Weise T, Chiong R, Tang K. Evolutionary optimization: Pitfalls and booby traps. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(5): 907–936 Sept. 2012. DOI 10.1007/s11390-012-1274-4

Evolutionary Optimization: Pitfalls and Booby Traps

Thomas Weise[1], *Member*, *IEEE*, Raymond Chiong[2], *Member*, *IEEE*, and Ke Tang[1], *Member*, *IEEE*

[1]*Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology University of Science and Technology of China, Hefei 230027, China*

[2]*Faculty of Higher Education, Swinburne University of Technology, 50 Melba Avenue, Lilydale, Victoria 3140, Australia*

E-mail: tweise@ustc.edu.cn; rchiong@swin.edu.au; ketang@ustc.edu.cn

Received August 31, 2011; revised February 23, 2012.

Abstract Evolutionary computation (EC), a collective name for a range of metaheuristic black-box optimization algorithms, is one of the fastest-growing areas in computer science. Many manuals and "how-to"s on the use of different EC methods as well as a variety of free or commercial software libraries are widely available nowadays. However, when one of these methods is applied to a real-world task, there can be many pitfalls and booby traps lurking — certain aspects of the optimization problem that may lead to unsatisfactory results even if the algorithm appears to be correctly implemented and executed. These include the convergence issues, ruggedness, deceptiveness, and neutrality in the fitness landscape, epistasis, non-separability, noise leading to the need for robustness, as well as dimensionality and scalability issues, among others. In this article, we systematically discuss these related hindrances and present some possible remedies. The goal is to equip practitioners and researchers alike with a clear picture and understanding of what kind of problems can render EC applications unsuccessful and how to avoid them from the start.

Keywords evolutionary computing, problem difficulty, optimization, meta-heuristics

1 Introduction

Every task with the goal of finding certain configuraticons considered as best in the context of pre-defined criteria can be viewed as an optimization problem. If these problems are formally specified, they can be solved algorithmically either with a dedicated, problem-specific algorithm (such as Dijkstra's algorithm for finding shortest path trees on graphs) or with a more general optimization method. The set of optimization algorithms ranges from mathematical (e.g., using Lagrange Multipliers), numerical (e.g., the Regula Falsi) and simple heuristic (e.g., A*-search) approaches to randomized metaheuristics such as the evolutionary computation (EC) methods. The latter is the focus of this special issue.

When skimming through the articles in this issue, the reader will find many successful examples and variants of different EC techniques (a detailed overview on EC can be found in [1-3]). However, questions such as these may arise: "Why are there so many different optimization methods?", "Is optimization a complicated process? If so, why?", "What makes an optimization problem difficult to solve?", "Which are the things I should consider when tackling a particular optimization task?", and so on. In this article, our aim is to provide some answers to these questions by discussing a list of fundamental issues that are often seen as "obstacles" in the evolutionary optimization domain.

To start with, there are many design decisions in implementing EC methods. For effective optimization, it is important to understand not only the problem being studied, but also how that problem interacts with the applied technique(s). Design choices that do not address issues related to convergence, ruggedness, deceptiveness and neutrality in the fitness landscape, epistasis, non-separability, noise, dimensionality, scalability and so on can hamper the effectiveness of the optimization effort. By using clear definitions and illustrations to describe these fundamental issues, we hope to increase awareness among computer scientists and practitioners about how to avoid pitfalls and how EC can be applied more efficiently in real-world environments (see [4]).

Regular Paper
This work was supported in part by the National Natural Science Foundation of China under Grant Nos. U0835002, 61175065, and 61150110488, the Natural Science Foundation of Anhui Province of China under Grant No. 1108085J16, the European Union 7th Framework Program under Grant No. 247619, the Chinese Academy of Sciences Fellowship for Young International Scientists under Grant No. CX05040000001, and Special Financial Grant from the China Postdoctoral Science Foundation under Grant No. 201104329.

It is necessary to note that this article is not intended to be a tutorial of how to apply a particular EC method, e.g., an evolutionary algorithm (EA), to specific problems[5-6] or how to address subject matters such as multi-objectivity[7], constraint handling[8], or the inclusion of problem-specific knowledge[9]. For the practical application of EC methods in general and EAs in particular, several books exist[4,10-12]. Instead, our aim is to take a closer look at what features of the problem or search space may decrease the solution quality even if the algorithm implementation appears to be correct and "make sense". Considering these features (and corresponding countermeasures) before developing an EA application (or, at least, when trying to improve its performance) may lead to significantly better results.

The article is also not a survey on problem complexity. Research studies on this topic are typically carried out from an analytical, mathematical, or theoretical perspective, with the goal to derive approximations for the expected runtime of the problem solvers[13-15]. These approaches usually focus on benchmark problems or specific classes of optimization tasks, but there is also progress towards developing more general theorems[15-16]. Here, we do not intend to provide a rigorous theoretical treatment of pitfalls and possible traps in evolutionary optimization, but simply to present a top-down view of some "complications" that may be encountered during the optimization process.

Our focus is therefore on the design decisions of EC methods. The effectiveness of these design decisions is often influenced by their actual implementation and the associated parameter values used in the optimization process. While parameter values are important for gaining the most benefit from an EC implementation, design decisions such as problem representation, operator design, and population structure are often considered to be even more critical[17-18].

In the remainder of this section, we will introduce the basic terminologies used throughout this article, describe some possible scenarios of the fitness landscape, and briefly discuss complexity theory. After which, we start off with the topic of convergence in Section 2, followed by other issues possibly leading to unsatisfying convergence, such as ruggedness (Section 3), deceptiveness (Section 4), or neutrality (Section 5) in the fitness landscape. One way ruggedness, neutrality, and deceptiveness can be manifested is from the genotype-phenotype mapping through a phenomenon known as epistasis (Section 6). Optimization can also become more complicated if solutions that are sought have to be robust against noise (Section 7). A high number of objective functions (Section 8) or a large problem scale (Section 9) increases the runtime requirement while also decreasing the expected quality of the solutions. As shown in the overview provided in Table 1 and Table 2, we discuss not only these interrelated issues in optimization, but also list their corresponding countermeasures (which is actually an m-to-n relation). If an optimization algorithm performs well in the presence of some of the problematic facets, this good performance has to be paid for with a loss of solution quality in a different situation — this fact has been formalized in the No Free Lunch Theorem, which we will discuss in Section 10. Finally, we conclude our review on the various issues with a summary in Section 11.

1.1 Basic Terminologies

Throughout this article, we will utilize terminologies commonly used in the EC community. Most of these terminologies are inspired from actual biological phenomena. Fig.1 shows the spaces involved in a typical evolutionary optimization scenario. The *candidate solutions* (or *phenotypes*) x of an optimization problem are elements of the *problem space* $\mathbb{X}$ (also called the *solution space*). Their utility is evaluated by $m \geqslant 1$ *objective functions* f, which embody the optimization criteria (usually subject to minimization). Together, these functions can be considered as one vector function $\boldsymbol{f}\colon \mathbb{X} \mapsto \mathbb{R}^m$.

The objective functions are the only direct source of information available to an EA. It uses this information to decide which candidate solutions are interesting and subsequently combines and/or modifies them in order to sample new points in the problem space. If these two processes can be conducted in a meaningful way, with a certain chance of finding better candidate solutions, the EA can progress towards an optimum — an issue which we discuss in Subsection 3.1 in more detail.

The search operations (such as the unary mutation or the binary recombination/crossover operation) utilized by the EA often do not work directly on the phenotypes. Instead, they are applied to the elements (the *genotypes*) of a search space $\mathbb{G}$ (the *genome*). The genotypes are encoded representations of the candidate solutions, which are mapped to the problem space by a *genotype-phenotype mapping gpm* : $\mathbb{G} \mapsto \mathbb{X}$. A traditional genetic algorithm (GA), for instance, may utilize a bit-string based encoding as the search space, which can be mapped to a real-valued problem space for function optimization[19-21]. In the common case that $\mathbb{G} = \mathbb{X}$, i.e., when the variables are processed in their "natural form"[22], the genotype-phenotype mapping is the identity mapping.

EAs manage a population, i.e., a set of individuals (genotype and the corresponding phenotype), which

Table 1. Overview on Topics and Measures (Part A)

Sections	Countermeasures	Problems											
		Premature Convergence	Slow Convergence	Bad Spread	Multi-Modality	Ruggedness	Deceptiveness	Neutrality	Needle-in-a-Haystack	Epistasis	Noise	Dimensionality ($\boldsymbol{f}$)	Scale ($\mathbb{G}$)
1.2, 3.2.4, 4.2.1, 5.1.2, 6.2.1, 5.2.2	Representation Design					✔	✔	✔	✔	✔			✔
2.1.3, 2.2.2, 3.2.4	Operator Design	✔				✔	✔	✔		✔			
2.1.3, 2.2.1	Increase Exploration	✔	✘	✔	✔		✔			✔			
2.1.3, 2.2.1	Increase Exploitation	✘	✔	✘	✘		✘			✘			
2.2.3	Restarting	✔			✔	✔							
2.1.3, 2.2.4	Lower Selection Pressure	✔	✘	✔	✔	✔	✔						
2.1.3, 6.2.2, 8.2.3	Higher Selection Pressure	✘	✔	✘	✘	✘		✔		✔		✔	
2.2.4, 8.2.1	Larger Population Size	✔	✘	✔	✔	✔		✔		✔		✔	
2.2.5, 4.2.2	Sharing & Niching	✔		✔	✔	✔	✔	✔					
2.2.6	Clustering of Population	✔		✔	✔	✔	✔						
2.2.7	Self-Adaptation	✔	✔				✔						✔
2.2.8	Multi-Objectivization	✔						✔					
3.2.2	Landscape Approximation					✔							
3.2.1	Hybrid Algorithms		✔			✔		✔	✔				
3.2.3	Combined Algorithms					✔							
4.2.2	Use of Memory						✔	✔					
4.2.3	Preventing Convergence	✔		✔	✔	✔	✔			✔			
4.2.4	Novelty Search	✔					✔						
6.2.3	Linkage Learning						✔			✔			✔
7.2	Randomizing Objectives										✔		
8.2.2	Use Multiple Archives											✔	
8.2.4	Use Indicator Functions											✔	
8.2.5	Scalarizing											✔	
8.2.6	Limiting Search Area											✔	
9.2.1	Parallelization		✔										✔
9.2.2	Developmental Represent.												✔
9.2.4	Adaptive Encodings												✔
9.2.5	Exploiting Separability												✔

Note: ✔ means that the given measure maybe useful if the problem in the same row is observed, ✘ means that the measure would likely be counter-productive.

Table 2. Overview on Topics and Measures (Part B)

Problems	Sections
Premature Convergence	2.1.1
Slow Convergence	2.1.1, 1.3
Bad Spread	2.1.1
Multi-Modality	3, 2.2.1
Ruggedness	3
Deceptiveness	4
Neutrality	5
Needle-in-a-Haystack	5.1.3
Epistasis	6
Noise	7
Dimensionality ($\boldsymbol{f}$)	8
Scale ($\mathbb{G}$)	9

undergo evaluation, selection, and reproduction in each iteration (generation). Before selection, a single scalar *fitness* value is assigned to each individual. The fitness denotes the priority of an individual for being selected as the parent for offspring in the next generation, i.e., its chance of being chosen as the input to a search operation. This fitness, in general, is determined by a fitness assignment process that usually relies on the objective value(s) of the candidate solution stored in the individual record. It often relates these objective values to those of other candidate solutions in the population, e.g., by computing the individual's (Pareto) rank among them. The fitness may, however, also include additional information[17] such as diversity metrics (see, e.g., Subsection 2.2.5).

If only a single objective function is to be optimized (i.e., $m = 1$ and $\boldsymbol{f} = f$), it is sometimes referred to as the *fitness function* as well, so there exists some ambiguity in the terminology[1]. From the latter, the term "fitness landscape" is derived, which refers to the visualization of an objective function (and not of the results of a fitness assignment process).

An illustration of the spaces and sets involved in (evolutionary) optimization is given in Figs. 1 and 2, where the candidate solutions are coordinate pairs decoded from bit strings (the genotypes) via the genotype-phenotype mapping. Each element of a genotype that can be modified by a search operation is called a *gene*. The term *building block* denotes groups of gene settings that together form an essential element of an individual.

