problems and constraints except for equality constraints, as they can be represented by two inequality constraints).

In the following, we provide a list of *what we subjectively consider* important developments over the past 30 years, and we elaborate on some of those in detail across the paper.

- The development of the *Covariance Matrix Adaptation Evolution Strategy*, in short CMA-ES (Hansen and Ostermeier, 1996), has defined a new state of the art and a whole branch of algorithms derived from the CMA-ES (see Bäck et al., 2013, for an overview). Section 2.1 provides more details about this development.
- Genetic Algorithms have been specialized into many powerful variants for combinatorial optimization problems, which are beyond the scope of this paper. When it comes to our problem definition (see Equation 1), the Genetic Algorithms often still exhibit strong similarities with the canonical Genetic Algorithm described by Bäck and Schwefel (1993), as discussed in Section 2.2.
- Parameter control methods, dating back to the early days of Evolution Strategies with Rechenberg's 1/5-success rule (Rechenberg, 1973) and Schwefel's self-adaptation methods (Schwefel, 1981), have now become a widespread technique in Evolutionary Computation. Section 2.2 discusses this in the context of Genetic Algorithms, and Section 3.1 provides a more general perspective.
- When it comes to identifying multiple local optima at the same time in so-called *multimodal optimization*, and its generalization to find solutions that exhibit behavioral properties as diverse as possible, new techniques in *Novelty Search* and *Quality-Diversity Search* have been developed. These algorithms are briefly discussed in Section 3.2.
- *Constraint handling methods* have been developed well beyond penalty-function and re-evaluation-based approaches that were popular in the early 1990s; see Section 3.3 for details.

- For expensive objective function evaluations, *Surrogate-Assisted Evolutionary Algorithms* use nonlinear regression methods from the field of machine learning to learn fast-to-evaluate proxy functions for objective(s) and constraints (Jin, 2011). Section 3.4 provides a few of the corresponding developments in this area of research.
- Multiobjective optimization (Deb, 2009; Coello Coello, 2006) has developed from only a few publications into a huge field within Evolutionary Computation due to the powerful ability of a population to approximate a Pareto-optimal frontier in its entirety by using concepts such as Pareto dominance or performance indicators. Section 3.5 explains some of the key developments and state of the art.
- Automated algorithm design using configurable frameworks, discussed in Section 3.6, is a promising new approach towards automatically creating new algorithms for parameter optimization and dynamically configuring algorithms.
- Particle swarm optimization and differential evolution, two relevant classes of evolutionary algorithms that did not exist yet in 1993, are discussed briefly in Sections 3.7 and 3.8.
- It is impossible to discuss the wealth of new theoretical results in the field, such that we can only give a very brief list of relevant works in Section 4.
- In Section 5, we briefly discuss the flood of "new" paradigms from nature that are proposed as optimization algorithms, and the urgent need for appropriate benchmarking procedures for qualifying new methods, and in Section 6 a brief outlook is provided.

Most recently, we have seen many tremendously important developments in the field, for instance, new benchmarking standards and test function sets, statistical analysis methods, and software tools for experimentation, visualization, and analysis of experimental results (Bäck et al., 2022). In Section 5.2, we address some key developments in benchmarking and argue that proper benchmarking and empirical analysis of optimization algorithms is critical to assessing their performance and their value to the community and, more importantly, to the domain expert who wants to solve an optimization problem but is not an expert in optimization algorithms. Related to this, we also need to stress that it is of paramount importance to properly benchmark any proposed "natural metaphor-based" optimization algorithm against established state-of-the-art methods and to refrain from an unjustified use of non-mathematical terminology to describe such algorithms. As outlined in Section 5.1, we fully concur with critical voices such as Sörensen (2015) and Campelo and Aranha (2018, 2021), and would like to encourage the research community to support the publication of new algorithms only if they are adequately defined using mathematical notation and appropriately benchmarked against well-known state-of-the-art algorithms.

Going beyond proposing expert-designed optimization algorithms, a fascinating idea is to automatically design optimization algorithms that are best-in-class for a given (set of) optimization problems. Approaches such as *hyper-heuristics* (Pillay and Qu, 2018; Drake et al., 2020), which can be based on genetic programming (Burke et al., 2009), have been proposed for such automated algorithm design. More recently, algorithm

configuration methods have also been proposed very successfully for genetic algorithms (Ye et al., 2022), modern CMA-ES-based evolution strategies (van Rijn et al., 2016), and differential evolution and particle swarm optimization (Boks et al., 2020; Camacho-Villalón et al., 2022b). Although these ideas are not completely new in the domain of genetic algorithms (see Grefenstette, 1986, for hyperparameter optimization; Bäck, 1994, for algorithm configuration and hyperparameter optimization), they can now be applied at a massive scale, with proper benchmarking procedures for comparing the algorithmic variants, and also facilitating both configuration and hyperparameter optimization as a systematic next step.

There are many important aspects of evolutionary computation that are not covered in this paper simply because we had to make choices.

2 Evolutionary Computation for Parameter Optimization—30 Years Later

We assume familiarity with the underlying ideas of evolutionary computation, including a set of candidate solutions (the *population*), a given optimization problem (as in Equation 1), *variation operators* such as *mutation* and *recombination*, and one or more *selection* operators—and an iteration of the operator loop until a given termination criterion is satisfied. In Bäck and Schwefel (1993), we have presented the general algorithmic loop of evolutionary algorithms and how it can be used to describe the various instances. In this section, we briefly summarize some of the key developments since 1993 concerning parameter optimization variants.

2.1 Evolution Strategies

Evolution strategies, like all other EAs, have seen tremendous development over more than 60 years, starting from the early work at the Technical University of Berlin (Schwefel, 1965; Rechenberg, 1965, 1973). The algorithm, a (1+1)-ES, was inspired by the process of biological evolution and used a simple iterative procedure that mutates the search point of a single individual $\mathbf{x} \in \mathbb{R}^n$ by a multivariate normal distribution, that is, $\mathbf{x}' = \mathbf{x} + \sigma \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$, and assigns $\mathbf{x} \leftarrow \mathbf{x}'$ iff $f(\mathbf{x}') \leq f(\mathbf{x})$. Experimenting with engineering optimization problems, Schwefel and Rechenberg quickly discovered the need for adaptively learning the mutation step size σ, a concept that then became central to evolution strategies: The *self-adaptation* of strategy parameters.

For the (1+1)-ES, it started with the so-called 1/5-success rule for updating σ, which was derived from theoretical results on the sphere and corridor models (Schwefel, 1977). With multimembered evolution strategies, for example, the (μ, λ)-ES and the $(\mu + \lambda)$-ES, which has μ parents and λ offspring, mutative self-adaptation was introduced. With these methods, individuals are represented as $(\mathbf{x}, \theta)$, where in addition to the search point $\mathbf{x}$, a set of endogenous parameters θ is evolved by the ES, based on the intuition that good search points sprout from good strategy parameters (Beyer and Schwefel, 2002). Schwefel's strategy extensions additionally allowed for the self-adaptation of coordinate-specific step sizes σ_i and covariances c_{ij} for mutations, using a lognormal distribution for their variation (Schwefel, 1981). In 1993, this (μ, λ)-mutational step size control (MSC)-ES (Schwefel, 1981) was considered state of the art and was the first evolution strategy that was able to adapt the parameters of the mutation distribution to an arbitrary covariance matrix and generate correlated mutations.

However, as pointed out by Hansen et al. (1995), this strategy strongly depends on the chosen coordinate system. It is not invariant to search space rotations—an insight that led to a development that again, after the discovery of self-adaptation, has

significantly advanced state of the art through *derandomization* (Section 2.1.1) and the *Covariance Matrix Adaptation ES* and its derivates (Section 2.1.2), which are now state of the art in the field.

2.1.1 Derandomized (DR) Evolution Strategies

Derandomized evolution strategies moved away from mutative self-adaptation and used global strategy parameters for all the individuals in the population. This was motivated by the observation that mutative self-adaptation of individual step sizes is unsatisfactory for small populations (Schwefel, 1987), which is attributed to two key reasons. Namely, (i) good strategy parameters do not necessarily cause a successful mutation of the search points, and (ii) there is a conflict in maintaining a diverse set of strategy parameters within the population and stability of the search behaviour (Ostermeier et al., 1994a). Derandomized evolution strategies aim to achieve self-adaptation without any independent stochastic variation of the strategy parameters. This is addressed in DR1 by using the length of the most successful mutation vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to update σ and by applying a dampening factor β in order to reduce strong oscillations of σ over time (Ostermeier et al., 1994a).

With DR2 (Ostermeier et al., 1994b), in addition to a global step size σ, local step sizes $\sigma \in \mathbb{R}_+^n$ are included, which control the variability of each component of the decision variables. The update of both the local and global step sizes is not only based on the best mutation vector $\mathbf{z}$ in the current generation, but also on the best moves of previous generations. This information is collected in a vector ζ, which is updated based on a global learning rate c.

DR3 (Hansen et al., 1995), or *Generating Set Adaptation Evolution Strategy* ($(1, \lambda)$-GSA-ES), includes a representation of the covariance matrix as a global strategy parameter and can produce mutations from an arbitrary multivariate normal distribution, similar to the (μ, λ)-MSC-ES. Moreover, the GSA-ES is able to learn problem scaling and is invariant to search space rotations. Instead of using a direct representation for the covariance matrix, a matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ is used, of which the column vectors form a so-called generating set. Correlated mutation vectors are generated by matrix multiplication of a random normal vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with $\mathbf{B}$. The number of columns $n^2 < m < 2n^2$ of the matrix can be used to control the speed of adaptation, where a smaller value of m yields faster adaption and a larger value increases the accuracy.

2.1.2 Completely Derandomized Self-Adaptation in Evolution Strategies

The algorithms described in the previous section, DR1-3, introduce the first level of derandomization of parameter control in evolution strategies, which reduces selection disturbance on the strategy level. The second level of derandomization, or complete derandomization, completely removes this disruption and aims to explicitly realize the original objective of mutative strategy parameter control: to favor strategy parameters that produce selected mutation steps with high probability (Hansen and Ostermeier, 2001). In addition, the second level of derandomization also aims to provide a direct control mechanism for the rate of change of strategy parameters and requires that strategy parameters are left unchanged in the case of random selection.

The *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)*, introduced in Hansen and Ostermeier (1996), satisfies these conditions through two key techniques, Covariance Matrix Adaptation (CMA) and Cumulative Step length Adaptation (CSA). A series of papers (Hansen and Ostermeier, 1996, 1997, 2001) refined the CMA-ES to use weighted recombination for $\mu > 1$. In this section, we discuss the (μ_W, λ)-CMA-ES

as introduced in Hansen and Ostermeier (2001), focusing on CMA and CSA. For a complete tutorial on the CMA-ES, we refer the interested reader to Hansen (2016).

The (μ_W, λ)-CMA-ES. As previously mentioned, evolution strategies (commonly) generate search points $\mathbf{x}_i$ for $i, \ldots, \lambda$ by sampling from a multivariate normal distribution, which for CMA-ES, at generation g, reads as:

$$\mathbf{y}_i^{(g+1)} = \mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}) \tag{2}$$

$$\mathbf{x}_i^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)}\mathbf{y}_i^{(g+1)}. \tag{3}$$

For generation $g+1$, self-adaptation of the CMA-ES involves the calculation of the next mean of the search distribution $\mathbf{m}^{(g+1)}$, step size $\sigma^{(g+1)}$ and covariance matrix $\mathbf{C}^{(g+1)}$. The mean of the search distribution is updated using the weighted average of μ-best individuals in the current generation, effectively performing weighted intermediate recombination:

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + \sum_{i=0}^{\mu} w_i(\mathbf{x}_i^{(g+1)} - \mathbf{m}^{(g)}). \tag{4}$$

Several variants for the recombination weights w_i have been proposed, and the CMA-ES from Hansen and Ostermeier (2001) uses exponentially decaying weights.

Covariance Matrix Adaptation. The covariance matrix is updated such that the likelihood of successful search steps is increased, and effectively involves an iterative principal component analysis of selected search steps. Since, in order to guarantee fast search on simple functions, the population size λ is required to be small, it is not feasible to produce a reliable estimator of the covariance matrix using only the information available in the current population at generation g. The CMA-ES thus uses the selected steps from previous generations to yield a good covariance matrix estimator. To put more weight on the information obtained in recent generations, exponential smoothing with learning rate c_μ is used in computing the so-called rank-μ update:

$$\mathbf{C}^{(g+1)} = (1 - c_\mu)\mathbf{C}^{(g)} + c_\mu \sum_{i=0}^{\mu} \mathbf{y}_i^{(g+1)}(\mathbf{y}_i^{(g+1)})^T. \tag{5}$$

Since the update of the covariance matrix uses the dot product of the selected steps, sign information is lost, as $\mathbf{y}_i(\mathbf{y}_i)^T = (\text{-}\mathbf{y}_i)(\text{-}\mathbf{y}_i)^T$, and a so-called evolution path $\mathbf{p}_c$ is used to reintroduce this information. The evolution path $\mathbf{p}_c$ aggregates the search path (selected steps) of the population through a number of successive generations. In practice, $\mathbf{p}_c$ is computed as an exponential moving average of the mean of the search distribution:

$$\mathbf{p}_c^{(g+1)} = (1 - c_c)\mathbf{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}. \tag{6}$$

Where c_c is the learning rate for the exponential smoothing of $\mathbf{p}_c$, and μ_{eff} denotes the variance effective selection mass. The full update of the covariance matrix then reads:

$$\mathbf{C}^{(g+1)} = \left(1 - c_1 - c_\mu \sum_{i=0}^{\mu} w_i\right)\mathbf{C}^{(g)} \tag{7}$$

$$+ c_1\mathbf{p}_c^{(g+1)}(\mathbf{p}_c^{(g+1)})^T \qquad \text{rank-one update} \tag{8}$$

$$+ c_\mu \sum_{i=0}^{\mu} w_i\mathbf{y}_i^{(g+1)}(\mathbf{y}_i^{(g+1)})^T \qquad \text{rank-}\mu\text{ update.} \tag{9}$$

This combines information from the current population through the rank-μ update and introduces the information of correlations between generations with the rank-one update by leveraging the evolution path $\mathbf{p}_c$.

