Why Is Optimization Difficult?

Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio J. Nebro

Abstract. This chapter aims to address some of the fundamental issues that are often encountered in optimization problems, making them difficult to solve. These issues include premature convergence, ruggedness, causality, deceptiveness, neutrality, epistasis, robustness, overfitting, oversimplification, multi-objectivity, dynamic fitness, the No Free Lunch Theorem, etc. We explain why these issues make optimization problems hard to solve and present some possible countermeasures for dealing with them. By doing this, we hope to help both practitioners and fellow researchers to create more efficient optimization applications and novel algorithms.

1 Introduction

Optimization, in general, is concerned with finding the best solutions for a given problem. Its applicability in many different disciplines makes it hard to give an exact definition. Mathematicians, for instance, are interested in finding the maxima or minima of a real function from within an allowable set of variables. In computing and engineering, the goal is to maximize the performance of a system or application with minimal runtime and resources.

Thomas Weise · Michael Zapf
Distributed Systems Group, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
e-mail: weise@vs.uni-kassel.de and zapf@vs.uni-kassel.de

Raymond Chiong
School of Computing & Design, Swinburne University of Technology (Sarawak Campus), 93350 Kuching, Sarawak, Malaysia
e-mail: rchiong@swinburne.edu.my

Antonio J. Nebro
Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga, Campus de Teatinos, 29071 Málaga, Spain
e-mail: antonio@lcc.uma.es

R. Chiong (Ed.): Nature-Inspired Algorithms for Optimisation, SCI 193, pp. 1–50.
springerlink.com

In the business industry, people aim to optimize the efficiency of a production process or the quality and desirability of their current products.

All these examples show that optimization is indeed part of our everyday life. We often try to maximize our gain by minimizing the cost we need to bear. However, are we really able to achieve an "optimal" condition? Frankly, whatever problems we are dealing with, it is rare that the optimization process will produce a solution that is truly optimal. It may be optimal for one audience or for a particular application, but definitely not in all cases.

As such, various techniques have emerged for tackling different kinds of optimization problems. In the broadest sense, these techniques can be classified into exact and stochastic algorithms. Exact algorithms, such as branch and bound, A* search, or dynamic programming can be highly effective for small-size problems. When the problems are large and complex, especially if they are either NP-complete or NP-hard, i.e., have no known polynomial-time solutions, the use of stochastic algorithms becomes mandatory. These stochastic algorithms do not guarantee an optimal solution, but they are able to find quasi-optimal solutions within a reasonable amount of time.

In recent years, metaheuristics, a family of stochastic techniques, has become an active research area. They can be defined as higher level frameworks aimed at efficiently and effectively exploring a search space [25]. The initial work in this area was started about half a century ago (see [175, 78, 24], and [37]). Subsequently, a lot of diverse methods have been proposed, and today, this family comprises many well-known techniques such as Evolutionary Algorithms, Tabu Search, Simulated Annealing, Ant Colony Optimization, Particle Swarm Optimization, etc.

There are different ways of classifying and describing metaheuristic algorithms. The widely accepted classification would be the view of nature-inspired vs. non nature-inspired, i.e., whether or not the algorithm somehow emulates a process found in nature. Evolutionary Algorithms, the most widely used metaheuristics, belong to the nature-inspired class. Other techniques with increasing popularity in this class include Ant Colony Optimization, Particle Swarm Optimization, Artificial Immune Systems, and so on. Scatter search, Tabu Search, and Iterated Local Search are examples of non nature-inspired metaheuristics. Unified models of metaheuristic optimization procedures have been proposed by Vaessens et al [220, 221], Rayward-Smith [169], Osman [158], and Taillard et al [210].

In this chapter, our main objective is to address some fundamental issues that make optimization problems difficult based on the nature-inspired class of metaheuristics. Apart from the reasons of being large, complex, and dynamic, we present a list of problem features that are often encountered and explain why some optimization problems are hard to solve. Some of the issues that will be discussed, such as multi-modality and overfitting, concern global optimization in general. We will also elaborate on other issues which are often linked to Evolutionary Algorithms, e.g., epistasis and neutrality, but can occur in virtually all metaheuristic optimization processes.

These concepts are important, as neglecting any one of them during the design of the search space and operations or the configuration of the optimization algorithms can render the entire invested effort worthless, even if highly efficient optimization methods are applied. To the best of our knowledge, to date there is not a single document in the literature comprising all such problematic features. By giving clear definitions and comprehensive introductions on them, we hope to create awareness among fellow scientists as well as practitioners in the industry so that they could perform optimization tasks more efficiently.

The rest of this chapter is organized as follows: In the next section, premature convergence to local minima is introduced as one of the major symptoms of failed optimization processes. Ruggedness (Section 3), deceptiveness (Section 4), too much neutrality (Section 5), and epistasis (Section 6), some of which have been illustrated in Fig. 1[1], are the main causes which may lead to this situation. Robustness, correctness, and generality instead are features which we expect from valid solutions. They are challenged by different types of noise discussed in Section 7 and the affinity of overfitting or overgeneralization (see Section 8). Some optimization tasks become further complicated because they involve multiple, conflicting objectives (Section 9) or dynamically changing ones (Section 10). In Section 11, we give a short introduction about the No Free Lunch Theorem, from which we can follow that no panacea, no magic bullet can exist against all of these problematic features. We will conclude our outline of the hardships of optimization with a summary in Section 12.

1.1 Basic Terminology

In the following text, we will utilize a terminology commonly used in the Evolutionary Algorithms community and sketched in Fig. 2 based on the example of a simple Genetic Algorithm. The possible solutions x of an optimization problem are elements of the problem space $\mathbb{X}$. Their utility as solutions is evaluated by a set $\mathbf{f}$ of objective functions f which, without loss of generality, are assumed to be subject to minimization. The set of search operations utilized by the optimizers to explore this space does not directly work on them. Instead, they are applied to the elements (the genotypes) of the search space $\mathbb{G}$ (the genome). They are mapped to the solution candidates by a genotype-phenotype mapping $\text{gpm} : \mathbb{G} \mapsto \mathbb{X}$. The term *individual* is used for both, solution candidates and genotypes.

[1] We include in Fig. 1 different examples of fitness landscapes, which relate solution candidates (or genotypes) to their objective values. The small bubbles in Fig. 1 represent solution candidates under investigation. An arrow from one bubble to another means that the second individual is found by applying one search operation to the first one. The objective values here are subject to minimization.

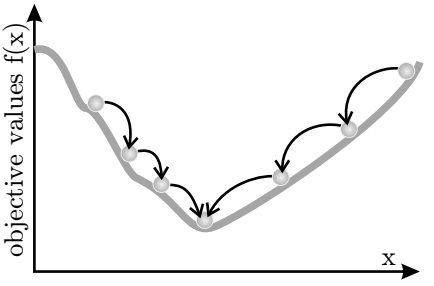


Fig. 1.a: Best Case

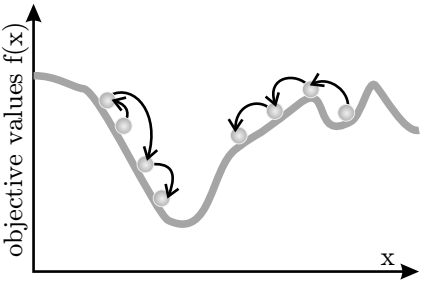


Fig. 1.b: Low Total Variation

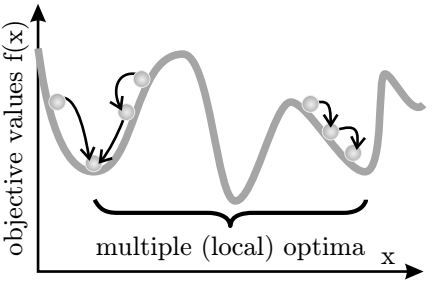


Fig. 1.c: Multimodal

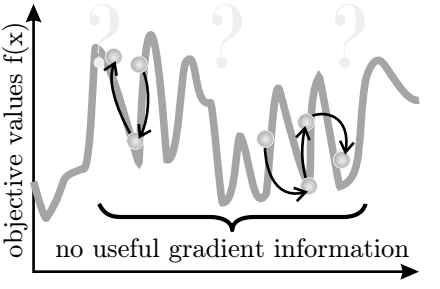


Fig. 1.d: Rugged

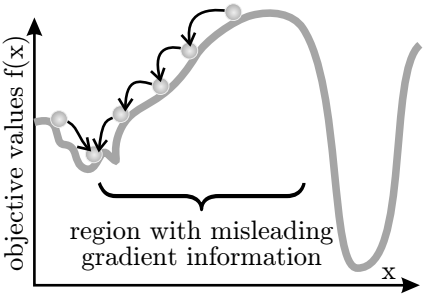


Fig. 1.e: Deceptive

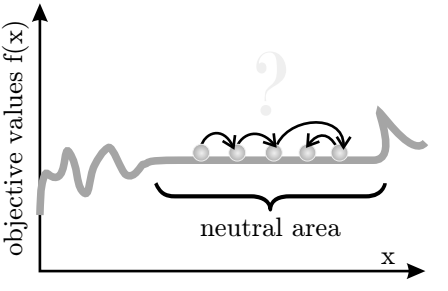


Fig. 1.f: Neutral

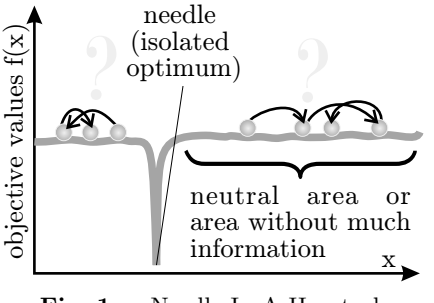


Fig. 1.g: Needle-In-A-Haystack

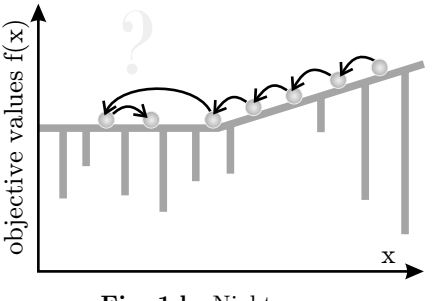


Fig. 1.h: Nightmare

Fig. 1 Different possible properties of fitness landscapes (minimization)

1.2 The Term "Difficult"

Before we go more into detail about what makes these landscapes *difficult*, we should establish the term in the context of optimization. The degree of difficulty of solving a certain problem with a dedicated algorithm is closely related to its *computational complexity*, i.e., the amount of resources such as time and memory required to do so. The computational complexity depends on the number of input elements needed for applying the algorithm. This dependency is often expressed in the form of approximate boundaries with the Big-O-family notations introduced by Bachmann [10] and made popular by Landau [122]. Problems can be further divided into *complexity classes*. One of the most difficult complexity classes owning to its resource requirements is NP, the set of all decision problems which are solvable in polynomial time by non-deterministic Turing machines [79]. Although many attempts have been

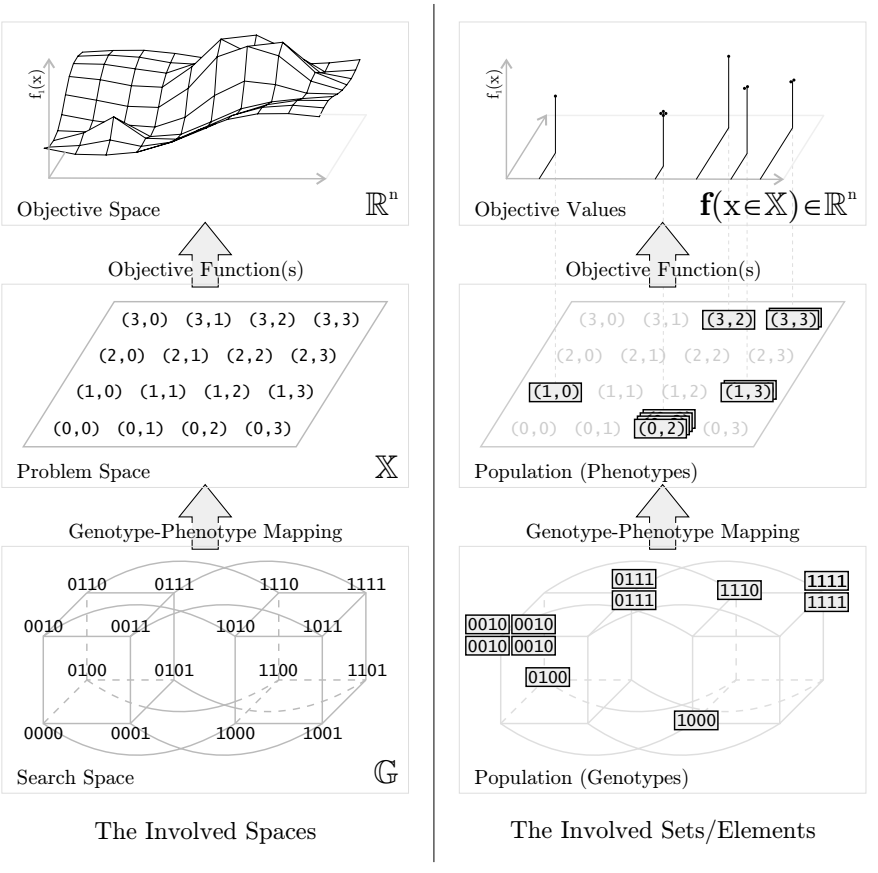


Fig. 2 The involved spaces and sets in optimization

made, no algorithm has been found which is able to solve an NP-complete [79] problem in polynomial time on a deterministic computer. One approach to obtaining near-optimal solutions for problems in NP in reasonable time is to apply metaheuristic, randomized optimization procedures.

As already stated, optimization algorithms are guided by objective functions. A function is *difficult* from a mathematical perspective in this context if it is not continuous, not differentiable, or if it has multiple maxima and minima. This understanding of difficulty comes very close to the intuitive sketches in Fig. 1.

In many real world applications of metaheuristic optimization, the characteristics of the objective functions are not known in advance. The problems are usually NP or have unknown complexity. It is therefore only rarely possible to derive boundaries for the performance or the runtime of optimizers in advance, let alone exact estimates with mathematical precision.

Most often, experience, rules of thumb, and empirical results based on the models obtained from related research areas such as biology are the only guides available. In this chapter we discuss many such models and rules, providing a better understanding of when the application of a metaheuristic is feasible and when not, as well as with indicators on how to avoid defining problems in a way that makes them *difficult*.

2 Premature Convergence

2.1 Introduction

An optimization algorithm has *converged* if it cannot reach new solution candidates anymore or if it keeps on producing solution candidates from a "small"[2] subset of the problem space. Global optimization algorithms will usually converge at some point in time. One of the problems in global optimization is that it is often not possible to determine whether the best solution currently known is situated on a local or a global optimum and thus, if convergence is acceptable. In other words, it is usually not clear whether the optimization process can be stopped, whether it should concentrate on refining the current optimum, or whether it should examine other parts of the search space instead. This can, of course, only become cumbersome if there are multiple (local) optima, i.e., the problem is multimodal as depicted in Fig. 1.c.

A mathematical function is multimodal if it has multiple maxima or minima [195, 246]. A set of objective functions (or a vector function) $\mathbf{f}$ is multimodal if it has multiple (local or global) optima – depending on the definition of "optimum" in the context of the corresponding optimization problem.