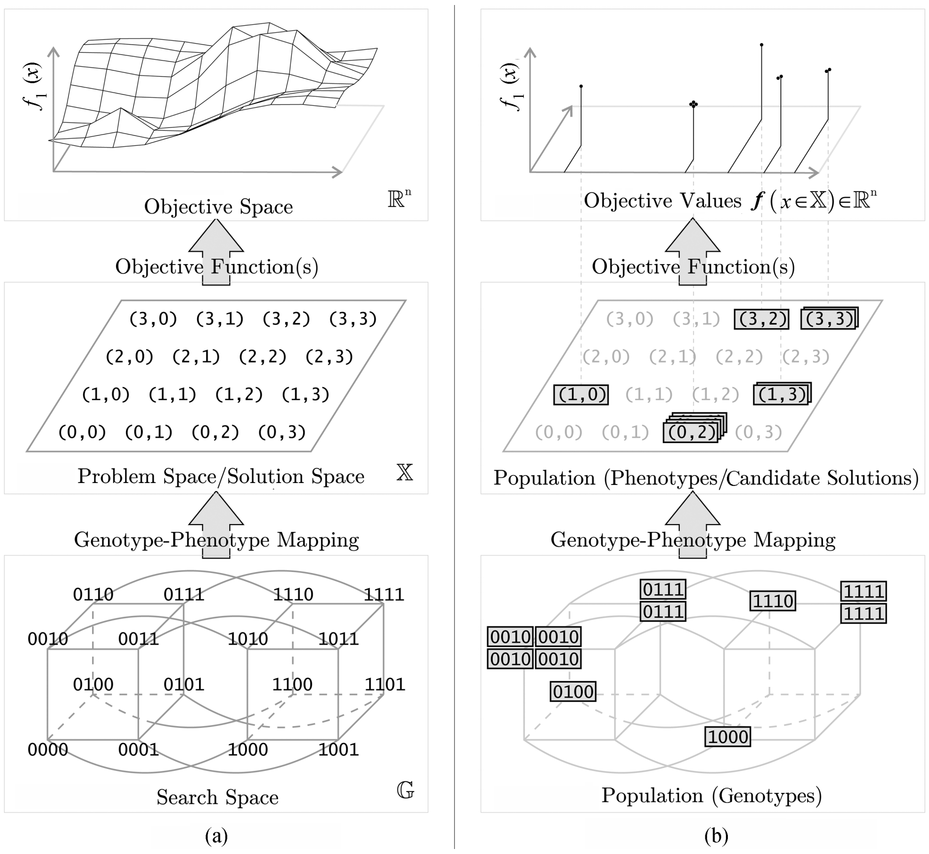
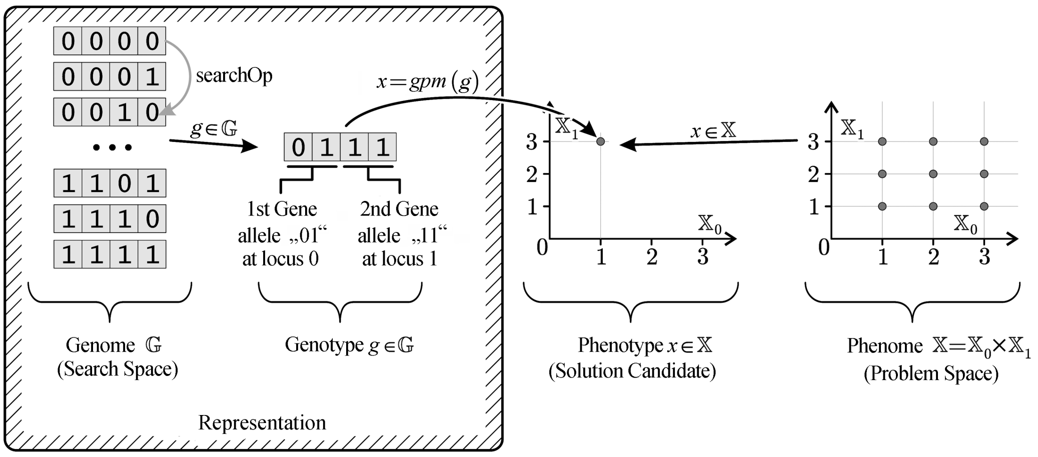


Fig.1. Involved spaces and sets in (evolutionary) optimization. (a) Involved spaces. (b) Involved sets/elements.



0 0 0 0
0 0 0 1
0 0 1 0
1 1 0 1
1 1 1 0
1 1 1 1
searchOp
g∈𝔾
0 1 1 1
x=gpm (g)
1st Gene allele „01" at locus 0
2nd Gene allele „11" at locus 1
x∈𝕏
𝕏_1
𝕏_0
Genome 𝔾 (Search Space)
Genotype g∈𝔾
Representation
Phenotype x∈𝕏 (Solution Candidate)
Phenome 𝕏=𝕏_0×𝕏_1 (Problem Space)

Fig.2. Relation between genotypes and phenotypes (candidate solutions) in EAs.

1.2 Fitness Landscapes

As aforementioned, the most important information sources for an optimization algorithm are the $m \geqslant 1$ objective functions that rate the quality of possible solutions to an optimization problem. 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 curves in Fig.3 where we sketch a number of possible scenarios of the fitness landscape (objective function plots) that we are going to discuss in this article. The objective values in the figure are subject to minimization and the small bubbles represent candidate solutions under investigation. An arrow from one bubble to another means that the second individual is found by applying a search operation to the first one. As can be seen, there are different objective function shapes that can pose to be difficult for an optimization algorithm to proceed its search in this manner. EAs typically work on multiple solutions simultaneously and, as a result, the search space navigation can be difficult to visualize. These graphs thus provide a simplified visualization of the theories discussed rather than an accurate depiction of the EA search process. The structure of EAs enables them to often overcome some local optima, deception, ruggedness, and neutrality.

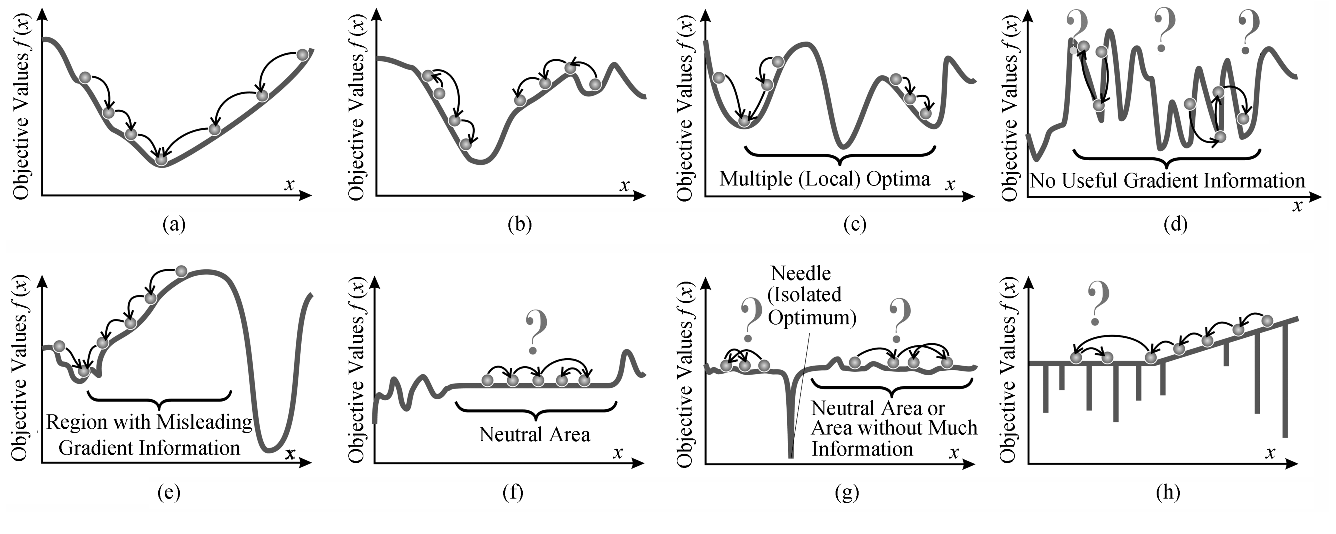
From these plots, it may also seem that the shape of the fitness landscape is defined by the objective function only. However, this is *not* true from the perspective of an EA. Here, the *representation*, i.e., the choice of search space, search operations, and the genotype-phenotype mapping, has a tremendous impact on the effective shape of the fitness landscape[23]. As outlined in the previous subsection, an EA conducts its search by applying the search operators to genotypes in a search space that are mapped to phenotypes in a problem space which, in turn, are evaluated by the objective functions, as illustrated in Fig.2. The concept of adjacency amongst candidate solutions from the viewpoint of an EA hence depends on the representation used and not on their proximity in the problem space (unless both spaces are the same, that is). In any case, many of the problematic issues which we will discuss in this article are closely related to the choice of representation, as can be seen directly in Tables 1 and 2 and, for instance in Subsections 3.2.4, 4.2.1, 5.1.2, and 6.2.1.

An important feature of the fitness landscape is that it may have different global and local structures. Fig.4 illustrates one objective function graph (in the top-left sub graph) from which regions are successively selected and "zoomed in". As can be seen, different sections of this function may exhibit different problematic features or issues. It is thus necessary to remember that the characteristic of an objective function may seem to be dynamic[24] and change during the course of optimization when the global optimum is approached.

Before going into the details of difficult fitness landscape features, we would like to briefly review the term *difficult* itself from the perspectives of both traditional, deterministic, exact algorithms as well as EAs.

1.3 Problem Hardness

One of the basic goals of computer science is to find the *best* algorithm for solving a given class of problems. The performance measures used to rate an algorithm's



Objective Values f(x)
x
(a)
(b)
Multiple (Local) Optima
(c)
?
No Useful Gradient Information
(d)
Region with Misleading Gradient Information
(e)
Neutral Area
(f)
Needle (Isolated Optimum)
Neutral Area or Area without Much Information
(g)
(h)

Fig.3. Examples of different possible scenarios in the fitness landscape (under minimization). (a) Best case. (b) Multi-modal with low total variation. (c) Multi-modal with higher total variation. (d) Rugged (multi-modal + high total variation). (e) Deceptive. (f) Neutral. (g) Needle-in-a-haystack. (h) Nightmare.

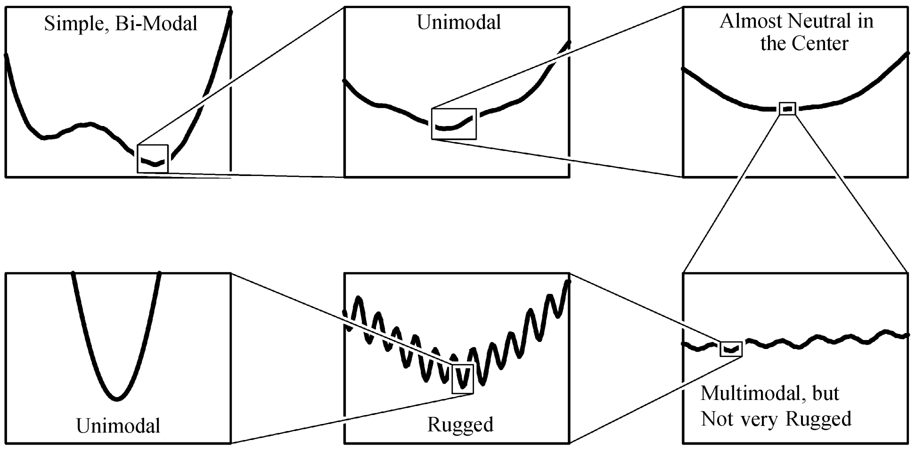


Fig.4. Artificial example of how landscape features may change depending on the selected region of the graph.

efficiency are 1) the time it takes to produce the desired outcome and 2) the storage space it needs for internal data, i.e., its time and space complexity. Both of these can be described as functions of the input size of the algorithm for best, average, and worst-case input situations, which are usually simplified using the big-$\boldsymbol{O}$ family notations.

The computational complexity of a problem is bounded by the best algorithm known for that problem. It states how much resources are necessary for solving the given problem, or, from the opposite point of view, tells whether the given resources are sufficient for this purpose.

The set of all problem classes that can be solved on a computer① within polynomial time is called $\mathcal{P}$. These are problems which are said to be exactly solvable in a feasible way. The set of problem classes that allows solution verification in polynomial time is called $\mathcal{NP}$, which also comprises all the problems from $\mathcal{P}$ ($\mathcal{P} \subseteq \mathcal{NP}$). A problem A is *hard* for a complexity class if every other problem in this class can be *reduced* to it, i.e., if the other problems can be re-formulated so that they can also be solved with an algorithm for A. Exactly solving any $\mathcal{NP}$-hard problem is difficult as it may require super-polynomial, exponential time.

Solving such a problem to optimality is thus not always possible. When dealing with $\mathcal{NP}$-hard problems that have more than a certain number of variables, we may need to give up some solution quality in order to make the problem computationally tractable. EAs use some random process in their execution. These stochastic algorithms (usually) trade in solution correctness, i.e., the guarantee to find the global optimum, for a lower runtime. In other words, if we apply an EA, we would normally not expect to find the global optima but some reasonably good approximations within feasible time. The limits of this speed-up are discussed in Subsection 9.1.

While $\mathcal{NP}$-hard problems can be considered to be difficult for any exact method, the question about which problems are *GA*- or *EA-hard* arises. This question has been considered from several perspectives[25-28], and the most notable discussions can be found in [15, 29]: Problem instance classes for which the expected worst case first hitting time, i.e., the number of steps required to find a global optimum, of a particular EA has an exponential lower bound are "EA-hard" (for that EA). In [15], two such classes have been proposed for (1 + 1) EAs: 1) wide-gap problems, where there is a very low probability that the EA can escape a local optimum towards a region with higher utility, and 2) long-path problems, where advancement towards better objective values has a reasonably high probability, but the necessary number of such steps is very high. It should be noted that some instance classes of $\mathcal{NP}$-hard problems can be EA-easy[30].

Finding out how hard certain problems are for EAs is an active research area and much work has been devoted to finding the asymptotical complexity of these stochastic algorithms in different scenarios[13-16].

2 Convergence

An optimization algorithm has *converged* 1) if it cannot reach new candidate solutions anymore or 2) if it keeps on producing candidate solutions from a "small"② subset of the problem space[2]. Optimization

①or Deterministic Turing Machine.

②According to a suitable metric like the number of modifications or mutations that need to be applied to a given solution in order to leave this subset.

processes will usually converge at some point in time. In the ideal case, convergence happens towards the global optimum. One of the problems in evolutionary optimization is that it is often not possible to determine whether the best solution currently known is situated on a local or global optimum and thus, if the convergence is acceptable. In other words, it is 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, of course, can only become cumbersome if there are multiple (local) optima, i.e., the problem is *multi-modal*[31], as depicted in Fig.3(c). It is worthwhile to note that convergence often occurs much more quickly in the objective space than in the search and solution spaces.