Cumulative Step Length Adaptation. The update of the global step size $\sigma^{(g+1)}$ is not addressed explicitly via CMA, and only the changes in scale for the respective directions are computed. The reasoning for maintaining a global step size in the CMA-ES algorithm is two-fold. First, the optimal step size cannot be well approximated by the CMA update alone, and second, the required rate of change for the global step length is much higher than the maximum rate of change of the covariance matrix (Hansen, 2016). Similar to CMA, the update of $\sigma^{(g+1)}$ utilizes an evolution path $\mathbf{p}_\sigma^{(g+1)}$, which represents the sum of successive steps over a given backward time horizon. Since the evolution path $\mathbf{p}_c^{(g+1)}$ depends on its direction, that is, covariance, a conjugate evolution path $\mathbf{p}_\sigma^{(g+1)}$ is constructed via:

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{(g)^{-\frac{1}{2}}}\frac{\mathbf{m}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}. \tag{10}$$

The inverse square root of the covariance matrix is computed via an eigendecomposition of $\mathbf{C} = \mathbf{B}(\mathbf{D}^2)\mathbf{B}^T$, where $\mathbf{C}^{-\frac{1}{2}} = \mathbf{B}(\mathbf{D}^{-1})\mathbf{B}^T$. Since this is a computationally expensive operation, the authors suggest only performing the eigendecomposition every $\max(1, \lfloor 1/(10n(c_1 + c\mu)) \rfloor)$ generations (Hansen, 2016).

For updating the global step size $\sigma^{(g+1)}$, the length of the conjugate evolution path is compared with its expected length under random selection, that is, $\mathbb{E}||\mathcal{N}(\mathbf{0}, \mathbf{I})||$. If the evolution path is too short, this means that single steps are not making enough progress and cancel each other out, thus the step size should be decreased. If the evolution path is long, the progress made in previous steps is correlated, and thus could have been made with fewer steps, and requires an increase in step size. With an additional dampening parameter d_σ, this allows $\sigma^{(g+1)}$ to be computed as:

$$\sigma^{(g+1)} = \sigma^{(g)}\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\mathbf{p}_\sigma^{(g+1)})}{\mathbb{E}||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right)\right). \tag{11}$$

Extensions. Several modifications to the CMA-ES have been proposed over the years, but the core algorithm remains the current state of the art in ES, ranking high in benchmarks for numerical optimization (Hansen et al., 2010), with many successful applications in real-world optimization problems (Hansen, Niederberger et al., 2009; Bredèche, 2008; Winter et al., 2008). Early extensions to the CMA-ES involved the introduction of local restart strategies (Auger et al., 2004), which uses multiple criteria to determine whether to restart the algorithm at certain points during the optimization run. This gave rise to the introduction of the IPOP-CMA-ES (Auger and Hansen, 2005), which upon a restart increases the size of the population, and the BIPOP-CMA-ES (Hansen et al., 2010), which balances between larger and smaller population sizes during restarts. The (1+1)-Cholesky CMA-ES (Igel et al., 2006) uses an implicit method for adapting the covariance matrix, without using an eigendecomposition but by leveraging the so-called Cholesky decomposition, which has the effect of reducing the runtime complexity in each generation from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$. This was later extended to the (μ, λ)-Cholesky CMA-ES (Krause et al., 2016), which uses triangular Cholesky factorization instead of the inverse square root of the covariance matrix. Since, in general, the CMA-ES scales poorly with increasing n, several variants have been proposed for large-scale optimization such as the sep-CMA-ES (Ros and Hansen, 2008) and dd-CMA-ES (Akimoto and

Hansen, 2020); see Varelas et al. (2018) for a review. The (μ_W, λ)-CMA-ES uses the information of the μ best individuals in the population to compute the update of $\mathbf{C}^{(g+1)}$. The Active-CMA-ES improves upon this idea by also including the information from the $\lambda - \mu$ worst individuals and scaling them with negative weights. Alternative strategies for updating the global step size have been proposed, such as the two-point step-size adaption (Hansen, 2008) and the median success rule (ElHara et al., 2013). Mirrored sampling and pairwise selection (Auger et al., 2011), seeks to improve the mutation operator by generating pairs of mirrored mutation steps and only selecting the best out of each mirrored pair for updating the strategy parameters. This was later extended to use orthogonal sampling (Wang et al., 2014). Many of the extensions mentioned in this section can be combined arbitrarily to produce novel CMA-ES variants (van Rijn et al., 2016; de Nobel et al., 2021a), even ones that are not originally proposed for use with the CMA-ES, such as the usage of quasi-random mutations (Auger et al., 2005). Hybrid CMA-ES algorithms, such as the HCMA, BIPOP-Active-CMA-STEP (Loshchilov et al., 2013), and IPOP-Active-CMA-ES (Faury et al., 2019) are currently ranking top on the well-known BBOB single objective noiseless benchmark (Hansen et al., 2021).

This brief overview clarifies how much the subfield of evolution strategies has diversified after the introduction of the CMA-ES in the second half of the 1990s. All variants based on the CMA-ES use historical information about successful mutations to support continual learning of the mutation distribution. Even for this branch of evolutionary computation, gaining a complete overview of the state of the art in the field becomes very difficult, as exemplified by the subset of algorithm variants that have been collected and empirically compared by Bäck et al. (2013).

2.2 Genetic Algorithms

The genetic algorithm (GA) is likely still the most widely known variant of evolutionary algorithms. It was initially conceived in Holland's work of general adaptive processes (Holland, 1975). The commonly known GA, which is referred to as *canonical* GA or *simple* GA (SGA), has been the first example of applying genetic algorithms to parameter optimization (De Jong, 1975). The SGA applies a binary representation and proportional selection, which is based on the fitness values of individuals. It emphasizes the use of recombination, that is, crossover, as the essential variator for generating offspring, while preferring a low mutation probability. After the *initialization* step, the procedure of an SGA can be briefly described as an optimization loop that successively applies the operators *selection*, *crossover*, *mutation*, and *evaluation* until a stopping criterion is met. Following years of development, the GA variants that have been proposed for applications in different domains still primarily follow this algorithm skeleton, but depending on the application domain (and, correspondingly, the search space), numerous variations have been proposed. A complete overview is beyond the scope of this paper, which focuses on continuous parameter optimization.

2.2.1 Decoding Is Not a Unique Option

A GA used to be characterized by its *encoding* and *decoding* system. The optimization operators execute on a *genotype* solution, while the problem-specific solutions are presented as the *phenotype*. The SGA applies a binary genotype representation, that is, $\mathbf{x} \in \{0, 1\}^n$, where n is the length of the bit string. When dealing with continuous and discrete parameter optimization problems, encoding and decoding methods are applied to transfer solutions between genotype and phenotype.

Nowadays, the GA supports the representations of arbitrary types of variables such as real, integer, and categorical values such that the encoding and decoding system is unnecessary for some variants, for example, in complex GA applications such as in engineering (Dasgupta and Michalewicz, 2013), neural networks (Leung et al., 2003), and permutation-based problems (Whitley, 2019; Larrañaga et al., 1999; Reeves and Yamada, 1998; Costa et al., 2010). Also, novel operators have been proposed to deal with the non-binary representation.

2.2.2 Real-Coded Operators

For continuous optimization, the GA variants with non-binary representation usually apply the following collection of real-coded crossover operators:

- Blend crossover (BLX) (Eshelman and Schaffer, 1992) samples values of offspring from a uniform distribution, of which the range depends on the values of the two parents.
- Simulated binary crossover (SBX) (Deb et al., 1995) simulates the one-point crossover of the binary representation. It creates two offspring based on the linear combination of two parents, and the scale of each parent's value depends on a distribution with a given spread factor.
- Unimodal normal distribution crossover (UMDX) (Ono et al., 1999) samples offspring values as linear combinations of the realization of two normal distributions. One distribution obtains the mean and the standard deviation, which is the average of two parents' values and is proportional to their distances, respectively. The other distribution's mean is 0, and its variance depends on the distance from the third parent to the line connecting the two parents.

Recent works mostly focus on the modification and self-adaptivity of the distributions that are used to sample offspring. For example, the simple crossover (SPX) (Tsutsui et al., 1999) samples offspring values as the combinations of values sampled uniformly at random based on pairs formed by multiple parents, which can be seen as an extension of BLX for multiple parents. Moreover, self-adaptive and dynamic models have been proposed for SBX (Deb et al., 2007; Chacón and Segura, 2018; Pan et al., 2021). The Parent-Centric crossover (PCX) (Deb, Anand, et al., 2002) modified the UNDX by introducing a parent-centric concept, and the PNX (Ballester and Carter, 2004) followed the idea of PCX and simplified the normal distribution used to sample offspring based on the deviation of two parents.

For the real-coded mutation operators, one of the earliest related work (Janikow et al., 1991; Neubauer, 1997) known as a non-uniform mutation increases and decreases variable values with equal probabilities. The mutated distance, that is, deviation of the parent and offspring values, is proportional to the deviation of the boundary and the parent values, and this deviation gets close to 0 after a large number of generations. Also, the recently proposed wavelet mutation (Ling and Leung, 2007) samples the mutated distance using the Morlet wavelet, which distributes symmetrically with an expected value of zero. Moreover, novel mutation operators (Korejo et al., 2010; Temby et al., 2005) search towards promising space based on the progress feedback from previous updatings of the population.

2.2.3 Developments in Binary Representation

At the same time, GAs with the binary representation are still applied in many domains, for example, pseudo-Boolean optimization. For the binary representation, mutation flips the chosen bits of a parent, and crossover swaps the chosen bits of the two parents.

The classic *standard bit mutation* flips each bit of an individual with probability $p_m \in [0, 1]$. It was seen as a "background operator" with a small $p_m \approx 10^{-3}$, later $p_m \approx 1/n$, in early GAs (Holland, 1975; Bäck and Schwefel, 1993). We rephrase the mutation procedure as flipping $\ell \in \{0, \ldots, n\}$ bits chosen uniformly at random. The standard bit mutation samples ℓ from a binomial distribution $\text{Bin}(n, p_m)$. Recent work of mutation-only EAs studies the choice of ℓ. For example, the so-called fast GA applies a heavy-tailed power-law distribution to sample ℓ (Doerr et al., 2017), and a normalized bit mutation samples ℓ using a normal distribution in Ye et al. (2019). Moreover, it has been shown that the value of ℓ depends on problem dimensionality n (Witt, 2013; Doerr et al., 2019).

Crossover, another key operator of the GA, obtains important characteristics that are not in mutation (Spears, 1993). Theoretical studies have tried to explain how crossover works in a GA and indicate that crossover could capitalize on population and mutation (Doerr et al., 2015; Dang et al., 2017; Corus and Oliveto, 2017; Sudholt, 2017; Pinto and Doerr, 2018; Antipov et al., 2020). The value of $p_c \in [0.6, 1.0]$ was mentioned in Bäck and Schwefel (1993) for the crossover probability, found in earlier empirical studies. However, a recent study shows that this suggested value is not optimal for some cases. For example, the optimal value changes with population size and dimensionality for the LeadingOnes problem (Ye et al., 2020).

Note that many theoretical studies exist about GAs for discrete optimization (Doerr and Neumann, 2020). Also, much evidence indicates that the optimal parameters (i.e., mutation rate, crossover probability, and population size) are dynamic, which relates to the topic of parameter control (Karafotias et al., 2014; Aleti and Moser, 2016).

2.2.4 Learning in Genetic Algorithms

The *schema theorem* plays an important role in the early theoretical study of genetic algorithms. It considers the hypothesis that good GAs combine *building blocks*, which are short, low-order, and above-average schemata, to form better solutions (Goldberg, 1989; White, 2014). Though the schema theorem shows some drawbacks (Eiben and Rudolph, 1999)—for example, this building blocks hypothesis implicitly assumes that the problems are separable, and the theorem cannot explain the dynamic behavior and limits of GAs—it reveals that one key issue of GA is destroying building blocks. Furthermore, to take "correct" actions, contemporary methods extract and apply useful information learning from the population of GAs, though this information may unnecessarily be about building blocks.

Linkage Learning was introduced to form a novel crossover operator (Harik and Goldberg, 1996), avoiding to destroy building blocks. Later on, the hierarchical clustering algorithm is applied to learn a linkage tree, creating building blocks for the crossover of the linkage tree GA (Thierens, 2010), which belongs to the family of gene-pool optimal mixing EAs (GOMEAs) (Thierens and Bosman, 2011). The GOMEA learns the maintained population's general linkage model, called the Family of Subset (FOS). A FOS is usually a set of variables, and for each FOS, GOMEA varies the values of the corresponding variables. Many variants exist for GOMEA, and the algorithms have been applied in both discrete and continuous optimization (Bosman and Thierens, 2013; Bouter et al., 2017).

The compact GA (CGA) (Harik et al., 1999) is another GA extension inspired by the idea of building blocks. However, CGA represents the population as a distribution over the set of solutions. New solutions are sampled based on the distribution, and the distribution is updated for each generation. Nowadays, this technique is known for the family of estimation of distribution algorithms (EDA) (Hauschild and Pelikan, 2011). EDAs maintain a stochastic model for the search space, and this model can be built for discrete, continuous, and permutation-based solutions.

2.2.5 Outlook

As before, the GA can still be represented by the iterative order *selection*, *variator*, and *evaluate*. A significant difference from the SGA introduced in Bäck and Schwefel (1993) is that the contemporary GAs do not necessarily follow the generational model, maintaining consistent population size. Moreover, more techniques are available for the variators to create new solutions. Novel operators have been proposed for mutation and crossover. In addition, contemporary variators learn to create new solutions using linkage information intelligently. GAs have been applied in various domains, although we cannot list all the applications, such as permutation-based problems (Oliver et al., 1987; Mathias and Whitley, 1992; Nagata and Kobayashi, 1997; Whitley et al., 2010; Goldberg and Lingle, 1985). While more and more techniques appear in various problem-specific domains, theoretical work is developing to help us understand how the GA works. However, providing guidelines for selecting the optimal operators for a given problem remains challenging while the GA community is putting effort towards this goal (Bartz-Beielstein et al., 2020; Ye, 2022).

2.3 Evolutionary Programming

The development of evolutionary programming (EP) dates back to the mid-1960s, too. Initial work was presented by Fogel et al. (1965, 1966) for evolving finite state machines (FSM) with the aim to solve prediction tasks generated from Markov processes and non-stationary time series. EP was originally proposed as an alternative method for generating machine intelligence (Fogel, 1994). Genetic variation was performed by mutating the corresponding discrete, finite alphabet with uniform random permutations. Each candidate machine in the parent population produced one offspring by mutation. The best performing half number of parents and offspring were then selected for the following generation. This selection method is analogous to the $(\mu + \lambda)$ strategy commonly used in ES. In addition, the original work for evolving finite state machines also offered the potential for exchanging parts of finite state machines, which one might consider as a proposal for a "crossover" operator (Fogel et al., 1966).

