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* [663, 652] (see Section 10.6 on page 256), 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. Increasing the proportion of exploration operations may also reduce the chance of premature convergence.

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 (see Section 2.3.4). The Strength Pareto Algorithms, which are widely accepted to be highly efficient, use another idea: they adapt the number of individuals that one solution candidate *dominates* as density measure [2329, 2332]. One very simple method aiming for convergence prevention is introduced in Section 2.4.8. 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) [1776, 1777, 1778].

1.4.3 Ruggedness and Weak Causality

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[49], 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. For short, one could say ruggedness is multi-modality plus steep ascends and descends in the fitness landscape. Examples of rugged landscapes are Kauffman's NK fitness landscape (see Section 21.2.1), the p-Spin model discussed in Section 21.2.2, Bergman and Feldman's jagged fitness landscape [182], and the sketch in Fig. 1.19.d on page 57.

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 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

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

properties in order to find out whether they can be improved any further[50]. Normally, such *exploitive* modifications should also lead to small changes in the objective values and hence, in the fitness of the solution candidate.

Definition 1.47 (Strong Causality). *Strong causality* (locality) means that small changes in the properties of an object also lead to small changes in its behavior [1713, 1714, 1759].

This principle (proposed by Rechenberg [1713, 1714]) 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 [1713, 597], structure evolution [1303, 1302], Genetic Programming [1758, 1759, 1007, 597], genotype-phenotype mappings [1854], search operators [597], and evolutionary algorithms in general [1955, 1765, 597].

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 degeneration of the performance of the optimizer [1168]. This does not necessarily mean that it is impossible to find good solutions, but it may take very long to do so.

Fitness Landscape Measures

As measures for the ruggedness of a fitness landscape (or their general difficulty), many different metrics have been proposed. Wedge and Kell [2164] and Altenberg [45] provide nice lists of them in their work[51], which we summarize here:

- Weinberger [2169] introduced the autocorrelation function and the correlation length of random walks.
- The correlation of the search operators was used by Manderick et al. [1354] in conjunction with the autocorrelation.
- Jones and Forrest [1070, 1069] proposed the fitness distance correlation (FDC), the correlation of the fitness of an individual and its distance to the global optimum. This measure has been extended by researchers such as Clergue et al. [416, 2103].
- The probability that search operations create offspring fitter than their parents, as defined by Rechenberg [1713] and Beyer [196] (and called evolvability by Altenberg [42]), will be discussed in Section 1.4.5 on page 65 in depth.
- Simulation dynamics have been researched by Altenberg [42] and Grefenstette [855].
- Another interesting metric is the fitness variance of formae (Radcliffe and Surry [1695]) and schemas (Reeves and Wright [1717]).
- The error threshold method from theoretical biology [625, 1552] has been adopted Ochoa et al. [1557] for evolutionary algorithms. It is the "critical mutation rate beyond which structures obtained by the evolutionary process are destroyed by mutation more frequently than selection can reproduce them" [1557].

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

[51] Especially the one of Wedge and Kell [2164] is beautiful and far more detailed than this summary here.

- The negative slope coefficient (NSC) by Vanneschi et al. [2104, 2105] may be considered as an extension of Altenberg's evolvability measure.
- Davidor [489] uses the epistatic variance as a measure of utility of a certain representation in genetic algorithms. We discuss the issue of epistasis in Section 1.4.6.
- The genotype-fitness correlation (GFC) of Wedge and Kell [2164] is a new measure for ruggedness in fitness landscape and has been shown to be a good guide for determining optimal population sizes in Genetic Programming.

Autocorrelation and Correlation Length

As example, let us take a look at the autocorrelation function as well as the correlation length of random walks [2169]. Here we borrow its definition from Verel et al. [2114]:

Definition 1.48 (Autocorrelation Function). Given a random walk $(x_i, x_{i+1}, \dots)$, the autocorrelation function ρ of an objective function f is the autocorrelation function of the time series $(f(x_i), f(x_{i+1}), \dots)$.

$$\rho(k, f) = \frac{E[f(x_i)\, f(x_{i+k})] - E[f(x_i)]\, E[f(x_{i+k})]}{D^2[f(x_i)]} \tag{1.41}$$

where $E[f(x_i)]$ and $D^2[f(x_i)]$ are the expected value and the variance of $f(x_i)$.

The correlation length $\tau = -\frac{1}{\log \rho(1,f)}$ measures how the autocorrelation function decreases and summarizes the ruggedness of the fitness landscape: the larger the correlation length, the lower the total variation of the landscape. From the works of Kinnear, Jr. [1141] and Lipsitch [1293] from 18, however, we also know that correlation measures do not always represent the hardness of a problem landscape full.

Countermeasures

To the knowledge of the author, 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 reduce the influence of ruggedness, but only up to a certain degree. Utilizing Lamarckian evolution [522, 2215] or the Baldwin effect [123, 929, 930, 2215], i. e., incorporating a local search into the optimization process, may further help to smoothen out the fitness landscape [864] (see Section 15.2 and Section 15.3, respectively).