[2] According to a suitable metric like numbers of modifications or mutations which need to be applied to a given solution in order to leave this subset.

2.2 The Problem

An optimization process has *prematurely converged* to a local optimum if it is no longer able to explore other parts of the search space than the area currently being examined *and* there exists another region that contains a superior solution [192, 219]. Fig. 3 illustrates examples of premature convergence.

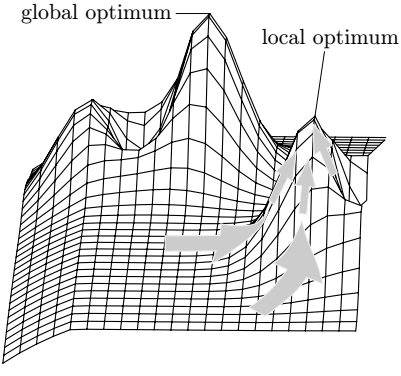


Fig. 3.a: Example 1: Maximization

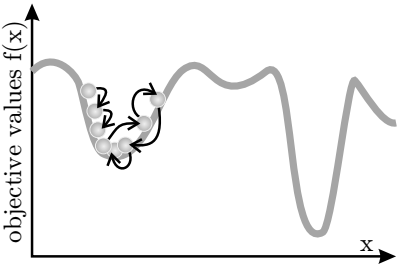


Fig. 3.b: Example 2: Minimization

Fig. 3 Premature convergence in the objective space

The existence of multiple global optima itself is not problematic and the discovery of only a subset of them can still be considered as successful in many cases (see Section 9). The occurrence of numerous local optima, however, is more complicated.

The phenomenon of *domino convergence* has been brought to attention by Rudnick [184] who studied it in the context of his BinInt problem [184, 213]. In principle, domino convergence occurs when the solution candidates have features which contribute significantly to different degrees of the total fitness. If these features are encoded in separate genes (or building blocks) in the genotypes, they are likely to be treated with different priorities, at least in randomized or heuristic optimization methods.

Building blocks with a very strong positive influence on the objective values, for instance, will quickly be adopted by the optimization process (i.e., "converge"). During this time, the alleles of genes with a smaller contribution are ignored. They do not come into play until the optimal alleles of the more "important" blocks have been accumulated. Rudnick [184] called this sequential convergence phenomenon *domino convergence* due to its resemblance to a row of falling domino stones [213].

In the worst case, the contributions of the less salient genes may almost look like noise and they are not optimized at all. Such a situation is also an instance of premature convergence, since the global optimum which would involve optimal configurations of all blocks will not be discovered. In this

situation, restarting the optimization process will not help because it will always turn out the same way. Example problems which are often likely to exhibit domino convergence are the Royal Road [139] and the aforementioned BinInt problem [184].

2.3 One Cause: Loss of Diversity

In biology, diversity is *the variety and abundance of organisms at a given place and time* [159, 133]. Much of the beauty and efficiency of natural ecosystems is based on a dazzling array of species interacting in manifold ways. Diversification is also a good investment strategy utilized by investors in the economy in order to increase their profit.

In population-based global optimization algorithms as well, maintaining a set of diverse solution candidates is very important. Losing diversity means approaching a state where all the solution candidates under investigation are similar to each other. Another term for this state is convergence. Discussions about how diversity can be measured have been provided by Routledge [183], Cousins [49], Magurran [133], Morrison and De Jong [148], and Paenke et al [159].

Preserving diversity is directly linked with maintaining a good balance between exploitation and exploration [159] and has been studied by researchers from many domains, such as

- Genetic Algorithms [156, 176, 177],
- Evolutionary Algorithms [28, 29, 123, 149, 200, 206],
- Genetic Programming [30, 38, 39, 40, 53, 93, 94],
- Tabu Search [81, 82], and
- Particle Swarm Optimization [238].

The operations which create new solutions from existing ones have a very large impact on the speed of convergence and the diversity of the populations [69, 203]. The step size in Evolution Strategy is a good example of this issue: setting it properly is very important and leads to the "exploration versus exploitation" problem [102] which can be observed in other areas of global optimization as well.[3]

In the context of optimization, *exploration* means finding new points in areas of the search space which have not been investigated before. Since computers have only limited memory, already evaluated solution candidates usually have to be discarded. Exploration is a metaphor for the procedure which allows search operations to find novel and maybe better solution structures. Such operators (like mutation in Evolutionary Algorithms) have a high chance of creating inferior solutions by destroying good building blocks but

[3] More or less synonymously to exploitation and exploration, the terms *intensifications* and *diversification* have been introduced by Glover [81, 82] in the context of Tabu Search.

also a small chance of finding totally new, superior traits (which, however, is not guaranteed at all).

Exploitation, on the other hand, is the process of improving and combining the traits of the currently known solution(s), as done by the crossover operator in Evolutionary Algorithms, for instance. Exploitation operations often incorporate small changes into already tested individuals leading to new, very similar solution candidates or try to merge building blocks of different, promising individuals. They usually have the disadvantage that other, possibly better, solutions located in distant areas of the problem space will not be discovered.

Almost all components of optimization strategies can either be used for increasing exploitation or in favor of exploration. Unary search operations that improve an existing solution in small steps can be built, hence being exploitation operators (as is done in Memetic Algorithms, for instance). They can also be implemented in a way that introduces much randomness into the individuals, effectively making them exploration operators. Selection operations in Evolutionary Computation choose a set of the most promising solution candidates which will be investigated in the next iteration of the optimizers. They can either return a small group of best individuals (exploitation) or a wide range of existing solution candidates (exploration).

Optimization algorithms that favor exploitation over exploration have higher convergence speed but run the risk of not finding the optimal solution and may get stuck at a local optimum. Then again, algorithms which perform excessive exploration may never improve their solution candidates well enough to find the global optimum or it may take them very long to discover it "by accident". A good example for this dilemma is the Simulated Annealing algorithm [117]. It is often modified to a form called *simulated quenching* which focuses on exploitation but loses the guaranteed convergence to the optimum [110]. Generally, optimization algorithms should employ at least one search operation of explorative character and at least one which is able to exploit good solutions further. There exists a vast body of research on the trade-off between exploration and exploitation that optimization algorithms have to face [7, 57, 66, 70, 103, 152].

2.4 Countermeasures

As we have seen, global optimization algorithms are optimization methods for finding the best possible solution(s) of an optimization problem instead of prematurely converging to a local optimum. Still, there is no general approach to ensure their success. The probability that an optimization process prematurely converges depends on the characteristics of the problem to be solved and the parameter settings and features of the optimization algorithms applied [215].

A very crude and yet, sometimes effective measure is restarting the optimization process at randomly chosen points in time. One example for this

method is *GRASP*s, *Greedy Randomized Adaptive Search Procedures* [71, 72], which continuously restart the process of creating an initial solution and refining it with local search. Still, such approaches are likely to fail in domino convergence situations.

In order to extend the duration of the evolution in Evolutionary Algorithms, many methods have been devised for steering the search away from areas which have already been frequently sampled. This can be achieved by integrating density metrics into the fitness assignment process. The most popular of such approaches are sharing and niching based on the Euclidean distance of the solution candidates in objective space [55, 85, 104, 138]. Using low selection pressure furthermore decreases the chance of premature convergence but also decreases the speed with which good solutions are exploited.

Another approach against premature convergence is to introduce the capability of self-adaptation, allowing the optimization algorithm to change its strategies or to modify its parameters depending on its current state. Such behaviors, however, are often implemented not in order to prevent premature convergence but to speed up the optimization process (which may lead to premature convergence to local optima) [185, 186, 187].

3 Ruggedness and Weak Causality

3.1 The Problem: Ruggedness

Optimization algorithms generally depend on some form of gradient in the objective or fitness space. The objective functions should be continuous and exhibit low total variation[4], so the optimizer can descend the gradient easily. If the objective functions are unsteady or fluctuating, i.e., going up and down, it becomes more complicated for the optimization process to find the right directions to proceed to. The more rugged a function gets, the harder it becomes to optimize it. From a simplified point of view, ruggedness is multimodality plus steep ascends and descends in the fitness landscape. Examples of rugged landscapes are Kauffman's NK fitness landscape [113, 115], the p-Spin model [6], Bergman and Feldman's jagged fitness landscape [19], and the sketch in Fig. 1.d.

3.2 One Cause: Weak Causality

During an optimization process, new points in the search space are created by the search operations. Generally we can assume that the genotypes which are the input of the search operations correspond to phenotypes which have previously been selected. Usually, the better or the more promising an individual is, the higher are its chances of being selected for further investigation. Reversing this statement suggests that individuals which are passed to the

[4] `http://en.wikipedia.org/wiki/Total_variation` [accessed 2008-04-23]

search operations are likely to have a good fitness. Since the fitness of a solution candidate depends on its properties, it can be assumed that the features of these individuals are not so bad either. It should thus be possible for the optimizer to introduce slight changes to their properties in order to find out whether they can be improved any further[5]. Normally, such modifications should also lead to small changes in the objective values and, hence, in the fitness of the solution candidate.

Definition 1 (Strong Causality). *Strong causality* (locality) means that small changes in the properties of an object also lead to small changes in its behavior [170, 171, 180].

This principle (proposed by Rechenberg [170, 171]) should not only hold for the search spaces and operations designed for optimization, but applies to natural genomes as well. The offspring resulting from sexual reproduction of two fish, for instance, has a different genotype than its parents. Yet, it is far more probable that these variations manifest in a unique color pattern of the scales, for example, instead of leading to a totally different creature.

Apart from this straightforward, informal explanation here, causality has been investigated thoroughly in different fields of optimization, such as Evolution Strategy [170, 65], structure evolution [129, 130], Genetic Programming [65, 107, 179, 180], genotype-phenotype mappings [193], search operators [65], and Evolutionary Algorithms in general [65, 182, 207].

In fitness landscapes with weak (low) causality, small changes in the solution candidates often lead to large changes in the objective values, i.e., ruggedness. It then becomes harder to decide which region of the problem space to explore and the optimizer cannot find reliable gradient information to follow. A small modification of a very bad solution candidate may then lead to a new local optimum and the best solution candidate currently known may be surrounded by points that are inferior to all other tested individuals.

The lower the causality of an optimization problem, the more rugged its fitness landscape is, which leads to a degradation of the performance of the optimizer [120]. This does not necessarily mean that it is impossible to find good solutions, but it may take very long to do so.

3.3 Countermeasures

To our knowledge, no viable method which can directly mitigate the effects of rugged fitness landscapes exists. In population-based approaches, using large population sizes and applying methods to increase the diversity can decrease the influence of ruggedness, but only up to a certain degree. Utilizing the Baldwin effect [13, 100, 101, 233] or Lamarckian evolution [54, 233], i.e., incorporating a local search into the optimization process, may further help to smoothen out the fitness landscape [89].

[5] We have already mentioned this under the subject of exploitation.

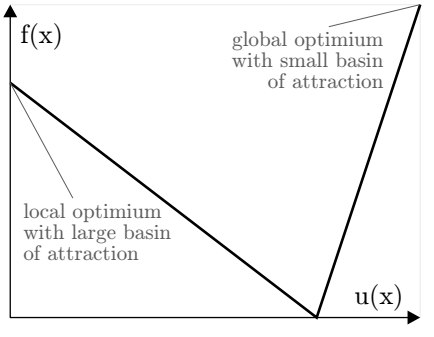
Weak causality is often a home-made problem: it results from the choice of the solution representation and search operations. Thus, in order to apply Evolutionary Algorithms in an efficient manner, it is necessary to find representations which allow for iterative modifications with bounded influence on the objective values.

4 Deceptiveness

4.1 Introduction

Especially annoying fitness landscapes show *deceptiveness* (or deceptivity). The gradient of deceptive objective functions leads the optimizer away from the optima, as illustrated in Fig. 1.e.

The term deceptiveness is mainly used in the Genetic Algorithm community in the context of the Schema Theorem. Schemas describe certain areas (hyperplanes) in the search space. If an optimization algorithm has discovered an area with a better average fitness compared to other regions, it will focus on exploring this region based on the assumption that highly fit areas are likely to contain the true optimum. Objective functions where this is not the case are called deceptive [20, 84, 127]. Examples for deceptiveness are the ND fitness landscapes [17], trap functions [1, 59, 112] like the one illustrated in Fig. 4, and the fully deceptive problems given by Goldberg et al [86, 60].



$$f(x) = \begin{cases} (8n/z)\,(z - u(x)) & \text{if } u(x) \leq z \\ (10n/\,(n-z))\,(u(x) - z) & \text{otherwise} \end{cases}$$

where $u(x)$ is the number of ones in the bit string x of length n and $z = \lfloor 3n/4 \rfloor$. $f(x)$ is subject to maximization.

Fig. 4 Ackley's "Trap" function [1, 112]

4.2 The Problem

If the information accumulated by an optimizer actually guides it away from the optimum, search algorithms will perform worse than a random walk or an exhaustive enumeration method. This issue has been known for a long time [228, 140, 141, 212] and has been subsumed under the No Free Lunch Theorem which we will discuss in Section 11.

4.3 Countermeasures

Solving deceptive optimization tasks perfectly involves sampling many individuals with very bad features and low fitness. This contradicts the basic ideas of metaheuristics and thus, there are no efficient countermeasures against deceptivity. Using large population sizes, maintaining a very high diversity, and utilizing linkage learning (see Section 6.3) are, maybe, the only approaches which can provide at least a small chance of finding good solutions.

5 Neutrality and Redundancy

5.1 The Problem: Neutrality

We consider the outcome of the application of a search operation to an element of the search space as neutral if it yields no change in the objective values [15, 172]. It is challenging for optimization algorithms if the best solution candidate currently known is situated on a plane of the fitness landscape, i.e., all adjacent solution candidates have the same objective values. As illustrated in Fig. 1.f, an optimizer then cannot find any gradient information and thus, no direction in which to proceed in a systematic manner. From its point of view, each search operation will yield identical individuals. Furthermore, optimization algorithms usually maintain a list of the best individuals found, which will then overflow eventually or require pruning.

The degree of neutrality ν is defined as the fraction of neutral results among all possible products of the search operations Op applied to a specific genotype [15]. We can generalize this measure to areas G in the search space $\mathbb{G}$ by averaging over all their elements. Regions where ν is close to one are considered as *neutral*.

$$\forall g_1 \in \mathbb{G} \Rightarrow \nu(g_1) = \frac{|\{g_2 | P(g_2 = Op(g_1)) > 0 \wedge \mathbf{f}(\text{gpm}(g_2)) = \mathbf{f}(\text{gpm}(g_1))\}|}{|\{g_2 | P(g_2 = Op(g_1)) > 0\}|} \quad (1)$$

$$\forall G \subseteq \mathbb{G} \Rightarrow \nu(G) = \frac{1}{|G|} \sum_{g \in G} \nu(g) \quad (2)$$

5.2 Evolvability

Another metaphor in global optimization borrowed from biological systems is evolvability [52]. Wagner [225, 226] points out that this word has two uses in biology: According to Kirschner and Gerhart [118], a biological system is evolvable if it is able to generate heritable, selectable phenotypic variations. Such properties can then be evolved and changed by natural selection. In its

second sense, a system is evolvable if it can acquire new characteristics via genetic change that help the organism(s) to survive and to reproduce. Theories about how the ability of generating adaptive variants has evolved have been proposed by Riedl [174], Altenberg [3], Wagner and Altenberg [227], and Bonner [26], amongst others. The idea of evolvability can be adopted for global optimization as follows:

Definition 2 (Evolvability). The evolvability of an optimization process in its current state defines how likely the search operations will lead to solution candidates with new (and eventually, better) objectives values.