D. B. Fogel later enhanced EP by adapting genetic variation on real-valued variables with normally distributed mutations (see e.g., Fogel et al., 1990; Fogel, 1992, 1993). EP traditionally did not use any form of crossover; therefore, the evolutionary process typically relied on a set of mutation techniques that have been proposed over the last decades. In general, EP viewed each candidate solution as a *reproducing population*, whereby the used mutation operator simulates all changes that occur between the current and the subsequent generation. Like evolutionary strategies, EP stressed the role of mutation in the evolutionary process. Traditionally, EP used normally distributed mutation as the primary search operator. During the course of the 1990s, the annual conference on EP was held until 1998. Several studies in the mid- and late-1990s showed that using a Cauchy random variable can enhance performance for certain parameter optimization problems (Yao and Liu, 1996). Moreover, a multi-operator approach to EP

was proposed in this period. It was shown that this multi-mutational self-adaptive EP strategy was able to adjust the use of both, the Gaussian and Cauchy mutations, and was thereby able to provide greater rates of optimization when compared to the sole use of Cauchy mutations (Saravanan and Fogel, 1997). A runtime analysis of EP using Cauchy mutations has been presented by Huang et al. (2010).

EP has been very successful in evolving strategies for certain games (see, e.g., work by Chellapilla and Fogel, 1999a). Most remarkable is the use of EP to evolve neural networks that were able to play checkers without relying on expert knowledge (Chellapilla and Fogel, 1999b; Fogel, 2000). In this approach, EP created a population of 15 artificial neural networks. Each candidate network created an offspring with Gaussian mutation that was controlled with a self-adaptive step size. To validate the strength of the best-evolved networks, the created artificial checkers players operated within a computer program called *Blondie24* on the online board gaming site Zone.com. Over a two-month period, 165 games against human opponents were played. The estimated rating of Blondie24 was in the mean 2045.85 with a standard error of 0.48 whereby the master level starts at a rating of 2200. The best result from the 165 games was seen when the network defeated a player with a rating of 2173, which is only 27 points away from the master level. The story of Blondie24 is described in David B. Fogel's book *Blondie 24—Playing at the Edge of AI* (Fogel, 2002). Later on, Fogel applied the approach to chess and created Blondie25, a chess program that was able to win over a nationally ranked human chess player and also showed some wins (one out of twelve games for black, two of twelve for white) over Fritz 8.0, which was the number five ranked program in the world at that time (Fogel et al., 2006; see also Fogel, 2023).

When considering parameter optimization problems as defined in Equation (1), early version of ES and EP shared quite a few similarities, as we already outlined in Bäck and Schwefel (1993). From the year 2000, algorithms based on the concept of CMA-ES are more prominent for this type of application than EP-based approaches, and today one finds that the term *evolutionary programming* is sometimes used to denote algorithms for continuous parameter optimization that are solely based on mutation (Abido and Elazouni, 2021) or the term is also sometimes confused with *genetic programming* (Jamei et al., 2022).

3 Other Important Developments

In this section, a few developments are discussed that we consider of strong importance for the field, however, again this is a subjective selection and a wide range of topics cannot be covered in this paper.

3.1 Parameter Control

It is truly amazing to see that early parameter control techniques such as the 1/5-th success rule (Rechenberg, 1973) and Hans-Paul's early approaches for adapting variance(s) and covariances of an n-dimensional Gaussian distribution (Schwefel, 1981), later on proposed in a similar way for genetic algorithms and binary search spaces $\{0,1\}^n$ (Bäck and Schütz, 1996; Kruisselbrink et al., 2011), as well as mixed search spaces (Li et al., 2013), have now turned into a standard approach in evolutionary computation. Thanks to the wonderful progress that was achieved in the theoretical analysis of evolutionary algorithms in the past 30 years, we also now have a fairly good understanding of why such approaches are helpful, how to analyze them, and what the time complexity and optimal parameter settings for some (classes of) test functions look like—both for

specific pseudo-Boolean optimization problems and for specific continuous parameter optimization problems. This field of research is huge now, and giving an overview is impossible, so a selection of just a few research works is incomplete and subjective, and therefore we abstain from trying to do so here.

3.2 Multimodal Optimization, Novelty Search, and Quality Diversity Algorithms

In this subsection, we consider the development in multimodal optimization in the last 30 years and a relatively new generalization of such optimization referred to as *Novelty Search* and *Quality Diversity* algorithms. At first, we introduce and motivate multimodal optimization, then highlight the development of niching as one of the most important concepts in multimodal optimization, and then proceed with describing several algorithms that use this concept including the most promising state-of-the-art algorithm. Secondly, we show how the idea of multimodal optimization developed to a more general setting by introducing and motivating Novelty Search and Quality Diversity. Thirdly, we introduce Multidimensional Archive of Phenotypic Elites (MAP-Elites) algorithm as one of the historically first and most influential methods in quality diversity and discuss its limitations and how they are overcome. We finish with a discussion of state-of-the-art quality diversity algorithms and the scenarios of their application.

3.2.1 Multimodal Optimization

Traditional single-objective optimization focuses on obtaining a solution with the best possible quality with regard to the given objective function in the smallest possible time. However, this exact setup may be not targeted by engineers and other practitioners who apply optimization at the beginning of the construction process. For this kind of application, it is relevant to explore possible trade-offs that give sufficiently good performance in acceptable time (Bradner et al., 2014). Such exploration may be seen as the task of obtaining a diverse set of solutions with a fitness value above the given threshold. This task is called multimodal optimization (MMO) since it is common for the considered objective functions to have many local optima or modes.

To approach MMO, a number of algorithms were developed. The first algorithms of this type were proposed back in the 1970s (De Jong, 1975). They were extensions of genetic algorithms by *niching* methods that aim to maintain the diversity in the populations (Preuss, 2015). Works of the mid-1990s followed where niching methods were improved. A *clearing* method was proposed (Pétrowski, 1996) where similar individuals are penalized when their number exceeds the given threshold. Restricted tournament selection modifies the selection procedure for creating the new population (Harik, 1995) by restricting the competition to take place only between similar offspring. This ensures that the population has a diverse set of individuals even if the quality of some of them is worse than others. Both those niching methods use distance in the search space as the criterion of similarity. The threshold (or niche radii) for this criterion in particular, and other parameters of those niching algorithms in general, cannot be efficiently determined a priori without assumptions about the optimized function. The series of works followed to solve this problem, in particular Shir and Bäck (2006) and Shir et al. (2007) propose to adaptively control this threshold during the optimization process. The latter work also introduced niching into the CMA-ES for the first time. The comprehensive survey on variations of niching can be found in Das et al. (2011).

Apart from niching in evolutionary algorithms, the same idea of diversity in population can be applied to other metaheuristics. Particle swarm optimization was extended

with niching by Qu et al. (2012). It uses the Euclidean distance-based neighborhood of every particle and picks several locally best-performing ones to guide the search for every particle. Thomsen (2004) proposed to extend differential evolution with niching using the classical idea to penalize individuals that are close. Surveys by Das et al. (2011) and Li et al. (2016) thoroughly describe other important methods and ideas for extending existing algorithms to MMO settings. According to Preuss et al. (2021), one of the leaders in competitions in MMO that were held on CEC/GECCO from 2016 to 2019 and the winner among niching algorithms when a greater budget is given is HillVallEA (Maree et al., 2018). This algorithm makes extra evaluation of the objective function to check the existence of worse solutions (hills) between the given pair of solutions to decide whether they belong to the same niche (valley).

3.2.2 Novelty Search

In the optimization community, mainly in evolutionary robotics, diversity with regard to the behavior of the solutions started to be considered around 2010 (Lehman and Stanley, 2008). In such domains, the individuals do not describe the complete solution, but the complete description may be obtained during the evaluation of the objective function, for example, the simulation of robot movements. The set of all possible descriptions is called *behavior space* (BS). The first algorithms that approached this modified goal aimed to explore this space, because the connection with genotype space is not known for the optimizer. Such approaches, known as novelty search algorithms (Lehman and Stanley, 2011a), changed the goal from obtaining high-quality solutions to the exploration of BS. Instead of optimizing the single given value of the objective function, they optimize the distance in BS between the sampled solution and solutions in the previously collected archive. Illumination of the landscape of BS produces a diverse set of solutions with different qualities. However, only the solutions of sufficiently good quality are interesting for practical applications.

3.2.3 Quality-Diversity

The family of algorithms that search for a distributed set of sufficiently good solutions in BS is known as *Quality-Diversity* (QD) algorithms (Pugh et al., 2015). Following Chatzilygeroudis et al. (2021), the goal of the QD algorithms may be seen as a generalization of MMO, such that the objective function is maximized as in the MMO case but the diversity is targeted with regard to the given behavior descriptor function. One of the first QD algorithms proposed in 2011 is an extension of novelty search and optimizes two objectives: novelty objective (the same as in the original novelty search) and local competition objective (number of closest solutions with lower objective value) (Lehman and Stanley, 2011b). Optimization of those objectives allows the generation of a set of diverse solutions with good local quality. Approximately at the same time, the MAP-Elites (Mouret and Clune, 2015) algorithm was proposed. Due to the simplicity of its idea and efficiency in practice, this algorithm gained a lot of attention in the optimization area and became a classical QD optimizer. Due to the great influence of this algorithm on QD optimization, we give its description in more detail in the next paragraph.

MAP-Elites maintains an archive of previously obtained solutions in a special container. The container discretizes the BS and splits it into cells. Every solution in the container belongs to the cell that corresponds to its behavior descriptor and eliminates from the container a worse solution (with respect to the objective function) that shares the same cell with it. To update the container MAP-Elites follows the general template of evolutionary algorithms. In the original version, random selection is applied to choose a

number of solutions from the container, then mutation and crossover are applied to the selected individuals to create a new generation of individuals, then the objective function is evaluated for each of them to obtain their behavior descriptors and performance, and then the container is updated.

The standard version of MAP-Elites was applied in a number of domains (see Section 3.1 of Chatzilygeroudis et al., 2021). However, it exhibits two limitations: (i) the high dimensionality of BS does not allow for efficient storage of the container; and (ii) the expensiveness of the objective function restricts the number of evaluations that the QD algorithm may afford to make in a reasonable time. Different modifications of this algorithm were proposed to overcome those problems; here we mention only the first ones. Vassiliades et al. (2017) addressed the problem of high-dimensional BS by using Voronoi diagrams to store the individuals in the container instead of the traditional grid, which occupies exponentially more memory with the growth of the number of dimensions. Gaier et al. (2017) addresses the problem of optimizing the expensive objective function by building a surrogate of the function and applying EGO for optimization.

QD algorithms are actively developed to satisfy the needs of modern applications. In the case when the BS is not simple to define before the optimization, Cully and Demiris (2018) proposed to apply autoencoders to automatically learn the latent space of digits images and use it as BS of a robotic hand that learns to write digits. In the case of a noisy function, Flageat and Cully (2020) proposed to store multiple individuals in every cell of the MAP-Elites container, select only the individuals with the highest quality, and change the randomly selected individual in the cell to the freshly added individual. This procedure ensures that the search is not misled by the deceptiveness of a noisy environment. Pierrot et al. (2022) considered the case of multiobjective functions and proposed to maintain the Pareto front in every cell of MAP-Elites' grid. The benchmarking study of this algorithm showed that it maintains a diverse set of individuals and at the same time manages to outperform (with respect to the global hypervolume) traditional multi-objective algorithms in many cases while not significantly losing on the rest of the considered functions. Urquhart and Hart (2018) first proposed a MAP-Elites based QD algorithm for discrete settings and applied it to workforce scheduling and routing problems, resulting in new best-known solutions for several problem instances.

All of these areas are strongly related, as analyzed and compared also in Hagg et al. (2020). Their use to support engineers in finding diverse solutions to engineering design problems (Hagg et al., 2018) and in involving the user experience in the ideation process (Hagg et al., 2019) are highly relevant to works in real-world optimization.

3.3 Constrained Optimization

Another highly relevant topic for parameter optimization is the handling of constraints, as these are typically occurring in real-world optimization settings. Consequently, the past 30 years have seen tremendous development in constraint-handling approaches (Coello, 2022), which, 30 years ago, was often limited to the simple "death penalty" approach of discarding infeasible offspring. Ordered from the most straightforward to more complex ones, we categorize contemporary approaches into six groups.

(i) Feasible solution preference is the most straightforward strategy. It works by always preferring feasible solutions over infeasible solutions (also known as the death penalty (Kramer, 2010), and preferring solutions with a small constraint violation over

solutions with large constraint violations (Deb, 2000; Mezura-Montes and Coello, 2005). A downside of this method is that little information is preserved in the population from infeasible solutions in the optimization process. Consequently, the algorithms using this method can get stuck in local optima.

(ii) Penalized unconstrained optimization uses a so-called penalty function that assigns a cost to constraint violations (Mezura-Montes and Coello, 2011). Penalty functions can be static, dynamic, adaptive, or even based on self-adaptive fitness formulations and stochastic ranking approaches. Penalty functions for constraints, however, lead to a strong bias toward feasible solutions. This is not always ideal, as for example, a high cost assigned by a bad penalty function could cause a selection operator to prefer a feasible solution far away from the optimum over a solution very close to the optimum which is only slightly infeasible. All the information from this promising solution would be lost and not maintained in the population. Unfortunately, the best penalty function cannot be known upfront, and tuning the hyper-parameters of the penalty function for each constraint requires a lot of additional function evaluations (Richardson et al., 1989).

(iii) Repair mechanisms aim to repair solutions that violate the constraints, and are often based on constraint-specific characteristics. Since they are often problem-dependent, a range of different repair mechanisms has been proposed (Salcedo-Sanz, 2009), for example, gradient-based repair mechanisms (Chootinan and Chen, 2006; Zahara and Kao, 2009) and strategies that reduce the probability to generate infeasible offspring (Liepins and Vose, 1990; Arnold and Hansen, 2012).

(iv) Constraint separation considers both the constraint and the original problem as separate optimization problems and aims to solve each of these independently. This can, for example, be achieved via coevolution, where two populations interact with each other (Paredis, 1994), where one population is focused on the fitness of solutions, and the other on constraint satisfaction. Another frequently used approach considers the constraints as additional objectives (Surry et al., 1997; Coello and Montes, 2002). After optimizing the multiobjective (see Section 3.5) problem, the solution with the best true objective function value without any constraint violation can be selected. A downside of this method is an unnecessarily more complex objective space and a strong bias towards the regions where the constraints are violated in the optimization process.