Weak causality is often a home-made problem because it results to some extent from the choice of the solution representation and search operations. We pointed out that exploration operations are important for lowering the risk of premature convergence. Exploitation operators are as same as important for refining solutions to a certain degree. In order to apply optimization algorithms in an efficient manner, it is necessary to find representations which allow for iterative modifications with bounded influence on the objective values, i. e., exploitation. In Section 1.5.2, we present some further rules-of-thumb for search space and operation design.

1.4.4 Deceptiveness

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. 1.19.e.

The term deceptiveness is mainly used in the genetic algorithm[52] 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 [190, 821, 1285]. Examples for deceptiveness are the ND fitness landscapes outlined in Section 21.2.3, trap functions (see Section 21.2.3), and the fully deceptive problems given by Goldberg et al. [825, 541].

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 [2159, 1433, 1434, 2034] and has been subsumed under the No Free Lunch Theorem which wewill discuss in Section 1.4.10.

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 1.4.6) are, maybe, the only approaches which can provide at least a small chance of finding good solutions.

1.4.5 Neutrality and Redundancy

The Problem: Neutrality

Definition 1.49 (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 [1718, 149].

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. 1.19.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 applied to a specific genotype [149]. 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*.

$$\forall g_1 \in \mathbb{G} \Rightarrow \nu(g_1) = \frac{|\{g_2 : P(g_2 = Op(g_1)) > 0 \wedge F(\text{gpm}(g_2)) = F(\text{gpm}(g_1))\}|}{|\{g_2 : P(g_2 = Op(g_1)) > 0\}|} \tag{1.42}$$

$$\forall G \subseteq \mathbb{G} \Rightarrow \nu(G) = \frac{1}{|G|} \sum_{g \in G} \nu(g) \tag{1.43}$$

[52] We are going to discuss genetic algorithms in Chapter 3 on page 141 and the Schema Theorem in Section 3.6 on page 150.

Evolvability

Another metaphor in global optimization borrowed from biological systems is evolvability[53] [500]. Wagner [2132, 2133] points out that this word has two uses in biology: According to Kirschner and Gerhart [1144], a biological system is evolvable if it is able to generate heritable, selectable phenotypic variations. Such properties can then be spread by natural selection and changed during the course of evolution. 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 [1732], Altenberg [43], Wagner and Altenberg [2134], Bonner [247], and Conrad [439], amongst others. The idea of evolvability can be adopted for global optimization as follows:

Definition 1.50 (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* [1713, 196], 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 [45, 42].

Neutrality: Problematic and Beneficial

The link between evolvability and neutrality has been discussed by many researchers [2300, 2133]. 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. 1.19.f on page 57. The work of Beaudoin et al. [161] on the ND fitness landscapes[54] 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 in biology have no selective influence [732, 980] and that the transformation from genotypes to phenotypes is a many-to-one mapping. Wagner [2133] 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 [2050] even go as far as declaring it a necessity for self-adaptation.

The theory of *punctuated equilibria*[55], in biology introduced by Eldredge and Gould [630, 629], states that species experience long periods of evolutionary inactivity which are interrupted by sudden, localized, and rapid phenotypic evolutions [118].[56] It is assumed that the populations explore neutral layers[57] during the time of stasis until, suddenly, a relevant change in a genotype leads to a better adapted phenotype [2098] which then reproduces quickly. Similar phenomena can be observed/are utilized in EAs [426, 1365].

"Uh?", you may think, "How does this fit together?" 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. [1913] have used illustrative examples similar to Figure 1.21 showing that a certain amount of neutral reproductions can foster the progress of optimization. In Fig. 1.21.a, basically the same scenario of premature convergence as in Fig. 1.20.a on page 59 is depicted. The optimizer is drawn to a local optimum from which it cannot escape anymore. Fig. 1.21.b shows that a little shot of neutrality

[53] http://en.wikipedia.org/wiki/Evolvability [accessed 2007-07-03]
[54] See Section 21.2.3 on page 333 for a detailed elaboration on the ND fitness landscape.
[55] http://en.wikipedia.org/wiki/Punctuated_equilibrium [accessed 2008-07-01]
[56] A very similar idea is utilized in the Extremal Optimization method discussed in Chapter 13.
[57] Or neutral networks, as discussed in Section 1.4.5.

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. 1.21.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. 1.19.f on page 57 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.