The direct *probability of success* [170, 22], i.e., the chance that search operators produce offspring fitter than their parents, is also sometimes referred to as *evolvability* in the context of Evolutionary Algorithms [2, 5].

5.3 Neutrality: Problematic and Beneficial

The link between evolvability and neutrality has been discussed by many researchers. The evolvability of neutral parts of a fitness landscape depends on the optimization algorithm used. It is especially low for Hill Climbing and similar approaches, since the search operations cannot directly provide improvements or even changes. The optimization process then degenerates to a random walk, as illustrated in Fig. 1.f. The work of Beaudoin et al [17] on the ND fitness landscapes shows that neutrality may "destroy" useful information such as correlation.

Researchers in molecular evolution, on the other hand, found indications that the majority of mutations have no selective influence [77, 106] and that the transformation from genotypes to phenotypes is a many-to-one mapping. Wagner [226] states that neutrality in natural genomes is beneficial if it concerns only a subset of the properties peculiar to the offspring of a solution candidate while allowing meaningful modifications of the others. Toussaint and Igel [214] even go as far as declaring it a necessity for self-adaptation.

The theory of *punctuated equilibria* in biology introduced by Eldredge and Gould [67, 68] states that species experience long periods of evolutionary inactivity which are interrupted by sudden, localized, and rapid phenotypic evolutions [47, 134, 12]. It is assumed that the populations explore neutral layers during the time of stasis until, suddenly, a relevant change in a genotype leads to a better adapted phenotype [224] which then reproduces quickly.

The key to differentiating between "good" and "bad" neutrality is its degree ν in relation to the number of possible solutions maintained by the optimization algorithms. Smith et al [204] have used illustrative examples similar to Fig. 5 showing that a certain amount of neutral reproductions can foster the progress of optimization. In Fig. 5.a, basically the same scenario of premature convergence as in Fig. 3.a is depicted. The optimizer is drawn to a local optimum from which it cannot escape anymore. Fig. 5.b shows

that a little shot of neutrality could form a bridge to the global optimum. The optimizer now has a chance to escape the smaller peak if it is able to find and follow that bridge, i.e., the evolvability of the system has increased. If this bridge gets wider, as sketched in Fig. 5.c, the chance of finding the global optimum increases as well. Of course, if the bridge gets too wide, the optimization process may end up in a scenario like in Fig. 1.f where it cannot find any direction. Furthermore, in this scenario we expect the neutral bridge to lead to somewhere useful, which is not necessarily the case in reality.

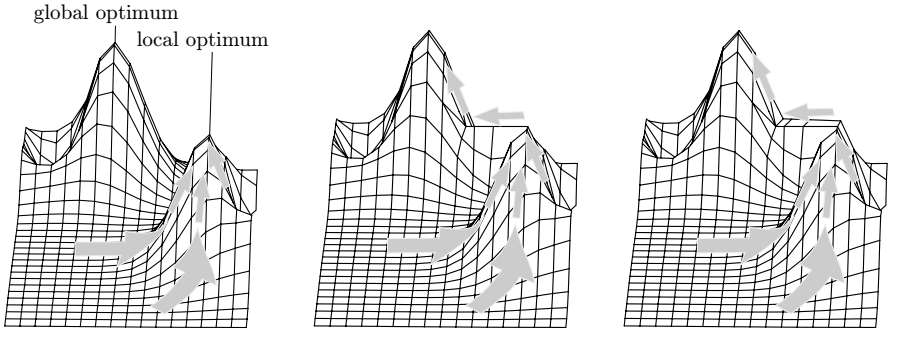


Fig. 5.a: Premature Convergence

Fig. 5.b: Small Neutral Bridge

Fig. 5.c: Wide Neutral Bridge

Fig. 5 Possible positive influence of neutrality

Examples for neutrality in fitness landscapes are the ND family [17], the NKp [15] and NKq [155] models, and the Royal Road [139]. Another common instance of neutrality is *bloat* in Genetic Programming [131].

5.4 Redundancy: Problematic and Beneficial

Redundancy in the context of global optimization is a feature of the genotype-phenotype mapping and means that multiple genotypes map to the same phenotype, i.e., the genotype-phenotype mapping is not injective. The role of redundancy in the genome is as controversial as that of neutrality [230]. There exist many accounts of its positive influence on the optimization process. Shackleton et al [194, 197], for instance, tried to mimic desirable evolutionary properties of RNA folding [106]. They developed redundant genotype-phenotype mappings using voting (both, via uniform redundancy and via a non-trivial approach), Turing machine-like binary instructions, Cellular automata, and random Boolean networks [114]. Except for the trivial voting mechanism based on uniform redundancy, the mappings induced neutral networks which proved beneficial for exploring the problem space. Especially the last approach provided particularly good results [194, 197]. Possibly converse

effects like epistasis (see Section 6) arising from the new genotype-phenotype mappings have not been considered in this study.

Redundancy can have a strong impact on the explorability of the problem space. When utilizing a one-to-one mapping, the translation of a slightly modified genotype will always result in a different phenotype. If there exists a many-to-one mapping between genotypes and phenotypes, the search operations can create offspring genotypes different from the parent which still translate to the same phenotype. The optimizer may now walk along a path through this neutral network. If many genotypes along this path can be modified to different offspring, many new solution candidates can be reached [197]. The experiments of Shipman et al [198, 196] additionally indicate that neutrality in the genotype-phenotype mapping can have positive effects.

Yet, Rothlauf [182] and Shackleton et al [194] show that simple uniform redundancy is not necessarily beneficial for the optimization process and may even slow it down. There is no use in introducing encodings which, for instance, represent each phenotypic bit with two bits in the genotype where `00` and `01` map to `0` and `10` and `11` map to `1`.

5.5 Summary

Different from ruggedness which is always bad for optimization algorithms, neutrality has aspects that may further as well as hinder the process of finding good solutions. Generally we can state that degrees of neutrality ν very close to 1 degenerate optimization processes to random walks. Some forms of neutral networks [14, 15, 27, 105, 208, 222, 223, 237] accompanied by low (nonzero) values of ν can improve the evolvability and hence, increase the chance of finding good solutions.

Adverse forms of neutrality are often caused by bad design of the search space or genotype-phenotype mapping. Uniform redundancy in the genome should be avoided where possible and the amount of neutrality in the search space should generally be limited.

6 Epistasis

6.1 Introduction

In biology, *epistasis* is defined as a form of interaction between different genes [163]. The term was coined by Bateson [16] and originally meant that one gene suppresses the phenotypical expression of another gene. In the context of statistical genetics, epistasis was initially called "epistacy" by Fisher [74]. According to Lush [132], the interaction between genes is epistatic if the effect on the fitness of altering one gene depends on the allelic state of other genes. This understanding of epistasis comes very close to another biological

expression: *Pleiotropy*, which means that a single gene influences multiple phenotypic traits [239]. In global optimization, such fine-grained distinctions are usually not made and the two terms are often used more or less synonymously.

Definition 3 (Epistasis). In optimization, Epistasis is the dependency of the contribution of one gene to the value of the objective functions on the allelic state of other genes [4, 51, 153].

We speak of minimal epistasis when every gene is independent of every other gene. Then, the optimization process equals finding the best value for each gene and can most efficiently be carried out by a simple greedy search [51]. A problem is maximally epistatic when no proper subset of genes is independent of any other gene [205, 153]. Examples of problems with a high degree of epistasis are Kauffman's NK fitness landscape [113, 115], the p-Spin model [6], and the tunable model of Weise et al [232].

6.2 The Problem

As sketched in Fig. 6, epistasis has a strong influence on many of the previously discussed problematic features. If one gene can "turn off" or affect the expression of many other genes, a modification of this gene will lead to a large change in the features of the phenotype. Hence, the *causality* will be weakened and *ruggedness* ensues in the fitness landscape. On the other hand, subsequent changes to the "deactivated" genes may have no influence on the phenotype at all, which would then increase the degree of *neutrality* in the search space. Epistasis is mainly an aspect of the way in which we define the genome $\mathbb{G}$ and the genotype-phenotype mapping gpm. It should be avoided where possible.

Generally, epistasis and conflicting objectives in multi-objective optimization should be distinguished from each other. Epistasis as well as pleiotropy

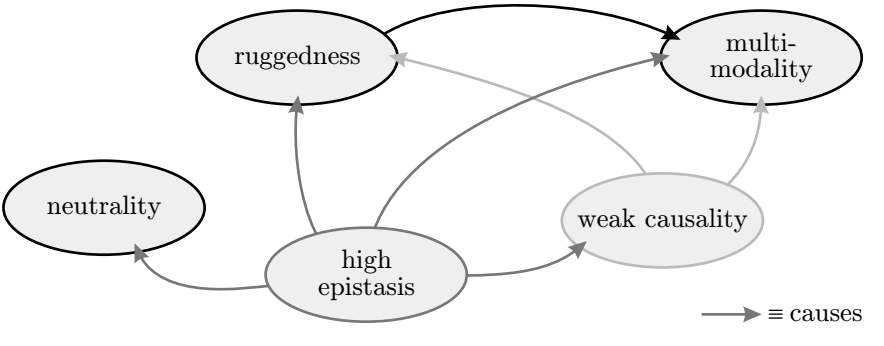


Fig. 6 The influence of epistasis on the fitness landscape

is a property of the influence of the elements (the genes) of the genotypes on the phenotypes. Objective functions can conflict *without* the involvement of any of these phenomena. We can, for example, define two objective functions $f_1(x) = x$ and $f_2(x) = -x$ which are clearly contradicting regardless of whether they are subject to maximization or minimization. Nevertheless, if the solution candidates x as well as the genotypes are simple real numbers and the genotype-phenotype mapping is simply an identity mapping, neither epistatic nor pleiotropic effects can occur.

Naudts and Verschoren [154] have shown for the special case of length-two binary string genomes that deceptiveness does not occur in situations with low epistasis and also that objective functions with high epistasis are not necessarily deceptive. Another discussion about different shapes of fitness landscapes under the influence of epistasis is given by Beerenwinkel et al [18].

6.3 Countermeasures

6.3.1 General

We have shown that epistasis is a root cause for multiple problematic features of optimization tasks. General countermeasures against epistasis can be divided into two groups. The symptoms of epistasis can be mitigated with the same methods which increase the chance of finding good solutions in the presence of ruggedness or neutrality – using larger populations and favoring explorative search operations. Epistasis itself is a feature which results from the choice of the search space structure, the search operations, and the genotype-phenotype mapping. Avoiding epistatic effects should be a major concern during their design. This can lead to a great improvement in the quality of the solutions produced by the optimization process [231]. General advice for good search space design is given in [84, 166, 178] and [229].

6.3.2 Linkage Learning

According to Winter et al [240], *linkage* is "the tendency for alleles of different genes to be passed together from one generation to the next" in genetics. This usually indicates that these genes are closely located in the same chromosome. In the context of Evolutionary Algorithms, this notation is not useful since identifying spatially close elements inside the genotypes is trivial. Instead, we are interested in alleles of different genes which have a joint effect on the fitness [150, 151].

Identifying these linked genes, i.e., learning their epistatic interaction, is very helpful for the optimization process. Such knowledge can be used to protect building blocks from being destroyed by the search operations. Finding approaches for *linkage learning* has become an especially popular discipline in the area of Evolutionary Algorithms with binary [99, 150, 46] and real [63] genomes. Two important methods from this area are the *messy Genetic*

Algorithm (mGA) by Goldberg et al [86] and the *Bayesian Optimization Algorithm* (BOA) [162, 41]. Module acquisition [8] may be considered as a similar effort in the area of Genetic Programming.

Let us take the mGA as an illustrative example for this family of approaches. By explicitly allowing the search operations to rearrange the genes in the genotypes, epistatically linked genes may get located closer to each other by time. As sketched in Fig. 7, the tighter the building blocks are packed, the less likely are they to be destroyed by crossover operations which usually split parent genotypes at randomly chosen points. Hence, the optimization process can strengthen the causality in the search space.

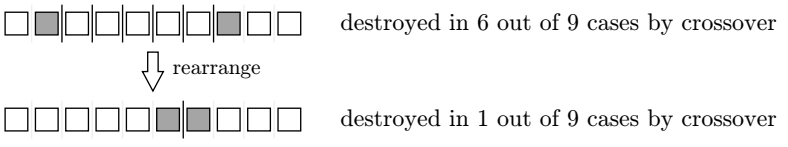


Fig. 7 Two linked genes and their destruction probability under single-point crossover

7 Noise and Robustness

7.1 Introduction – Noise

In the context of optimization, three types of noise can be distinguished. The first form is noise in the training data used as basis for learning *(i)*. In many applications of machine learning or optimization where a model m for a given system is to be learned, data samples including the input of the system and its measured response are used for training. Some typical examples of situations where training data is the basis for the objective function evaluation are

- the usage of global optimization for building classifiers (for example for predicting buying behavior using data gathered in a customer survey for training),
- the usage of simulations for determining the objective values in Genetic Programming (here, the simulated scenarios correspond to training cases), and
- the fitting of mathematical functions to (x, y)-data samples (with artificial neural networks or symbolic regression, for instance).

Since no measurement device is 100% accurate and there are always random errors, noise is present in such optimization problems.

Besides inexactnesses and fluctuations in the input data of the optimization process, perturbations are also likely to occur during the application of its results. This category subsumes the other two types of noise: perturbations that may arise from inaccuracies in *(ii)* the process of realizing the solutions

and *(iii)* environmentally induced perturbations during the applications of the products.

This issue can be illustrated using the process of developing the perfect tire for a car as an example. As input for the optimizer, all sorts of material coefficients and geometric constants measured from all known types of wheels and rubber could be available. Since these constants have been measured or calculated from measurements, they include a certain degree of noise and imprecision *(i)*.

The result of the optimization process will be the best tire construction plan discovered during its course and it will likely incorporate different materials and structures. We would hope that the tires created according to the plan will not fall apart if, accidently, an extra 0.0001% of a specific rubber component is used *(ii)*. During the optimization process, the behavior of many construction plans will be simulated in order to find out about their utility. When actually manufactured, the tires should not behave unexpectedly when used in scenarios different from those simulated *(iii)* and should instead be applicable in all driving scenarios likely to occur.

The effects of noise in optimization have been studied by various researchers; Miller and Goldberg [136, 137], Lee and Wong [125], and Gurin and Rastrigin [92] are some of them. Many global optimization algorithms and theoretical results have been proposed which can deal with noise. Some of them are, for instance, specialized

- Genetic Algorithms [75, 119, 188, 189, 217, 218],
- Evolution Strategies [11, 21, 96], and
- Particle Swarm Optimization [97, 161] approaches.

7.2 The Problem: Need for Robustness

The goal of global optimization is to find the global optima of the objective functions. While this is fully true from a theoretical point of view, it may not suffice in practice. Optimization problems are normally used to find good parameters or designs for components or plans to be put into action by human beings or machines. As we have already pointed out, there will always be noise and perturbations in practical realizations of the results of optimization.