2.1 The Issues

There are at least three basic problems related to the convergence of an optimization algorithm: premature, non-uniform, and domino convergence. The first one is considerably the most important in optimization, but the latter ones may cause a lot of inconveniences too.

2.1.1 Premature and Non-Uniform Convergence

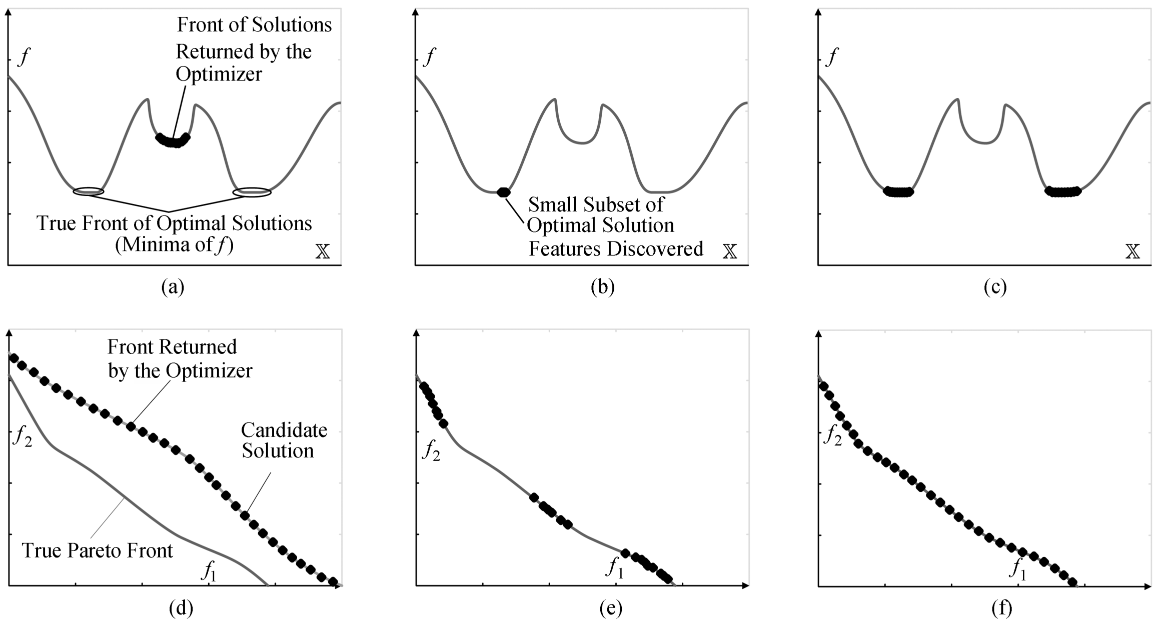
The main goal in EC is to find solutions that are as close to the true global optimum as possible. An optimization process is considered to have *prematurely converged* to a local optimum if it is no longer able (or extremely unlikely) to explore other parts of the search space than the area currently being examined and there exists another region that contains a superior solution. In case that there is more than one global optimum, then the second goal is to discover as many of them as possible.

In single-objective optimization, all the global optima have the same objective values but reside on different peaks (or hyperplanes) of the objective function. The presence of multiple such optima is the focus of research on multi-modal optimization[32].

In multi-objective optimization, there are usually many global optima due to the trade-off of the objectives. Take the task of finding a good car, for example, where the criteria *speed* and *fuel consumption* would be traded-off. The optimization process should discover both, slower, environmentally friendly cars and fast cars that need more gasoline.

In some optimization problems, the number of (globally) optimal solutions is too large to provide all of them to the human operator. On these cases, the subset of delivered solutions should well represent the range of possible results, i.e., it should be a uniform sample of all possible optimal features. If only some of the optimal features are presented to the human operator, e.g., only the fast cars in the above example, the convergence is said to be *non-uniform*[33].

Fig.5 illustrates these issues on the examples of a



Front of Solutions Returned by the Optimizer
f
True Front of Optimal Solutions (Minima of f)
𝕏
(a)
f
Small Subset of Optimal Solution Features Discovered
𝕏
(b)
f
𝕏
(c)
Front Returned by the Optimizer
Candidate Solution
f2
True Pareto Front
f1
(d)
f2
f1
(e)
f2
f1
(f)

Fig.5. Optimal solution approximation sets. (a) Bad convergence and good spread (single-objective). (b) Good convergence and bad spread (single-objective). (c) Good convergence and spread (single-objective). (d) Bad convergence and good spread (bi-objective). (e) Good convergence and bad spread (bi-objective). (f) Good convergence and spread (bi-objective).

single-objective (Figs. 5(a)$\sim$5(c)) and a bi-objective optimization task (Figs. 5(d)$\sim$5(f)); objectives are subject to minimization. Fig.5(a) shows the result of having a very good spread (or diversity) of solutions, but the points are far away from the optima. Fig.5(d) is a sketch of the same issue for a bi-objective problem: the discovered solutions are diverse, but distant from the true Pareto front of best trade-offs. Such results are not attractive because they do not provide optimal solutions and we would consider the convergence to be premature in this case. The second examples (Fig.5(b) and Fig.5(e)) contain solution sets that are very close to the true optima but cover them only partially, so the decision maker could lose important options. Finally, the optimization results depicted in Fig.5(c) and Fig.5(f) have the two desirable properties of good convergence (i.e., the solutions are very close to optimal) and spread (i.e., the whole trade-off curve between the two objectives is covered).

2.1.2 Domino Convergence

The phenomenon of *domino convergence*[34-35] occurs when the candidate solutions have features contributing to significantly different degrees to the total fitness. If these features are encoded separately, they are likely to be treated with different priorities. If, for example, optimization takes place over $\mathbb{R}^\ell$ and the first element of a solution vector is much more important (from the perspective of the objective function) than the second one, its priority during the optimization process will be much higher too.

Although this seems to be legit, it can prevent us from finding the global optimum: gene values with strong positive influence on the objective values, for instance, will quickly be adopted by the optimization process (i.e., "converge"). During this time, the values of the genes with smaller contribution are ignored. Their state may remain rather random and *hitchhike* through the generations in genotypes with good configurations of the more salient genes[36]. They do not receive evolutionary pressure until the optimal configurations of these genes have been accumulated. This sequential convergence phenomenon is called *domino convergence* due to its resemblance to a row of falling domino stones[35].

In the worst case, the contributions of the less influential genes may look almost like noise and they are not optimized at all. This leads to premature convergence, since the global optimum which would involve optimal configurations of all genes will not be discovered. Here, restarting the optimization process will not help because it will turn out the same way with very high probability.

2.1.3 Diversity, Exploration, and Exploitation

In biology, diversity is referred to as the variety and abundance of organisms at a given place and time[37]. Genetic diversity is the fuel of evolution and essential for a species' robustness against and adaptivity to environmental changes. In EAs, maintaining a diverse population is very important as well. Losing diversity means approaching a state where all the candidate solutions under investigation become similar to each other. Consequently, no new areas in the search space will be explored and the optimization process will not make any further progress.

The process of finding points in new areas of the search space that are rather distant from the currently investigated candidate solutions is called *exploration*[38]. Exploration increases diversity but often leads to the creation of solutions inferior to those that have already been investigated. However, like in biology, there is a small chance that new genetic material can lead to the discovery of superior traits.

On the other hand, *exploitation* is the process of improving and combining the traits of the (best) currently known solutions. Exploitation-based search operations often perform small changes in individuals, producing new, very similar candidate solutions. This would give rise to some steady improvement in fitness for a period of time, but it also reduces diversity in the population since offspring and parents become more and more similar to each other. Another problem with exploitation is that possibly existing better solutions which may be located in distant areas of the problem space will not be discovered.

Exploration versus exploitation[39-40] is therefore the dilemma of deciding which of the two principles to apply and to which degree at a certain stage of optimization. It is sketched in Fig.6 and can be observed in many areas of optimization. More or less synonymous to exploitation and exploration are the terms *intensification* and *diversification*[41]. Optimization algorithms that favor exploitation over exploration have higher convergence speed but run the risk of not finding the optimal

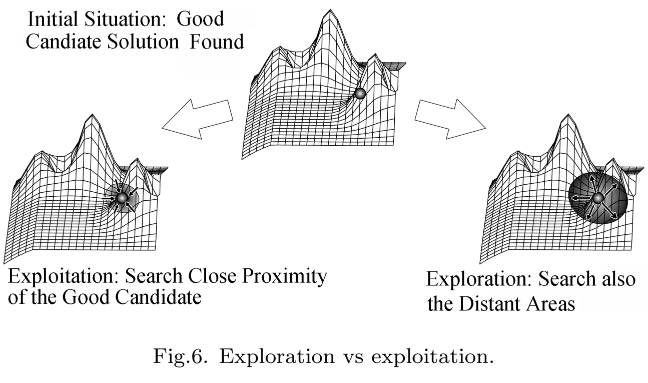

Initial Situation: Good Candiate Solution Found
Exploitation: Search Close Proximity of the Good Candidate
Exploration: Search also the Distant Areas

Fig.6. Exploration vs exploitation.

solution and may get stuck at a local optimum. Then again, algorithms that perform excessive exploration may never improve their candidate solutions well enough to find the global optimum or it may take them very long to discover it.

Almost all components of optimization strategies can either be used for increasing exploitation or in favor of exploration. Exploitation can be achieved by building unary search operations (e.g., mutation operators) that improve an existing solution in small steps. However, mutation in an EA can also be implemented in a way that introduces much randomness into the individuals, effectively turning it into an exploration operator. Selection operations choose a set of the most promising candidate solutions that will be investigated in the next iteration of the algorithm. They can either return a small group of best individuals (exploitation) or a wide range of existing candidate solutions (exploration).

A good example for the exploration vs exploitation dilemma is the simulated annealing algorithm[42]. It is often modified to a faster form called *simulated quenching*, which focuses on exploitation but loses the guaranteed convergence to the optimum[43]. Another good example is given in [44-45], where it is shown that for some problems, the selection pressure and mutation rate of an EA must be balanced extremely well in order to achieve a polynomial expected runtime. Too much exploitation or exploration may both lead to an exponential expected first hitting time.

2.2 Countermeasures

There is no general approach to prevent unsatisfying convergence as this phenomenon may have a variety of different causes. The probability of an optimization process getting caught in a local optimum depends on the characteristics of the problem at hand and the parameter settings as well as on features of the optimization algorithms applied[33].

2.2.1 Balanced Exploration and Exploitation

Generally, optimization algorithms should employ at least one search operation of explorative character and at least one that 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[38], ranging from targeted initialization of the population[46], mining data from the optimization process[47], to devising specialized population structures[48] and specialized search operators[49].

2.2.2 Search Operator Design

A very basic measure to decrease the probability of premature convergence is to make sure that the search operations are *complete*, i.e., to make sure that they can (theoretically at least) reach every point in the search space from every other point. Then, it is possible to escape arbitrary local optima with non-zero probability.

A good example for this is the modification to evolutionary programming (EP) introduced in [50]: By replacing the usually applied normally distributed mutations with Lévy distributed ones, the probability to reach distant points in a real-coded search space within a single mutation step is increased and better results could be obtained. In [51], the large impact of search operator design on the solution quality for a combinatorial problem is confirmed.

2.2.3 Restarting

A very crude yet sometimes effective measure is to restart the optimization process at randomly or strategically chosen points in time. One example for this is the Greedy Randomized Adaptive Search Procedure (GRASP)[52], which continuously restarts the process of creating an initial solution and refines it with local search. Still, this approach is likely to fail in domino convergence situations.

2.2.4 Low Selection Pressure and/or Larger Population Size

Generally, the higher the chance that candidate solutions with bad fitness are investigated instead of being discarded in favor of seemingly better ones, the lower the chance of getting stuck at a local optimum. This is the exact idea which distinguishes simulated annealing from hill climbing. It is known that simulated annealing can find the global optimum, whereas simple hill climbers are likely to prematurely converge since they always proceed with the best candidate solution discovered so far.

In an EA, too, using low selection pressure decreases the chance of premature convergence and can lead to a better approximation of the true global optima. However, such an approach also decreases the speed with which good solutions are exploited and thus, increases the runtime. Also, too low of a selection pressure may cause genetic drift, which we will put into the context of neutrality and evolvability in Subsection 5.1.1.

Increasing the population size may be useful as well, since larger populations can maintain more individuals and hence, cover many different solutions. This coverage can lead to a lower selection pressure. However, the idea that larger populations will lead to better optimization results does not always hold[53-54]. For these reasons, both population-sizing[54-55] and selection[14] are highly-active research areas in the EC community.

2.2.5 *Sharing, Niching, and Clearing*

As opposed to increasing the population size, it is also possible to "gain more" from the smaller populations. In order to extend the duration of the evolution in EAs, many methods have been devised for steering the search away from areas which have already been frequently sampled. In steady-state EAs it is common to remove duplicate genotypes from the population[56].

More generally, the exploration capabilities of an optimizer can be improved by integrating density metrics into the fitness assignment process. The most popular of such approaches are sharing and niching[39,57-59]. The strength pareto-type algorithms, which are widely accepted to be highly efficient, use another idea: they adapt the number of individuals a candidate solution *dominates* as the density measure[60-61]. In the *simple convergence prevention* method[2,62-63], candidate solutions with the same objective values are deleted based on a given probability. In the *clearing* approach[64], all individuals are grouped according to their distance in the phenotypic or genotypic space and all but a certain number of individuals from each group receive the worst possible fitness. The efficiency of all these diversity preservation methods strongly depends on the situation — a method suitable for one scenario may cause problems in another[65].