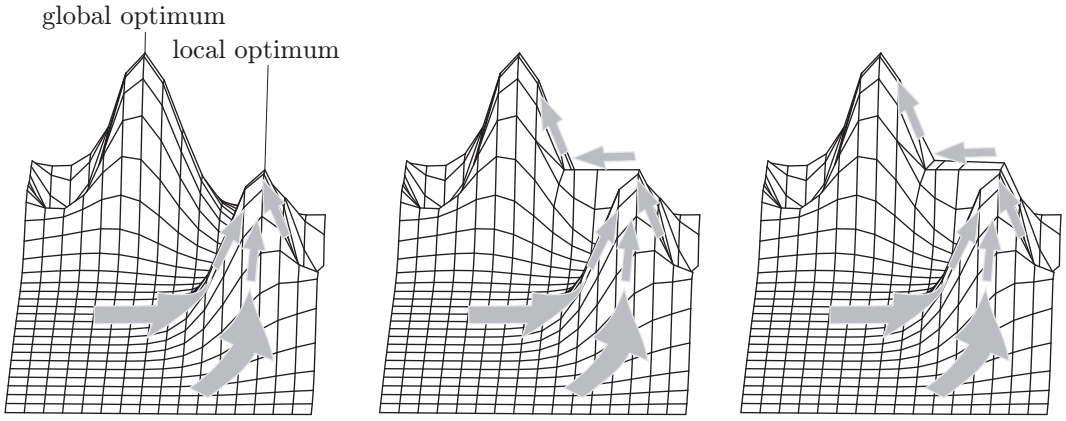

Fig. 1.21.a: Premature Convergence

Fig. 1.21.b: Small Neutral Bridge

Fig. 1.21.c: Wide Neutral Bridge

Figure 1.21: Possible positive influence of neutrality.

Recently, the idea of utilizing the processes of molecular[58] and evolutionary[59] biology as complement to Darwinian evolution for optimization gains interest [144]. Scientists like Hu and Banzhaf [967, 968] have begun to study the application of metrics such as the evolution rate of gene sequences [2281, 2257] to evolutionary algorithms. Here, the degree of neutrality (synonymous vs. non-synonymous changes) seems to play an important role.

Examples for neutrality in fitness landscapes are the ND family (see Section 21.2.3), the NKp and NKq models (discussed in Section 21.2.1), and the Royal Road (see Section 21.2.4). Another common instance of neutrality is *bloat* in Genetic Programming, which is outlined in Section 4.10.3 on page 224.

Neutral Networks

From the idea of neutral bridges between different parts of the search space as sketched by Smith et al. [1913], we can derive the concept of neutral networks.

Definition 1.51 (Neutral Network). Neutral networks are equivalence classes K of elements of the search space $\mathbb{G}$ which map to elements of the problem space $\mathbb{X}$ with the same objective values and are connected by chains of applications of the search operators Op [149].

$$\forall g_1, g_2 \in \mathbb{G} : g_1 \in K(g_2) \subseteq \mathbb{G} \Leftrightarrow \exists k \in \mathbb{N}_0 : P\big(g_2 = Op^k(g_1)\big) > 0 \wedge F(\text{gpm}(g_1)) = F(\text{gpm}(g_2)) \tag{1.44}$$

Barnett [149] states that a neutral network has the *constant innovation property* if

[58] http://en.wikipedia.org/wiki/Molecular_biology [accessed 2008-07-20]
[59] http://en.wikipedia.org/wiki/Evolutionary_biology [accessed 2008-07-20]

1. the rate of discovery of innovations keeps constant for a reasonably large amount of applications of the search operations [981], and
2. if this rate is comparable with that of an unconstrained random walk.

Networks with this property may prove very helpful if they connect the optima in the fitness landscape. Stewart [1962] utilizes neutral networks and the idea of punctuated equilibria in his *extrema selection*, a genetic algorithm variant that focuses on exploring individuals which are far away from the centroid of the set of currently investigated solution candidates (but have still good objective values). Then again, Barnett [148] showed that populations in genetic algorithm tend to dwell in neutral networks of high dimensions of neutrality regardless of their objective values, which (obviously) cannot be considered advantageous.

The convergence on neutral networks has furthermore been studied by Bornberg-Bauer and Chan [251], van Nimwegen et al. [2097, 2096], and Wilke [2225]. Their results show that the topology of neutral networks strongly determines the distribution of genotypes on them. Generally, the genotypes are "drawn" to the solutions with the highest degree of neutrality ν on the neutral network Beaudoin et al. [161].

Redundancy: Problematic and Beneficial

Definition 1.52 (Redundancy). 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.

$$\exists g_1, g_2 : g_1 \neq g_2 \wedge \text{gpm}(g_1) = \text{gpm}(g_2) \tag{1.45}$$

The role of redundancy in the genome is as controversial as that of neutrality [2168]. There exist many accounts of its positive influence on the optimization process. Shipman et al. [1871, 1856], for instance, tried to mimic desirable evolutionary properties of RNA folding [980]. 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 [1099]. 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 [1871, 1856]. Possibly converse effects like epistasis (see Section 1.4.6) 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 [1871]. One example for beneficial redundancy is the extradimensional bypass idea discussed in Section 1.5.2.

The experiments of Shipman et al. [1872, 1870] additionally indicate that neutrality in the genotype-phenotype mapping can have positive effects. In the Cartesian Genetic Programming method, neutrality is explicitly introduced in order to increase the evolvability (see Section 4.7.4 on page 201) [2110, 2297].

Yet, Rothlauf [1765] and Shackleton et al. [1856] 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`. Another example for this issue is given in Fig. 1.31.b on page 86.

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 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.

Needle-In-A-Haystack

One of the worst cases of fitness landscapes is the *needle-in-a-haystack* (NIAH) problem sketched in Fig. 1.19.g on page 57, where the optimum occurs as isolated spike in a plane. In other words, small instances of extreme ruggedness combine with a general lack of information in the fitness landscape. Such problems are extremely hard to solve and the optimization processes often will converge prematurely or take very long to find the global optimum. An example for such fitness landscapes is the all-or-nothing property often inherent to Genetic Programming of algorithms [2058], as discussed in Section 4.10.2 on page 223.