(v) Model-assisted constraint optimization uses machine learning to identify feasible solutions, by training a classification model or fitting a surrogate model with the already evaluated solutions of the constraint function. The trained models can then be used to classify or predict which solutions are feasible and which are not. Simple classification algorithms (such as SVMs; see Poloczek and Kramer, 2013) can be used in this approach, but more advanced regression or surrogate models (see Section 3.4) can also be used. A benefit of model-assisted constraint optimization is that no information is lost from evaluated solutions and that a lot of function evaluations can be saved. However, this often comes at the cost of a substantial increase in computational resources.

(vi) Hybrid methods exist that make use of a combination of the above-mentioned techniques. Noteworthy mentions are, for example, SACOBRA, which uses a repair mechanism in combination with model-assisted constraint optimization techniques (Bagheri et al., 2017) and automatic constraint-handling technique selection based on problem characteristics (Mallipeddi and Suganthan, 2010).

These constraint-handling strategies are not necessarily limited to being used solely in single-objective optimization, but can also be applied to find feasible solutions for multiobjective problems. The only clear difference between them is a different way of handling the objectives (see Section 3.5 for more information).

3.4 Surrogate-Assisted Approaches

Optimization based on computer simulations or real-world experiments can be very expensive in terms of time and money. Therefore, the automotive, marine, aviation, and other engineering fields that make use of computer simulations make use of *surrogate-assisted* optimization algorithms.

Surrogate-assisted evolutionary computation is a variant of EC that utilizes a surrogate model, also known as a *metamodel* or *response surface model*, to guide the search process. In traditional EC, the objective function, which maps a set of input parameters to a single output value, is evaluated directly to determine the fitness of candidate solutions. In surrogate-assisted EC, the objective function is approximated using a surrogate model, which is trained on a subset of the evaluated points and can be used to predict the objective function value at any point in the search space. The use of a surrogate model allows the search process to be more efficient by reducing the number of function evaluations required to find good solutions. This is particularly useful when the objective function is computationally expensive to evaluate or when the search space is large and complex.

The original idea of using surrogate models in optimization comes from the *Efficient Global Optimization* (EGO) framework (*Bayesian optimization*) by Jones et al. (1998). In EGO, a Gaussian Process Regression (GPR) model is used as a surrogate model where the GPR is exploited to select a promising candidate solution. At each iteration, the most promising (yet unseen) candidate solution is evaluated on the real objective function. In surrogate-assisted EC, similar techniques are used to reduce the number of real objective evaluations by using the surrogate models as a filter to remove the most likely bad individuals from the population before the real evaluation. Surrogate-assisted EC was mainly motivated by reducing the time-complexity of expensive problems such as aerodynamic design optimization (e.g., Jin and Sendhoff, 2009) or drug design (e.g., Douguet, 2010).

Different strategies for using surrogate models are often known as model management or evolution control. This is specifically challenging when the problems are of high dimension. The first works in this field date from the mid-1990s (e.g., Ratle, 1998; Yang and Flockton, 1995). Surrogate models can be applied to almost all operations of EAs, such as population initialization, mutation (Abboud and Schoenauer, 2001), crossover (Anderson and Hsu, 1999), and fitness evaluations (Rasheed and Hirsh, 2000).

3.4.1 Advances in Model Management Strategies

Model management strategies for fitness evaluations can generally be divided into population-based, individual-based, and generation-based strategies (Jin, 2005). Population-based means that more than one subpopulation coevolves. Each subpopulation uses its own surrogate model for fitness evaluations. Individual-based methods use a surrogate for some of the individuals in a generation and use the real function on the rest of the individuals (Jin et al., 2000). Generation-based means that the surrogate is used for fitness evaluations in some of the generations, and the real objective function in the rest (Ratle, 1998; Jin et al., 2002).

3.4.2 Advances in Surrogate Models

The original EGO framework uses Gaussian Process Regression, or Kriging (Rasmussen and Williams, 2006), as the default surrogate model due to its capability of providing uncertainty quantification. However, for surrogate-assisted EC this uncertainty is not always required and other machine-learning models can be used as well. There are also many advances in EGO surrogate models that use Kriging approximation methods to deal with larger numbers of dimensions or samples (Van Stein et al., 2020; Raponi et al., 2020; Antonov et al., 2022). Next to Kriging(-like) methods, popular choices are Radial Basis Functions (Sun et al., 2018), artificial neural networks (Jin et al., 2002), polynomial repression (Zhou et al., 2005), and support vector regression (Wang et al., 2015). Recent advances also include the use of surrogate ensembles and incorporate automated hyperparameter optimization methods (AutoML) for selecting a well-fitting surrogate model (Yu et al., 2020, 2022; Huang et al., 2022; de Winter et al., 2022).

3.5 Multiobjective Optimization

While multiobjective optimization almost was in its infancy 30 years ago, nowadays it is almost impossible to imagine evolutionary computation without this important subfield. Evolutionary multiobjective optimization now defines its own research field and requires other approaches than single-objective optimization. The main reason for this is that in multiobjective optimization, we are often not interested in a single solution, but in finding the set of Pareto-optimal solutions (Konak et al., 2006). The first multiobjective optimization algorithm (VEGA, see Schaffer, 1985) splits the population into multiple subpopulations where each population would be optimized for a different objective. This strategy, however, tends to only converge to the extremes of each objective. In the 30 years following, researchers realized that there was a need for diversity mechanisms, elitism in the population, and other fitness functions to find well-spread solutions along the entire Pareto frontier. Three evolutionary multiobjective optimization algorithms that had these characteristics quickly started to define the state of the art (Deb, 2007):

1. *Nondominated Sorting Genetic Algorithm II (NSGA-II)* by Deb, Agrawal et al. (2002) became popular because it uses an elite-preservation mechanism, enhances a diversity mechanism, and emphasizes nondominated solutions. The emphasis on the nondominated solutions is done with a selection operator that selects only the top N nondominated solutions for crossover to generate new individuals.
2. *Strength Pareto Evolutionary Algorithm 2 (SPEA2)* by Zitzler et al. (2001) keeps a fixed archive of nondominated solutions where nondominated solutions are replaced with solutions that dominate more individuals. The search for new solutions is guided by a k-th nearest neighbor density estimation technique.
3. *Pareto Envelope-based Selection Algorithm (PESA2)* by Corne et al. (2001) is an evolutionary strategy that compares a parent selected from a hyperbox with its offspring child. If the child dominates the parent, the child is accepted as the next parent. All nondominated solutions are put in an archive, and to keep the archive small only the solutions in the least crowded regions are kept.

In the years following, the multiobjective optimization research field was enhanced with strategies for many-objective optimization (e.g., Wagner et al., 2007; Deb and Jain, 2013; Ishibuchi et al., 2008), for problems with expensive function evaluations (e.g., Ponweiser et al., 2008; Knowles, 2006; Santana-Quintero et al., 2010), interactive

multiobjective optimization (e.g., Miettinen, 2014; Xin et al., 2018), and constraint multiobjective optimization (e.g., Qu and Suganthan, 2011; Blank and Deb, 2021; de Winter et al., 2022).

3.6 Automated Algorithm Design

As the number of available algorithms to solve parameter optimization problems keeps growing, as evidenced by the many developments in the algorithms described in Section 2, the question of how to choose one of them is gaining considerable importance. While the problem of *algorithm selection* (see Kerschke et al., 2019; Muñoz et al., 2015 for surveys), where we want to pick one algorithm to solve a particular problem, has been around for decades (Rice, 1976), extensions on this original framework have gained more traction in the last decade.

The question of *algorithm configuration* or meta-optimization (see Huang et al., 2019 for a survey), where we want to tune parameters for a specific algorithm, has become more feasible as computational resources are more readily available. Simultaneously, the number of parameters of most methods in evolutionary computation is growing as more modifications are proposed. As such, tuning of parameters is becoming more prevalent (Eiben and Smit, 2011), and tools for algorithm configuration are gaining popularity (e.g., see López-Ibáñez et al., 2016; Lindauer et al., 2022).

While tuning parameters for an overall performance gain on a specific function or set of functions is natural, further benefits are expected when the choice of algorithms or their parameters can be changed during the optimization process. In contrast to self-adaptation, this notion of *Dynamic Algorithm Configuration* (DAC; see Biedenkapp et al., 2020; Adriaensen et al., 2022) often relies on methods like reinforcement learning, which enable more general adaptation of not just a single parameter, but potentially switching multiple operators at once (see Eimer et al., 2021; Sun et al., 2021). Results with DAC have been promising, showing the potential to mimic complex traditional self-adaptation schemes (Shala et al., 2020).

These online configuration tasks are feasible only if we consider our algorithms to be modular in nature, so changes between operators are as simple as swapping a module, as we have recently shown (de Nobel et al., 2021a; Vermetten et al., 2019). However, even when this modularity is not present, advantages could be gained by switching between algorithms to exploit differences in their behavior (Vermetten et al., 2020). While this notion of switching between algorithms has shown promise in the context of hybrid algorithm design, online policies to switch between algorithms on a per-run basis, for example, based on the internal state and the landscape as seen by the algorithm, are promising future directions (e.g., Kostovska, Jankovic et al., 2022).

3.7 Particle Swarm Optimization

Particle Swarm Optimization (PSO), first introduced by Eberhart and Kennedy (1995), is inspired by the swarming behavior of animals, observed in flocks of birds and schools of fish. In PSO, a group of candidate solutions, called particles, are modeled as moving in a multidimensional search space. Each particle i maintains its own position $\mathbf{x}_i \in \mathbb{R}^n$ and velocity $\mathbf{v}_i \in \mathbb{R}^n$, which it updates based on its own experience $\mathbf{p}_i \in \mathbb{R}^n$ and the experiences of other particles in the swarm $\mathbf{p}_s \in \mathbb{R}^n$. In canonical PSO, after random initialization of the position and velocity of each particle, the algorithm loops until termination criteria are met. In each iteration, the velocity of each particle is updated using:

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \mathcal{U}(0, \phi_1) \otimes (\mathbf{p}_i - \mathbf{x}_i) + \mathcal{U}(0, \phi_2) \otimes (\mathbf{p}_s - \mathbf{x}_i). \tag{12}$$

Here, $\mathcal{U}(0, \phi)$ represents a vector of random numbers, uniformly distributed in $[0, \phi]^n$, and $\otimes$ is component-wise multiplication. The parameters ϕ_1 and ϕ_2 are often referred to as the social and cognitive components and control the impact of the local best value $\mathbf{p}_i$ and the global best value $\mathbf{p}_s$. Following the update of $\mathbf{v}_i$, the position of each particle is updated by adding the velocity to the current position, i.e., $\mathbf{x}_i + \mathbf{v}_i$. Several variations to this general scheme exist; see Boks et al. (2020) for an overview. In the original PSO algorithm, hard velocity bounds are used (minimum and maximum velocity) to prevent particles from leaving the search space. It is, however, not a trivial task to choose these constraints, and particles may fail to converge using this strategy. Attempts to better control the scope of the search without using hard velocity constraints include Shi and Eberhart (1998), which introduces an *inertia weight* ω, that dampens the rate of change of $\mathbf{v}_i$. Large values of ω result in more exploratory behavior while a small value results in exploitative behavior. It was proposed by Shi and Eberhart (1998) to reduce the value of ω adaptively from 0.9 to 0.4 during the algorithm run. Notable variants of PSO are *Fully Informed Particle Swarm (FIPS)* (see Kennedy and Mendes, 2002), which uses the best previous positions of all its neighbors, and *Bare-Bones PSO* (see Kennedy, 2003), which does not use velocity at all but updates the position using a Gaussian probability distribution.

3.8 Differential Evolution

Originally proposed in 1995 for parameter optimization problems, *Differential Evolution (DE)* (Storn and Price, 1995) has rapidly gained popularity over the subsequent years due to the simplicity of its implementation, low number of parameters, low space complexity, and competitive performance on a wide range of problems (Das and Suganthan, 2011).

The general framework of DE is as follows: the algorithm starts with a population of N solutions typically initialized at random as real-valued vectors inside problem boundaries. Then, within the generational loop of DE, all solutions in the population sequentially one-by-one undergo a multi-stage perturbation made up of: (i) *mutation*: first, a number of "donor" solutions is selected uniformly at random from the population for each "current" solution (now called "target"), then a new "mutant" solution is generated as a sum of a selected solution (called "base") selected following the mutation strategy and a scaled difference between pairs of abovementioned selected solutions; (ii) *crossover/recombination*: components of the new "trial" solution are produced via exchanging components, with some probability, between mutant and target solutions; (iii) *selection*: trial and target solutions deterministically compete for a spot in the next generation in terms of their values of the objective function. Similar to other EC algorithms, the termination criterion can be either based on the precision or the budget of function evaluations.

All classic DE variants can be described via a general notation *DE/a/b/c* where *a* stands for mutation strategy, *b* denotes the number of difference vectors employed in the mutation strategy (e.g., *rand/1* Storn and Price, 1995; *rand/2*, *best/1*, *best/2* originally proposed in Price et al., 2005; and more advanced versions like *target-to-pbest/1*, Tanabe and Fukunaga, 2013 and Zhang and Sanderson, 2007; *target-to-best/2*, Gong and Cai, 2013; *target-to-rand/1*, Qijang, 2014, Sharma et al., 2020, Qin et al., 2009; *2-Opt/1* Chiang et al., 2010), and *c* specifies the crossover strategy (e.g., originally proposed *bin*, Storn and Price, 1995; *exp*, Price et al., 2005; or more advanced options from Das et al., 2016). Initialization (uniform, Halton, Gaussian, etc.) and feasibility-preserving (Boks et al.,

2021) operators are traditionally not included in such notation, typically leaving the algorithm's specification somewhat ambiguous (Kononova et al., 2022).

3.8.1 Differential Evolution in the Context of EC

DE is *closely related* to a number of other heuristic algorithms, namely: (i) the standard EA, due to the same computational steps involved in both algorithms; (ii) annealing-inspired algorithms [1] (Kirkpatrick et al., 1983), due to built-in scaling down of the "step size" in both algorithms; (iii) the Nelder-Mead algorithm (Nelder and Mead, 1965) and controlled random search (CRS) (Price, 1977), both of which also operationally rely heavily on the difference vectors to perturb the current trial solutions.