2.2.6 *Clustering of Candidate Solutions*

A more explicit method to prevent premature convergence is to cluster the search space or population of an EA. This allows the optimization method to track multiple different basins of attraction at the same time and increases the chance of finding the global optimum in one of them. Particularly in the context of estimation of distribution algorithms (EDAs), various such methods have been proposed[66-69].

2.2.7 *Self-Adaptation*

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)[70-71].

2.2.8 *Multi-Objectivization*

Recently, the idea of using helper objectives[72] has emerged. Here, a single-objective problem is transformed into a multi-objective one by adding new objective functions[73-77]. In some cases, such changes can speed up the optimization process[78]. The new objectives are often derived from the main objective by decomposition[78] or from certain characteristics of the problem[75]. They are then optimized together with the original objective function with some multi-objective techniques.

3 Ruggedness

Optimization algorithms generally depend on some form of trends[③] in the fitness landscape. Ideally, the objective functions would be continuous and exhibit low total variation[④] (as sketched in Fig.3(b)), so that the optimizer can track the trend easily. If an objective function is unsteady or goes up and down frequently, it becomes more complicated to find the right directions to proceed during the optimization process (see Fig.7 and Fig.3(d)). The more rugged the function gets, the harder it is to optimize it. In short, one could say ruggedness is multi-modality (see Fig.3(c)) plus steep ascends and descends in the fitness landscape.

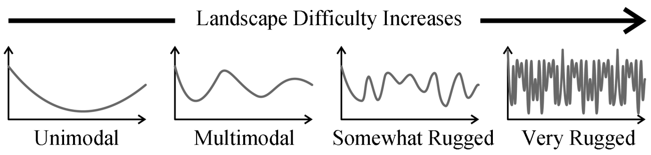


Fig.7. Landscape difficulty increases with increasing ruggedness.

3.1 Issue: Weak Causality

During an optimization process, new points in the search space are created by the search operations. Generally, we can assume that the inputs of the search operations correspond to points that 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 being passed to the search operations are likely to have good fitness. Since the fitness of a candidate solution depends on its features, it can be assumed that the features of these individuals are promising, too. It should thus be possible for the optimizer to introduce small changes to

[③]Using the word "gradient" here would be too restrictive and mathematical.

[④]http://en.wikipedia.org/wiki/Total_variation, Nov.25, 2011.

these features (by modifying the genes encoding them slightly) in order to find out whether they can be improved any further. Normally, such *exploitive* modifications should also lead to small changes in the objective values and hence, in the fitness of the candidate solution.

Strong causality (locality) means that small changes in the features of an object also lead to small changes in its behavior[79-80]. In fitness landscapes with weak (low) causality, small changes in the candidate solutions often lead to large changes in the objective values. It then becomes harder to decide which region of the problem space to explore and the optimizer cannot find reliable trend information to follow. The lower the causality of an optimization problem, the more rugged its fitness landscape is, which leads to degeneration of the performance of the optimizer[81]. This does not necessarily mean that it is impossible to find good solutions, but it may take longer time to do so.

3.2 Countermeasures

Ruggedness in the fitness landscape is hard to mitigate. In population-based approaches, using large population sizes and applying methods to increase diversity can reduce the influence of ruggedness, but only up to a certain degree.

3.2.1 Hybridization with Local Search

Often, EAs are combined with a local search technique applied to each individual in the population before presenting it to the evolutionary process. Two such common approaches are Lamarckian evolution[82] (performing a local search on the genotype level) and the Baldwin effect[82-83] (local search on the phenotype level). Memetic algorithms[84-86] and other hybrid approaches[22,87-88] also fall into this category. Since the EA only receives individuals residing in local optima resulting from the local search procedure(s), the fitness landscape may seem to be less rugged from its perspective[89-90]. However, local search can also lead to much higher selection pressure and thus swing the pendulum to the problem of premature convergence[3].

3.2.2 Landscape Approximation

In order to smoothen out a rugged landscape, it can be approximated by parameterizing a function based on the knowledge gathered from previously sampled candidate solutions. The optimization process can then be performed on this smooth approximation, which, in turn, is updated in each step. The goal here is not to find a function that perfectly represents the fitness landscape, but to work on a much smoother function without changing the location of the global optimum. In [90], for example, a k-dimensional quadratic polynomial is used to approximate the fitness function. The second advantage of this idea is that a new candidate solution can be created by directly solving the approximation function analytically.

3.2.3 Two-Staged Optimization

Another approach is to apply a two-staged optimization process[91] where two different algorithms are applied sequentially. Here, the first optimization method should be an algorithm with strong global optimization abilities, which discovers the most promising area in the search space and is not easily distracted from rugged objectives (e.g., an EDA). Then, an algorithm that is quick to exploit and follow the trend in a landscape, such as differential evolution (DE), is applied to the subspace discovered by the first algorithm.

3.2.4 Better Operator and Search Space Design

Weak causality is often caused, to some extent, by bad design of the solution representation and search operations. We pointed out that exploration operations are important for minimizing the risk of premature convergence. Exploitation operators are equally important for refining the solution quality. In order to apply optimization algorithms in an efficient manner, it is necessary to find representations that allow for iterative modifications with bounded influence on the objective values[62-63,92-93], i.e., exploitation. This can eventually lead to better candidate solutions. Fortunately, many problems where their formulation is inspired by a real-world problem share the feature that improved solutions can often be built from other good solutions, i.e., often exhibit strong causality. A comprehensive collection of examples for representations that exhibit this property in real-world application domains can be found in [4].

4 Deceptiveness

Especially annoying fitness landscapes show *deceptiveness* (or deceptivity). The gradient of deceptive objective functions leads the optimization process away from the optima, as illustrated in Fig.3(e) as well as Fig.8. The term deceptiveness is mainly used for the GA in the context of the Schema Theorem[2,94-95]. Schemas describe certain areas (hyperplanes) in the search space. If an optimization algorithm has discovered an area with 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

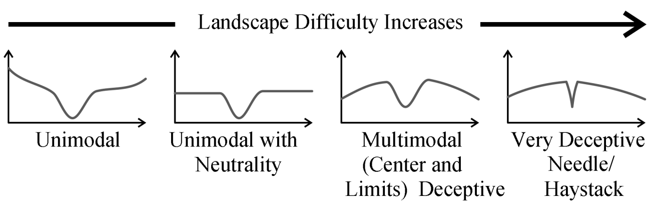

Landscape Difficulty Increases
Unimodal
Unimodal with Neutrality
Multimodal (Center and Limits) Deceptive
Very Deceptive Needle/ Haystack

Fig.8. Increasingly difficult landscapes caused by deceptivity.

the case are considered to be deceptive[22,96]. It is interesting that some problems with the highest level of deceptiveness appear to be easy for GAs[22], whereas an increasing amount of deceptiveness generally leads to a steep increase in problem hardness[97].

4.1 The Issue

An objective function is *deceptive* if a greedy local search algorithm would be steered in a direction leading away from all global optima in large parts of the search space. The basic problem caused by deceptiveness is that the information accumulated by an optimizer actually guides it away from the optimum. Search algorithms that strictly follow a path towards improving fitness will not be able to discover the global optimum in this case. In other words, they may perform worse than non-repeating random sampling, a random walk, or an exhaustive enumeration method in terms of the first hitting time of the global optimum. These most primitive search methods sample new candidate solutions without taking into account the utility of the already investigated solutions and hence are not vulnerable to deceptiveness.

4.2 Countermeasures

Solving tasks with deceptive objective functions perfectly involves sampling many individuals with very bad features and fitness. This contradicts the basic ideas of metaheuristics and thus, there are no really efficient countermeasures against high degrees of objective function deceptivity. Using large population sizes, maintaining high diversity (see, e.g., Subsections 2.2.5 and 2.2.6), and utilizing linkage learning (see Subsection 6.2.3) provide at least a small chance of finding good solutions.

4.2.1 Representation Design

Like weak causality, deceptiveness can also be caused by the design of the representation. Utilizing a more suitable search space, search operations, and genotype-phenotype mapping may make an optimization problem much less deceptive. Notice that the representation is a part of the optimization algorithm which produces the inputs of the objective function (see Fig.1). Changing it can change the behavior of the objective function from the perspective of the optimization process significantly. Combining different representations in an EA may lead to better results as shown in [98]. This can be a feasible approach if the nature of the problem is too complex to manually design a non-deceptive representation.

4.2.2 Niching and Memory

Applying the diversity increasing methods mentioned in Subsection 2.2.5 (such as niching and the simple convergence prevention method) can delay the convergence of the optimization process and thus, increase the chance to escape from deceptive local optima. Recent studies of particle swarm optimization (PSO)[99] show that the local memory property of the simulated particles can lead to some niching behavior, which is especially suitable for this purpose as well. Here, the *lbest* PSO with ring topology discussed in [99] is noteworthy.

4.2.3 Preventing Convergence

Another approach to counteract deceptiveness is to stop the optimization algorithm from converging altogether. If the population of an EA is prevented from collapsing to a certain area of the search space and is always kept "moving", deceptive basins of attraction will be left eventually.

The Fitness Uniform Selection Scheme[100-101] takes the idea a step further. Instead of selecting the most promising candidate solutions, a diverse population with individuals from all fitness levels is maintained in order to avoid getting stuck at a local optimum. To achieve this, in each generation the best and worst individuals (with the smallest and largest fitness, say f_s and f_l) in the population are first determined. For each slot in the new population, a random value r uniformly distributed between f_s and f_l is drawn and the individual with the fitness closest to r will be selected. The selected candidate solutions will be diverse in terms of fitness and the population basically maintains a path of individuals out of the current local optimum.

If the optimization problem lacks causality and the fitness landscape is very rugged, however, this method may fail. If structurally similar points within a small subset of the search space may possess very different fitness, the search may get trapped within that subset.

4.2.4 Novelty Search

In Novelty Search[102-104], the objective function f is completely abandoned. The reason is that, on one hand, in the case of deceptivity f may be misleading and guide the search away from the global optima. On the other hand, it is also not clear whether f would reward stepping stones, i.e., the intermediate solutions between the initially chosen starting points and the global optimum. In many genetic programming (GP) applications[92,105], for example, the intermediate steps obtained by modifying a bad program iteratively towards a perfect solution rarely form a sequence of improving fitness and even needle-in-a-haystack situations (see Subsection 5.1.3) are common.

Novelty Search thus does not employ a traditional fitness measure since it may not help the optimizer to discover and combine building blocks anyway. Instead, an archive of past candidate solutions is kept and updated and selection will choose the individuals that differ the most from the archived ones. As more and more candidate solutions with different behaviors are discovered, chances are that one amongst them is an acceptable solution. This method led to good results in the evolution of virtual creatures[104], walking behaviors[103], and navigation control[102-103].

5 Neutrality

The outcome of the application of a search operation to an element of the search space is *neutral* if it yields no change in the objective values[107-108]. It is challenging for optimization algorithms if the best candidate solution currently known is situated on a plane of the fitness landscape, i.e., all adjacent candidate solutions have the same objective values. As illustrated in Fig.3(f) and Fig.9, an optimizer cannot find any gradient information in this case and thus there is no direction as to which way 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 eventually overflow and require pruning.

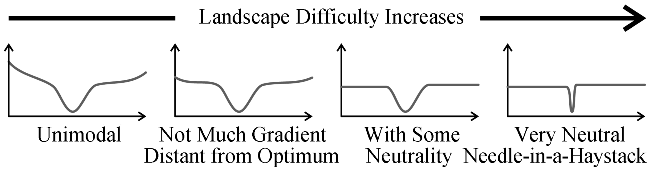

Landscape Difficulty Increases
Unimodal
Not Much Gradient Distant from Optimum
With Some Neutrality
Very Neutral Needle-in-a-Haystack

Fig.9. Landscape difficulty caused by neutrality.

5.1 The Issues

5.1.1 Evolvability

Another metaphor in EC that has been borrowed from biological systems is *evolvability*[109]. In biology, the meaning of this word is twofold[110]: 1) a biological system is evolvable if it is able to generate heritable, selectable phenotypic variations[111]; and 2) a system is evolvable if it can acquire new characteristics via genetic changes that help the organism(s) to survive and to reproduce. In the optimization domain, the evolvability of an optimization process defines how likely the search operations will lead to candidate solutions with new (and eventually, better) objective values. (The direct *probability of success*[79,112], i.e., the chance of search operators producing offspring fitter than their parents, is also sometimes referred to as *evolvability* in the context of EAs[113-114].) In Subsections 4.2.3, 4.2.4, and (part of) 2.2.5, we already argued that preventing an optimization process from converging, i.e., keeping it in an evolvable state, may enable it to discover better results.

The link between evolvability and neutrality has been discussed by many researchers[110,115]. The evolvability of neutral parts of a fitness landscape depends on the optimization algorithm used. For example, the evolvability of hill climbing-like approaches can be especially low, since the search operations cannot directly provide improvements or even changes in fitness. This could then degenerate the optimization process to a random walk, as illustrated in Fig.3(f). Using the ND fitness landscapes, i.e., landscapes with a well-defined degree of neutrality, it has been shown that neutrality may "destroy" useful information such as correlation[116].

Researchers in molecular evolution, on the other hand, found indications that the majority of mutations in biology have no selective influence[117], and that the transformation from genotypes to phenotypes is a many-to-one mapping. Neutrality in natural genomes is often considered as beneficial if it concerns only a subset of the properties peculiar to the offspring while allowing meaningful modifications of the others[110,118].

The theory of *punctuated equilibria*[119] states that species experience long periods of evolutionary inactivity, which are interrupted by sudden, localized, and rapid phenotypic evolutions. It is assumed that the populations explore networks of neutral genetic changes during the time of stasis until, suddenly, a relevant change in a genotype leads to a better adapted phenotype[120] and reproduces quickly. Similar phenomena can be observed and have been utilized in EAs[121-122].