Definition 4 (Robustness). A system in engineering or biology is *robust* if it is able to function properly in the face of genetic or environmental perturbations [225].

Therefore, a local optimum (or even a non-optimal element) for which slight deviations only lead to gentle performance degenerations is usually favored over a global optimum located in a highly rugged area of the fitness landscape [31]. In other words, local optima in regions of the fitness landscape with

strong causality are sometimes better than global optima with weak causality. Of course, the level of this acceptability is application-dependent. Fig. 8 illustrates the issue of local optima which are robust vs. global optima which are not. More examples from the real world are:

- When optimizing the control parameters of an airplane or a nuclear power plant, the global optimum is certainly not used if a slight perturbation can have hazardous effects on the system [218].
- Wiesmann et al [234, 235] bring up the topic of manufacturing tolerances in multilayer optical coatings. It is no use to find optimal configurations if they only perform optimal when manufactured to a precision which is either impossible or too hard to achieve on a constant basis.
- The optimization of the decision process on which roads should be precautionary salted for areas with marginal winter climate is an example of the need for dynamic robustness. The global optimum of this problem is likely to depend on the daily (or even current) weather forecast and may therefore be constantly changing. Handa et al [98] point out that it is practically infeasible to let road workers follow a constantly changing plan and circumvent this problem by incorporating multiple road temperature settings in the objective function evaluation.
- Tsutsui et al [218, 217] found a nice analogy in nature: The phenotypic characteristics of an individual are described by its genetic code. During the interpretation of this code, perturbations like abnormal temperature, nutritional imbalances, injuries, illnesses and so on may occur. If the phenotypic features emerging under these influences have low fitness, the organism cannot survive and procreate. Thus, even a species with good genetic material will die out if its phenotypic features become too sensitive to perturbations. Species robust against them, on the other hand, will survive and evolve.

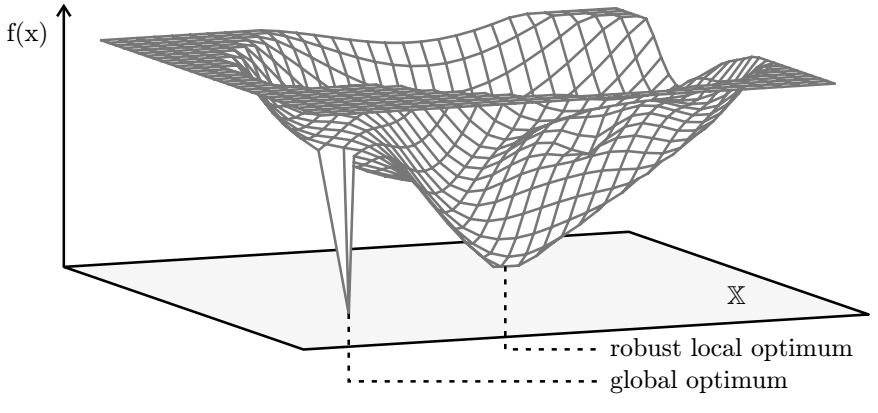


Fig. 8 A robust local optimum vs. a "unstable" global optimum

7.3 Countermeasures

For the special case where the problem space corresponds to the real vectors ($\mathbb{X} \subseteq \mathbb{R}^n$), several approaches for dealing with the problem of robustness have been developed. Inspired by Taguchi methods[6] [209], possible disturbances are represented by a vector $\boldsymbol{\delta} = (\delta_1, \delta_2, .., \delta_n)^T, \delta_i \in \mathbb{R}$ in the method of Greiner [87, 88]. If the distribution and influence of the δ_i are known, the objective function $f(\mathbf{x}) : \mathbf{x} \in \mathbb{X}$ can be rewritten as $\tilde{f}(\mathbf{x}, \boldsymbol{\delta})$ [235]. In the special case where $\boldsymbol{\delta}$ is normally distributed, this can be simplified to $\tilde{f}\Big((x_1 + \delta_1, x_2 + \delta_2, .., x_n + \delta_n)^T\Big)$. It would then make sense to sample the probability distribution of $\boldsymbol{\delta}$ a number of t times and to use the mean values of $\tilde{f}(\mathbf{x}, \boldsymbol{\delta})$ for each objective function evaluation during the optimization process. In cases where the optimal value y of the objective function f is known, Equation 3 can be minimized. This approach is also used in the work of Wiesmann et al [234, 235] and basically turns the optimization algorithm into something like a maximum likelihood estimator.

$$f'(\mathbf{x}) = \frac{1}{t} \sum_{i=1}^{t} \Big(y - \tilde{f}(\mathbf{x}, \boldsymbol{\delta}_i)\Big)^2 \tag{3}$$

This method corresponds to using multiple, different training scenarios during the objective function evaluation in situations where $\mathbb{X} \not\subseteq \mathbb{R}^n$. By adding random noise and artificial perturbations to the training cases, the chance of obtaining robust solutions which are stable when applied or realized under noisy conditions can be increased.

8 Overfitting and Oversimplification

In all scenarios where optimizers evaluate some of the objective values of the solution candidates by using training data, two additional phenomena with negative influence can be observed: overfitting and oversimplification.

8.1 Overfitting

8.1.1 The Problem

Definition 5 (Overfitting). Overfitting is the emergence of an overly complicated model (solution candidate) in an optimization process resulting from the effort to provide the best results for as much of the available training data as possible [64, 80, 190, 202].

A model (solution candidate) $m \in \mathbb{X}$ created with a finite set of training data is considered to be overfitted if a less complicated, alternative model

[6] `http://en.wikipedia.org/wiki/Taguchi_methods` [accessed 2008-07-19]

$m' \in \mathbb{X}$ exists which has a smaller error for the set of all possible (maybe even infinitely many), available, or (theoretically) producible data samples. This model m' may, however, have a larger error in the training data.

The phenomenon of overfitting is best known and can often be encountered in the field of artificial neural networks or in curve fitting [124, 128, 181, 191, 211]. The latter means that we have a set A of n training data samples (x_i, y_i) and want to find a function f that represents these samples as well as possible, i.e., $f(x_i) = y_i \; \forall (x_i, y_i) \in A$.

There exists exactly one polynomial of the degree $n - 1$ that fits to each such training data and goes through all its points. Hence, when only polynomial regression is performed, there is exactly one perfectly fitting function of minimal degree. Nevertheless, there will also be an infinite number of polynomials with a higher degree than $n - 1$ that also match the sample data perfectly. Such results would be considered as overfitted.

In Fig. 9, we have sketched this problem. The function $f_1(x) = x$ shown in Fig. 9.b has been sampled three times, as sketched in Fig. 9.a. There exists no other polynomial of a degree of two or less that fits to these samples than f_1. Optimizers, however, could also find overfitted polynomials of a higher degree such as f_2 which also match the data, as shown in Fig. 9.c. Here, f_2 plays the role of the overly complicated model m which will perform as good as the simpler model m' when tested with the training sets only, but will fail to deliver good results for all other input data.

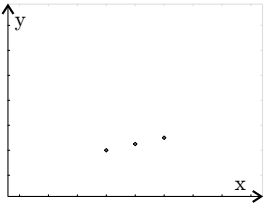


Fig. 9.a: Three sample points of f_1

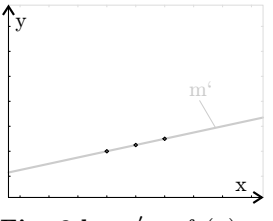


Fig. 9.b: $m' \equiv f_1(x) = x$

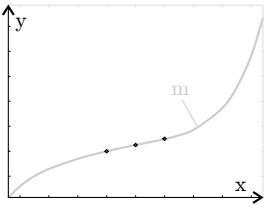


Fig. 9.c: $m \equiv f_2(x)$

Fig. 9 Overfitting due to complexity

A very common cause for overfitting is noise in the sample data. As we have already pointed out, there exists no measurement device for physical processes which delivers perfect results without error. Surveys that represent the opinions of people on a certain topic or randomized simulations will exhibit variations from the true interdependencies of the observed entities, too. Hence, data samples based on measurements will always contain some noise.

In Fig. 10 we have sketched how such noise may lead to overfitted results. Fig. 10.a illustrates a simple physical process obeying some quadratic equation. This process has been measured using some technical equipment

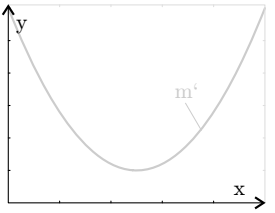


Fig. 10.a: The original physical process

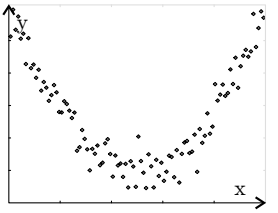


Fig. 10.b: The measurement/training data

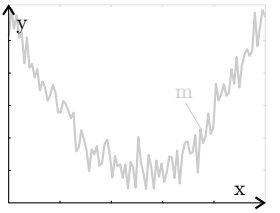


Fig. 10.c: The overfitted result

Fig. 10 Fitting noise

and the 100 noisy samples depicted in Fig. 10.b has been obtained. Fig. 10.c shows a function resulting from an optimization that fits the data perfectly. It could, for instance, be a polynomial of degree 99 that goes right through all the points and thus, has an error of zero. Although being a perfect match to the measurements, this complicated model does not accurately represent the physical law that produced the sample data and will not deliver precise results for new, different inputs.

From the examples we can see that the major problem that results from overfitted solutions is the loss of generality.

Definition 6 (Generality). A solution of an optimization process is general if it is not only valid for the sample inputs $a_1, a_2, \ldots, a_n$ which were used for training during the optimization process, but also for different inputs $a \neq a_i \; \forall i : 0 < i \leq n$ if such inputs a exist.

8.1.2 Countermeasures

There exist multiple techniques that can be utilized in order to prevent overfitting to a certain degree. It is most efficient to apply multiple such techniques together in order to achieve best results.

A very simple approach is to restrict the problem space $\mathbb{X}$ in a way that only solutions up to a given maximum complexity can be found. In terms of function fitting, this could mean limiting the maximum degree of the polynomials to be tested. Furthermore, the functional objective functions which solely concentrate on the error of the solution candidates should be augmented by penalty terms and non-functional objective functions putting pressure in the direction of small and simple models [64, 116].

Large sets of sample data, although slowing down the optimization process, may improve the generalization capabilities of the derived solutions. If arbitrarily many training datasets or training scenarios can be generated, there are two approaches which work against overfitting:

1. The first method is to use a new set of (randomized) scenarios for each evaluation of a solution candidate. The resulting objective values may differ

largely even if the same individual is evaluated twice in a row, introducing incoherence and ruggedness into the fitness landscape.
2. At the beginning of each iteration of the optimizer, a new set of (randomized) scenarios is generated which is used for all individual evaluations during that iteration. This method leads to objective values which can be compared without bias.

In both cases it is helpful to use more than one training sample or scenario per evaluation and to set the resulting objective value to the average (or better median) of the outcomes. Otherwise, the fluctuations of the objective values between the iterations will be very large, making it hard for the optimizers to follow a stable gradient for multiple steps.

Another simple method to prevent overfitting is to limit the runtime of the optimizers [190]. It is commonly assumed that learning processes normally first find relatively general solutions which subsequently begin to overfit because the noise "is learned", too.

For the same reason, some algorithms allow to decrease the rate at which the solution candidates are modified by time. Such a decay of the learning rate makes overfitting less likely.

If only one finite set of data samples is available for training/optimization, it is common practice to separate it into a set of training data A_t and a set of test cases A_c. During the optimization process, only the training data is used. The resulting solutions are tested with the test cases afterwards. If their behavior is significantly worse when applied to A_c than when applied to A_t, they are probably overfitted.

The same approach can be used to detect when the optimization process should be stopped. The best known solution candidates can be checked with the test cases in each iteration without influencing their objective values which solely depend on the training data. If their performance on the test cases begins to decrease, there are no benefits in letting the optimization process continue any further.

8.2 Oversimplification

8.2.1 The Problem

Oversimplification (also called overgeneralization) is the opposite of overfitting. Whereas overfitting denotes the emergence of overly-complicated solution candidates, oversimplified solutions are not complicated enough. Although they represent the training samples used during the optimization process seemingly well, they are rough overgeneralizations which fail to provide good results for cases not part of the training.

A common cause for oversimplification is sketched in Fig. 11: The training sets only represent a fraction of the set of possible inputs. As this is normally the case, one should always be aware that such an incomplete coverage may fail to represent some of the dependencies and characteristics of the data,

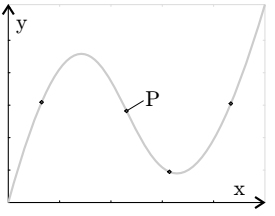


Fig. 11.a: The "real system" and the points describing it

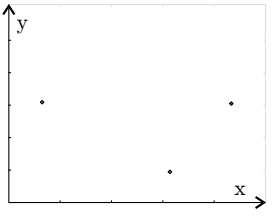


Fig. 11.b: The sampled training data

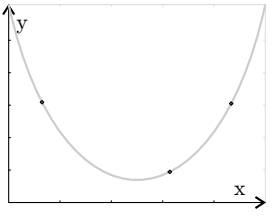


Fig. 11.c: The oversimplified result

Fig. 11 Oversimplification

which then may lead to oversimplified solutions. Another possible reason is that ruggedness, deceptiveness, too much neutrality, or high epistasis in the fitness landscape may lead to premature convergence and prevent the optimizer from surpassing a certain quality of the solution candidates. It then cannot completely adapt them even if the training data perfectly represents the sampled process. A third cause is that a problem space which does not include the correct solution was chosen.

Fig. 11.a shows a cubic function. Since it is a polynomial of degree three, four sample points are needed for its unique identification. Maybe not knowing this, only three samples have been provided in Fig. 11.b. By doing so, some vital characteristics of the function are lost. Fig. 11.c depicts a square function – the polynomial of the lowest degree that fits exactly to these samples. Although it is a perfect match, this function does not touch any other point on the original cubic curve and behaves totally differently at the lower parameter area.

However, even if we had included point P in our training data, it would still be possible that the optimization process would yield Fig. 11.c as a result. Having training data that correctly represents the sampled system does not mean that the optimizer is able to find a correct solution with perfect fitness – the other, previously discussed problematic phenomena can prevent it from doing so. Furthermore, if it was not known that the system which was to be modeled by the optimization process can best be represented by a polynomial of the third degree, one could have limited the problem space $\mathbb{X}$ to polynomials of degree two and less. Then, the result would likely again be something like Fig. 11.c, regardless of how many training samples are used.

8.2.2 Countermeasures

In order to counter oversimplification, its causes have to be mitigated. Generally, it is not possible to have training scenarios which cover the complete input space of the evolved programs. By using multiple scenarios for each individual evaluation, the chance of missing important aspects is decreased. These scenarios can be replaced with new, randomly created ones in each

generation, which will decrease this chance even more. The problem space, i.e., the representation of the solution candidates, should further be chosen in a way which allows constructing a correct solution to the problem defined. Then again, releasing too many constraints on the solution structure increases the risk of overfitting and thus, careful proceeding is recommended.