At the same time, DE is *remarkably different* from traditional EC algorithms (Das and Suganthan, 2011): (i) population-derived differences used for perturbation in DE automatically adapt to the natural scaling of the problem, unlike, for example, an ES which requires explicit rules for adaptation of step size per variable over time; (ii) perturbations to solutions are generated from evolving nonzero population difference vectors rather than a predefined density function in, for example, Estimation of Distribution Algorithms (EDA)—the so-called *contour matching* property of DE (Price et al., 2005) which allows both a mutation step's size and its orientation to adapt automatically to the landscape of the objective function; (iii) donor solutions in the mutation step of DE are selected without any reference to their fitness values, unlike a GA, where typically some sort of fitness-based parent selection takes place; (iv) while both algorithms ensure *elitism*, one-to-one competition for survival in DE between parent and child mostly resembles that of a $(\mu + \lambda)$-ES, where surviving solutions are selected from the joined pool of parents and children; (v) survivor selection in DE is local as solutions are compared only against a (relatively local) mutated version of itself and not against solutions that have evolved in distant parts of solution space—apart from multi-population versions, no standard EC algorithm behaves in a similar fashion (Price et al., 2005). Finally, the latter property prevents premature convergence and makes DE perfectly suitable for parallelization.

3.8.2 Parameter Control

Population size. To employ the full potential of the chosen mutation operator, the population size N_p should allow choices of non-repeating indices of donor solutions (i.e., $N_p \geq 4$ for *rand/1*). At the same time, too large populations should be avoided as potentially deleterious for the convergence process, at the expense of an adequate exploitation phase to refine promising solutions locally.

Control parameters. DE mutation is parameterized via *scale factor* $F \in (0, 2]$, whose value in practice should be sufficiently high to counteract selection pressure and generate sufficiently diverse trial solutions to avoid premature convergence (Zaharie, 2002). While values $1 < F < 2$ are generally less reliable, their occasional utilization can be beneficial within adaptive schemes described below. DE crossover is parameterized via *crossover rate* $C_r \in [0, 1]$, which controls the proportion of components of the mutant solution copied into the trial solution and, counterintuitively at first, defines a mutation rate—the probability that a component is inherited from a mutant (Price et al., 2005).

[1] In fact, the first version of DE originated as a floating-point-arithmetic modification of a genetic annealing algorithm (Price et al., 2005) proposed in 1994 in a popular programmer's magazine, *Dr. Dobb's Journal*.

(Self-)adaptation. Population size clearly influences population diversity: larger population size means higher diversity and therefore a better exploration of the search space (Zaharie and Micota, 2017). Such reasoning gave rise to algorithms with adaptive schemes, controlling a gradual reduction of *population size* over the optimization run, to lower chances of stagnation (Brest et al., 2008; Zamuda and Brest, 2012; Tanabe and Fukunaga, 2013), following similar developments in other subfields of EC.

4 Theoretical Aspects

Huge efforts have been devoted in the past 30 years to develop a rigorous theoretical analysis of evolutionary computation, in particular for genetic algorithms and evolution strategies. Following the pioneering work of John Holland's schema theorem (Holland, 1975) on genetic algorithms, Wegener (2001) used a fitness level-based method to analyze the expected running time of the $(1 + 1)$ EA. This work presented the first proof that crossover can reduce the running time of EAs on the so-called royal road functions (Garnier et al., 1999). The analysis of the $(1 + 1)$ EA for more theoretical problems (e.g., Linear and LeadingOnes) was presented in the work of Droste et al. (2002). Recently, attention was also drawn to the impact of hyperparameters on the running time, for instance, for EAs with parameterized population sizes (Jansen et al., 2005; Witt, 2006; Rowe and Sudholt, 2014; Antipov et al., 2019) and mutation rates (Böttcher et al., 2010; Doerr et al., 2013; Gießen and Witt, 2017). Furthermore, extensive work on the analysis of the crossover operator has also been done in Doerr and Doerr (2015); Doerr et al. (2015); Dang et al. (2017); Corus and Oliveto (2017); Sudholt (2017); Pinto and Doerr (2018); and Antipov et al. (2020).

For continuous optimization, many contributions focus on the asymptotic analysis of evolution strategies, such as the inspiring and fundamental analysis of Beyer (2001) and Auger and Hansen (2011). Moreover, the well-known drift analysis has been introduced by He and Yao (2001, 2003), serving as a general theoretical framework. It was employed for several challenging theoretical questions. For instance, Doerr et al. (2012) provides an elegant proof for the $\mathcal{O}(n \log n)$ running time of the $(1 + 1)$ EA on any linear function. In Witt (2013) tight runtime bounds of $(1 + 1)$ EA are presented, and Akimoto et al. (2018) uses drift analysis to analyze the runtime bound for the $(1 + 1)$ evolution strategy with the 1/5-success rule. Lengler has summarized some examples of using drift analysis for EAs (Lengler, 2020).

For multiobjective evolutionary algorithms (MOEAs), theoretical results are mainly derived for simple MOEAs on discrete problems. For example, the runtime analyses for (global) simple evolutionary multi-objective optimizers on classic bi-objective pseudo-Boolean optimization problems (Laumanns et al., 2004; Giel and Lehre, 2006; Osuna et al., 2020). Very recently, the first theoretical results were presented for the famous NSGA-II algorithm (Zheng et al., 2022; and Doerr and Qu, 2022a, b).

For a more complete overview of the many fundamental theoretical results of EC in recent years, we refer the interested reader to Auger and Hansen (2011); Doerr and Neumann (2020); Krejca and Witt (2020); and Doerr and Neumann (2021).

5 The More the Better? Not Always

In this section, we briefly address the excessive number of optimization algorithms with potentially similar performance in the field, referring to the ever-growing number of paradigms gleaned from nature, claimed to be distinct models for optimization

algorithms (Section 5.1). A possible investigation of this problem can be achieved by proper benchmarking, as we argue in Section 5.2.

5.1 The "Bestiary" and the Problem of Too Many Algorithms

Although the rising popularity of evolutionary computation has led to a vast number of positive contributions in the optimization domain at large, there have also been some rather negative developments over the past 30 years. In particular, the trend of devising "novel" nature-inspired optimization heuristics has been a concern in the last decades, and the trend does not seem to slow down (Hussain et al., 2019).

While early evolutionary computation techniques effectively use metaphors to motivate the algorithmic concepts, the use of metaphor has expanded to the point where it is (un)intentionally obfuscating the underlying algorithmic components (Sörensen, 2015). There is now a veritable zoo of metaphor-based optimization heuristics (Campelo and Aranha, 2021, 2018), each with its own terminology, which is hard to decipher for those with a more standard background in optimization heuristics, and for those who are looking for a well-defined formal algorithmic or mathematical description. These "novel" metaphor-based methods still gain attention, and even though many of them have been exposed as providing no novelty, new ones are still devised and promoted (Weyland, 2010; Camacho Villalón et al., 2020; Camacho-Villalón et al., 2022a). For practitioners who are not familiar with the field, but looking for the best possible choice of an optimization algorithm for solving a real-world problem, the ever-growing zoo of metaphor-based heuristics often confuses them and results in the impossibility of making a sensible choice for a specific problem to solve.

We, therefore, strongly advocate (i) the use of mathematical language and formal pseudocode to describe new algorithms; (ii) avoiding the excessive use of analogies with nature and the corresponding language, and (iii) proper benchmarking comparisons (see Section 5.2) against the state-of-the-art optimization algorithms to justify any proposal for new ones.

5.2 Benchmarking for Algorithm Performance Comparison

With an ever-increasing variety of optimization algorithms, questions about when a particular algorithm is useful become difficult to answer. Despite numerous advances in the theoretical analysis of optimization algorithms, it is infeasible to answer our question with only theory. This is due to the sheer number of variations of evolutionary algorithms and the fact that modern runtime analyses only apply to a limited number of theoretical problems and algorithms. There is, however, still a need to gain insight into the behavior of algorithms to decide when they might apply to a particular real-world scenario. As such, benchmarking has grown from an ad-hoc procedure into a field with many toolboxes to help researchers perform their experiments (Bartz-Beielstein et al., 2020).

The standards for experimentation have improved a lot in the last 30 years, and the importance of not just benchmarking, but standardized and reproducible benchmarking, has become increasingly apparent to the community (López-Ibáñez et al., 2021). Various benchmarking frameworks have been developed, providing access to curated sets of problems (Bennet et al., 2021; Hansen et al., 2021; de Nobel et al., 2021b). This way, performance data can be shared, promoting data reuse and enabling straightforward comparisons to state-of-the-art algorithms. One particular example of benchmarking tools promoting this idea is the IOHprofiler (Doerr et al., 2020), a collaboration between Leiden University, Sorbonne Université, Tel-Hai College, and others, which collects

performance data from a wide range of sources and makes it directly accessible via a web service,[2] such that anyone can analyze the data (Wang et al., 2022).

As noted in Section 3, there is a great variety in the types of optimization problems that can be formulated, and each of them can be applicable for a specific real-world use case (van der Blom et al., 2020). As such, benchmark problems have to be developed for each of these domains. The most used suites in the context of parameter optimization are for single-objective, noiseless optimization without additional constraints, where benchmark collections have largely become standardized in the last decade (Hansen, Finck et al., 2009; Suganthan et al., 2005), but other areas (e.g., constrained optimization) have also seen significant efforts to ensure reliable benchmark setups (see e.g., Hellwig and Beyer, 2019). Some tools, such as the COCO framework, have expanded on a single suite of commonly used problems (Hansen et al., 2010) to include variations with constraints, noise, discrete variables, and multiple objectives (Hansen et al., 2021).

While the specific demands of different communities within EC lead to variations in benchmarking problems, the overall procedures have much in common with each other (Beiranvand et al., 2017). To ensure fair comparisons, evaluation counts provide a hardware-independent measure of computational effort, which allows data generated decades ago to still be used today. While performance data formats have not yet been standardized, moves towards interoperability of benchmark tools are gaining steam, with, for example, analysis pipelines that inherently support data from many different sources (Wang et al., 2022). Further developments in terms of data representation and reuse are promising future directions (Kostovska et al., 2021).

While benchmarking plays a key role in understanding the strengths of optimization algorithms, benchmarking data is useful only when the problem being benchmarked can be clearly understood. While many benchmark problems are built from the ground up to have specific high-level properties (e.g., strong global structure), the way in which the problem and algorithm interact in a lower-level context is often not as easily described. To aid with this process, the field of *Exploratory Landscape Analysis* (ELA) has been developed (Mersmann et al., 2011). This field aims to characterize problems based on sets of low-level features, such as information content of a set of samples. This type of analysis (Kerschke and Trautmann, 2019) has proven useful in areas like algorithm selection (Bischl et al., 2012), but also for benchmarking and understanding the impact of algorithmic components of modular algorithms (Kostovska, Vermetten, et al., 2022).

6 A Subjective Outlook (Not for 30 Years)

In the domain of continuous parameter optimization problems, as defined in Equation (1), it is extremely difficult today for practitioners and researchers to pick the best possible optimization algorithm for a given problem and to parameterize it correctly. First steps in the direction of problem characterization, for example, by means of *Exploratory Landscape Analysis*, and problem-specific *Combined Algorithm Selection and Hyperparameter Optimization* have been made recently, with promising results. Methods such as dynamic algorithm configuration, metaheuristics, and modular algorithm frameworks, as discussed in Section 3.6, clearly indicate one path towards the future of the field—namely, automatically designing the optimal optimization algorithm for a given problem or problem class.

[2]https://iohanalyzer.liacs.nl

Related to this, we would like to emphasize that there are too many nature-inspired optimization algorithms around, with new ones being proposed continuously. We strongly propose for the future to consolidate the field, understand which algorithms are good for which optimization problem characteristics, and develop configurable algorithm frameworks that allow for the generation, experimentation, and proper benchmarking of existing and new algorithms. A few more topics we propose are the following:

- In addition to the common "competitive" form of benchmarking, a focus should also be placed on the explainability of achieved results by understanding the interaction between algorithmic components and problem properties better.
- Where possible, reproducibility should become a requirement for all evolutionary computation publications. We should, however, also leave possibilities to publish about real-world applications, which often use proprietary data or simulators but facilitate a better understanding of the requirements of real-world applications.
- It would be good to use configurable algorithm frameworks with straightforward interfaces to allow easy use of state-of-the-art algorithms.
- We need to get rid of "novel" nature-inspired algorithms and use a standardized description and formalization approach for new algorithms, to easily see whether they are similar or identical to existing methods.

Acknowledgments

It would be a paper on its own to acknowledge all the wonderful people we have had the pleasure to work with during the last 30 years, at the Technical University of Dortmund, at Leiden University, and within our worldwide research network. Thomas Bäck would like to mention one person, Hans-Paul Schwefel, who was the reason that I joined this exciting field of research, and have always enjoyed and continue to enjoy this journey. Thank you very much for this, Hans-Paul, and for your inspiration and continuous support, especially of course for the first years of my journey, as a young PhD student under your guidance. Our special thanks also go to David B. Fogel for his very useful input on Section 2.3, and to Emma Hart for encouraging this project and providing suggestions that helped improve the paper.

References

Abboud, K., and Schoenauer, M. (2001). Surrogate deterministic mutation: Preliminary results. In *Artificial Evolution, 5th International Conference, Evolution Artificielle, EA 2001, Le Creusot, France, October 29–31, 2001, Selected Papers*, pp. 104–116. Lecture Notes in Computer Science, Vol. 2310.

Abido, M. A., and Elazouni, A. (2021). Modified multi-objective evolutionary programming algorithm for solving project scheduling problems. *Expert Systems with Applications*, 183:115338. 10.1016/j.eswa.2021.115338

Adriaensen, S., Biedenkapp, A., Shala, G., Awad, N. H., Eimer, T., Lindauer, M., and Hutter, F. (2022). Automated dynamic algorithm configuration. *Journal of Artificial Intelligence Research*, 75:1633–1699. 10.1613/jair.1.13922

Akimoto, Y., Auger, A., and Glasmachers, T. (2018). Drift theory in continuous search spaces: expected hitting time of the (1 + 1)-ES with 1/5 success rule. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 801–808.