Another example for neutrality in biology is *degeneracy*: the ability of elements that are structurally different to perform the same function or yield the same output[123] while also having additional, unique features. Similarly, degeneracy of the properties of candidate solutions introduced by the chosen solution representation in an optimization process can improve its robustness and ability to adapt[124].

The key to differentiating between "good" and "bad" neutrality is its degree in relation to the number of possible solutions maintained by an optimization algorithm. The illustrative example in Fig.10 shows that a certain amount of neutral reproduction can foster the progress of optimization. In Fig.10(a), a scenario of premature convergence is depicted. Fig.10(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.10(c), the chance of finding the global optimum increases as well. Then again, if the bridge gets too wide (see Fig.10(d)), the optimization process may end up in a scenario like Fig.3(f) where it cannot find any direction.

Drift, a term stemming from the area of population genetics, describes the loss of population diversity resulting from the stochastic nature of selection in a finite population (in both nature and EAs)[117,126]. In neutral parts of the fitness landscape or under low selection pressure, this effect is very likely. A reduction of diversity in the population generally is a negative effect (see Subsection 2.1.3).

Unlike ruggedness, which is always bad for the performance of optimization algorithms, neutrality has aspects that may further as well as hinder the process of finding good solutions. Generally, we can state that very high degrees of neutrality degenerate optimization processes to random walk. On the other hand, some forms of neutral pathways can improve evolvability and hence increase the chance of finding good solutions.

5.1.2 *Redundancy*

Redundancy in the context of EAs is a feature of the genotype-phenotype mapping and it means that multiple genotypes are mapped 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[127]. There exist many accounts of its positive influence on the optimization process. In [128-129], redundant genotype-phenotype mappings are developed using voting (via uniform redundancy as well as a non-trivial approach), Turing machine-like binary instructions, cellular automata, and random Boolean networks (RBNs)[130]. Except for the trivial voting mechanism based on uniform redundancy, the mappings could induce neutral pathways that were beneficial for exploring the problem space. The RBN approach in particular provided very good results[128-129].

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 that are different from their parents but 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 candidate solutions can be reached[128].

In the Cartesian GP method, neutrality is explicitly introduced to increase evolvability[131-132]. Yet, simple uniform redundancy is not necessarily beneficial for the optimization process and may even slow it down[23,129]. If the population of individuals under investigation contains many *isomorphic* genotypes, i.e., genotypes that encode the same phenotype, a slow-down may also occur[56]. If this isomorphism can be identified and removed, a significant speed-up may be gained[56].

5.1.3 *Needle-in-a-Haystack Problems*

Besides fully deceptive problems, one of the worst cases found in fitness landscapes is the *needle-in-a-haystack* problem[27] (see Fig.9 and Fig.3(g)), where the

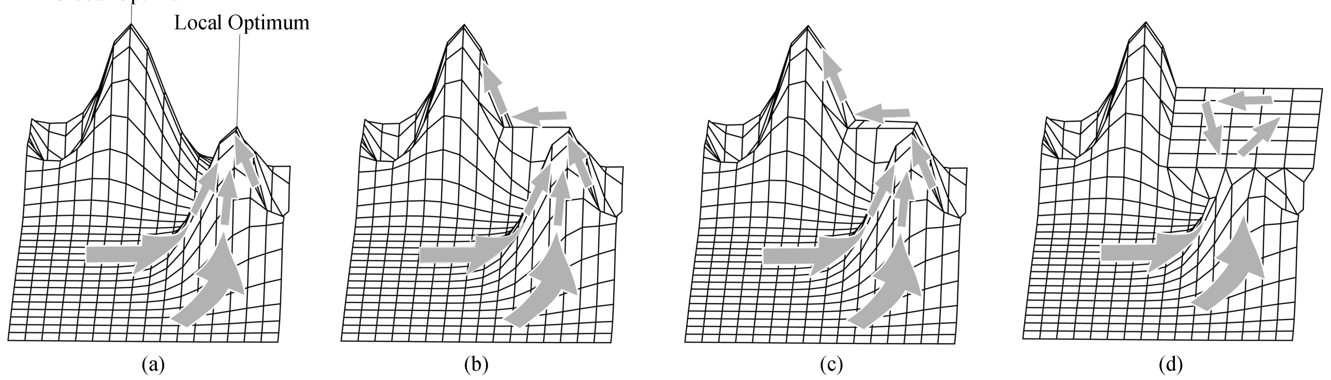


Fig.10. Possible positive and negative influence of neutrality (inspired by [125]). (a) Premature convergence. (b) Small neutral bridge. (c) Wide neutral bridge. (d) Neutral bridge too wide.

optimum occurs as an isolated spike in a plane[93,133]. In other words, this is the combination of small instances of extreme ruggedness with a general lack of information in the fitness landscape. Such problems are extremely hard to solve and the optimization process often will converge prematurely or take very long to find the global optimum. An example of this kind of fitness landscapes is the all-or-nothing property often inherent to GP[92-93,134-135].

5.2 Countermeasures

Extreme cases of neutrality, especially the needle-in-a-haystack-type fitness landscapes, are hard to combat. Hybridization of an EA with local search is sometimes recommended in such situations[83]. Multi-objectivization (see Subsection 8.2.2) and increasing the population size can possibly reduce the impact of neutrality too.

5.2.1 Selection Pressure

Higher selection pressure may be useful if the neutral regions in the fitness landscape still exhibit marginally different objective values that could be exploited to find a way out. It should be noted that fitness proportionate selection methods (e.g., "Roulette-Wheel Selection") may perform very badly in such a case, since they will assign the essentially same reproduction probability to all individuals. Other methods such as Tournament Selection, which only consider the *less-then* relation instead of absolute fitness values and proportions, will be not affected.

In the case where all objective values in the neutral regions are identical, a strong emphasis on diversity, possibly achieved by sharing and niching in the problem or search space (see Subsection 2.2.5), may drive the search out of the neutral region faster.

5.2.2 Representation

Uniform redundancy in the genome should be avoided as it causes adverse forms of neutrality. In [22, 136], it is stated that the representation of phenotypic traits in the search space should be as short as possible. The length of different genes and the numbers of their alleles should be as small as possible. However, as we discussed earlier, non-trivial representations with a well-adjusted degree of redundancy may exhibit a higher evolvability and thus lead to a more robust and steadily improving optimization process[132].

5.2.3 Memory

In Tabu Search, recently performed search steps are memorized and not performed again. This allows the algorithm to escape small neutral areas. Similar techniques could be applied in EAs as well.

6 Epistasis, Pleiotropy, and Separability

In biology, *epistasis* is defined as a form of interaction between different genes[137]. According to [138], the interaction between genes is epistatic if the effect of altering one gene on the fitness depends on the allelic state of other genes. In (evolutionary) optimization, epistasis is the non-linear interaction of two or more genes of the genotypes as expressed in objective function values after the genotype-phenotype mapping. Two genes interact epistatically if the contribution of one of these genes to the objective value depends on the value of the other gene[2,139-141]. Epistasis can also be considered as the higher-order or non-main effects in a model predicting fitness values based on the interactions of the genes from the viewpoint of Design of Experiments[77].

On one hand, 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 iteratively applied to each gene while keeping the others constant[139]. On the other hand, a problem is maximally epistatic when no proper subset of genes is independent of any other gene[141]. The effects of epistasis are closely related to another biological phenomenon: *Pleiotropy*, which denotes that a single gene is responsible for multiple phenotypical traits[109].

Like epistasis, pleiotropy can sometimes lead to unexpected improvements but often is harmful for an evolutionary system[114]. Both phenomena may easily intertwine. If one gene epistatically influences, for instance, two others that are responsible for distinct phenotypical traits, it has both epistatic and pleiotropic effects. We will therefore consider pleiotropy and epistasis together, and when discussing the effects of the latter, we also implicitly refer to the former.

In Fig.11, we illustrate a fictional dinosaur along with a snippet of its fictional genome consisting of four genes. Gene 1 influences the color of the creature and is neither pleiotropic nor has any epistatic relations. Gene 2, however, exhibits pleiotropy since it determines the length of the hind legs and forelegs. At the same time, it is epistatically connected with gene 3, which also influences the length of the forelegs — maybe preventing them from looking exactly like the hind legs. The fourth gene is again pleiotropic by determining the shape of the bone armors on the top of the dinosaur's skull and on its snout.

In the area of optimization over continuous problem spaces, epistasis and pleiotropy are closely related to

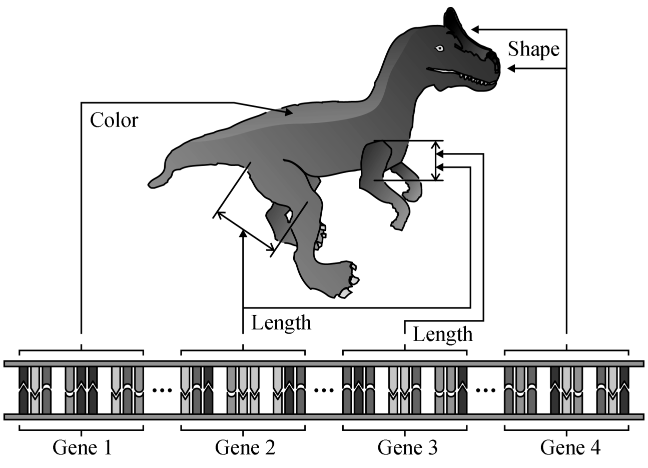


Fig.11. Pleiotropy and epistasis in a dinosaur's genome.

the term *separability*. Separability is a feature of the objective function(s) of an optimization problem[142]. A function of $\ell_{\mathbb{X}}$ variables is separable if it can be rewritten as a sum of $\ell_{\mathbb{X}}$ functions of just one variable[49,143]. Hence, the genes involved in the problem can be optimized independently of each other, i.e., are minimally epistatic, and the problem is said to be separable. A function f: $\mathbb{R}^{\ell_{\mathbb{X}}} \mapsto \mathbb{R}$ is separable[144] if and only if the condition given in (1) holds.

$$\arg\min_{x_1,\ldots,x_{\ell_{\mathbb{X}}}} f(x_1,\ldots,x_{\ell_{\mathbb{X}}}) = (\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_{\ell_{\mathbb{X}}}} f(\ldots,x_{\ell_{\mathbb{X}}})). \quad (1)$$

Otherwise, $f(\boldsymbol{x})$ is called a non-separable function. If a function $f(\boldsymbol{x})$ is separable, the parameters $x_1,\ldots,x_{\ell_{\mathbb{X}}}$ forming the candidate solution $\boldsymbol{x}$ are called independent. A separable problem is *decomposable*. A function f: $\mathbb{R}^{\ell_{\mathbb{X}}} \mapsto \mathbb{R}$ is k-non-separable if at most k of its parameters x_i are not independent. A non-separable function $f(\boldsymbol{x})$ is called fully non-separable if any two of its parameters x_i are not independent. The higher the degree of non-separability, the harder a function will usually become for optimization[144-145]. Often, the term *non-separable* is used in the sense of *fully non-separable*. In between separable and fully non-separable problems, a variety of *partially* separable problems exist.

6.1 The Issue

As sketched in Fig.12, epistasis has a strong influence on many of the previously discussed issues. If one variable (gene) of a point (genotype) in the search space can "turn off" or affect the expression of other genes, modifying 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 an 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. Representations and genotypes with low pleiotropy often lead to better and more robust solutions[146].

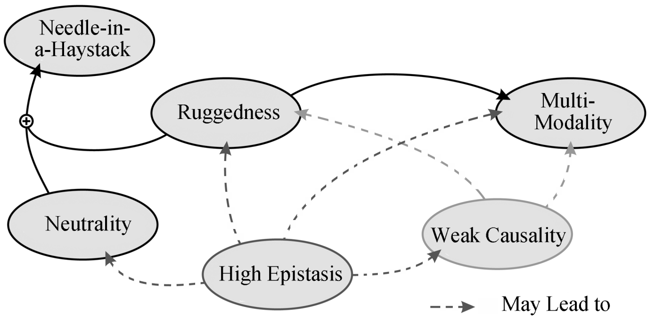


Fig.12. Influence of epistasis on the fitness landscape.

6.2 Countermeasures

Epistasis is a root cause for multiple related issues in optimization tasks. The symptoms of epistasis can be mitigated with the same methods that increase the chance of finding good solutions in the presence of ruggedness or neutrality. Other methods are discussed in the following.

6.2.1 *Choice of the Representation*

Epistasis itself is again an issue resulting from the choice of the search space structure, the search operations, the genotype-phenotype mapping, and the structure of the problem space. Avoiding epistatic effects should be a major concern during the design phase. Choosing the solution space and the genotype-phenotype mapping correctly can lead to great improvements in the quality of the solutions produced by the optimization process[92-93,135,147]. Introducing specialized search operations can achieve similar effects[148].

6.2.2 *Adjusting Selection Pressure*

Using larger populations and favoring explorative search operations could be helpful in epistatic problems, since these are ways to increase diversity. On the other hand, applying 1) higher selection pressure, i.e., increasing the chance of picking the best candidate solutions for further investigation instead of the weaker ones, and 2) *extinctive selection*, i.e., only working with the newest produced set of candidate solutions while discarding their parents, can also increase the reliability of an optimizer to find good solutions[148]. These two concepts are slightly contradicting, so careful adjustment of the algorithm settings appears to be vital in epistatic environments. Higher selection pressure also leads to earlier convergence[148], a fact we already discussed in Section 2.

6.2.3 Linkage and Interaction Learning

According to [149], *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 EAs, this notation is not useful since identifying spatially close elements inside the genotypes is trivial. Instead, we are interested in different genes that have a joint effect on the fitness[150].