1.4.6 Epistasis

Introduction

In biology, *epistasis*[60] is defined as a form of interaction between different genes [1640]. The term was coined by Bateson [157] 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 [677]. According to Lush [1335], 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*[61], which means that a single gene influences multiple phenotypic traits [2227]. In the area of global optimization, such fine-grained distinctions are usually not made and the two terms are often used more or less synonymously.

Definition 1.53 (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. [491, 44, 1503]

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 (see Section 17.4.1) [491]. A problem is maximally epistatic when no proper subset of genes is independent of any other gene [1924, 1503]. Examples of problems with a high degree of epistasis are Kauffman's NK fitness landscape [1098, 1100] (Section 21.2.1), the p-Spin model [48] (Section 21.2.2), and the tunable model of Weise et al. [2185] (Section 21.2.7).

The Problem

As sketched in Figure 1.22, epistasis has a strong influence on many of the previously discussed problematic features. If one gene can "turn off" or affect the expression of other

[60] http://en.wikipedia.org/wiki/Epistasis [accessed 2008-05-31]
[61] http://en.wikipedia.org/wiki/Pleiotropy [accessed 2008-03-02]

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. It also becomes harder to define search operations with exploitive character. Moreover, 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 the genome $\mathbb{G}$ and the genotype-phenotype mapping are defined. It should be avoided where possible.

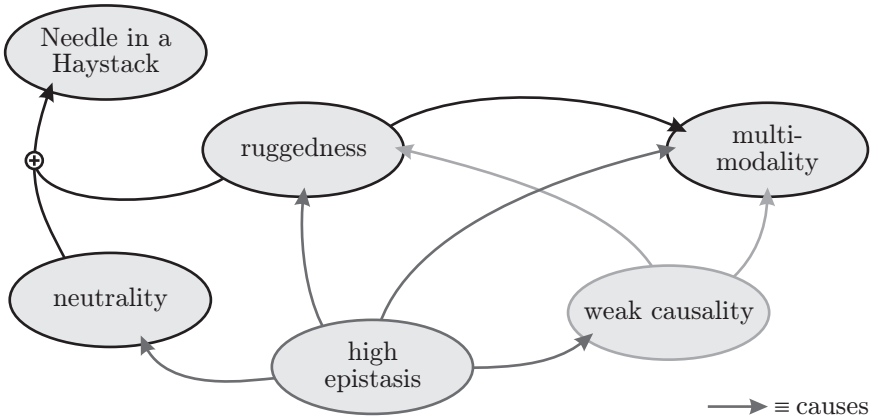

Figure 1.22: The influence of epistasis on the fitness landscape.

Generally, epistasis and conflicting objectives in multi-objective optimization should be distinguished from each other. Epistasis as well as pleiotropy is a property of the influence of the editable 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 both are subject to maximization or minimization. Nevertheless, if the solution candidates x and the genotypes are simple real numbers and the genotype-phenotype mapping is an identity mapping, neither epistatic nor pleiotropic effects can occur.

Naudts and Verschoren [1504] 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. [167].

Countermeasures

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 [2181]. Some general rules for search space design are outlined in Section 1.5.2.

Linkage Learning

According to Winter et al. [2242], *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 $g \in \mathbb{G}$ is trivial. Instead, we are interested in alleles of different genes which have a joint effect on the fitness [1486, 1485].

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[62] from being destroyed by the search operations (such as crossover in genetic algorithms), for instance. Finding approaches for *linkage learning* has become an especially popular discipline in the area of evolutionary algorithms with binary [896, 1486, 1647] and real [546] genomes. Two important methods from this area are the *messy GA* (mGA, see Section 3.7) by Goldberg et al. [825] and the *Bayesian Optimization Algorithm* (BOA) [1633, 333]. Module acquisition [66] may be considered as such an effort.

1.4.7 Noise and Robustness

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 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

1. the usage of global optimization for building classifiers (for example for predicting buying behavior using data gathered in a customer survey for training),
2. the usage of simulations for determining the objective values in Genetic Programming (here, the simulated scenarios correspond to training cases), and
3. 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 *(ii)* inaccuracies in the process of realizing the solutions and *(iii)* environmentally induced perturbations during the applications of the products.

This issue can be illustrated by 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

[62] See Section 3.6.5 for information on the Building Block Hypothesis.

in scenarios different from those simulated *(iii)* and should instead be applicable in all driving situations likely to occur.

The effects of noise in optimization have been studied by various researchers; Miller and Goldberg [1416, 1415], Lee and Wong [1268], and Gurin and Rastrigin [870] 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

1. genetic algorithms [685, 2062, 2060, 1799, 1800, 1146],
2. Evolution Strategies [195, 100, 881], and
3. Particle Swarm Optimization [1606, 884] approaches.

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. There is no process in the world that is 100% accurate and the optimized parameters, designs, and plans have to tolerate a certain degree of imprecision.