9 Multi-objective Optimization

9.1 Introduction

Many optimization problems in the real world have k possibly contradictory objectives f_i which must be optimized simultaneously. Furthermore, the solutions must satisfy m inequality constraints g and p equality constraints h. A solution candidate x is *feasible*, if and only if $g_i(x) \geq 0 \; \forall i = 1, 2, .., m$ and $h_i(x) = 0 \; \forall i = 1, 2, .., p$ holds. A *multi-objective optimization problem* (MOP) can then be formally defined as follows:

Definition 7 (MOP). Find a solution candidate $x^\star$ in $\mathbb{X}$ which minimizes (or maximizes) the vector function $\mathbf{f}(x^\star) = (f_i(x^\star), f_2(x^\star), .., f_k(x^\star))^T$ and is feasible, (i.e., satisfies the m inequality constraints $g_i(x^\star) \geq 0 \; \forall i = 1, 2, .., m$, the p equality constraints $h_i(x^\star) = 0 \; \forall i = 1, 2, .., p$).

As in single-objective optimization, nature-inspired algorithms are popular techniques to solve such problems. The fact that there are two or more objective functions implies additional difficulties. Due to the contradictory feature of the functions in a MOP and the fact that there exists no total order in $\mathbb{R}^n$ for $n > 1$, the notions of "better than" and "optimum" have to be redefined. When comparing any two solutions x_1 and x_2, solution x_1 can have a better value in objective f_i, i.e., $f_i(x_1) < f_i(x_2)$, while solution x_2 can have a better value in objective f_j. The concepts commonly used here are *Pareto dominance* and *Pareto optimality*.

Definition 8 (Pareto Dominance). In the context of multi-objective global optimization, a solution candidate x_1 is said to *dominate* another solution candidate x_2 (denoted by $x_1 \preccurlyeq x_2$) if and only if $\mathbf{f}(x_1)$ is partially less than $\mathbf{f}(x_2)$, i.e., $\forall i \in \{1, .., k\} \;\; f_i(x_1) \leq f_i(x_2) \;\wedge\; \exists j \in \{1, .., k\} : \; f_j(x_1) < f_j(x_2)$.

The dominance notion allows us to assume that if solution x_1 dominates solution x_2, then x_1 is preferable to x_2. If both solution are non-dominated (such as candidate ① and ② in Fig. 12), some additional criteria have to be used to choose one of them.

Definition 9 (Pareto Optimality). A feasible point $x^\star \in \mathbb{X}$ is *Pareto-optimal* if and only if there is no feasible $x_b \in \mathbb{X}$ with $x_b \preccurlyeq x^\star$.

This definition states that $x^\star$ is Pareto-optimal if there is no other feasible solution x_b which would improve some criterion without causing a simultaneous worsening in at least one other criterion. The solution to a MOP,

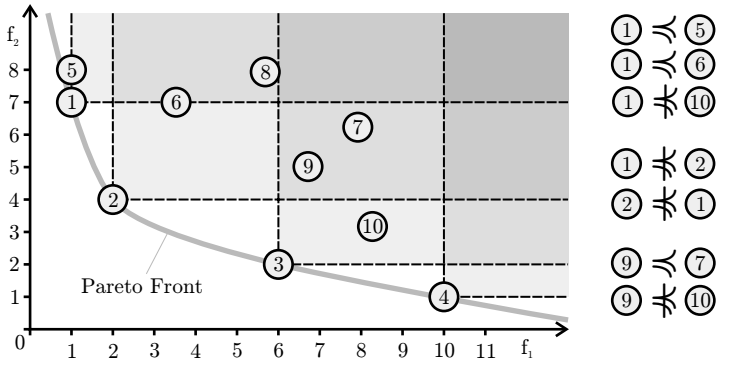


Fig. 12 Some examples for the dominance relation

considering Pareto optimality, is the set of feasible, non-dominated solutions which is known as *Pareto-optimal set*:

Definition 10 (Pareto-Optimal Set). For a given MOP $\mathbf{f}(x)$, the Pareto optimal set is defined as $\mathcal{P}^\star = \{x^\star \in \mathbb{X} | \neg\exists x \in \mathbb{X} : x \preccurlyeq x^\star\}$.

When the solutions in the Pareto-optimal set are plotted in the objective space (as sketched in Fig. 12), they are collectively known as the *Pareto front*:

Definition 11 (Pareto Front). For a given MOP $\mathbf{f}(x)$ and its Pareto-optimal set $\mathcal{P}^\star$, the Pareto front is defined as $\mathcal{PF}^\star = \{\mathbf{f}(x) \,| x \in \mathcal{P}^\star\}$.

Obtaining the Pareto front of a MOP is the main goal of multi-objective optimization. In a real scenario, the solutions in the Pareto front are sent to an expert in the MOP, the decision maker, who will be responsible for choosing the best tradeoff solution among all of them. Fig. 13 depicts the Pareto front of a bi-objective MOP. In a real problem example, f_1 could

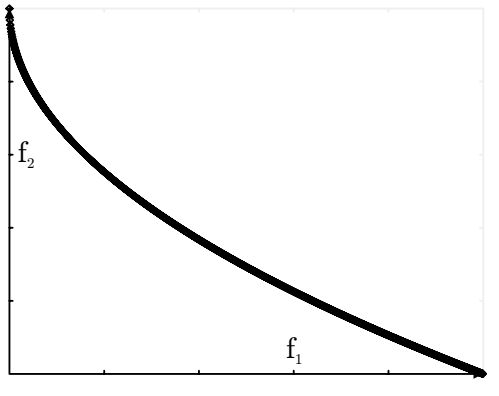


Fig. 13 Example of Pareto front of a bi-objective MOP

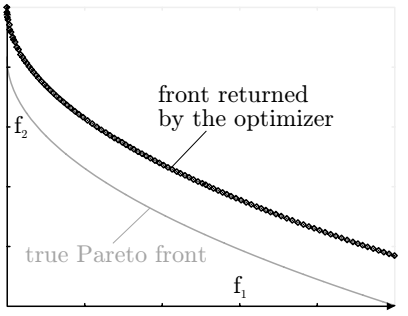


Fig. 14.a: Bad Convergence and Good Spread

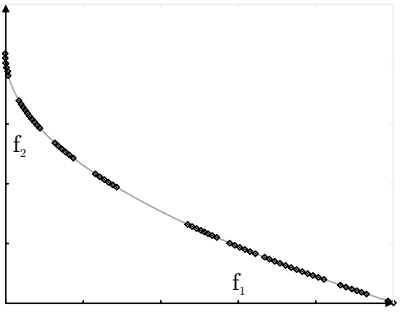


Fig. 14.b: Good Convergence and Bad Spread

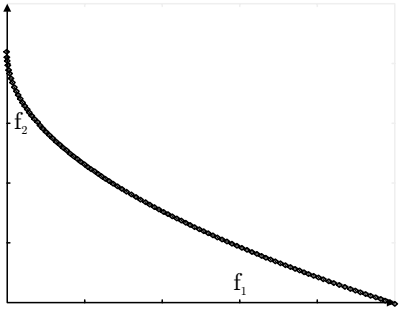


Fig. 14.c: Good Convergence and Spread

Fig. 14 Pareto front approximation sets

represent the time required by a car to cover a given distance, while f_2 could be the fuel consumption.

The Pareto front of a MOP can contain a large (possibly infinite) number of points. Usually, the goal of optimization is to obtain a fixed-size set of solutions called *Pareto front approximation set*. Population-based algorithms, such as Genetic Algorithms, are very popular to solve MOPs because they can provide an approximation set in a single run.

Given that the goal is to find a Pareto front approximation set, two issues arise. First, the optimization process should *converge* to the true Pareto front and return solutions as close to it as possible. Second, they should be uniformly spread along this front.

Let us examine the three fronts included in Fig. 14. The first picture (Fig. 14.a) shows an approximation set having a very good spread[7] of

[7] In MO optimization, this property is usually called *diversity*. In order to avoid confusion with the (related) diversity property from Section 2.3, we here use the term *spread* instead.

solutions, but the points are far away from the true Pareto front. Such results are not attractive because they do not provide Pareto-optimal solutions. The second example (Fig. 14.b) contains a set of solutions which are very close to the true Pareto front but cover it only partially, so the decision maker could lose important trade-off solutions. Finally, the front depicted in Fig. 14.c has the two desirable properties of good convergence and spread.

9.2 The Problem

Features such as multi-modality, deceptiveness, or epistasis found in single-objective optimization also affect MOPs, making them more difficult to solve. However, there are some characteristics that are particular to MOPs. Here we comment on two of them: geometry and dimensionality.

The Pareto front in Fig. 13 has a convex geometry, but there are other different shapes as well. In Fig. 15 we show some examples, including non-convex (concave), disconnected, linear, and non-uniformly distributed Pareto

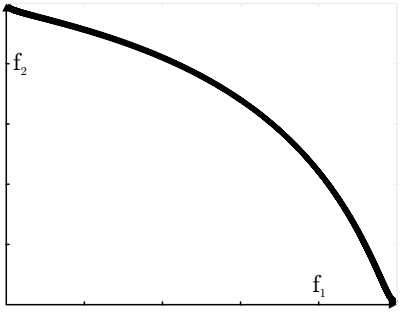


Fig. 15.a: Non-Convex (Concave)

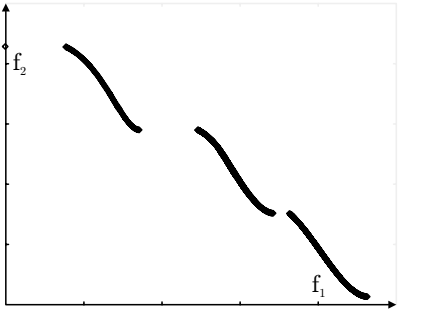


Fig. 15.b: Disconnected

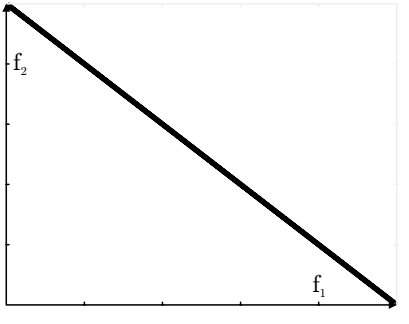


Fig. 15.c: linear

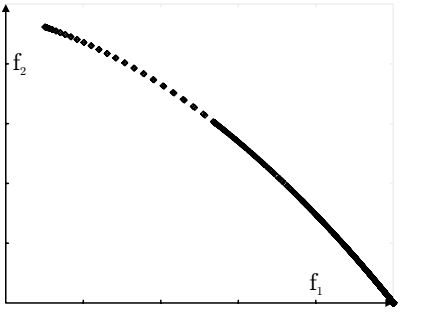


Fig. 15.d: Non-Uniformly Distributed

Fig. 15 Examples of Pareto fronts

fronts. Besides Pareto optimization, there is a wide variety of other concepts for defining what optima are in the presence of multiple objective functions [45]. The simplest approach is maybe to use a weighted sum of all objective values and set $v(x) = \sum_{i=1}^{k} f_i(x)$. Then the optima would be the element(s) $x^\star$ with $\neg\exists x \in \mathbb{X} : v(x) < v(x^\star)$. However, an optimization process driven by such a linear aggregating function will not find portions of Pareto fronts with non-convex geometry as shown by Das and Dennis [50].

Many studies in the literature consider mainly bi-objective MOPs. As a consequence, many algorithms are designed to deal with that kind of problems. However, MOPs having a higher number of objective functions are common in practice, leading to the so-called *many-objective optimization* [165], which is currently a hot research topic. Most of the optimization algorithms applied today utilize the Pareto dominance relation. When the dimension of the MOPs increases, the majority of solution candidates are non-dominated. As a consequence, traditional nature-inspired algorithms have to be redesigned.

9.3 Countermeasures

In order to obtain an accurate approximation to the true Pareto front, many nature-inspired multi-objective algorithms apply a fitness assignment scheme based on the concept of Pareto dominance, as commented before. For example, NSGA-II [61, 62], the most well-known multi-objective technique, assigns to each solution a rank depending on the number of solutions dominating it. Thus, solutions with rank 1 are non-dominated, solutions with rank 2 are dominated by one solution, and so on. Other algorithms, such as SPEA2 [247, 248] introduce the concept of strength, which is similar to the ranking but also considers the number of dominated solutions.

While the use of Pareto-based ranking methods allows the techniques to search in the direction of finding approximations with good convergence, additional strategies are needed to promote spread. The most commonly adopted approach is to include a kind of density estimator in order to select those solutions which are in the less crowded regions of the objective space. Thus, NSGA-II employs the crowding distance [61] and SPEA2 the distance to the k-nearest neighbor [62].

9.4 Constraint Handling

How the constraints mentioned in Definition 7 are handled is a whole research area in itself with roots in single-objective optimization. Maybe one of the most popular approach for dealing with constraints goes back to Courant [48] who introduced the idea of *penalty functions* [73, 44, 201] in 1943: Consider, for instance, the term $f'(x) = f(x) + v\,[h(x)]^2$ where f is the original objective

function, h is an equality constraint, and $v > 0$. If f' is minimized, an infeasible individual will always have a worse fitness than a feasible one with the same objective values.

Besides such static penalty functions, *dynamic* terms incorporating the generation counter [111, 157] or *adaptive* approaches utilizing additional population statistics [95, 199] have been proposed. Rigorous discussions on penalty functions have been contributed by Fiacco and McCormick [73] and Smith and Coit [201].

During the last fifteen years, many approaches have been developed which incorporate constraint handling and multi-objectivity. Instead of using penalty terms, Pareto ranking can also be extended by additionally comparing individuals according to their feasibility, for instance. Examples for this approach are the Method of Inequalities (MOI) of Zakian [245] as used by Pohlheim [164] and the Goal Attainment method defined in [76]. Deb [56, 58] even suggested to simply turn constraints into objective functions in his MOEA version of Goal Programming.

10 Dynamically Changing Fitness Landscape

It should also be mentioned that there exist problems with dynamically changing fitness landscapes [33, 32, 36, 147, 173]. The task of an optimization algorithm is, then, to provide solution candidates with momentarily optimal objective values for each point in time. Here we have the problem that an optimum in iteration t will possibly not be an optimum in iteration $t + 1$ anymore.

The moving peaks benchmarks by Branke [33, 32] and Morrison and De Jong [147] are good examples for dynamically changing fitness landscapes. Such problems with dynamic characteristics can, for example, be tackled with special forms [244] of

- Evolutionary Algorithms [9, 34, 35, 145, 146, 216, 236],
- Genetic Algorithms [83, 119, 142, 143, 144],
- Particle Swarm Optimization [23, 42, 43, 126, 160],
- Differential Evolution [135, 243], and
- Ant Colony Optimization [90, 91]

11 The No Free Lunch Theorem

By now, we know the most important problems that can be encountered when applying an optimization algorithm to a given problem. Furthermore, we have seen that it is arguable what actually an optimum is if multiple criteria are optimized at once. The fact that there is most likely no optimization method that can outperform all others on all problems can, thus, easily be accepted. Instead, there exist a variety of optimization methods specialized

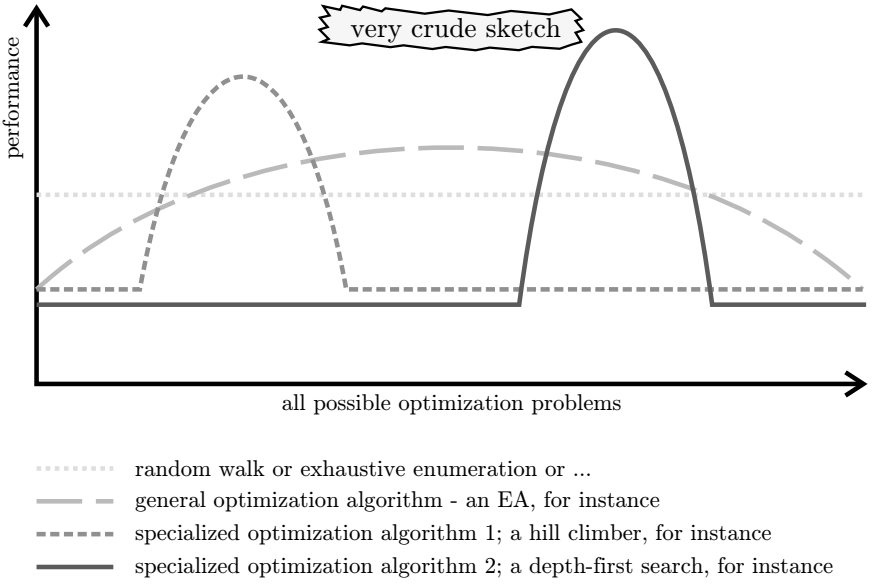


Fig. 16 A visualization of the No Free Lunch Theorem

in solving different types of problems. There are also algorithms which deliver good results for many different problem classes, but may be outperformed by highly specialized methods in each of them.