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 (such as crossover in GAs), for instance. Finding approaches for *linkage learning* for binary[150-151] and real-valued[152] genomes has become a popular research area. Two important methods derived from this research are the messy GA (mGA)[153] and the Bayesian Optimization Algorithm (BOA)[154].

Module acquisition[155] may be considered as such an effort too. Here, an additional reproduction operation can group connected components of a genotype together into an atomic group, which becomes immune to modification by other reproduction operators. In GP, this is similar to adding a new automatically defined function that represents a subtree of the program individual.

Especially promising in numerical optimization is the Variable Interaction Learning (VIL) technique[156] that can detect which genes have non-separable relations. These are then grouped together and the resulting division of the genotypes can be optimized separately in a cooperative-coevolution approach[156-157], see Subsection 9.2.5.

7 Noise and Robustness

Noise is an undesired and unpredictable random disturbance to a signal. In the context of optimization, three types of noise can be distinguished[158]. The first form is noise in the objective functions or in the training data used[159]. 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. Besides inexactnesses and fluctuations in the input data of the optimization process, perturbations are also likely to occur during the application of its results, which takes place after the optimization has finished. This category subsumes the other two types of noise: perturbations that may arise from inaccuracies in the process of realizing the solutions *and* environmentally induced perturbations during the applications of the products. The effects of noise in optimization have been the subject of many studies[160-161]. Many optimization algorithms and theoretical results have been proposed to deal with noise. Some of them are, for instance, specialized GAs[162-163], Evolution Strategics (ESs)[164-165], and PSO algorithms[166].

7.1 Issue: Need for Robustness

The goal of 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 discussed, there will always be noise and perturbations in practical realizations of the results of optimization. Designs, plans, and procedures must address the fact that no process is perfect. As a result, practitioners may desire a relatively good and yet predictable solution that can tolerate a certain degree of imprecision during its application in lieu of a less predictable but globally optimal solution.

A system in engineering or biology is *robust* if it is able to function properly in the face of genetic or environmental perturbations[109]. 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[167]. 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.13 illustrates the issue of local optima which are robust vs global optima which are not.

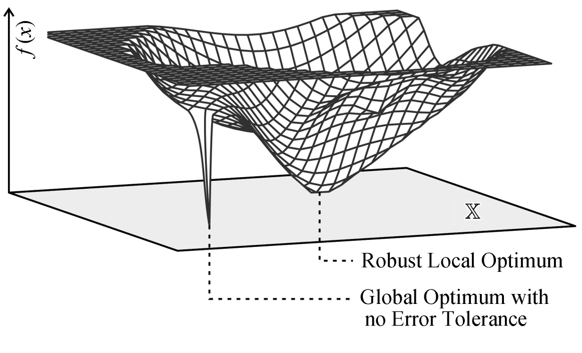


Fig.13. A robust local optimum vs an "unstable" global optimum.

7.2 Countermeasures

For the special case where the solution space is a real vector space, several approaches for dealing with the need for robustness have been developed. Inspired by Taguchi methods[168], possible disturbances are represented by a vector $\boldsymbol{\delta}$ in the method suggested in [169]. The objective function can be rewritten as $\tilde{f}(\boldsymbol{x}, \boldsymbol{\delta})$[170] if $\boldsymbol{\delta}$ follows a stochastic distribution with known (measured, approximated) parameters. The probability distribution of $\boldsymbol{\delta}$ then can be sampled a number of t times and the mean values of $\tilde{f}(\boldsymbol{x}, \boldsymbol{\delta})$ are used during the optimization process[170].

This method turns the optimization algorithm into something like a maximum likelihood estimator and also corresponds to using multiple, different training scenarios during the objective function evaluation. By adding random noise and artificial perturbations to the training cases, the chance of obtaining robust solutions that are stable when applied or realized under noisy conditions can be higher.

8 Dimensionality

Many engineering or scheduling problems involve multiple, often conflicting, optimization criteria. In logistic planning tasks[62-63], for instance, the goals are 1) to fulfill as many transportation orders within their respective time windows as possible, 2) at the lowest possible cost, and 3) with as little CO_2 emissions as possible. We refer to the number m of objective functions of an optimization problem as its *dimension* (or dimensionality). Later in this article, we will discuss issues arising from a large number of decision variables, which we put under the heading *scalability* in Section 9.

The most common way to define optima in multi-objective problems (MOPs) is to use the Pareto domination relation. A candidate solution x_1 is said to dominate another candidate solution x_2 $(x_1 \prec x_2)$ in an m-objective optimization problem if and only if its corresponding vector of objective values $\boldsymbol{f}(x_1)$ is (partially) less than the one of x_2, i.e., $i \in 1..m \Rightarrow f_i(x_1) \leqslant f_i(x_2)$ and $\exists i \in 1..m : f_i(x_1) < f_i(x_2)$, in minimization problems. More precisely, this is called *weak* dominance; *strong* dominance requires x_1 to be strictly better than x_2 in all objectives. However, the latter notion is usually not applied in the optimization domain. The solutions in the Pareto optimal set (also called *Pareto set* or *Pareto efficient frontier*) are not (weakly) dominated by any other solution in the problem space, i.e., globally optimal with respect to the dominance relation[171-173]. These are the elements we would like to find, or at least approximate as closely as possible, with optimization (see Fig.5 in Subsection 2.1.1).

Many studies in the literature consider mainly bi-objective problems[174]. Consequently, many algorithms have been designed to deal with that kind of problems. However, MOPs having a higher number of objective functions are common in practice — sometimes the number of objectives reaches double figures[175] — leading to the so-called *many-objective optimization*[33,174,176-178]. This term has been coined by the Operations Research community to denote problems with more than two or three objective functions[179].

8.1 Issue: Many-Objective Optimization

When the dimension of MOPs increases, the majority of the candidate solutions become non-dominated. Traditional multi-objective EAs (MOEAs), however, assign fitness mainly based on information about the Pareto domination relation in the population, usually combined with some diversity metric. Examples include the NSGA-II[180] (Pareto rank combined with the crowding distance in the objective space), SPEA-2[61] (Pareto-domination based strength together with distance to the k nearest neighbor in the objective space), and PESA[181] (Pareto domination and number of other individuals in the same hyper-box in a grid defined over the search space). It thus can be assumed that Pareto-based optimization approaches (maybe extended with diversity preservation methods) will not perform well in problems with four or more objectives[182]. Results from the application of such algorithms to two or three objectives cannot simply be extrapolated to larger numbers of optimization criteria[174]. In [183], Pareto optimality is considered as unfair and imperfect in many-objective problems and [182] indicated that:

1) an optimizer that produces an entire Pareto set in one run is better than generating the Pareto set through many single-objective optimizations using an aggregation approach if the number of objective function evaluations is fixed, and that

2) optimizers that use Pareto ranking based methods to sort the population will be very effective for small numbers of objectives, but not perform as effectively for many-objective optimization in comparison with methods based on other approaches.

The results in [174, 176, 184] further demonstrated the degeneration of the performance of traditional multi-objective metaheuristics in many-objective problems in comparison with single-objective approaches. Various elements distant from the true Pareto frontier may survive as hardly-dominated solutions and lead to a decrease in the probability of producing new candidate solutions dominating the existing ones[185]. This phenomenon is called *dominance resistance*. The problem of *redundant solutions* is recognized and demonstrated with an example function (provided as part of a test function suite for continuous multi-objective optimization) in [186].

In addition to these algorithm-sided limitations, [187] suggested that a human mind[188] will not be able to make efficient decisions if more than a dozen of objectives are involved. Visualizing the solutions in a human-understandable way becomes more complex with the rising number of dimensions too[189].

The number of non-dominated elements in random samples increases quickly with the dimension[190]. The hyper-surface of the Pareto frontier may increase exponentially with the number of objective functions[189]. Like in [189], we would like to illustrate this issue with an experiment.

Assume that a population-based optimization approach is used to solve a many-objective problem. The algorithm will fill the initial population with n randomly created individuals. The distribution of the probability $P(\#\text{dom} = o|m, n)$ that a randomly selected individual from this initial population is non-dominated (in this population) depends on the population size n and the number of objective functions m. We have *approximated* this probability distribution using experiments with n m-dimensional vectors where each element is drawn from the same uniform distribution for several values of m spanning from $m = 2$ to $m = 20$ and with $n = 3$ to $n = 3\,600$.

The fraction of non-dominated elements in the random populations is illustrated in Fig.14, based on the arithmetic means of 100 000 runs for each configuration. It rises (roughly) exponentially with m, whereas the population size n seems to have only an approximately logarithmically positive influence. If we list the population sizes required to keep the fraction of non-dominated candidate solutions at the same level as in the case of $n = 5$ and $m = 2$ (at around 0.457), we find that for $m = 3 \Rightarrow n \approx 12$, for $m = 4 \Rightarrow n \approx 35$, for $m = 5 \Rightarrow n \approx 90$, for $m = 6 \Rightarrow n \approx 250$, for $m = 7 \Rightarrow n \approx 650$, and for $m = 8 \Rightarrow n \approx 1\,800$. An extremely coarse rule of thumb here would hence be that around $0.6e^m$ individuals are required in the population to hold the proportion of non-dominated candidate solutions at around 46% in this experiment.

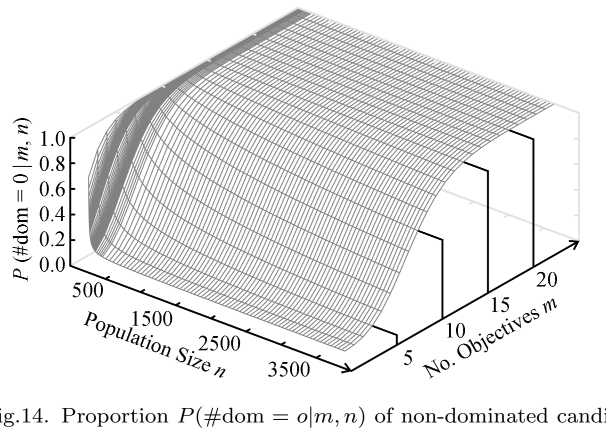


Fig.14. Proportion $P(\#\text{dom} = o|m, n)$ of non-dominated candidate solutions for several population sizes n and dimensionalities m.

The increasing dimensionality of the objective space leads to three main problems[189]:

1) The performance of traditional approaches based solely on Pareto comparisons deteriorates.

2) The utility of the solutions cannot be understood by the human operator anymore.

3) The number of possible Pareto-optimal solutions may increase exponentially.

8.2 Countermeasures

Various countermeasures have been proposed against the problem of dimensionality. Surveys on current approaches to many-objective optimization with EC methods, to the difficulties arising in many-objective and on benchmark problems, have been provided in [189, 191]. In the following we list a number of approaches for many-objective optimization, some of which are based on the information provided in [189].

8.2.1 *Increasing the Population Size*

The most trivial measure is to increase the population size. This, however, works only for a few objective functions and we have to "throw" in exponentially more individuals in order to neutralize the influence of many objectives (as can be seen in Fig.14). Hence, increasing the population size will not get us far.

8.2.2 *Multi-Archive Approaches*

On large numbers of objectives, traditional MOEAs usually exhibit *either* convergence close to the Pareto front *or* a good spread alongside it[174,176]. One possible solution is to use two archives[192-193] in the algorithms: one for diversity and one for convergence, with the goal to combine the two positive features.

8.2.3 *Increasing the Selection Pressure*

The way multi-objective approaches scale with increasing dimensionality can be improved by increasing the selection pressure into the direction of the Pareto frontier. Ishibuchi *et al.*[189] distinguished approaches that modify the definition of *domination* in order to reduce the number of non-dominated candidate solutions in the population[194] and methods that assign different ranks to non-dominated solutions[195-198]. Relying on fuzzy Pareto methods instead of pure Pareto comparisons is proposed in [179].

8.2.4 *Indicator Function-Based Approaches*

Fitness assignment methods not based on Pareto dominance can also be applied[189]. One approach is to use indicator functions such as those involving hypervolume metrics[199-200]. Hypervolume metrics have been shown to be able to approximate the Pareto frontier[201].

8.2.5 *Scalarizing Approaches*

Another possible countermeasure is to use scalarizing functions[189] for fitness assignment in order to treat many-objective problems with single-objective style methods. Several studies[182,199] showed that this method can produce better results than applying traditional MOEAs such as NSGA-II[180] or SPEA 2[61], but also refuted the idea that Pareto-based algorithms cannot cope with their performance in general. Other scalarizing methods can be found in [202-203].

8.2.6 *Limiting the Search Area in the Objective Space*

Furthermore, we can limit the search area in the objective space. This leads to a decrease in the number of non-dominated points[189] and can be achieved by either incorporating preference information[204-205] or by reducing the dimensionality[206-208].

8.2.7 *Visualization Methods*

Approaches for visualizing solutions of many-objective problems in order to make them more comprehensible have been provided in [209-210].

9 Scalability

An increasing number of objective functions can threaten the performance of optimization algorithms. We referred to this as the *dimensionality* problem, i.e., the dimension of the objective space. There is another space-related issue — the *"curse of dimensionality"* of the search space, i.e., the exponential increase of its volume with the number of genes (or decision variables)[211-212]. To better distinguish between the dimensionality of the objective space and the search space, we will refer to the latter as *scale*.

As an example, we illustrate small-scale versus large-scale problems using discrete or continuous vector-based search spaces. If we search, for instance, on one gene having values in the natural interval 1..10, there are ten points that could be the optimal solution. When the search space is composed of two such genes, i.e., $(1..10)^2$, there exist one hundred possible results and for $(1..10)^3$, it is already one thousand. In other words, the number of elements that could be a solution to an optimization problem grows exponentially with the number of genes.