Definition 1.54 (Robustness). A system in engineering or biology is*robust* if it is able to function properly in the face of genetic or environmental perturbations [2132].

Therefore, a local optimum (or even a non-optimal element) for which slight disturbances only lead to gentle performance degenerations is usually favored over a global optimum located in a highly rugged area of the fitness landscape [276]. 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. Figure 1.23 illustrates the issue of local optima which are robust vs. global optima which are not. More examples from the real world are:

1. 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 [2062].
2. Wiesmann et al. [2218, 2217] 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.
3. 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. [886] 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.
4. Tsutsui et al. [2062, 2060] 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.

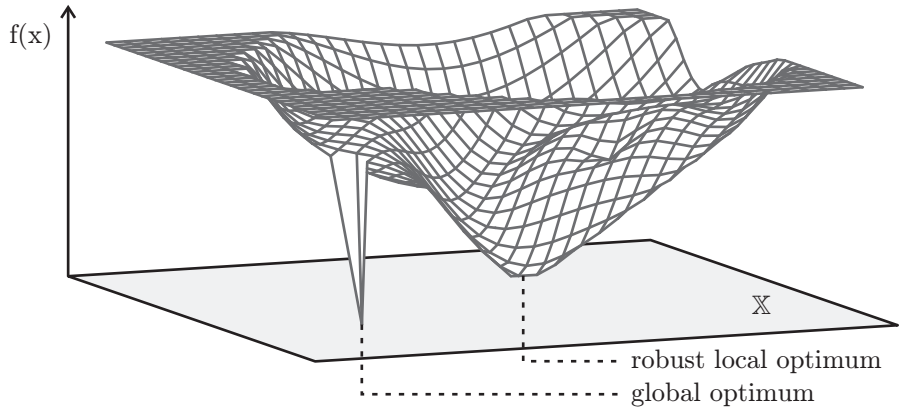

Figure 1.23: A robust local optimum vs. a "unstable" global optimum.

Countermeasures

For the special case where the phenome is a real vector space ($\mathbb{X} \subseteq \mathbb{R}^n$), several approaches for dealing with the need for robustness have been developed. Inspired by Taguchi methods[63] [1995], possible disturbances are represented by a vector $\boldsymbol{\delta} = (\delta_1, \delta_2, .., \delta_n)^T, \delta_i \in \mathbb{R}$ in the method suggested by Greiner [859, 860]. If the distributions and influences of the δ_i are known, the objective function $f(\mathbf{x}) : \mathbf{x} \in \mathbb{X}$ can be rewritten as $\tilde{f}(\mathbf{x}, \boldsymbol{\delta})$ [2218]. 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 $y^\star$ of the objective function f is known, Equation 1.46 can be minimized. This approach is also used in the work of Wiesmann et al. [2217, 2218] and basically turns the optimization algorithm into something like a maximum likelihood estimator (see Section 28.7.2 and Equation 28.252 on page 502).

$$f'(\mathbf{x}) = \frac{1}{t}\sum_{i=1}^{t}\left(y^\star - \tilde{f}(\mathbf{x}, \boldsymbol{\delta}_i)\right)^2 \tag{1.46}$$

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.

1.4.8 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.

Overfitting

The Problem

Definition 1.55 (Overfitting). Overfitting[64] 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 [1805, 1905, 785, 564].

[63] http://en.wikipedia.org/wiki/Taguchi_methods [accessed 2008-07-19]
[64] http://en.wikipedia.org/wiki/Overfitting [accessed 2007-07-03]

A model (solution candidate) $m \in \mathbb{X}$ optimized based on a finite set of training data is considered to be overfitted if a less complicated, alternative model $m' \in \mathbb{X}$ 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[65] [2019, 1291, 1265, 1806, 1761]. 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[66] of the degree $n-1$ that fits to each such training data and goes through all its points.[67] 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 Figure 1.24, we have sketched this problem. The function $f_1(x) = x$ shown in Fig. 1.24.b has been sampled three times, as sketched in Fig. 1.24.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. 1.24.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.

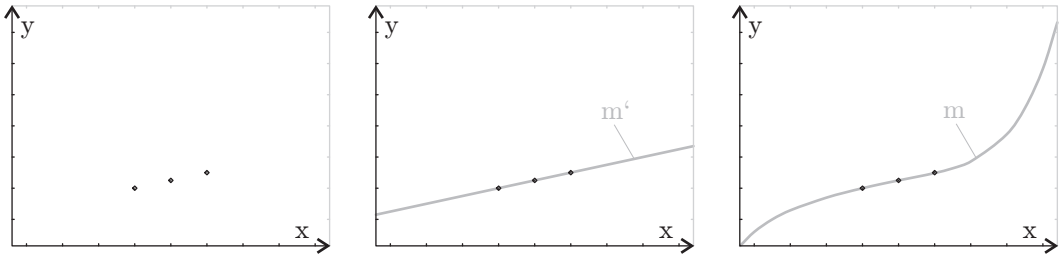

Fig. 1.24.a: Three sample points of f_1.

Fig. 1.24.b: $m' \equiv f_1(x) = x$.