These facts have been formalized by Wolpert and Macready [241, 242] in their *No Free Lunch Theorems* (NFL) for search and optimization algorithms. Wolpert and Macready [242] focus on single-objective optimization and prove that the sum of the values of any performance measure (such as the objective value of the best solution candidate discovered until a time step m) over all possible objective functions f is always identical for all optimization algorithms.

From this theorem, we can immediately follow that, in order to outperform the optimization method a_1 in one optimization problem, the algorithm a_2 will necessarily perform worse in another. Fig. 16 visualizes this issue. The higher the value of the performance measure illustrated there, the faster will the corresponding problem be solved. The figure shows that general optimization approaches (like Evolutionary Algorithms) can solve a variety of problem classes with reasonable performance. Hill Climbing approaches, for instance, will be much faster than Evolutionary Algorithms if the objective functions are steady and monotonous, that is, in a smaller set of optimization tasks. Greedy search methods will perform fast on all problems with matroid structure. Evolutionary Algorithms will most often still be able to solve these problems, it just takes them longer to do so. The performance of Hill Climbing and greedy approaches degenerates in other classes of optimization tasks as a trade-off for their high utility in their "area of expertise".

One interpretation of the No Free Lunch Theorem is that it is impossible for any optimization algorithm to outperform random walks or exhaustive enumerations on all possible problems. For every problem where a given method leads to good results, we can construct a problem where the same method has exactly the opposite effect (see Section 4). As a matter of fact, doing so is even a common practice to find weaknesses of optimization algorithms and to compare them with each other.

Another interpretation is that every useful optimization algorithm utilizes some form of problem-specific knowledge. Radcliffe [167] states that without such knowledge, search algorithms cannot exceed the performance of simple enumerations. Incorporating knowledge starts with relying on simple assumptions like "if x is a good solution candidate, than we can expect other good solution candidates in its vicinity", i.e., strong causality. The more (correct) problem specific knowledge is integrated (correctly) into the algorithm structure, the better will the algorithm perform. On the other hand, knowledge correct for one class of problems is, quite possibly, misleading for another class. In reality, we use optimizers to solve a given set of problems and are not interested in their performance when (wrongly) applied to other classes.

Today, there exists a wide range of work on No Free Lunch Theorems for many different aspects of machine learning. The website `http://www.no-free-lunch.org/`[8] gives a good overview about them. Further summaries and extensions have been provided by Köppen et al [121] and Igel and Toussaint [108, 109]. Radcliffe and Surry [168] discuss the NFL in the context of Evolutionary Algorithms and the representations used as search spaces. The No Free Lunch Theorem is furthermore closely related to the Ugly Duckling Theorem proposed by Watanabe [228] for classification and pattern recognition.

12 Concluding Remarks

The subject of this introductory chapter was the question about what makes optimization problems hard, especially for metaheuristic approaches. We have discussed numerous different phenomena which can affect the optimization process and lead to disappointing results. If an optimization process has converged prematurely, it has been trapped in a non-optimal region of the search space from which it cannot "escape" anymore (Section 2). Ruggedness (Section 3) and deceptiveness (Section 4) in the fitness landscape, often caused by epistatic effects (Section 6), can misguide the search into such a region. Neutrality and redundancy (Section 5) can either slow down optimization because the application of the search operations does not lead to a gain in information or may also contribute positively by creating neutral networks from which the search space can be explored and local optima can be escaped

[8] Accessed: 2008-03-28

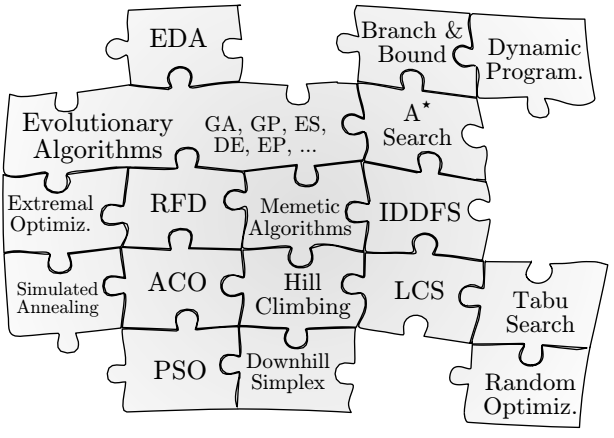


Fig. 17 The puzzle of optimization algorithms

from. The solutions that are derived, even in the presence of noise, should be robust (Section 7). Also, they should neither be too general (oversimplification, Section 8.2) nor too specifically aligned only to the training data (overfitting, Section 8.1). Furthermore, many practical problems are multi-objective, i.e., involve the optimization of more than one criterion at once (Section 9), or concern objectives which may change over time (Section 10).

In the previous section, we discussed the No Free Lunch Theorem and argued that it is not possible to develop the *one* optimization algorithm, the problem-solving machine which can provide us with near-optimal solutions in short time for every possible optimization task. This must sound very depressing for everybody new to this subject.

Actually, quite the opposite is the case, at least from the point of view of a researcher. The No Free Lunch Theorem means that there will always be new ideas, new approaches which will lead to better optimization algorithms to solve a given problem. Instead of being doomed to obsolescence, it is far more likely that most of the currently known optimization methods have at least one niche, one area where they are excellent. It also means that it is very likely that the "puzzle of optimization algorithms" will never be completed. There will always be a chance that an inspiring moment, an observation in nature, for instance, may lead to the invention of a new optimization algorithm which performs better in some problem areas than all currently known ones.

Acknowledgements. We gratefully acknowledge comments on early drafts of this chapter by Peter J Bentley and Patrick Siarry. The last author acknowledges support from the "Consejería de Innovación, Ciencia y Empresa", Junta de Andalucía under contract `P07-TIC-03044 DIRICOM` project, `http://diricom.lcc.uma.es`.

References

1. Ackley, D.H.: A connectionist machine for genetic hillclimbing. The Springer International Series in Engineering and Computer Science, vol. 28. Kluwer Academic Publishers, Dordrecht (1987)
2. Altenberg, L.: The schema theorem and price's theorem. Foundations of Genetic Algorithms 3, 23–49 (1994)
3. Altenberg, L.: Genome growth and the evolution of the genotype-phenotype map. In: Evolution and Biocomputation – Computational Models of Evolution, pp. 205–259. Springer, Heidelberg (1995)
4. Altenberg, L.: Nk fitness landscapes. In: Handbook of Evolutionary Computation, ch.. B2.7.2. Oxford University Press, Oxford (1996)
5. Altenberg, L.: Fitness distance correlation analysis: An instructive counterexample. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 57–64 (1997)
6. Amitrano, C., Peliti, L., Saber, M.: Population dynamics in a spin-glass model of chemical evolution. Journal of Molecular Evolution 29(6), 513–525 (1989)
7. Amor, H.B., Rettinger, A.: Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1531–1538 (2005) doi:10.1145/1068009.1068250
8. Angeline, P.J., Pollack, J.: Evolutionary module acquisition. In: The Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, pp. 154–163 (1993)
9. Aragón, V.S., Esquivel, S.C.: An evolutionary algorithm to track changes of optimum value locations in dynamic environments. Journal of Computer Science & Technology (JCS&T) 4(3), 127–133 (2004); invited paper
10. Bachmann, P.G.H.: Die Analytische Zahlentheorie / Dargestellt von Paul Bachmann, Zahlentheorie: Versuch einer Gesamtdarstellung dieser Wissenschaft in ihren Haupttheilen, vol. Zweiter Theil. B. G. Teubner, Leipzig, Germany (1894)
11. Bäck, T., Hammel, U.: Evolution strategies applied to perturbed objective functions. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1, pp. 40–45 (1994) doi:10.1109/ICEC.1994.350045
12. Bak, P., Sneppen, K.: Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters 71, 4083–4086 (1993)
13. Baldwin, J.M.: A new factor in evolution. The American Naturalist 30, 441–451 (1896)
14. Barnett, L.: Tangled webs: Evolutionary dynamics on fitness landscapes with neutrality. Master's thesis, School of Cognitive Science, University of East Sussex, Brighton, UK (1997)
15. Barnett, L.: Ruggedness and neutrality – the nkp family of fitness landscapes. In: Artificial Life VI: Proceedings of the sixth international conference on Artificial life, pp. 18–27 (1998)
16. Bateson, W.: Mendel's Principles of Heredity. Cambridge University Press, Cambridge (1909)
17. Beaudoin, W., Verel, S., Collard, P., Escazut, C.: Deceptiveness and neutrality the nd family of fitness landscapes. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 507–514 (2006) doi:10.1145/1143997.1144091

18. Beerenwinkel, N., Pachter, L., Sturmfels, B.: Epistasis and shapes of fitness landscapes. Eprint arXiv:q-bio/0603034 (Quantitative Biology, Populations and Evolution) (accessed 2007-08-05) (2006), http://arxiv.org/abs/q-bio.PE/0603034
19. Bergman, A., Feldman, M.W.: Recombination dynamics and the fitness landscape. Physica D: Nonlinear Phenomena 56, 57–67 (1992)
20. Bethke, A.D.: Genetic algorithms as function optimizers. PhD thesis, University of Michigan, Ann Arbor, MI, USA (1980)
21. Beyer, H.-G.: Toward a theory of evolution strategies: Some asymptotical results from the $(1, +\lambda)$-theory. Evolutionary Computation 1(2), 165–188 (1993)
22. Beyer, H.-G.: Toward a theory of evolution strategies: The (μ, λ)-theory. Evolutionary Computation 2(4), 381–407 (1994)
23. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary Computation in Dynamic and Uncertain Environments, ch. 2, pp. 29–52. Springer, Heidelberg (2007)
24. Bledsoe, W.W., Browning, I.: Pattern recognition and reading by machine. In: Proceedings of the Eastern Joint Computer Conference (EJCC) – Papers and Discussions Presented at the Joint IRE - AIEE - ACM Computer Conference, pp. 225–232 (1959)
25. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35(3), 268–308 (2003)
26. Bonner, J.T.: On Development: The Biology of Form, new ed edn. Commonwealth Fund Publications, Harvard University Press (1974)
27. Bornberg-Bauer, E., Chan, H.S.: Modeling evolutionary landscapes: Mutational stability, topology, and superfunnels in sequence space. Proceedings of the National Academy of Science of the United States of Americs (PNAS) – Biophysics 96(19), 10689–10694 (1999)
28. Bosman, P.A.N., Thierens, D.: Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. International Journal Approximate Reasoning 31(3), 259–289 (2002)
29. Bosman, P.A.N., Thierens, D.: A thorough documentation of obtained results on real-valued continuous and combinatorial multi-objective optimization problems using diversity preserving mixture-based iterated density estimation evolutionary algorithms. Tech. Rep. UU-CS-2002-052, Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands (2002)
30. Brameier, M.F., Banzhaf, W.: Explicit control of diversity and effective variation distance in linear genetic programming. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278, pp. 37–49. Springer, Heidelberg (2002)
31. Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 119–128. Springer, Heidelberg (1998)
32. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 3, pp. 1875–1882 (1999) doi:10.1109/CEC.1999.785502
33. Branke, J.: The moving peaks benchmark. Tech. rep., Institute AIFB, University of Karlsruhe, Germany (accessed 2007-08-19) (1999), http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/ Presented in [32]

34. Branke, J.: Evolutionary optimization in dynamic environments. PhD thesis, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften (2000)
35. Branke, J.: Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Kluwer Academic Publishers, Dordrecht (2001)
36. Branke, J., Salihoğlu, E., Uyar, Ş.: Towards an analysis of dynamic environments. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1433–1440 (2005)
37. Bremermann, H.J.: Optimization through evolution and recombination. Self-Organizing systems pp. 93–100 (1962)
38. Burke, E.K., Gustafson, S.M., Kendall, G.: Survey and analysis of diversity measures in genetic programming. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 716–723 (2002)
39. Burke, E.K., Gustafson, S.M., Kendall, G., Krasnogor, N.: Is increasing diversity in genetic programming beneficial? an analysis of the effects on fitness. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 1398–1405 (2003)
40. Burke, E.K., Gustafson, S.M., Kendall, G.: Diversity in genetic programming: An analysis of measures and correlation with fitness. IEEE Transactions on Evolutionary Computation 8(1), 47–62 (2004)
41. Cantú-Paz, E., Pelikan, M., Goldberg, D.E.: Linkage problem, distribution estimation, and bayesian networks. Evolutionary Computation 8(3), 311–340 (2000)
42. Carlisle, A.J.: Applying the particle swarm optimizer to non-stationary environments. PhD thesis, Graduate Faculty of Auburn University (2002)
43. Carlisle, A.J., Dozier, G.V.: Tracking changing extrema with adaptive particle swarm optimizer. In: Proceedings of the 5th Biannual World Automation Congress, WAC 2002, Orlando, Florida, USA, vol. 13, pp. 265–270 (2002) doi:10.1109/WAC.2002.1049555
44. Carroll, C.W.: An operations research approach to the economic optimization of a kraft pulping process. PhD thesis, Institute of Paper Chemistry, Appleton, Wisconsin, USA (1959)
45. Ceollo Coello, C.A., Lamont, G.B., van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, Genetic and Evolutionary Computation. Genetic and Evolutionary Computation (1st edn., 2002, 2nd edn., 2007), vol. 5. Kluwer Academic Publishers, Springer (2007)
46. Chen, Y.p.: Extending the Scalability of Linkage Learning Genetic Algorithms – Theory & Practice. Studies in Fuzziness and Soft Computing, vol. 190. Springer, Heidelberg (2006)
47. Cohoon, J.P., Hegde, S.U., Martin, W.N., Richards, D.: Punctuated equilibria: a parallel genetic algorithm. In: Proceedings of the Second International Conference on Genetic algorithms and their Application, pp. 148–154 (1987)
48. Courant, R.: Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Society 49(1), 1–23 (1943)
49. Cousins, S.H.: Species diversity measurement: Choosing the right index. Trends in Ecology and Evolution (TREE) 6(6), 190–192 (1991)