9.1 The Issue

The issue of scale has already been introduced in Subsection 1.3, where we discussed the computational complexity as a measure of how many algorithm steps are needed to solve a problem consisting of $\ell_{\mathbb{X}}$ decision variables. As can be seen in Fig.15, if the number $t(\ell_{\mathbb{X}})$ of algorithm steps, i.e., the runtime, needed to solve a problem grows exponentially with the problem size $\ell_{\mathbb{X}}$, it quickly exceeds any feasible bound. However, in Subsection 1.3, the issue was considered from the perspective of deterministic algorithms, which are supposed to solve a problem to optimality. As a remedy for the infeasible runtime of these algorithms, we then suggested to apply stochastic optimization methods. Although these may be able to solve problems with a several magnitudes higher scale in a close-to-optimal way, their performance deteriorates with rising scales too.

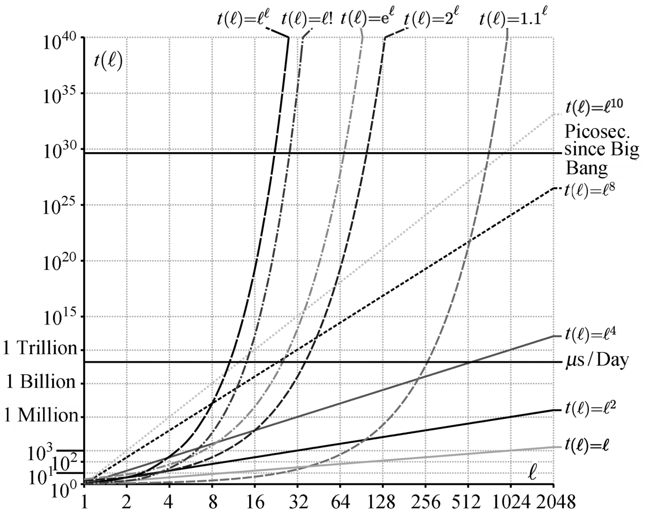


Fig.15. Illustration of the rising speed of some functions, inspired by [213].

9.2 Countermeasures

9.2.1 *Parallelization and Distribution*

When facing problems of large scale, the main "obstacle" is the high runtime requirement. Thus, any measure of using as much computational power as available can be a remedy. Obviously, there (currently) exists no way to solve large-scale $\mathcal{NP}$-hard problems exactly within feasible time. With more computers, cores, or hardware, a linear (actually, only sublinear[214]) improvement of runtime can be achieved at most. However, for problems residing in the grey area between feasible and infeasible, distributed computing[215] may be the method of choice.

There is a long tradition of parallelizing and distributing the computational workload in EC[216-219]. The basic ways to parallelize an EA are:

1) *Local Parallelization.* To parallelize the execution by using hardware with multiple CPUs[220] in a single computer, or, as is the current trend.

2) by utilizing modern graphics processing inits (GPUs)[221-222] to evaluate and process the individuals in a population in parallel.

3) *Parallel Restarts.* It is also possible to run different instances of the same algorithm on multiple CPUs or computers in a network at the same time, which would be a parallel version of the restarting strategy.

4) *Master/Slave Approach*[223]. If the evaluation of a candidate solution is very time consuming, this step can be parallelized to several workers (threads or computers in a network), which receive their task from a single central server maintaining a global population.

5) *Island Model*[224]. Alternatively, each node (or thread) may maintain an own population and, from time to time, exchange promising candidate solutions with neighboring nodes in the topology.

6) Of course, any combination of the above is possible[219].

9.2.2 Generative Representations

Another way, possibly the best way, to tackle a large-scale problem is to "solve" it as a small-scale problem. For some optimization tasks, it is possible to choose a search space $\mathbb{G}$ having a smaller size (e.g., a small number $\ell_{\mathbb{G}}$ of genes) than the problem space $\mathbb{X}$ (i.e., having $\ell_{\mathbb{X}} > \ell_{\mathbb{G}}$ decision variables). Indirect genotype-phenotype mappings can link the spaces together.

Here, one option is the generative mapping, which step-by-step constructs a complex phenotype by translating a genotype according to some static rules. Grammatical evolution[225], for instance, unfolds a start symbol according to a grammar with rules identified in a genotype. This recursive process can basically lead to arbitrarily complex phenotypes.

9.2.3 Developmental Representations

Applying a developmental, ontogenic mapping[18,226] that uses feedback from simulations or objective functions in the process of building a candidate solution is another possible countermeasure. If, for instance, a rigid truss composed of $\ell_{\mathbb{X}} = 600$ beams is to be found, instead of optimizing the volumes of each of the beams directly, the goal would be to find a suitable function that receives as a parameter the mechanical stress on a given beam and returns how much the cross section of the beam should be increased.

Beginning with a basic beam structure, the mechanical stress is evaluated and the function is applied to each of the beams. The updated truss is simulated again and the process is repeated a couple of times. The resulting structure would be the phenotype. The genotype can be an artificial neural network representing the function, encoded as real vectors containing the neural weights, thus having much fewer variables (e.g., $\ell_{\mathbb{G}} = 12$). Moreover, $\ell_{\mathbb{G}}$ is independent from $\ell_{\mathbb{X}}$, and therefore, much larger problem spaces can become tangible and excellent results may be obtained in reasonable time, likely with better quality and faster than using generative mappings[18].

9.2.4 Adaptive Encodings

Somewhat in between purely generative and a developmental approach is the Dynamic Parameter Encoding (DPE) method[21], which is basically a dynamic genotype-phenotype mapping for binary-encoded real vectors. Traditionally, the number of bits in each gene is fixed and corresponds to the desired precision. In DPE, the interval in the problem space represented by each gene is assigned dynamically, iteratively shrinking down from the full range: If the GA used for optimization has converged to some values of the bits of a gene, a *zooming* operation changes the meaning of that gene to now represent the corresponding sub-interval only. This way, the number of bits needed to achieve a given solution precision can be significantly reduced. In other words, although it still needs one gene in the genotype per decision variable in the phenotype, the genes themselves only consist of a few bits (e.g., three) and are thus much more compact than in fixed genotype-phenotype mappings.

9.2.5 Exploiting Separability

If a large-scale problem cannot be solved as a single small-scale problem, solving it as multiple small-scale problems may be another option for saving runtime. Sometimes, parts of candidate solutions are independent from each other and can be optimized more or less separately. In such a case (low epistasis, see Section 6), a large-scale problem can be divided into several components of smaller-scale to be optimized separately. If solving a problem of scale $\ell_{\mathbb{X}}$ takes $2^{\ell_{\mathbb{X}}}$ algorithm steps, solving two problems of scale $0.5\ell_{\mathbb{X}}$ will clearly lead to a great runtime reduction. Such a reduction may even be worth the sacrifice of some solution quality. If the optimization problem at hand exhibits low epistasis or is separable, such a sacrifice may even be avoided.

Coevolution has shown to be an efficient approach in combinatorial optimization[227]. If extended with a cooperative component (i.e., to Cooperative Coevolution[156-157]), it can efficiently exploit separability in numerical problems and lead to better results[156,228].

9.2.6 *Combination of Techniques*

Generally speaking, it can be a good idea to concurrently use different sets of algorithms[229] or portfolios[230] to work on the same or different populations. This way, the strengths of different optimization methods can be combined. In the beginning, for instance, an algorithm with good convergence speed may be granted more runtime. Later, the focus can shift towards methods that can retain diversity and are not prone to premature convergence. Alternatively, a sequential approach can be performed, which starts with one algorithm and switches to another one when no further improvements can be found[91]. By doing this, an interesting area in the search space can first be discovered, and then be investigated more thoroughly.

10 No Free Lunch Theorem

So far, we have discussed various difficulties that could arise when applying an optimization algorithm to a given problem. The fact that not a single optimization method is likely to be able to outperform all other methods on all problems can easily be accepted. Instead, we see a variety of optimization methods specialized in solving different types of problems. There are also algorithms that may deliver good results for many different problem classes, but could be outperformed by highly specialized methods in each of them. These facts have been formalized by Wolpert and Macready[231] in their *No Free Lunch Theorems* for search and optimization algorithms.

The performance of an algorithm a executed for p steps on an optimization problem can be defined as the conditional probability of finding a particular sample (such as the global optimum). Wolpert and Macready[231] proved that the sum of such probabilities over all possible optimization problems on *finite* domains is always identical for all optimization algorithms. This means that the average performance over all finite problems is independent of the algorithm applied. From this theorem, we can immediately follow that, in order to outperform algorithm a_1 in one optimization problem, algorithm a_2 will necessarily perform worse in another problem, as sketched in Fig.16. This implies that it is impossible for any optimization algorithm to always outperform non-repeating random walks or exhaustive enumerations.

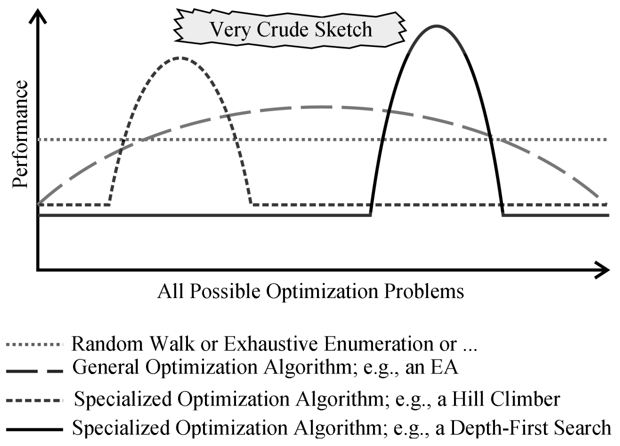


Fig.16. A visualization of the No Free Lunch Theorem.

In practice, an optimizer is not applied to all possible problems but to only some, restricted classes. In terms of these classes, it is well possible to perform comparisons and to make statements regarding which algorithms perform the best (which, by the way, is often the topic of challenges and competitions[142-143]). Furthermore, it was recently shown that the No Free Lunch Theorem holds only in a weaker form for countable infinite and not for continuous domains[232].

Another interpretation of the No Free Lunch Theorem is that every useful optimization algorithm utilizes some form of problem-specific knowledge. In [233], it is stated that without such knowledge, search algorithms cannot exceed the performance of simple enumerations. Incorporating knowledge starts with relying on simple assumptions like causality (see Subsection 3.1). The more problem specific knowledge is integrated into the algorithm structure, the better the algorithm can perform[18].

11 Concluding Remarks

The subject of this article is to address questions about issues that make optimization problems difficult to solve, with a particular focus on evolutionary optimization. We have discussed a variety of scenarios that can influence/affect the optimization process and lead to disappointing results.

If an optimization process has converged prematurely, it is said to be 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) may either slow down optimization or contribute positively. Noise is present in virtually all practical optimization problems. The solutions that are derived for them should thus be robust (Section 7). Also, many practical problems are multi-objective in nature, i.e., involve the optimization of more than one criterion at a time (see Section 8).

The No Free Lunch Theorem argues that it is not possible to develop a *universal* optimization algorithm, the problem-solving machine that can provide us with near-optimal solutions in short time for every possible optimization task in finite domains. Such a statement may sound depressing for those who are 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 that 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 could excel in. This fact has contributed to the emergence of memetic, hybrid and the new area of portfolio-type algorithms[230], which combine different optimization methods.

It is most likely that the "puzzle"[⑤] of optimization algorithms as sketched in Fig.17 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 that performs better in some problem areas than all the currently known ones.

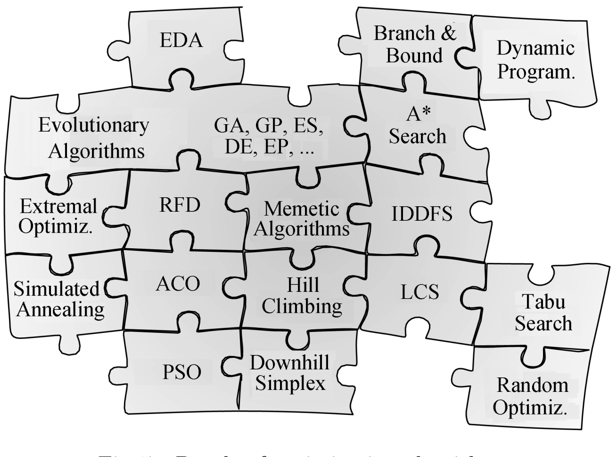


Fig.17. Puzzle of optimization algorithms.

References

[1] Blum C, Chiong R, Clerc M, De Jong K A, Michalewicz Z, Neri F, Weise T. Evolutionary optimization. In *Variants of Evolutionary Algorithms for Real-World Applications*, Chiong R, Weise T, Michalewicz Z (eds.), Berlin/Heidelberg: Springer-Verlag, 2011, pp.1-29.

[2] Weise T. Global Optimization Algorithms – Theory and Application. Germany: it-weise.de (self-published), 2009. http://www.it-weise.de/projects/book.pdf.

[3] Eiben Á E, Smith J E. Introduction to Evolutionary Computing (Natural Computing Series). New York, USA: Springer New York, 2003.

[4] Chiong R, Weise T, Michalewicz Z (eds.). Variants of Evolutionary Algorithms for Real-World Applications. Berlin/Heidelberg: Springer-Verlag, 2011.

[5] Whitley L D. A genetic algorithm tutorial. *Statistics and Computing*, 1994, 4(2): 65-85.

[6] Michalewicz Z. Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Germany: Springer-Verlag GmbH, 1996.

[7] Coello Coello C A. A short tutorial on evolutionary multiobjective optimization. In *Proc. the 1st International Conference on Evolutionary Multi-Criterion Optimization* (*EMO2001*), Zürich, Switzerland, March 7-9, 2001, pp.21-40.