Fig. 1.24.c: $m \equiv f_2(x)$.

Figure 1.24: 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 Figure 1.25 we have sketched how such noise may lead to overfitted results. Fig. 1.25.a illustrates a simple physical process obeying some quadratic equation. This process has been measured using some technical equipment and the 100 noisy samples depicted in Fig. 1.25.b has been obtained. Fig. 1.25.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

[65] We will discuss overfitting in conjunction with Genetic Programming-based symbolic regression in Section 23.1 on page 397.

[66] http://en.wikipedia.org/wiki/Polynomial [accessed 2007-07-03]

[67] http://en.wikipedia.org/wiki/Polynomial_interpolation [accessed 2008-03-01]

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.

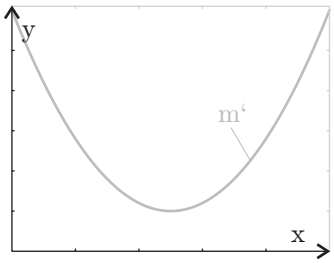

Fig. 1.25.a: The original physical process.

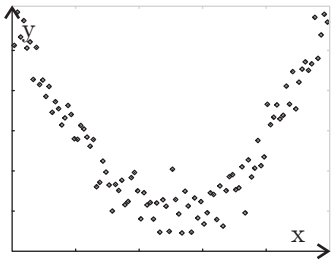

Fig. 1.25.b: The measurement/training data.

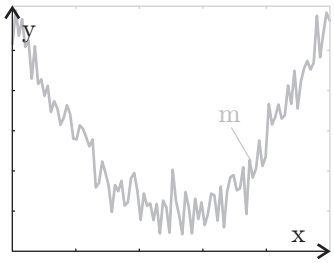

Fig. 1.25.c: The overfitted result.

Figure 1.25: Fitting noise.

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

Definition 1.56 (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.

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 [564, 1108].

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 each solution candidate. The resulting objective values then 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. They can be made even more comparable if the objective functions are always normalized into some fixed interval, say $[0, 1]$.

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 [1805]. 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.

Dividing Data into Training and Test Sets 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.

Oversimplification

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 Figure 1.26: 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, which then may lead to oversimplified solutions. Another possible reason for oversimplification 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 adapt them completely even if the training data perfectly represents the sampled process. A third possible cause is that a problem space could have been chosen which does not include the correct solution.

Fig. 1.26.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. 1.26.b. By doing so, some vital characteristics of the function are lost. Fig. 1.26.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. 1.26.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. 1.26.c, regardless of how many training samples are used.

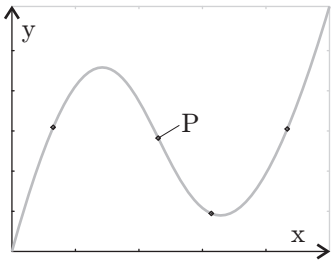

Fig. 1.26.a: The "real system" and the points describing it.

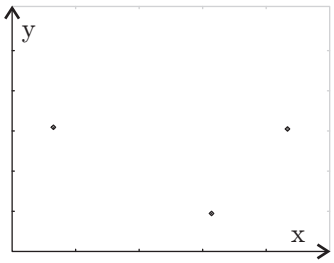

Fig. 1.26.b: The sampled training data.

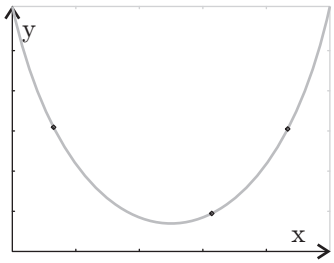

Fig. 1.26.c: The oversimplified result.

Figure 1.26: Oversimplification.

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, in order to 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.

1.4.9 Dynamically Changing Fitness Landscape

It should also be mentioned that there exist problems with dynamically changing fitness landscapes [282, 1465, 1729, 277, 278]. 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.

Problems with dynamic characteristics can, for example, be tackled with special forms [2280] of

1. evolutionary algorithms [2053, 2224, 279, 280, 1463, 1464, 82],
2. genetic algorithms [817, 1457, 1458, 1459, 1146],
3. Particle Swarm Optimization [343, 344, 1280, 1605, 211],
4. Differential Evolution [1391, 2266], and
5. Ant Colony Optimization [868, 869]

The moving peaks benchmarks by Branke [277, 278] and Morrison and De Jong [1465] are good examples for dynamically changing fitness landscapes. You can find them discussed in Section 21.1.3 on page 328.

1.4.10 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 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 [2244, 2245] in their *No Free Lunch Theorems*[68] (NFL) for search and optimization algorithms.

Initial Definitions

Wolpert and Macready [2245] consider single-objective optimization and define an optimization problem $\phi(g) \equiv f(\text{gpm}(g))$ as a mapping of a search space $\mathbb{G}$ to the objective space $\mathbb{Y}$.[69] Since this definition subsumes the problem space and the genotype-phenotype mapping, only skipping the possible search operations, it is very similar to our Definition 1.34 on page 46. They further call a time-ordered set d_m of m distinct visited points in $\mathbb{G} \times \mathbb{Y}$ a "sample" of size m and write $d_m \equiv \{(d_m^g(1), d_m^y(1)), (d_m^g(2), d_m^y(2)), \ldots, (d_m^g(m), d_m^y(m))\}$. $d_m^g(i)$ is the genotype and $d_m^y(i)$ the corresponding objective value visited at time step i. Then, the set $D_m = (\mathbb{G} \times \mathbb{Y})^m$ is the space of all possible samples of length m and $D = \cup_{m \geq 0} D_m$ is the set of all samples of arbitrary size.