50. Das, I., Dennis, J.E.: A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Structural optimization 14(1), 63–69 (1997)
51. Davidor, Y.: Epistasis variance: A viewpoint on GA-hardness. In: Proceedings of the First Workshop on Foundations of Genetic Algorithms, pp. 23–35 (1990)
52. Dawkins, R.: The evolution of evolvability. In: ALIFE – Artificial Life: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, pp. 201–220 (1987)
53. de Jong, E.D., Watson, R.A., Pollack, J.B.: Reducing bloat and promoting diversity using multi-objective methods. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 11–18 (2001)
54. de Lamarck, J.B.P.A.d.C.: Philosophie zoologique – ou Exposition des considérations relatives à l'histoire naturelle des Animaux. Dentu / G. Baillière, Paris, France/Harvard University (1809)
55. Deb, K.: Genetic algorithms in multimodal function optimization. Master's thesis, The Clearinghouse for Genetic algorithms, University of Alabama, Tuscaloosa, tCGA Report No. 89002 (1989)
56. Deb, K.: Solving goal programming problems using multi-objective genetic algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 77–84 (1999) doi:10.1109/CEC.1999.781910
57. Deb, K.: Genetic algorithms for optimization. KanGAL Report 2001002, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur, PIN 208 016, India (2001)
58. Deb, K.: Nonlinear goal programming using multi-objective genetic algorithms. Journal of the Operational Research Society 52(3), 291–302 (2001)
59. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Foundations of Genetic Algorithms 2, pp. 93–108 (1993)
60. Deb, K., Goldberg, D.E.: Sufficient conditions for deceptive and easy binary functions. Annals of Mathematics and Artificial Intelligence 10(4), 385–408 (1994)
61. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp. 849–858 (2000); KanGAL Report No. 200001
62. Deb, K., Pratab, A., Agrawal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
63. Deb, K., Sinha, A., Kukkonen, S.: Multi-objective test problems, linkages, and evolutionary methodologies. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1141–1148. ACM, New York (2006)
64. Dietterich, T.: Overfitting and undercomputing in machine learning. ACM Computing Surveys (CSUR) 27(3), 326–327 (1995)
65. Droste, S., Wiesmann, D.: On representation and genetic operators in evolutionary algorithms. Tech. Rep. CI–41/98, Fachbereich Informatik, Universität Dortmund (1998)
66. Eiben, Á.E., Schippers, C.A.: On evolutionary exploration and exploitation. Fundamenta Informaticae 35(1-4), 35–50 (1998)

67. Eldredge, N., Gould, S.J.: Punctuated equilibria: an alternative to phyletic gradualism. In: Schopf, T.J.M. (ed.) Models in Paleobiology, ch. 5, pp. 82–115. W.H. Freeman, New York (1972)
68. Eldredge, N., Gould, S.J.: Punctuated equilibria: The tempo and mode of evolution reconsidered. Paleobiology 3(2), 115–151 (1977)
69. Eshelman, L.J., Schaffer, J.D.: Preventing premature convergence in genetic algorithms by preventing incest. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 115–122 (1991)
70. Eshelman, L.J., Caruana, R.A., Schaffer, J.D.: Biases in the crossover landscape. In: Proceedings of the third international conference on Genetic algorithms, pp. 10–19 (1989)
71. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133 (1995)
72. Festa, P., Resende, M.G.: An annotated bibliography of grasp. AT&T Labs Research Technical Report TD-5WYSEW, AT&T Labs (2004)
73. Fiacco, A.V., McCormick, G.P.: Nonlinear Programming: Sequential Unconstrained Minimization Techniques. John Wiley & Sons Inc., Chichester (1968)
74. Fisher, S.R.A.: The correlations between relatives on the supposition of mendelian inheritance. Philosophical Transactions of the Royal Society of Edinburgh 52, 399–433 (1918)
75. Fitzpatrick, J.M., Grefenstette, J.J.: Genetic algorithms in noisy environments. Machine Learning 3(2–3), 101–120 (1988)
76. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416–423 (1993)
77. Forst, C.V., Reidys, C., Weber, J.: Evolutionary dynamics and optimization: Neutral networks as model-landscapes for RNA secondary-structure foldinglandscapes. In: European Conference on Artificial Life, pp. 128–147 (1995)
78. Friedberg, R.M.: A learning machine: Part i. IBM Journal of Research and Development 2, 2–13 (1958)
79. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Series of Books in the Mathematical Sciences. W. H. Freeman & Co., New York (1979)
80. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation 4(1), 1–58 (1992)
81. Glover, F.: Tabu search – part ii. Operations Research Society of America (ORSA) Journal on Computing 2(1), 190–206 (1990)
82. Glover, F., Taillard, É.D., de Werra, D.: A user's guide to tabu search. Annals of Operations Research 41(1), 3–28 (1993)
83. Gobb, H.G., Grefenstette, J.J.: Genetic algorithms for tracking changing environments. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 523–529 (1993)
84. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Longman Publishing Co., Amsterdam (1989)
85. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic algorithms and their Application, pp. 41–49 (1987)
86. Goldberg, D.E., Deb, K., Korb, B.: Messy genetic algorithms: motivation, analysis, and first results. Complex Systems 3, 493–530 (1989)

87. Greiner, H.: Robust filter design by stochastic optimization. Proceedings of SPIE (The International Society for Optical Engineering) 2253, 150–161 (1994)
88. Greiner, H.: Robust optical coating design with evolutionary strategies. Applied Optics 35, 5477–5483 (1996)
89. Gruau, F., Whitley, L.D.: Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. Evolutionary Computation 1(3), 213–233 (1993)
90. Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
91. Guntsch, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 860–867 (2001)
92. Gurin, L.S., Rastrigin, L.A.: Convergence of the random search method in the presence of noise. Automation and Remote Control 26, 1505–1511 (1965)
93. Gustafson, S.M.: An analysis of diversity in genetic programming. PhD thesis, University of Nottingham, School of Computer Science & IT (2004)
94. Gustafson, S.M., Ekárt, A., Burke, E.K., Kendall, G.: Problem difficulty and code growth in genetic programming. Genetic Programming and Evolvable Machines 5(3), 271–290 (2004)
95. Hadj-Alouane, A.B., Bean, J.C.: A genetic algorithm for the multiple-choice integer program. Tech. Rep. 92-50, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbour, MI 48109-2117, USA (1992)
96. Hammel, U., Bäck, T.: Evolution strategies on noisy functions: How to improve convergence properties. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 159–168. Springer, Heidelberg (1994)
97. Han, L., He, X.: A novel opposition-based particle swarm optimization for noisy problems. In: ICNC 2007: Proceedings of the Third International Conference on Natural Computation, vol. 3, pp. 624–629 (2007) doi:10.1109/ICNC.2007.119
98. Handa, H., Lin, D., Chapman, L., Yao, X.: Robust solution of salting route optimisation using evolutionary algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 3098–3105 (2006) doi:10.1109/CEC.2006.1688701
99. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. PhD thesis, University of Michigan, Ann Arbor (1997)
100. Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. Complex Systems 1, 495–502 (1987)
101. Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. In: Adaptive individuals in evolving populations: models and algorithms, pp. 447–454. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (1996)
102. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The University of Michigan Press, Ann Arbor (1975); reprinted by MIT Press, NetLibrary, Inc. (April 1992)

103. Holland, J.H.: Genetic algorithms. Scientific American 267(1), 44–50 (1992)
104. Horn, J., Nafpliotis, N., Goldberg, D.E.: A niched pareto genetic algorithm for multiobjective optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 1, pp. 82–87 (1994) doi:10.1109/ICEC.1994.350037
105. Huynen, M.A.: Exploring phenotype space through neutral evolution. Journal of Molecular Evolution 43(3), 165–169 (1996)
106. Huynen, M.A., Stadler, P.F., Fontana, W.: Smoothness within ruggedness: The role of neutrality in adaptation. Proceedings of the National Academy of Science, USA 93, 397–401 (1996)
107. Igel, C.: Causality of hierarchical variable length representations. In: Proceedings of the 1998 IEEE World Congress on Computational Intelligence, pp. 324–329 (1998)
108. Igel, C., Toussaint, M.: On classes of functions for which no free lunch results hold. Information Processing Letters 86(6), 317–321 (2003)
109. Igel, C., Toussaint, M.: Recent results on no-free-lunch theorems for optimization. ArXiv EPrint arXiv:cs/0303032 (Computer Science, Neural and Evolutionary Computing) (accessed 2008-03-28) (2003), http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0303032
110. Ingber, L.: Adaptive simulated annealing (asa): Lessons learned. Control and Cybernetics 25(1), 33–54 (1996)
111. Joines, J.A., Houck, C.R.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. In: Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 579–584 (1994) doi:10.1109/ICEC.1994.349995
112. Jones, T.: Evolutionary algorithms, fitness landscapes and search. PhD thesis, The University of New Mexico (1995)
113. Kauffman, S.A.: Adaptation on rugged fitness landscapes. In: Stein, D.L. (ed.) Lectures in the Sciences of Complexity: The Proceedings of the 1988 Complex Systems Summer School. Santa Fe Institute Studies in the Sciences of Complexity, vol. Lecture I, pp. 527–618. Addison-Wesley, Reading (1988)
114. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
115. Kauffman, S.A., Levin, S.A.: Towards a general theory of adaptive walks on rugged landscapes. Journal of Theoretical Biology 128(1), 11–45 (1987)
116. Kearns, M.J., Mansour, Y., Ng, A.Y., Ron, D.: An experimental and theoretical comparison of model selection methods. In: COLT 1995: Proceedings of the eighth annual conference on Computational learning theory, pp. 21–30. ACM Press, New York (1995)
117. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
118. Kirschner, M., Gerhart, J.: Evolvability. Proceedings of the National Academy of Science of the USA (PNAS) 95(15), 8420–8427 (1998)
119. Kita, H., Sano, Y.: Genetic algorithms for optimization of noisy fitness functions and adaptation to changing environments. In: 2003 Joint Workshop of Hayashibara Foundation and 2003 Workshop on Statistical Mechanical Approach to Probabilistic Information Processing (SMAPIP) (2003)

120. Kolarov, K.: Landscape ruggedness in evolutionary algorithms. In: Proceedings of the IEEE Conference on Evolutionary Computation, pp. 19–24 (1997)
121. Köppen, M., Wolpert, D.H., Macready, W.G.: Remarks on a recent paper on the "no free lunch" theorems. IEEE Transactions on Evolutionary Computation 5(3), 295–296 (2001)
122. Landau, E.: Handbuch der Lehre von der Verteilung der Primzahlen. B. G. Teubner, Leipzig (1909); reprinted by Chelsea, New York (1953)
123. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms. Tech. Rep. 108, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich and Kanpur Genetic Algorithms Laboratory (KanGAL), Department of Mechanical Engineering, Indian Institute of Technology Kanpur (2001)
124. Lawrence, S., Giles, C.L.: Overfitting and neural networks: Conjugate gradient and backpropagation. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), vol. 1, pp. 1114–1119. IEEE Computer Society, Los Alamitos (2000)
125. Lee, J.Y.B., Wong, P.C.: The effect of function noise on gp efficiency. In: Progress in Evolutionary Computation, pp. 1–16 (1995)
126. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 51–58 (2006) doi:10.1145/1143997.1144005
127. Liepins, G.E., Vose, M.D.: Deceptiveness and genetic algorithm dynamics. In: Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA), pp. 36–50 (1991)
128. Ling, C.X.: Overfitting and generalization in learning discrete patterns. Neurocomputing 8(3), 341–347 (1995)
129. Lohmann, R.: Structure evolution and neural systems. In: Dynamic, Genetic, and Chaotic Programming: The Sixth-Generation, pp. 395–411. Wiley Interscience, Hoboken (1992)
130. Lohmann, R.: Structure evolution and incomplete induction. Biological Cybernetics 69(4), 319–326 (1993)
131. Luke, S., Panait, L.: A comparison of bloat control methods for genetic programming. Evolutionary Computation 14(3), 309–344 (2006)
132. Lush, J.L.: Progeny test and individual performance as indicators of an animal's breeding value. Journal of Dairy Science 18(1), 1–19 (1935)
133. Magurran, A.E.: Biological diversity. Current Biology Magazine 15, R116–R118 (2005)
134. Martin, W.N., Lienig, J., Cohoon, J.P.: Island (migration) models: Evolutionary algorithms based on punctuated equilibria. In: Handbook of Evolutionary Computation, ch. 6.3. Oxford University Press, Oxford (1997)
135. Mendes, R., Mohais, A.S.: Dynde: a differential evolution for dynamic optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 3, pp. 2808–2815 (2005)
136. Miller, B.L., Goldberg, D.E.: Genetic algorithms, tournament selection, and the effects of noise. IlliGAL Report 95006, Illinois Genetic Algorithms Laboratory, Department of General Engineering, University of Illinois (1995)

137. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation 4(2), 113–131 (1996)
138. Miller, B.L., Shaw, M.J.: Genetic algorithms with dynamic niche sharing for multimodal function optimization. IlliGAL Report 95010, Department of General Engineering, University of Illinois at Urbana-Champaign (1995)
139. Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In: Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, pp. 245–254 (1991)
140. Mitchell, T.M.: Generalization as search. In: Webber, B.L., Nilsson, N.J. (eds.) Readings in Artificial Intelligence, 2nd edn., pp. 517–542. Tioga Pub. Co. Press, Morgan Kaufmann Publishers, Elsevier Science & Technology Books (1981)
141. Mitchell, T.M.: Generalization as search. Artificial Intelligence 18(2), 203–226 (1982)
142. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 513–522. Springer, Heidelberg (1996)
143. Mori, N., Imanishi, S., Kita, H., Nishikawa, Y.: Adaptation to changing environments by means of the memory based thermodynamical genetic algorithm. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 299–306 (1997)
144. Mori, N., Kita, H., Nishikawa, Y.: Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 149–158. Springer, Heidelberg (1998)
145. Morrison, R.W.: Designing evolutionary algorithms for dynamic environments. PhD thesis, George Mason University, USA (2002)
146. Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Natural Computing 24(1), 143–144 (2004)
147. Morrison, R.W., De Jong, K.A.: A test problem generator for non-stationary environments. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 3, pp. 2047–2053 (1999) doi:10.1109/CEC.1999.785526
148. Morrison, R.W., De Jong, K.A.: Measurement of population diversity. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 1047–1074. Springer, Heidelberg (2002)
149. Mostaghim, S.: Multi-objective evolutionary algorithms: Data structures, convergence and, diversity. PhD thesis, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn, Deutschland, Germany (2004)
150. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detection for overlapping functions. Evolutionary Computation 7(4), 377–398 (1999)
151. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detection for overlapping functions. IlliGAL Report 99005, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana-Champaign (1999)
152. Muttil, N., Liong, S.-Y.: Superior exploration–exploitation balance in shuffled complex evolution. Journal of Hydraulic Engineering 130(12), 1202–1205 (2004)