Akimoto, Y., and Hansen, N. (2020). Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary Computation*, 28(3):405–435. 10.1162/evco_a_00260

Aleti, A., and Moser, I. (2016). A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Computing Surveys*, 49(3):1–35. 10.1145/2996355

Anderson, K. S., and Hsu, Y. (1999). Genetic crossover strategy using an approximation concept. In *Proceedings of the 1999 Congress on Evolutionary Computation*, Vol. 1, pp. 527–533. 10.1109/CEC.1999.781978

Antipov, D., Buzdalov, M., and Doerr, B. (2020). Fast mutation in crossover-based algorithms. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1268–1276.

Antipov, D., Doerr, B., and Yang, Q. (2019). The efficiency threshold for the offspring population size of the (μ, λ) EA. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1461–1469.

Antonov, K., Raponi, E., Wang, H., and Doerr, C. (2022). High dimensional Bayesian optimization with kernel principal component analysis. In *International Conference on Parallel Problem Solving from Nature*, pp. 118–131.

Arnold, D. V., and Hansen, N. (2012). A (1+1)-CMA-ES for constrained optimisation. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 297–304.

Auger, A., Brockhoff, D., and Hansen, N. (2011). Mirrored sampling in evolution strategies with weighted recombination. In *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO)*, pp. 861–868.

Auger, A., and Hansen, N. (2005). A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1769–1776.

Auger, A., and Hansen, N. (2011). Theory of evolution strategies: A new perspective. In A. Auger and B. Doerr (Eds.), *Theory of randomized search heuristics: Foundations and recent developments*, Vol. 1, *Series on theoretical computer science*, pp. 289–325. World Scientific. 10.1142/9789814282673_0010

Auger, A., Jebalia, M., and Teytaud, O. (2005). Algorithms (x, sigma, eta): Quasi-random mutations for evolution strategies. In *Artificial Evolution, 7th International Conference, Evolution Artificielle, EA 2005, Lille, France, October 26–28, 2005, Revised Selected Papers*, pp. 296–307. Lecture Notes in Computer Science, Vol. 3871.

Auger, A., Schoenauer, M., and Vanhaecke, N. (2004). LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In *Proceedings of Parallel Problem Solving from Nature*, pp. 182–191. Lecture Notes in Computer Science, Vol. 3242.

Bäck, T. (1994). Parallel optimization of evolutionary algorithms. In *Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature*, pp. 418–427. Lecture Notes in Computer Science, Vol. 866.

Bäck, T., Doerr, C., Sendhoff, B., and Stützle, T. (2022). Special issue on benchmarking sampling-based optimization heuristics: Methodology and software. *IEEE Transactions on Evolutionary Computation*, 26(6):1202–1205.

Bäck, T., Fogel, D., and Michalewicz, Z. (1997). *Handbook of evolutionary computation*. Oxford University Press.

Bäck, T., Foussette, C., and Krause, P. (2013). *Contemporary evolution strategies*. Springer.

Bäck, T., and Schütz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms. In *Proceedings of the 9th International Symposium on Foundations of Intelligent Systems*, pp. 158–167. Lecture Notes in Computer Science, Vol. 1079.

Bäck, T., and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23.

Bagheri, S., Konen, W., Emmerich, M., and Bäck, T. (2017). Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, 61: 377–393. 10.1016/j.asoc.2017.07.060

Ballester, P. J., and Carter, J. N. (2004). An effective real-parameter genetic algorithm with parent centric normal crossover for multimodal optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 901–913. Lecture Notes in Computer Science, Vol. 3102.

Bartz-Beielstein, T., Doerr, C., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., López-Ibáñez, M., Malan, K. M., Moore, J. H., Naujoks, B., Orzechowski, P., Volz, V., Wagner, M., and Weise, T. (2020). Benchmarking in optimization: Best practice and open issues. *CoRR*, abs/2007.03488.

Beiranvand, V., Hare, W., and Lucet, Y. (2017). Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4):815–848. 10.1007/s11081-017-9366-1

Bennet, P., Doerr, C., Moreau, A., Rapin, J., Teytaud, F., and Teytaud, O. (2021). Nevergrad: Blackbox optimization platform. *ACM SIGEVOlution*, 14(1):8–15. 10.1145/3460310.3460312

Beyer, H. (2001). *The theory of evolution strategies*. Natural computing series. Springer.

Beyer, H., and Schwefel, H. (2002). Evolution strategies—A comprehensive introduction. *Natural Computing*, 1(1):3–52. 10.1023/A:1015059928466

Biedenkapp, A., Bozkurt, H. F., Eimer, T., Hutter, F., and Lindauer, M. (2020). Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *Proceedings of the 24th European Conference on Artificial Intelligence*, pp. 427–434. Frontiers in Artificial Intelligence and Applications, Vol. 325.

Bischl, B., Mersmann, O., Trautmann, H., and Preuß, M. (2012). Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 313–320.

Blank, J., and Deb, K. (2021). Constrained bi-objective surrogate-assisted optimization of problems with heterogeneous evaluation times: Expensive objectives and inexpensive constraints. In *Proceedings of the 11th International Conference on Evolutionary Multi-Criterion Optimization*, pp. 257–269. Lecture Notes in Computer Science, Vol. 12654. 10.1007/978-3-030-72062-9_21

Boks, R., Kononova, A. V., and Wang, H. (2021). Quantifying the impact of boundary constraint handling methods on differential evolution. In *Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1199–1207.

Boks, R., Wang, H., and Bäck, T. (2020). A modular hybridization of particle swarm optimization and differential evolution. In *Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1418–1425.

Bosman, P.A.N., and Thierens, D. (2013). More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 359–366.

Böttcher, S., Doerr, B., and Neumann, F. (2010). Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, pp. 1–10. Lecture Notes in Computer Science, Vol. 6238.

Bouter, A., Alderliesten, T., Witteveen, C., and Bosman, P. A. (2017). Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 705–712.

Bradner, E., Iorio, F., and Davis, M. (2014). Parameters tell the design story: Ideation and abstraction in design optimization. In *Proceedings of the Symposium on Simulation for Architecture & Urban Design*, Article 26, pp. 1–8.

Bredèche, N. (2008). A multi-cellular developmental system in continuous space using cell migration. In *Tenth International Conference on Simulation of Adaptive Behavior, From Animals to Animats*, pp. 260–269. Lecture Notes in Computer Science, Vol. 5040.

Brest, J., and Maučec, M. S. (2008). Population size reduction for the differential evolution algorithm. *Applied Intelligence*, 29:228–247. 10.1007/s10489-007-0091-x

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., and Woodward, J. R. (2009). Exploring hyper-heuristic methodologies with genetic programming. In C. L. Mumford and L. C. Jain (Eds.), *Computational Intelligence: Collaboration, Fusion and Emergence*, pp. 177–201. Springer.

Camacho Villalón, C. L., Stützle, T., and Dorigo, M. (2020). Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. In *International Conference on Swarm Intelligence*, pp. 121–133.

Camacho-Villalón, C. L., Dorigo, M., and Stützle, T. (2022a). An analysis of why cuckoo search does not bring any novel ideas to optimization. *Computers & Operations Research*, 142:105747.

Camacho-Villalón, C. L., Dorigo, M., and Stützle, T. (2022b). PSO-X: A component-based framework for the automatic design of particle swarm optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 26(3):402–416.

Campelo, F., and Aranha, C. (2018). EC Bestiary: A bestiary of evolutionary, swarm and other metaphor-based algorithms. Retrieved from https://zenodo.org/record/1293352#.ZCLYqnbMJPY

Campelo, F., and Aranha, C. d. C. (2021). Sharks, zombies and volleyball: Lessons from the evolutionary computation bestiary. In *Proceedings of the CEUR Workshop*, p. 6. Vol. 3007.

Chacón, J., and Segura, C. (2018). Analysis and enhancement of simulated binary crossover. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8.

Chatzilygeroudis, K., Cully, A., Vassiliades, V., and Mouret, J.-B. (2021). Quality-diversity optimization: a novel branch of stochastic optimization. In *Black box optimization, machine learning, and no-free lunch theorems*, pp. 109–135. Springer.

Chellapilla, K., and Fogel, D. B. (1999a). Evolution, neural networks, games, and intelligence. *Proceedings of the IEEE*, 87(9):1471–1496. 10.1109/5.784222

Chellapilla, K., and Fogel, D. B. (1999b). Evolving neural networks to play checkers without relying on expert knowledge. *IEEE Transactions on Neural Networks*, 10(6):1382–1391. 10.1109/72.809083

Chiang, C.-W., Lee, W.-P., and Heh, J.-S. (2010). A 2-Opt based differential evolution for global optimization. *Applied Soft Computing*, 10(4):1200–1207. 10.1016/j.asoc.2010.05.012

Chootinan, P., and Chen, A. (2006). Constraint handling in genetic algorithms using a gradient-based repair method. *Computers & Operations Research*, 33(8):2263–2281.

Coello, C.A.C. (2022). Constraint-handling techniques used with evolutionary algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1310–1333.

Coello, C.A.C., and Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203. 10.1016/S1474-0346(02)00011-3

Coello Coello, C. (2006). Evolutionary multi-objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36. 10.1109/MCI.2006.1597059

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 283–290.

Corus, D., and Oliveto, P. S. (2017). Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 22(5):720–732. 10.1109/TEVC.2017.2745715

Costa, A., Celano, G., Fichera, S., and Trovato, E. (2010). A new efficient encoding/decoding procedure for the design of a supply chain network with genetic algorithms. *Computers & Industrial Engineering*, 59(4):986–999.

Cully, A., and Demiris, Y. (2018). Hierarchical behavioral repertoires with unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 69–76.

Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2017). Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22(3):484–497. 10.1109/TEVC.2017.2724201

Das, S., Maity, S., Qu, B.-Y., and Suganthan, P. N. (2011). Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88. 10.1016/j.swevo.2011.05.005

Das, S., Mullick, S. S., and Suganthan, P. N. (2016). Recent advances in differential evolution—An updated survey. *Swarm and Evolutionary Computation*, 27:1–30. 10.1016/j.swevo.2016.01.004

Das, S., and Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31. 10.1109/TEVC.2010.2059031

Dasgupta, D., and Michalewicz, Z. (2013). *Evolutionary algorithms in engineering applications*. Springer Science & Business Media.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan.

de Nobel, J., Vermetten, D., Wang, H., Doerr, C., and Bäck, T. (2021a). Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In *Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1375–1384.

de Nobel, J., Ye, F., Vermetten, D., Wang, H., Doerr, C., and Bäck, T. (2021b). IOHexperimenter: Benchmarking platform for iterative optimization heuristics. arXiv:2111.04077

de Winter, R., Bronkhorst, P., van Stein, B., and Bäck, T. (2022). Constrained multi-objective optimization with a limited budget of function evaluations. *Memetic Computing*, 14(2):151–164. 10.1007/s12293-022-00363-y

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338. 10.1016/S0045-7825(99)00389-8

Deb, K. (2007). Current trends in evolutionary multi-objective optimization. *International Journal for Simulation and Multidisciplinary Design Optimization*, 1(1):1–8. 10.1051/ijsmdo:2007001

Deb, K. (2009). *Multi-objective optimization using evolutionary algorithms*. Wiley.

Deb, K., and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197. 10.1109/4235.996017

Deb, K., Anand, A., and Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, 10(4):371–395. 10.1162/106365602760972767

Deb, K., and Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601. 10.1109/TEVC.2013.2281535

Deb, K., Sindhya, K., and Okabe, T. (2007). Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1187–1194.

Doerr, B., and Doerr, C. (2015). A tight runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on OneMax. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1423–1430.

Doerr, B., Doerr, C., and Ebel, F. (2015). From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104. 10.1016/j.tcs.2014.11.028

Doerr, B., Gießen, C., Witt, C., and Yang, J. (2019). The (1+λ) evolutionary algorithm with self-adjusting mutation rate. *Algorithmica*, 81(2):593–631. 10.1007/s00453-018-0502-x

Doerr, B., Jansen, T., Sudholt, D., Winzen, C., and Zarges, C. (2013). Mutation rate matters even when optimizing monotonic functions. *Evolutionary Computation*, 21(1):1–27. 10.1162/EVCO_a_00055

Doerr, B., Johannsen, D., and Winzen, C. (2012). Multiplicative drift analysis. *Algorithmica*, 64(4):673–697. 10.1007/s00453-012-9622-x

Doerr, B., Le, H. P., Makhmara, R., and Nguyen, T. D. (2017). Fast genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 777–784.

Doerr, B., and Neumann, F. (Eds.). (2020). *Theory of evolutionary computation—Recent developments in discrete optimization*. Natural Computing Series. Springer.

Doerr, B., and Neumann, F. (2021). A survey on recent progress in the theory of evolutionary algorithms for discrete optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 1(4):16:1–16:43. 10.1145/3472304

Doerr, B., and Qu, Z. (2022a). A first runtime analysis of the NSGA-II on a multimodal problem. In *Proceedings of the 17th International Conference on Parallel Problem Solving from Nature*, pp. 399–412. Lecture Notes in Computer Science, Vol. 13399. 10.1007/978-3-031-14721-0_28

Doerr, B., and Qu, Z. (2022b). From understanding the population dynamics of the NSGA-II to the first proven lower bounds. *CoRR*, abs/2209.13974.

Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O. M., and Bäck, T. (2020). Benchmarking discrete optimization heuristics with IOHprofiler. *Applied Soft Computing*, 88:106027. 10.1016/j.asoc.2019.106027

Douguet, D. (2010). e-LEA3D: A computational-aided drug design web server. *Nucleic acids research*, 38(Web-Server-Issue):615–621. 10.1093/nar/gkq322

Drake, J. H., Kheiri, A., Özcan, E., and Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405–428. 10.1016/j.ejor.2019.07.073

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1–2):51–81. 10.1016/S0304-3975(01)00182-7

Eberhart, R., and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *International Symposium on Micro Machine and Human Science*, pp. 39–43.

Eiben, A. E., and Rudolph, G. (1999). Theory of evolutionary algorithms: A bird's eye view. *Theoretical Computer Science*, 229(1–2):3–9. 10.1016/S0304-3975(99)00089-4

Eiben, A. E., and Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31. 10.1016/j.swevo.2011.02.001

Eimer, T., Biedenkapp, A., Reimer, M., Adriaensen, S., Hutter, F., and Lindauer, M. (2021). DACBench: A benchmark library for dynamic algorithm configuration. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1668–1674.

ElHara, O. A., Auger, A., and Hansen, N. (2013). A median success rule for non-elitist evolution strategies: study of feasibility. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 415–422.

Eshelman, L. J., and Schaffer, J. D. (1992). Real-coded genetic algorithms and interval-schemata. In *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*, pp. 187–202.