An optimization algorithm a can now be considered to be a mapping of the previously visited points in the search space (i. e., a sample) to the next point to be visited. Formally, this means $a : D \mapsto \mathbb{G}$. Without loss of generality, Wolpert and Macready [2245] only regard unique visits and thus define $a : d \in D \mapsto g : g \notin d$.

Performance measures Ψ can be defined independently from the optimization algorithms only based on the values of the objective function visited in the samples d_m. If the objective function is subject to minimization, $\Psi(d_m^y) = \min\{d_m^y : i = 1..m\}$ would be the appropriate measure.

Often, only parts of the optimization problem ϕ are known. If the minima of the objective function f were already identified beforehand, for instance, its optimization would be useless. Since the behavior in wide areas of ϕ is not obvious, it makes sense to define a probability $P(\phi)$ that we are actually dealing with ϕ and no other problem. Wolpert and Macready [2245] use the handy example of the travelling salesman problem in order to illustrate this issue. Each distinct TSP produces a different structure of ϕ. Yet, we would use the same optimization algorithm a for all problems of this class without knowing the exact shape of ϕ. This corresponds to the assumption that there is a set of very similar optimization problems which we may encounter here although their exact structure is not known. We act as if there was a probability distribution over all possible problems which is non-zero for the TSP-alike ones and zero for all others.

The Theorem

The performance of an algorithm a iterated m times on an optimization problem ϕ can then be defined as $P(d_m^y | \phi, m, a)$, i. e., the conditional probability of finding a particular sample d_m^y. Notice that this measure is very similar to the value of the problem landscape $\Phi(x, \tau)$ introduced in Definition 1.38 on page 48 which is the cumulative probability that the optimizer has visited the element $x \in \mathbb{X}$ until (inclusively) the τ^{th} evaluation of the objective function(s).

Wolpert and Macready [2245] prove that the sum of such probabilities over all possible optimization problems ϕ is always identical for all optimization algorithms. For two optimizers a_1 and a_2, this means that

[68] http://en.wikipedia.org/wiki/No_free_lunch_in_search_and_optimization [accessed 2008-03-28]

[69] Notice that we have partly utilized our own notations here in order to be consistent throughout the book.

$$\sum_{\forall\phi} P(d_m^y\,|\phi, m, a_1\,) = \sum_{\forall\phi} P(d_m^y\,|\phi, m, a_2\,) \tag{1.47}$$

Hence, the average over all ϕ of $P(d_m^y\,|\phi, m, a\,)$ is independent of a.

Implications

From this theorem, we can immediately follow that, in order to outperform a_1 in one optimization problem, a_2 will necessarily perform worse in another. Figure 1.27 visualizes this issue. It shows that general optimization approaches like evolutionary algorithms can solve a variety of problem classes with reasonable performance. In this figure, we have chosen a performance measure Φ subject to maximization, i. e., the higher its values, the faster will the problem be solved. 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[70] 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".

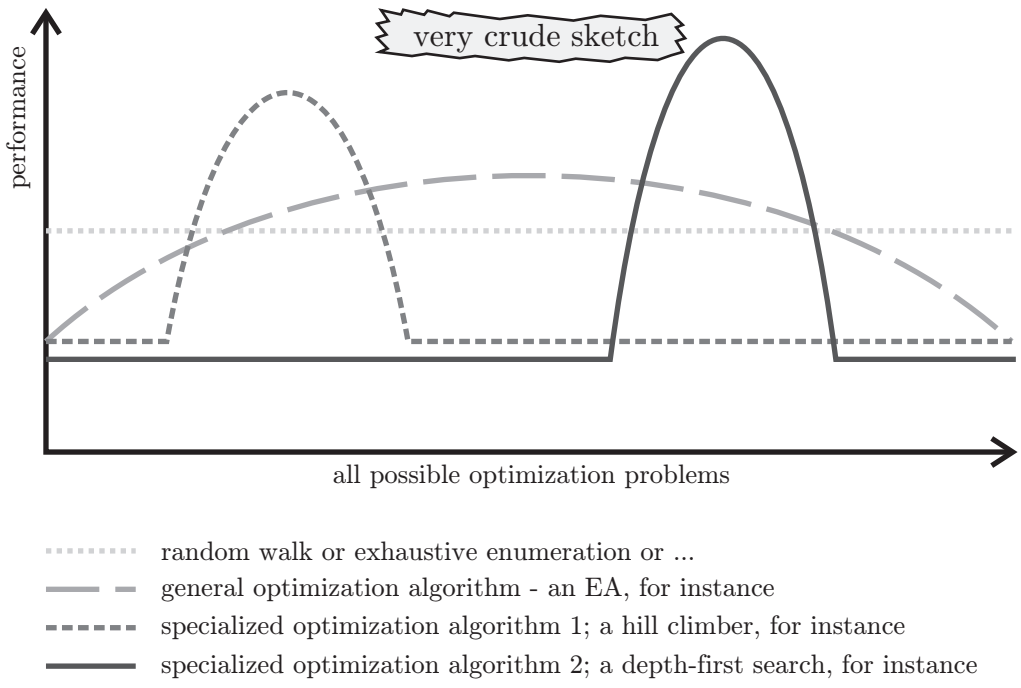

Figure 1.27: A visualization of the No Free Lunch Theorem.