153. Naudts, B., Verschoren, A.: Epistasis on finite and infinite spaces. In: Proceedings of the 8th International Conference on Systems Research, Informatics and Cybernetics, pp. 19–23 (1996)
154. Naudts, B., Verschoren, A.: Epistasis and deceptivity. Bulletin of the Belgian Mathematical Society 6(1), 147–154 (1999)
155. Newman, M.E.J., Engelhardt, R.: Effect of neutral selection on the evolution of molecular species. Proceedings of the Royal Society of London B (Biological Sciences) 256(1403), 1333–1338 (1998)
156. Oei, C.K., Goldberg, D.E., Chang, S.J.: Tournament selection, niching, and the preservation of diversity. IlliGAl Report 91011, Illinois Genetic Algorithms Laboratory (IlliGAL), Department of Computer Science, Department of General Engineering, University of Illinois at Urbana-Champaign (1991)
157. Olsen, A.L.: Penalty functions and the knapsack problem. In: Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 2, pp. 554–558 (1994)
158. Osman, I.H.: An introduction to metaheuristics. In: Lawrence, M., Wilsdon, C. (eds.) Operational Research Tutorial Papers, pp. 92–122. Stockton Press, Hampshire (1995); publication of the Operational Research Society, Birmingham, UK
159. Paenke, I., Branke, J., Jin, Y.: On the influence of phenotype plasticity on genotype diversity. In: First IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007), pp. 33–40 (2007)
160. Pan, G., Dou, Q., Liu, X.: Performance of two improved particle swarm optimization in dynamic optimization environments. In: ISDA 2006: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA 2006), vol. 2, pp. 1024–1028. IEEE Computer Society Press, Los Alamitos (2006)
161. Pan, H., Wang, L., Liu, B.: Particle swarm optimization for function optimization in noisy environment. Applied Mathematics and Computation 181(2), 908–919 (2006)
162. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: Boa: The bayesian optimization algorithm. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 525–532 (1999)
163. Phillips, P.C.: The language of gene interaction. Genetics 149(3), 1167–1171 (1998)
164. Pohlheim, H.: Geatbx introduction – evolutionary algorithms: Overview, methods and operators. Tech. rep., documentation for GEATbx version 3.7 (2005) (accessed, 2007-07-03), `http://www.GEATbx.com`
165. Purshouse, R.C.: On the evolutionary optimisation of many objectives. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield (2003)
166. Radcliffe, N.J.: Non-linear genetic representations. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp. 259–268. Elsevier, Amsterdam (1992)
167. Radcliffe, N.J.: The algebra of genetic algorithms. Annals of Mathematics and Artificial Intelligence 10(4) (1994) doi:10.1007/BF01531276
168. Radcliffe, N.J., Surry, P.D.: Fundamental limitations on search algorithms: Evolutionary computing in perspective. In: van Leeuwen, J. (ed.) Computer Science Today. LNCS, vol. 1000, pp. 275–291. Springer, Heidelberg (1995)

169. Rayward-Smith, V.J.: A unified approach to tabu search, simulated annealing and genetic algorithms. In: Rayward-Smith, V.J. (ed.) Applications of Modern Heuristic Methods – Proceedings of the UNICOM Seminar on Adaptive Computing and Information Processing, Brunel University Conference Centre, London, UK, vol. I, pp. 55–78. Alfred Waller Ltd / Nelson Thornes Ltd / Unicom Seminars Ltd (1994)
170. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart (1973)
171. Rechenberg, I.: Evolutionsstrategie 1994. Werkstatt Bionik und Evolutionstechnik, vol. 1. Frommann Holzboog (1994)
172. Reidys, C.M., Stadler, P.F.: Neutrality in fitness landscapes. Applied Mathematics and Computation 117(2–3), 321–350 (2001)
173. Richter, H.: Behavior of evolutionary algorithms in chaotically changing fitness landscapes. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 111–120. Springer, Heidelberg (2004)
174. Riedl, R.J.: A systems-analytical approach to macroevolutionary phenomena. Quarterly Review of Biology, 351–370 (1977)
175. Robbins, H., Monro, S.: A stochastic approximation method. Annals of Mathematical Statistics 22(3), 400–407 (1951)
176. Ronald, S.: Preventing diversity loss in a routing genetic algorithm with hash tagging. Complexity International 2 (1995) (accessed 2008-12-07), http://www.complexity.org.au/ci/vol02/sr_hash/
177. Ronald, S.: Genetic algorithms and permutation-encoded problems. diversity preservation and a study of multimodality. PhD thesis, University Of South Australia. Department of Computer and Information Science (1996)
178. Ronald, S.: Robust encodings in genetic algorithms: A survey of encoding issues. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 43–48 (1997) doi:10.1109/ICEC.1997.592265
179. Rosca, J.P.: An analysis of hierarchical genetic programming. Tech. Rep. TR566, The University of Rochester, Computer Science Department (1995)
180. Rosca, J.P., Ballard, D.H.: Causality in genetic programming. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 256–263 (1995)
181. Rosin, P.L., Fierens, F.: Improving neural network generalisation. In: Proceedings of the International Geoscience and Remote Sensing Symposium, Quantitative Remote Sensing for Science and Applications, IGARSS 1995, vol. 2, pp. 1255–1257. IEEE, Los Alamitos (1995)
182. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Physica-Verlag (2006) (1st edn., 2002)
183. Routledge, R.D.: Diversity indices: Which ones are admissible? Journal of Theoretical Biology 76, 503–515 (1979)
184. Rudnick, W.M.: Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks. PhD thesis, Oregon Graduate Institute of Science & Technology (1992)
185. Rudolph, G.: Self-adaptation and global convergence: A counter-example. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1, pp. 646–651 (1999)

186. Rudolph, G.: Self-adaptive mutations may lead to premature convergence. IEEE Transactions on Evolutionary Computation 5(4), 410–414 (2001)
187. Rudolph, G.: Self-adaptive mutations may lead to premature convergence. Tech. Rep. CI–73/99, Fachbereich Informatik, Universität Dortmund (2001)
188. Sano, Y., Kita, H.: Optimization of noisy fitness functions by means of genetic algorithms using history of search. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 571–580. Springer, Heidelberg (2000)
189. Sano, Y., Kita, H.: Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 360–365 (2002)
190. Sarle, W.: What is overfitting and how can i avoid it? Usenet FAQs: compaineural-nets FAQ 3: Generalization(3) (2007)
191. Sarle, W.S.: Stopped training and other remedies for overfitting. In: Proceedings of the 27th Symposium on the Interface: Computing Science and Statistics, pp. 352–360 (1995)
192. Schaffer, J.D., Eshelman, L.J., Offutt, D.: Spurious correlations and premature convergence in genetic algorithms. In: Proceedings of the First Workshop on Foundations of Genetic Algorithms (FOGA), pp. 102–112 (1990)
193. Sendhoff, B., Kreutz, M., von Seelen, W.: A condition for the genotype-phenotype mapping: Causality. In: Proceedings of the International Conference on Genetic Algorithms, ICGA, pp. 73–80 (1997)
194. Shackleton, M., Shipman, R., Ebner, M.: An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 493–500 (2000)
195. Shekel, J.: Test functions for multimodal search techniques. In: Proceedings of the Fifth Annual Princeton Conference on Information Science and Systems, pp. 354–359. Princeton University Press, Princeton (1971)
196. Shipman, R.: Genetic redundancy: Desirable or problematic for evolutionary adaptation? In: Proceedings of the 4^{th} International Conference on Artificial Neural Nets and Genetic Algorithms, pp. 1–11 (1999)
197. Shipman, R., Shackleton, M., Ebner, M., Watson, R.: Neutral search spaces for artificial evolution: a lesson from life. In: Bedau, M., McCaskill, J.S., Packard, N.H., Rasmussen, S., McCaskill, J., Packard, N. (eds.) Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life. The MIT Press, Bradford Books, Complex Adaptive Systems (2000)
198. Shipman, R., Shackleton, M., Harvey, I.: The use of neutral genotype-phenotype mappings for improved evolutionary search. BT Technology Journal 18(4), 103–111 (2000)
199. Siedlecki, W.W., Sklansky, J.: Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In: Proceedings of the third international conference on Genetic algorithms, pp. 141–150 (1989)
200. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 1305–1312 (2006)
201. Smith, A.E., Coit, D.W.: Penalty functions. In: Handbook of Evolutionary Computation, ch. 5.2. Oxford University Press, Oxford (1997)

202. Smith, M.: Neural Networks for Statistical Modeling. John Wiley & Sons, Inc. International Thomson Computer Press (1993/1996)
203. Smith, S.S.F.: Using multiple genetic operators to reduce premature convergence in genetic assembly planning. Computers in Industry 54(1), 35–49 (2004)
204. Smith, T., Husbands, P., Layzell, P., O'Shea, M.: Fitness landscapes and evolvability. Evolutionary Computation 10(1), 1–34 (2002)
205. Spatz, B.M., Rawlins, G.J.E. (eds.): Proceedings of the First Workshop on Foundations of Genetic Algorithms. Morgan Kaufmann Publishers, Inc., San Francisco (1990)
206. Spieth, C., Streichert, F., Speer, N., Zell, A.: Utilizing an island model for ea to preserve solution diversity for inferring gene regulatory networks. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1, pp. 146–151 (2004)
207. Stagge, P., Igel, C.: Structure optimization and isomorphisms. In: Theoretical Aspects of Evolutionary Computing, pp. 409–422. Springer, Heidelberg (2000)
208. Stewart, T.: Extrema selection: accelerated evolution on neutral networks. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, vol. 1 (2001)
209. Taguchi, G.: Introduction to Quality Engineering: Designing Quality into Products and Processes. Asian Productivity Organization / American Supplier Institute Inc. / Quality Resources / Productivity Press Inc., translation of Sekkeisha no tame no hinshitsu kanri (1986)
210. Taillard, É.D., Gambardella, L.M., Gendrau, M., Potvin, J.-Y.: Adaptive memory programming: A unified view of metaheuristics. European Journal of Operational Research 135(1), 1–16 (2001)
211. Tetko, I.V., Livingstone, D.J., Luik, A.I.: Neural network studies, 1. comparison of overfitting and overtraining. Journal of Chemical Information and Computer Sciences 35(5), 826–833 (1995)
212. Thierens, D.: On the scalability of simple genetic algorithms. Tech. Rep. UU-CS-1999-48, Department of Information and Computing Sciences, Utrecht University (1999)
213. Thierens, D., Goldberg, D.E., Pereira, Â.G.: Domino convergence, drift, and the temporal-salience structure of problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 535–540 (1998), doi:10.1109/ICEC.1998.700085
214. Toussaint, M., Igel, C.: Neutrality: A necessity for self-adaptation. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC, pp. 1354–1359 (2002)
215. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6), 317–325 (2003)
216. Trojanowski, K.: Evolutionary algorithms with redundant genetic material for non-stationary environments. PhD thesis, Instytut Podstaw Informatyki PAN, Institute of Computer Science, Warsaw, University of Technology, Poland (1994)
217. Tsutsui, S., Ghosh, A.: Genetic algorithms with a robust solution searching scheme. IEEE Transactions on Evolutionary Computation 1, 201–208 (1997)
218. Tsutsui, S., Ghosh, A., Fujimoto, Y.: A robust solution searching scheme in genetic search. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 543–552. Springer, Heidelberg (1996)

219. Ursem, R.K.: Models for evolutionary algorithms and their applications in system identification and control optimization. PhD thesis, Department of Computer Science, University of Aarhus, Denmark (2003)
220. Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K.: A local search template. In: Proceedings of the International Conference on Parallel Problem Solving from Nature, PPSN, pp. 67–76 (1992)
221. Vaessens, R.J.M., Aarts, E.H.L., Lenstra, J.K.: A local search template. Computers and Operations Research 25(11), 969–979 (1998)
222. van Nimwegen, E., Crutchfield, J.P.: Optimizing epochal evolutionary search: Population-size dependent theory. Machine Learning 45(1), 77–114 (2001)
223. van Nimwegen, E., Crutchfield, J.P., Huynen, M.: Neutral evolution of mutational robustness. Proceedings of the National Academy of Science of the United States of Americs (PNAS) – Evolution 96(17), 9716–9720 (1999)
224. van Nimwegen, E., Crutchfield, J.P., Mitchell, M.: Statistical dynamics of the royal road genetic algorithm. Theoretical Computer Science 229(1–2), 41–102 (1999)
225. Wagner, A.: Robustness and Evolvability in Living Systems. Princeton Studies in Complexity. Princeton University Press, Princeton (2005)
226. Wagner, A.: Robustness, evolvability, and neutrality. FEBS Lett 579(8), 1772–1778 (2005)
227. Wagner, G.P., Altenberg, L.: Complex adaptations and the evolution of evolvability. Evolution 50(3), 967–976 (1996)
228. Watanabe, S.: Knowing and Guessing: A Quantitative Study of Inference and Information. John Wiley & Sons, Chichester (1969)
229. Weicker, K.: Evolutionäre Algorithmen. Leitfäden der Informatik, B. G. Teubner GmbH (2002)
230. Weicker, K., Weicker, N.: Burden and benefits of redundancy. In: Sixth Workshop on Foundations of Genetic Algorithms (FOGA), pp. 313–333. Morgan Kaufmann, San Francisco (2000)
231. Weise, T., Zapf, M., Geihs, K.: Rule-based Genetic Programming. In: Proceedings of BIONETICS 2007, 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (2007)
232. Weise, T., Niemczyk, S., Skubch, H., Reichle, R., Geihs, K.: A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 795–802 (2008)
233. Whitley, L.D., Gordon, V.S., Mathias, K.E.: Lamarckian evolution, the baldwin effect and function optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 6–15. Springer, Heidelberg (1994)
234. Wiesmann, D., Hammel, U., Bäck, T.: Robust design of multilayer optical coatings by means of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 2, 162–167 (1998)
235. Wiesmann, D., Hammel, U., Bäck, T.: Robust design of multilayer optical coatings by means of evolutionary strategies. Sonderforschungsbereich (sfb) 531, Universität Dortmund (1998)
236. Wilke, C.O.: Evolutionary dynamics in time-dependent environments. PhD thesis, Fakultät für Physik und Astronomie, Ruhr-Universität Bochum (1999)
237. Wilke, C.O.: Adaptive evolution on neutral networks. Bulletin of Mathematical Biology 63(4), 715–730 (2001)

238. Wilke, D.N., Kok, S., Groenwold, A.A.: Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. International Journal for Numerical Methods in Engineering 70(8), 962–984 (2007)
239. Williams, G.C.: Pleiotropy, natural selection, and the evolution of senescence. Evolution 11(4), 398–411 (1957)
240. Winter, P.C., Hickey, G.I., Fletcher, H.L.: Instant Notes in Genetics, 3rd edn. Springer, New York (2006) (1st edn. 1998, 2nd edn. 2002)
241. Wolpert, D.H., Macready, W.G.: No free lunch theorems for search. Tech. Rep. SFI-TR-95-02-010, The Santa Fe Institute (1995)
242. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
243. Wu, N.: Differential evolution for optimisation in dynamic environments. Tech. rep., School of Computer Science and Information Technology, RMIT University (2006)
244. Yang, S., Ong, Y.S., Jin, Y.: Evolutionary Computation in Dynamic and Uncertain Environments. Studies in Computational Intelligence, vol. 51(XXIII). Springer, Heidelberg (2007)
245. Zakian, V.: New formulation for the method of inequalities. Proceedings of the Institution of Electrical Engineers 126(6), 579–584 (1979)
246. Žilinskas, A.: Algorithm as 133: Optimization of one-dimensional multimodal functions. Applied Statistics 27(3), 367–375 (1978)
247. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2001)
248. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems. Proceedings of the EUROGEN 2001 Conference, pp. 95–100 (2001)