Faury, L., Calauzènes, C., and Fercoq, O. (2019). Benchmarking GNN-CMA-ES on the BBOB noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1928–1936.

Flageat, M., and Cully, A. (2020). Fast and stable MAP-Elites in noisy domains using deep grids. In *Proceedings of 2020 Conference on Artificial Life*, pp. 273–282.

Fogel, D. (1992). An analysis of evolutionary programming. In *Proceedings of the 1st Annual Conference on Evolutionary Programming*, pp. 43–51.

Fogel, D. B. (1993). Using evolutionary programing to create neural networks that are capable of playing tic-tac-toe. In *Proceedings of International Conference on Neural Networks*, pp. 875–880.

Fogel, D. B. (1994). Evolutionary programming: An introduction and some current directions. *Statistics and Computing*, 4(2):113–129. 10.1007/BF00175356

Fogel, D. B. (2000). Evolving a checkers player without relying on human experience. *Intelligence*, 11(2):20–27. 10.1145/337897.337996

Fogel, D. B. (2002). *Blondie24: Playing at the edge of AI*. Morgan Kaufmann.

Fogel, D. B. (2023). Personal reflections on some early work in evolving strategies in the iterated prisoner's dilemma. *Evolutionary Computation*, 31(2):157–161.

Fogel, D. B., Fogel, L. J., and Porto, V. W. (1990). Evolutionary programming for training neural networks. In *International Joint Conference on Neural Networks*, pp. 601–605.

Fogel, D. B., Hays, T. J., Hahn, S. L., and Quon, J. (2006). The Blondie25 chess program competes against Fritz 8.0 and a human chess master. In *2006 IEEE Symposium on Computational Intelligence and Games*, pp. 230–235.

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial intelligence through simulated evolution*. Wiley.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1965). Intelligent decision-making through a simulation of evolution. *IEEE Transactions on Human Factors in Electronics*, HFE-6(1):13–23. 10.1109/THFE.1965.6591252

Gaier, A., Asteroth, A., and Mouret, J.-B. (2017). Aerodynamic design exploration through surrogate-assisted illumination. In *18th AIAA/ISSMO Conference on Multidisciplinary Analysis and Optimization*, p. 3330.

Garnier, J., Kallel, L., and Schoenauer, M. (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7(2):173–203. 10.1162/evco.1999.7.2.173

Giel, O., and Lehre, P. K. (2006). On the effect of populations in evolutionary multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 651–658.

Gießen, C., and Witt, C. (2017). The interplay of population size and mutation probability in the (1+λ) EA on OneMax. *Algorithmica*, 78(2):587–609.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley.

Goldberg, D. E., and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 154–159.

Gong, W., and Cai, Z. (2013). Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics*, 43(6):2066–2081. 10.1109/TCYB.2013.2239988

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128. 10.1109/TSMC.1986.289288

Hagg, A., Asteroth, A., and Bäck, T. (2018). Prototype discovery using quality-diversity. In *Parallel Problem Solving from Nature*, pp. 500–511.

Hagg, A., Asteroth, A., and Bäck, T. (2019). Modeling user selection in quality diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 116–124.

Hagg, A., Preuss, M., Asteroth, A., and Bäck, T. (2020). An analysis of phenotypic diversity in multi-solution optimization. In B. Filipič, E. Minisci, and M. Vasile (Eds.), *Bioinspired optimization methods and their applications*, pp. 43–55. Springer.

Hansen, N. (2008). CMA-ES with two-point step-size adaptation. *CoRR*, abs/0805.0231.

Hansen, N. (2016). The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772.

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 1689–1696.

Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., and Brockhoff, D. (2021). COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144. 10.1080/10556788.2020.1808977

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009). Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Retrieved from https://hal.inria.fr/inria-00362633/document/

Hansen, N., Niederberger, A. S. P., Guzzella, L., and Koumoutsakos, P. (2009). A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197. 10.1109/TEVC.2008.924423

Hansen, N., and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 312–317.

Hansen, N., and Ostermeier, A. (1997). Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I, \lambda)$-es. In *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, pp. 650–654.

Hansen, N., and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195. 10.1162/106365601750190398

Hansen, N., Ostermeier, A., and Gawelczyk, A. (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 57–64.

Harik, G. (1995). Finding multiple solutions using restricted tournament selection. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 24–31.

Harik, G. R., and Goldberg, D. E. (1996). Learning linkage. In *Proceedings of the 4th Workshop on Foundations of Genetic Algorithms*, pp. 247–262.

Harik, G. R., Lobo, F. G., and Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3(4):287–297. 10.1109/4235.797971

Hauschild, M., and Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128. 10.1016/j.swevo.2011.08.003

He, J., and Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85. 10.1016/S0004-3702(01)00058-3

He, J., and Yao, X. (2003). Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1–2):59–97. 10.1016/S0004-3702(02)00381-8

Hellwig, M., and Beyer, H.-G. (2019). Benchmarking evolutionary algorithms for single objective real-valued constrained optimization—A critical review. *Swarm and Evolutionary Computation*, 44:927–944. 10.1016/j.swevo.2018.10.002

Holland, J. H. (1992, 1st edition: 1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. MIT Press.

Huang, C., Li, Y., and Yao, X. (2019). A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation*, 24(2):201–216. 10.1109/TEVC.2019.2921598

Huang, H., Hao, Z., Cai, Z., and Zhu, Y. (2010). Runtime analysis of evolutionary programming based on Cauchy mutation. In *Proceedings of the First International Conference on Swarm, Evolutionary, and Memetic Computing*, pp. 222–229. Lecture Notes in Computer Science, Vol. 6466. 10.1007/978-3-642-17563-3_27

Huang, Q., de Winter, R., van Stein, B., Bäck, T., and Kononova, A. V. (2022). Multi-surrogate assisted efficient global optimization for discrete problems. In *IEEE Symposium Series on Computational Intelligence*, pp. 1650–1658.

Hussain, K., Mohd Salleh, M. N., Cheng, S., and Shi, Y. (2019). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4):2191–2233. 10.1007/s10462-017-9605-z

Igel, C., Suttorp, T., and Hansen, N. (2006). A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 453–460.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimization: A short review. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2419–2426.

Jamei, M., Mohammed, A. S., Ahmadianfar, I., Sabri, M.M.S., Karbasi, M., and Hasanipanah, M. (2022). Predicting rock brittleness using a robust evolutionary programming paradigm and regression-based feature selection model. *Applied Sciences*, 12(14). 10.3390/app12147101

Janikow, C. Z., Michalewicz, Z., et al., (1991). An experimental comparison of binary and floating point representations in genetic algorithms. In *International Conference on Genetic Algorithms*, pp. 31–36.

Jansen, T., Jong, K.A.D., and Wegener, I. (2005). On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13(4):413–440. 10.1162/106365605774666921

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12. 10.1007/s00500-003-0328-5

Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70. 10.1016/j.swevo.2011.05.001

Jin, Y., Olhofer, M., and Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494. 10.1109/TEVC.2002.800884

Jin, Y., Olhofer, M., Sendhoff, B., et al., (2000). On evolutionary optimization with approximate fitness functions. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 786–793.

Jin, Y., and Sendhoff, B. (2009). A systems approach to evolutionary multiobjective structural optimization and beyond. *IEEE Computational Intelligence Magazine*, 4(3):62–76. 10.1109/MCI.2009.933094

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492. 10.1023/A:1008306431147

Karafotias, G., Hoogendoorn, M., and Eiben, Á. E. (2014). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187. 10.1109/TEVC.2014.2308294

Kennedy, J. (2003). Bare bones particle swarms. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 80–87.

Kennedy, J., and Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 2, pp. 1671–1676.

Kerschke, P., Hoos, H. H., Neumann, F., and Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45. 10.1162/evco_a_00242

Kerschke, P., and Trautmann, H. (2019). Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco. In *Applications in statistical computing*, pp. 93–123. Springer.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. 10.1126/science.220.4598.671

Knowles, J. D. (2006). ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66. 10.1109/TEVC.2005.851274

Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, 91(9):992–1007. 10.1016/j.ress.2005.11.018

Kononova, A. V., Vermetten, D., Caraffini, F., Mitran, M.-A., and Zaharie, D. (2022). The importance of being constrained: Dealing with infeasible solutions in differential evolution and beyond. Retrieved from arXiv:2203.03512v1

Korejo, I., Yang, S., and Li, C. (2010). A directed mutation operator for real coded genetic algorithms. In *Proceedings of Applications of Evolutionary Computation: EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC*, pp. 491–500.

Kostovska, A., Jankovic, A., Vermetten, D., de Nobel, J., Wang, H., Eftimov, T., and Doerr, C. (2022). Per-run algorithm selection with warm-starting using trajectory-based features. In *Proceedings of the 17th International Conference on Parallel Problem Solving from Nature*, pp. 46–60. Lecture Notes in Computer Science, Vol. 13398. 10.1007/978-3-031-14714-2_4

Kostovska, A., Vermetten, D., Doerr, C., Džeroski, S., Panov, P., and Eftimov, T. (2021). Option: Optimization algorithm benchmarking ontology. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 239–240.

Kostovska, A., Vermetten, D., Džeroski, S., Doerr, C., Korosec, P., and Eftimov, T. (2022). The importance of landscape features for performance prediction of modular CMA-ES variants. In *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 648–656.

Kramer, O. (2010). A review of constraint-handling techniques for evolution strategies. *Applied Computational Intelligence and Soft Computing*, 2010(3):1–19. 10.1155/2010/185063

Krause, O., Arbonès, D. R., and Igel, C. (2016). CMA-ES with optimal covariance update and storage complexity. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), *Advances in neural information processing systems*, Vol. 29. Curran Associates.

Krejca, M. S., and Witt, C. (2020). Theory of estimation-of-distribution algorithms. In B. Doerr and F. Neumann (Eds.), *Theory of evolutionary computation—Recent developments in discrete optimization*. Natural Computing Series, pp. 405–442. Springer.

Kruisselbrink, J. W., Li, R., Reehuis, E., Eggermont, J., and Bäck, T. (2011). On the log-normal self-adaptation of the mutation rate in binary search spaces. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 893–900.

Larrañaga, P., Kuijpers, C.M.H., Murga, R. H., Inza, I., and Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170. 10.1023/A:1006529012972

Laumanns, M., Thiele, L., and Zitzler, E. (2004). Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182. 10.1109/TEVC.2004.823470

Lehman, J., and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on the Synthesis and Simulation of Living Systems*, pp. 329–336.

Lehman, J., and Stanley, K. O. (2011a). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223. 10.1162/EVCO_a_00025

Lehman, J., and Stanley, K. O. (2011b). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 211–218.

Lengler, J. (2020). Drift analysis. In B. Doerr and F. Neumann (Eds.), *Theory of evolutionary computation—Recent developments in discrete optimization*. Natural Computing Series, pp. 89–131. Springer.

Leung, F. H.-F., Lam, H.-K., Ling, S.-H., and Tam, P. K.-S. (2003). Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, 14(1):79–88. 10.1109/TNN.2002.804317

Li, R., Emmerich, M. T., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., and Reiber, J. (2013). Mixed integer evolution strategies for parameter optimization. *Evolutionary Computation*, 21(1):29–64. 10.1162/EVCO_a_00059

Li, X., Epitropakis, M. G., Deb, K., and Engelbrecht, A. (2016). Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538. 10.1109/TEVC.2016.2638437

Liepins, G. E., and Vose, M. D. (1990). Representational issues in genetic optimization. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(2):101–115.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23:54–1.

Ling, S.-H., and Leung, F. H. (2007). An improved genetic algorithm with average-bound crossover and wavelet mutation operations. *Soft Computing*, 11:7–31. 10.1007/s00500-006-0049-7

López-Ibáñez, M., Branke, J., and Paquete, L. (2021). Reproducibility in evolutionary computation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(4):1–21.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Loshchilov, I., Schoenauer, M., and Sèbag, M. (2013). Bi-population CMA-ES algorithms with surrogate models and line searches. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, pp. 1177–1184.

Mallipeddi, R., and Suganthan, P. N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14(4):561–579. 10.1109/TEVC.2009.2033582

Maree, S. C., Alderliesten, T., Thierens, D., and Bosman, P. A. (2018). Real-valued evolutionary multi-modal optimization driven by hill-valley clustering. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 857–864.

Mathias, K. E., and Whitley, L. D. (1992). Genetic operators, the fitness landscape and the traveling salesman problem. In *Parallel Problem Solving from Nature 2*, pp. 221–230.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, pp. 829–836.

Mezura-Montes, E., and Coello, C. A. C. (2005). A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1):1–17. 10.1109/TEVC.2004.836819

Mezura-Montes, E., and Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194. 10.1016/j.swevo.2011.10.001

Miettinen, K. (2014). Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, 36(1):3–37. 10.1007/s00291-012-0297-0

Mouret, J.-B., and Clune, J. (2015). Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909.

Muñoz, M. A., Sun, Y., Kirley, M., and Halgamuge, S. K. (2015). Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245.

Nagata, Y., and Kobayashi, S. (1997). Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem. In *Proceedings of International Conference on Genetic Algorithms*, pp. 450–457.

Nelder, J., and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7:308–313. 10.1093/comjnl/7.4.308

Neubauer, A. (1997). A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, pp. 93–96.

Oliver, I. M., Smith, D. J., and Holland, J.R.C. (1987). A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of International Conference on Genetic Algorithms*, pp. 224–230.

Ono, I., Kita, H., and Kobayashi, S. (1999). A robust real-coded genetic algorithm using unimodal normal distribution crossover augmented by uniform crossover: Effects of self-adaptation of crossover probabilities. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Companion (GECCO)*, Vol. 1, pp. 496–503.

Ostermeier, A., Gawelczyk, A., and Hansen, N. (1994a). A derandomized approach to self-adaptation of evolution strategies. *Evolutionary Computation*, 2(4):369–380. 10.1162/evco.1994.2.4.369

Ostermeier, A., Gawelczyk, A., and Hansen, N. (1994b). Step-size adaptation based on non-local use of selection information. In *Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature*, pp. 189–198.

Osuna, E. C., Gao, W., Neumann, F., and Sudholt, D. (2020). Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*, 832:123–142. 10.1016/j.tcs.2018.06.009

Pan, L., Xu, W., Li, L., He, C., and Cheng, R. (2021). Adaptive simulated binary crossover for rotated multi-objective optimization. *Swarm and Evolutionary Computation*, 60:100759.