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 1.4.4). 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, see Section 21.2.6, for example.

[70] http://en.wikipedia.org/wiki/Matroid [accessed 2008-03-28]

Another interpretation is that every useful optimization algorithm utilizes some form of problem-specific knowledge. Radcliffe [1696] 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.

The rough meaning of the NLF is that all black-box optimization methods perform equally well over the complete set of all optimization problems [1563]. In practice, we do not want to apply an optimizer to all possible problems but to only some, restricted classes. In terms of these classes, we can make statements about which optimizer performs better.

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/`[71] gives a good overview about them. Further summaries, extensions, and criticisms have been provided by Köppen et al. [1173], Droste et al. [602, 601, 599, 600], Oltean [1563], and Igel and Toussaint [1008, 1009]. Radcliffe and Surry [1694] 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[72] proposed by Watanabe [2159] for classification and pattern recognition.

1.4.11 Conclusions

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 1.4.2). Ruggedness (Section 1.4.3) and deceptiveness (Section 1.4.4) in the fitness landscape, often caused by epistatic effects (Section 1.4.6), can misguide the search into such a region. Neutrality and redundancy (Section 1.4.5) 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 from. Noise is present in virtually all practical optimization problems. The solutions that are derived for them should be robust (Section 1.4.7). Also, they should neither be too general (oversimplification, Section 1.4.8) nor too specifically aligned only to the training data (overfitting, Section 1.4.8). Furthermore, many practical problems are multi-objective, i. e., involve the optimization of more than one criterion at once (partially discussed in Section 1.2.2), or concern objectives which may change over time (Section 1.4.9).

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

[71] accessed: 2008-03-28

[72] `http://en.wikipedia.org/wiki/Ugly_duckling_theorem` [accessed 2008-08-22]

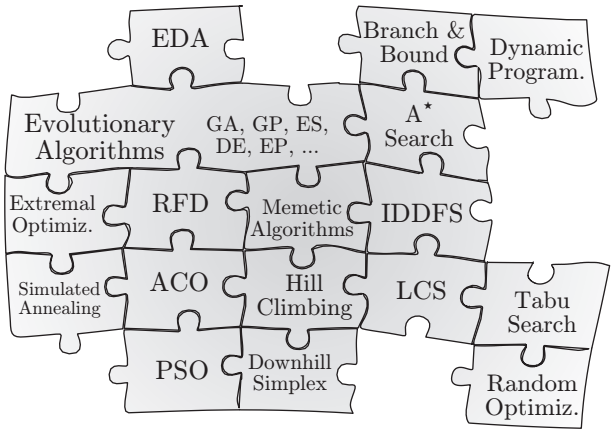

Figure 1.28: The puzzle of optimization algorithms.

is very likely that the "puzzle of optimization alorithms" 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.

1.5 Formae and Search Space/Operator Design

Most global optimization algorithms share the premise that solutions to problems are either elements of a somewhat continuous space that can be approximated stepwise or that they can be composed of smaller modules which have good attributes even when occurring separately.

The design of the search space (or genome) $\mathbb{G}$ and the genotype-phenotype mapping gpm is vital for the success of the optimization process. It determines to what degree these expected features can be exploited by defining how the properties and the behavior of the solution candidates are encoded and how the search operations influence them. In this chapter, we will first discuss a general theory about how properties of individuals can be defined, classified, and how they are related. We will then outline some general rules for the design of the genome which are inspired by our previous discussion of the possible problematic aspects of fitness landscapes.

1.5.1 Forma Analysis

The Schema Theorem has been stated for genetic algorithms by Holland [940] in its seminal work [940, 512, 945]. In this section, we are going to discuss it in the more general version from Weicker [2167] as introduced by Radcliffe and Surry [1695] and Surry [1983] in [1692, 1696, 1691, 1691, 1695].

The different individuals p in the population Pop of the search and optimization algorithms are characterized by their properties ϕ. Whereas the optimizers themselves focus mainly on the phenotypical properties since these are evaluated by the objective functions, the properties of the genotypes may be of interest in an analysis of the optimization performance.

A rather structural property ϕ_1 of formulas $f : \mathbb{R} \mapsto \mathbb{R}$ in symbolic regression[73] would be whether it contains the mathematical expression $x+1$ or not. We can also declare a behavioral property ϕ_2 which is `true` if $|f(0) - 1| \leq 0.1$ holds, i. e., if the result of f is close to a value

[73] More information on symbolic regression can be found in Section 23.1 on page 397.