Paredis, J. (1994). Co-evolutionary constraint satisfaction. In *International Conference on Parallel Problem Solving from Nature*, pp. 46–55.

Pétrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 798–803.

Pierrot, T., Richard, G., Beguir, K., and Cully, A. (2022). Multi-objective quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 139–147.

Pillay, N., and Qu, R. (2018). *Hyper-heuristics: Theory and applications*. Natural Computing Series. Springer.

Pinto, E. C., and Doerr, C. (2018). A simple proof for the usefulness of crossover in black-box optimization. In *Proceedings of Conference on Parallel Problem Solving from Nature*, pp. 29–41.

Poloczek, J., and Kramer, O. (2013). Local SVM constraint surrogate models for self-adaptive evolution strategies. In *Annual Conference on Artificial Intelligence*, pp. 164–175.

Ponweiser, W., Wagner, T., Biermann, D., and Vincze, M. (2008). Multiobjective optimization on a limited budget of evaluations using model-assisted -metric selection. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature*, pp. 784–794. Lecture Notes in Computer Science, Vol. 5199.

Preuss, M. (2015). Groundwork for niching. In *Multimodal optimization by means of evolutionary algorithms*. Natural Computing Series, pp. 55–73. Springer.

Preuss, M., Epitropakis, M., Li, X., and Fieldsend, J. E. (2021). Multimodal optimization: Formulation, heuristics, and a decade of advances. In *Metaheuristics for finding multiple solutions*, pp. 1–26. Springer.

Price, K., Storn, R., and Lampinen, J. (2005). *Differential evolution: A practical approach to global optimization*. Natural Computing Series. Springer.

Price, W. L. (1977). A controlled random search procedure for global optimisation. *Computer Journal*, 20:367–370. 10.1093/comjnl/20.4.367

Pugh, J. K., Soros, L. B., Szerlip, P. A., and Stanley, K. O. (2015). Confronting the challenge of quality diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 967–974.

Qijang, J. (2014). A unified differential evolution algorithm for global optimization. *IEEE Transactions on Evolutionary Computation*.

Qin, A. K., Huang, V. L., and Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417. 10.1109/TEVC.2008.927706

Qu, B. Y., and Suganthan, P. N. (2011). Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods. *Engineering Optimization*, 43(4):403–416. 10.1080/0305215X.2010.493937

Qu, B. Y., Suganthan, P. N., and Das, S. (2012). A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, 17(3):387–402. 10.1109/TEVC.2012.2203138

Raponi, E., Wang, H., Bujny, M., Boria, S., and Doerr, C. (2020). High dimensional Bayesian optimization assisted by principal component analysis. In *International Conference on Parallel Problem Solving from Nature*, pp. 169–183.

Rasheed, K., and Hirsh, H. (2000). Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 628–635.

Rasmussen, C., and Williams, C. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning series. University Press Group.

Ratle, A. (1998). Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *International Conference on Parallel Problem Solving from Nature*, pp. 87–96.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Library Translation 1122*.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer System nach Prinzipien der biologischen Evolution*. frommann-holzboog.

Reeves, C. R., and Yamada, T. (1998). Genetic algorithms, path relinking, and the flowshop sequencing problem. *Evolutionary Computation*, 6(1):45–60. 10.1162/evco.1998.6.1.45

Rice, J. R. (1976). The algorithm selection problem. In *Advances in computers*, Vol. 15, pp. 65–118. Elsevier. 10.1016/S0065-2458(08)60520-3

Richardson, J. T., Palmer, M. R., Liepins, G. E., and Hilliard, M. R. (1989). Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 191–197.

Ros, R., and Hansen, N. (2008). A simple modification in CMA-ES achieving linear time and space complexity. In *International Conference on Parallel Problem Solving from Nature*, pp. 296–305.

Rowe, J. E., and Sudholt, D. (2014). The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38. 10.1016/j.tcs.2013.09.036

Salcedo-Sanz, S. (2009). A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer Science Review*, 3(3):175–192. 10.1016/j.cosrev.2009.07.001

Santana-Quintero, L. V., Montano, A. A., and Coello, C.A.C. (2010). A review of techniques for handling expensive functions in evolutionary multi-objective optimization. *Computational Intelligence in Expensive Optimization Problems*, pp. 29–59.

Saravanan, N., and Fogel, D. B. (1997). Multi-operator evolutionary programming: A preliminary study on function optimization. In *Proceedings of the 6th International Conference on Evolutionary Programming VI, 6th International Conference*, pp. 215–222. Lecture Notes in Computer Science, Vol. 1213.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100.

Schwefel, H.-P. (1965). Kybernetische evolution als strategie der experimentellen forschung in der stromungstechnik. Diploma thesis, Technical University of Berlin.

Schwefel, H.-P. (1977). *Numerische optimierung von computer-modellen mittels der evolutionsstrategie.(Teil 1, Kap. 1-5)*. Birkhäuser.

Schwefel, H.-P. (1981). *Numerical optimization of computer models*. John Wiley & Sons.

Schwefel, H.-P. (1987). Collective phenomena in evolutionary systems. In *31st Annual Meeting of the International Society for General Systems Research*.

Shala, G., Biedenkapp, A., Awad, N., Adriaensen, S., Lindauer, M., and Hutter, F. (2020). Learning step-size adaptation in CMA-ES. In *International Conference on Parallel Problem Solving from Nature*, pp. 691–706.

Sharma, M., Lopez-Ibanez, M., and Kazakov, D. (2020). Unified framework for the adaptive operator selection of discrete parameters. arXiv:2005.05613

Shi, Y., and Eberhart, R. (1998). A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)*, pp. 69–73.

Shir, O. M., and Bäck, T. (2006). Niche radius adaptation in the CMA-ES niching algorithm. In *Parallel Problem Solving from Nature*, pp. 142–151.

Shir, O. M., Emmerich, M., and Bäck, T. (2007). Self-adaptive niching CMA-ES with Mahalanobis metric. In *2007 IEEE Congress on Evolutionary Computation*, pp. 820–827.

Sörensen, K. (2015). Metaheuristics—The metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18.

Spears, W. M. (1993). Crossover or mutation? In *Foundations of genetic algorithms*, Vol. 2, pp. 221–237. Elsevier.

Storn, R., and Price, K. (1995). Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 23.

Sudholt, D. (2017). How crossover speeds up building block assembly in genetic algorithms. *Evolutionary Computation*, 25(2):237–274. 10.1162/EVCO_a_00171

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., and Tiwari, S. (2005). *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization*. KanGAL Report No. 2005005.

Sun, C., Ding, J., Zeng, J., and Jin, Y. (2018). A fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. *Memetic Computing*, 10(2):123–134. 10.1007/s12293-016-0199-9

Sun, J., Liu, X., Bäck, T., and Xu, Z. (2021). Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Transactions on Evolutionary Computation*, 25(4):666–680. 10.1109/TEVC.2021.3060811

Surry, P. D., and Radcliffe, N. J. (1997). The COMOGA method: Constrained optimisation by multi-objective genetic algorithms. *Control and Cybernetics*, 26:391–412.

Tanabe, R., and Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation*, pp. 71–78.

Temby, L., Vamplew, P., and Berry, A. (2005). Accelerating real-valued genetic algorithms using mutation-with-momentum. In *Proceedings of the 18th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, pp. 1108–1111.

Thierens, D. (2010). The linkage tree genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pp. 264–273.

Thierens, D., and Bosman, P. A. (2011). Optimal mixing evolutionary algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 617–624.

Thomsen, R. (2004). Multimodal optimization using crowding-based differential evolution. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, Vol. 2, pp. 1382–1389. 10.1109/CEC.2004.1331058

Tsutsui, S., Yamamura, M., and Higuchi, T. (1999). Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 657–664.

Urquhart, N., and Hart, E. (2018). Optimisation and illumination of a real-world workforce scheduling and routing application (WSRP) via Map-Elites. In *Proceedings of the 15th International Conference on Parallel Problem Solving from Nature*, pp. 488–499. Lecture Notes in Computer Science, Vol. 11101. 10.1007/978-3-319-99253-2_39

van der Blom, K., Deist, T. M., Volz, V., Marchi, M., Nojima, Y., Naujoks, B., Oyama, A., and Tušar, T. (2020). Identifying properties of real-world optimisation problems through a questionnaire. arXiv:2011.05547

van Rijn, S., Wang, H., van Leeuwen, M., and Bäck, T. (2016). Evolving the structure of evolution strategies. In *2016 IEEE Symposium Series on Computational Intelligence*, pp. 1–8.

Van Stein, B., Wang, H., Kowalczyk, W., Emmerich, M., and Bäck, T. (2020). Cluster-based Kriging approximation algorithms for complexity reduction. *Applied Intelligence*, 50(3):778–791. 10.1007/s10489-019-01549-7

Varelas, K., Auger, A., Brockhoff, D., Hansen, N., ElHara, O. A., Semet, Y., Kassab, R., and Barbaresco, F. (2018). A comparative study of large-scale variants of CMA-ES. In *Parallel Problem Solving from Nature*, pp. 3–15.

Vassiliades, V., Chatzilygeroudis, K., and Mouret, J.-B. (2017). Using centroidal Voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation*, 22(4):623–630. 10.1109/TEVC.2017.2735550

Vermetten, D., van Rijn, S., Bäck, T., and Doerr, C. (2019). Online selection of CMA-ES variants. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 951–959.

Vermetten, D., Wang, H., Bäck, T., and Doerr, C. (2020). Towards dynamic algorithm selection for numerical black-box optimization: investigating BBOB as a use case. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 654–662.

Wagner, T., Beume, N., and Naujoks, B. (2007). Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization*, pp. 742–756. Lecture Notes in Computer Science, Vol. 4403.

Wang, D.-J., Liu, F., Wang, Y.-Z., and Jin, Y. (2015). A knowledge-based evolutionary proactive scheduling approach in the presence of machine breakdown and deterioration effect. *Knowledge-Based Systems*, 90:70–80.

Wang, H., Emmerich, M., and Bäck, T. (2014). Mirrored orthogonal sampling with pairwise selection in evolution strategies. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pp. 154–156.

Wang, H., Vermetten, D., Ye, F., Doerr, C., and Bäck, T. (2022). IOHanalyzer: Detailed performance analyses for iterative optimization heuristics. *ACM Transactions on Evolutionary Learning and Optimization*, 2(1):1–29. 10.1145/3510426

Wegener, I. (2001). Theoretical aspects of evolutionary algorithms. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, pp. 64–78. Lecture Notes in Computer Science, Vol. 2076.

Weyland, D. (2010). A rigorous analysis of the Harmony Search algorithm: How the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing*, 1(2):50–60. 10.4018/jamc.2010040104

White, D. (2014). An overview of schema theory. arXiv:1401.2651

Whitley, D. (2019). *Next generation genetic algorithms: A user's guide and tutorial*, pp. 245–274. Springer.

Whitley, L. D., Hains, D., and Howe, A. E. (2010). A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, pp. 566–575. Lecture Notes in Computer Science, Vol. 6238.

Winter, S., Brendel, B., Pechlivanis, I., Schmieder, K., and Igel, C. (2008). Registration of CT and intraoperative 3-D ultrasound images of the spine using evolutionary and gradient-based methods. *IEEE Transactions on Evolutionary Computation*, 12:284–296. 10.1109/TEVC.2007.907558

Witt, C. (2006). Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14(1):65–86.

Witt, C. (2013). Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing*, 22(2):294–318.

Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K., and Liu, B. (2018). Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access*, 6:41256–41279. 10.1109/ACCESS.2018.2856832

Yang, D., and Flockton, S. J. (1995). Evolutionary algorithms with a coarse-to-fine function smoothing. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, Vol. 2, pp. 657–662. 10.1109/ICEC.1995.487462

Yao, X., and Liu, Y. (1996). Fast evolutionary programming. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pp. 451–460.

Ye, F. (2022). Benchmarking discrete optimization heuristics: From building a sound experimental environment to algorithm configuration. PhD thesis, Leiden University.

Ye, F., Doerr, C., and Bäck, T. (2019). Interpolating local and global search by controlling the variance of standard bit mutation. In *IEEE Congress on Evolutionary Computation*, pp. 2292–2299.

Ye, F., Doerr, C., Wang, H., and Bäck, T. (2022). Automated configuration of genetic algorithms by tuning for anytime performance. *IEEE Transactions on Evolutionary Computation*, 26(6):1526–1538. 10.1109/TEVC.2022.3159087

Ye, F., Wang, H., Doerr, C., and Bäck, T. (2020). Benchmarking a $(\mu + \lambda)$ genetic algorithm with configurable crossover probability. In *Proceedings of the 16th International Conference on Parallel Problem Solving from Nature*, pp. 699–713. Lecture Notes in Computer Science, Vol. 12270. 10.1007/978-3-030-58115-2_49

Yu, M., Li, X., and Liang, J. (2020). A dynamic surrogate-assisted evolutionary algorithm framework for expensive structural optimization. *Structural and Multidisciplinary Optimization*, 61(2):711–729. 10.1007/s00158-019-02391-8

Yu, M., Liang, J., Wu, Z., and Yang, Z. (2022). A twofold infill criterion-driven heterogeneous ensemble surrogate-assisted evolutionary algorithm for computationally expensive problems. *Knowledge-Based Systems*, 236:107747. 10.1016/j.knosys.2021.107747

Zahara, E., and Kao, Y.-T. (2009). Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36(2):3880–3886. 10.1016/j.eswa.2008.02.039

Zaharie, D. (2002). Critical values for control parameters of differential evolution algorithm. In *Proceedings of 8th International Mendel Conference on Soft Computing*, pp. 62–67.

Zaharie, D., and Micota, F. (2017). Revisiting the analysis of population variance in differential evolution algorithms. In *2017 IEEE Congress on Evolutionary Computation*, pp. 1811–1818.

Zamuda, A., and Brest, J. (2012). Population reduction differential evolution with multiple mutation strategies in real world industry challenges. In *Swarm and evolutionary computation*, pp. 154–161. Springer.

Zhang, J., and Sanderson, A. C. (2007). JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In *2007 IEEE Congress on Evolutionary Computation*, pp. 2251–2258.

Zheng, W., Liu, Y., and Doerr, B. (2022). A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pp. 10408–10416.

Zhou, Z., Ong, Y. S., Nguyen, M. H., and Lim, D. (2005). A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 3, pp. 2832–2839. 10.1109/CEC.2005.1555050

Zitzler, E., Laumanns, M., and Thiele, L. (2001). *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. TIK-Report 103.