[8] Coello Coello C A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 2002, 191(11-12): 1245-1287.

[9] Trojanowski K, Michalewicz Z. Evolutionary algorithms and the problem-specific knowledge. In *Proc. the 2nd National Conference on Evolutionary Computation and Global Optimization*, Rytro, Poland, September 16-19, 1997, pp.281-292.

[10] Chiong R, Dhakal S (eds.). Natural Intelligence for Scheduling, Planning and Packing Problems. Berlin/Heidelberg: Springer-Verlag, 2009.

[11] Chiong R (ed.). Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science and Engineering. Hershey, PA, USA: Information Science Reference, 2009.

[12] Chiong R (ed.). Nature-Inspired Algorithms for Optimisation. Berlin/Heidelberg: Springer-Verlag, 2009.

[13] Chen T, Tang K, Chen G, Yao X. Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 2010, 14(1): 1-22.

[14] Chen T, He J, Chen G, Yao X. Choosing selection pressure for wide-gap problems. *Theoretical Computer Science*, 2010, 411(6): 926-934.

[15] He J, Yao X. Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 2003, 145(1-2): 59-97.

[16] He J, Reeves C R, Witt C, Yao X. A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. *Evolutionary Computation*, 2007, 15(4): 435-443.

[17] Lochtefeld D F, Ciarallo F W. A diversity classification scheme for genetic algorithms. In *Proc. the 61st Annual IIE Conference and Expo* (IERC2011), Reno, NV, USA, May 21-25, 2011.

[18] Devert A, Weise T, Tang K. A study on scalable representations for evolutionary optimization of ground structures. *Evolutionary Computation*, 2012, 20(3): 453-472.

[19] De Jong K A. An analysis of the behavior of a class of genetic adaptive systems [Ph.D. Thesis]. University of Michigan, 1975.

[20] Wright A H. Genetic algorithms for real parameter optimization. In *Proc. the 1st Workshop on Foundations of Genetic*

[⑤]So far undefined abbreviations: RFD — river formation ynamics, IDDFS — iteratively deepening depth-first search, ACO — ant colony optimization, LCS — learning classifier systems.

Algorithms (*FOGA1990*), Bloomington, IN, USA, July 15-18, 1990, pp.205-218.

[21] Schraudolph N N, Belew R K. Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 1992, 9(1): 9-21.

[22] Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[23] Rothlauf F. Representations for Genetic and Evolutionary Algorithms. Heidelberg, Germany: Physica-Verlag GmbH & Co., 2002.

[24] Grefenstette J J. Deception considered harmful. In *Proc. the 2nd Workshop on Foundations of Genetic Algorithms* (*FOGA1992*), Vail, CO, USA, July 26-29, 1992, pp.75-91.

[25] Leblanc B, Lutton E. Bitwise regularity and GA-hardness. In *Proc. the 1998 IEEE International Conference on Evolutionary Computation* (*CEC1998*), Anchorage, AK, USA, May 4-9, 1998, pp.517-522.

[26] Naudts B, Kallel L. Some facts about so called GA-hardness measures. Rapport Interne (R.I.) 379, Centre de Mathématiques APpliquées (CMAP), 1998.

[27] Borenstein Y, Poli R. Fitness distributions and GA hardness. In *Proc. the 8th International Conference on Parallel Problem Solving from Nature*, Birmingham, UK, September 18-22, 2008, pp.11-20.

[28] Guo H, Hsu W H. GA-hardness revisited. In *Proc. the Genetic and Evolutionary Computation Conference* (*GECCO2003*), Part I, Chicago, IL, USA, July 12-16, 2003, pp.1584-1585.

[29] Oliveto P S, He J, Yao X. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 2007, 4(3): 281-293.

[30] Oliveto P S, He J, Yao X. Analysis of the (1+1)-ea for finding approximate solutions to vertex cover problems. *IEEE Trans. Evolutionary Computation*, 2009, 13(5): 1006-1029.

[31] Horn J, Goldberg D E. Genetic algorithm difficulty and the modality of the fitness landscape. In *Proc. the 3rd Workshop on Foundations of Genetic Algorithms*, Estes Park, CO, USA, July 31-August 2, 1994, pp.243-269.

[32] Singh G, Deb K. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *Proc. the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, USA, July 8-12, 2006, pp.1305-1312.

[33] Weise T, Zapf M, Chiong R, Nebro Urbaneja A J. Why is optimization difficult? In *Nature-Inspired Algorithms for Optimisation, Studies in Computational Intelligence 193/2009*, Chiong R (ed.), Berlin/Heidelberg: Springer-Verlag, 2009, pp.1-50.

[34] Rudnick W M. Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks [Ph.D. Thesis]. Oregon Graduate Institute of Science & Technology, 1992.

[35] Thierens D, Goldberg D E, Pereira Â G. Domino convergence, drift, and the temporal-salience structure of problems. In *Proc. the 1998 IEEE International Conference on Evolutionary Computation*, Anchorage, AK, USA, May 4-9, 1998, pp.535-540.

[36] Mitchell M, Forrest S, Holland J H. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Proc. the 1st European Conference on Artificial Life* (*ECAL1991*), Paris, France, December 11-13, 1991, pp.245-254.

[37] Paenke I, Branke J, Jin Y. On the influence of phenotype plasticity on genotype diversity. In *Proc. the 1st IEEE Symposium on Foundations of Computational Intelligence* (*FOCI2007*), Honolulu, HI, USA, April 1-5, 2007, pp.33-40.

[38] Holland J H. Genetic algorithms — Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, 1992, 267(1): 44-50.

[39] Holland J H. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[40] Eiben Á E, Schippers C A. On evolutionary exploration and exploitation. *Fundamenta Informaticae*, 1998, 35(1-4): 35-50.

[41] Glover F. Tabu search — Part ii. *ORSA Journal on Computing*, 1990, 2(1): 190-206.

[42] Nolte A, Schrader R. A note on the finite time behaviour of simulated annealing. *Mathematics of Operations Research*, 2000, 25(3): 476-484.

[43] Ingber L. Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 1993, 18(11): 29-57.

[44] Lehre P K, Yao X. On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2012, 16(2): 225-241.

[45] Oliveto P S, Lehre P K, Neumann F. Theoretical analysis of rank-based mutation — Combining exploration and exploitation. In *Proc. the 10th IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 18-21, 2009, pp.1455-1462.

[46] Muttil N, Liong S. Superior exploration-exploitation balance in shuffled complex evolution. *Journal of Hydraulic Engineering*, 2004, 130(12): 1202-1205.

[47] Amor H B, Rettinger A. Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation. In *Proc. the Genetic and Evolutionary Computation Conference* (*GECCO2005*), Washington, DC, USA, June 25-27, 2005, pp.1531-1538.

[48] Nguyen Q H, Ong Y, Lim M H, Krasnogor N. Adaptive cellular memetic algorithms. *Evolutionary Computation*, 2009, 17(2): 231-256.

[49] Ortiz-Boyer D, Hervás-Martínez C, García C A R. Cixl2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research*, 2005, 24: 1-48.

[50] Lee C, Yao X. Evolutionary programming using the mutations based on the Lévy probability distribution. *IEEE Transactions on Evolutionary Computation*, 2004, 8(1): 1-13.

[51] Yao X. An empirical study of genetic operators in genetic algorithms. *Microprocessing and Microprogramming*, 1993, 38(1-5): 707-714.

[52] Feo T A, Resende M G. Greedy randomized adaptive search procedures. *J. Global Optimization*, 1995, 6(2): 109-133.

[53] van Nimwegen E, Crutchfield J P. Optimizing epochal evolutionary search: Population-size dependent theory. *Machine Learning*, 2001, 45(1): 77-114.

[54] Chen T, Tang K, Chen G, Yao X. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 2012, 436, June: 54-70.

[55] He J, Yao X. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2002, 6(5): 495-511.

[56] Ronald S, Asenstorfer J, Vincent M. Representational redundancy in evolutionary algorithms. In *Proc. the 2nd IEEE International Conference on Evolutionary Computation* (*CEC1995*), Perth, WA, Australia, November 29-December 1, 1995, pp.631-637.

[57] Goldberg D E, Richardson J T. Genetic algorithms with sharing for multimodal function optimization. In *Proc. the 2nd International Conference on Genetic Algorithms and their Applications*, Cambridge, MA, USA, July 28-31, 1987, pp.41-49.

[58] Deb K, Goldberg D E. An investigation of niche and species formation in genetic function optimization. In *Proc. the 3rd International Conference on Genetic Algorithms* (*ICGA 1989*), Fairfax, VA, USA, June 4-7, 1989, pp.42-50.

[59] Goldberg D E, Deb K, Horn J. Massive multimodality, deception, and genetic algorithms. In *Proc. the 2nd Parallel Problem Solving from Nature 2*, Brussels, Belgium, September 28-30, 1992, pp.37-48.

[60] Zitzler E, Thiele L. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. TIK-Report 43, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Eidgenössische Technische Hochschule (ETH) Zürich, 1998.

[61] Zitzler E, Laumanns M, Thiele L. Spea 2: Improving the strength pareto evolutionary algorithm. TIK-Report 101, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Eidgenössische Technische Hochschule (ETH) Zürich, 2001.

[62] Weise T, Podlich A, Gorldt C. Solving real-world vehicle routing problems with evolutionary algorithms. In *Natural Intelligence for Scheduling, Planning and Packing Problems* (Studies in Computational Intelligence 250), Chiong R, Dhakal S (eds.), Berlin/Heidelberg: Springer-Verlag, 2009, pp.29-53.

[63] Weise T, Podlich A, Reinhard K, Gorldt C, Geihs K. Evolutionary freight transportation planning. In *Lecture Notes in Computer Science 5484*, Giacobini M, Brabazon A, Cagnonj S *et al.* (eds.), Springer-Verlag, 2009, pp.768-777.

[64] Pétrowski A. A clearing procedure as a niching method for genetic algorithms. In *Proc. the IEEE International Conference on Evolutionary Computation* (*CEC1996*), Nagoya, Japan, May 1996, pp.798-803.

[65] Darwen P J, Yao X. Every niching method has its niche: Fitness sharing and implicit sharing compared. In *Proc. the 4th International Conference on Parallel Problem Solving from Nature*, Berlin, Germany, September 22-24, 1996, pp.398-407.

[66] Weise T, Niemczyk S, Chiong R, Wan M. A framework for multi-model edas with model recombination. In *Proc. the EvoApplications 2011*, Part 1, Torino, Italy, April 27-29, 2011, pp.304-313.

[67] Lu Q, Yao X. Clustering and learning gaussian distribution for continuous optimization. *IEEE Transactions on Systems, Man, and Cybernetics — Part C: Applications and Reviews*, 2005, 35(2): 195-204.

[68] Cao A, Chen Y, Wei J, Li J. A hybrid evolutionary algorithm based on edas and clustering analysis. In *Proc. the 26th Chinese Control Conference* (*CCC2007*), Zhangjiajie, Hunan, China, July 26-31, 2007, pp.754-758.

[69] Pelikan M, Sastry K, Goldberg D E. Multiobjective HBOA, clustering, and scalability. In *Proc. the Genetic and Evolutionary Computation Conference* (*GECCO2005*), Washington, DC, USA, June 25-27, 2005, pp.663-670.

[70] Rudolph G. Self-adaptation and global convergence: A counter-example. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC1999*), Washington, DC, USA, July 6-9, 1999, pp.646-651.

[71] Rudolph G. Self-adaptive mutations may lead to premature convergence. *IEEE Transactions on Evolutionary Computation*, 2001, 5(4): 410-414.

[72] Lochtefeld D F, Ciarallo F W. Helper-objective optimization strategies for the job-shop scheduling problem. *Applied Soft Computing*, 2011, 11(6): 4161-4174.

[73] Knowles J D, Watson R A, Corne D W. Reducing local optima in single-objective problems by multi-objectivization. In *Proc. the 1st International Conference on Evolutionary Multi-Criterion Optimization* (*EMO2001*), Zürich, Switzerland, March 7-9, 2001, pp.269-283.

[74] Jensen M T. Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. *Journal of Mathematical Modelling and Algorithms*, 2004, 3(4): 323-347.

[75] Jähne M, Li X, Branke J. Evolutionary algorithms and multi-objectivization for the travelling salesman problem. In *Proc. the 11th Annual Conference on Genetic and Evolutionary Computation*, Montréal, QC, Canada, July 8-12, 2009, pp.595-602.

[76] Lochtefeld D F, Ciarallo F W. Deterministic helper objective sequence applied to the job-shop scheduling problem. In *Proc. the Genetic and Evolutionary Computation Conference*, Portland, OR, USA, July 7-11, 2010, pp.431-438.

[77] Lochtefeld D F, Ciarallo F W. Multiobjectivization via helper-objectives with the tunable objectives problem. *IEEE Transactions on Evolutionary Computation*, 2011, 16(3): 373-390.

[78] Handl J, Lovell S C, Knowles J D. Multiobjectivization by decomposition of scalar cost functions. In *Proc. the 10th International Conference on Parallel Problem Solving from Nature*, Dortmund, North Rhine-Westphalia, Germany, September 13-17, 2008, pp.31-40.

[79] Rechenberg I. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution [Ph.D. Thesis]. Technische Universität Berlin, Stuttgart, Germany, 1971.

[80] Rechenberg I. Evolutionsstrategie '94. Bad Cannstadt, Stuttgart, Baden-Württemberg, Germany: Frommann-Holzboog Verlag, 1994.

[81] Kolarov K. Landscape ruggedness in evolutionary algorithms. In *Proc. the IEEE International Conference on Evolutionary Computation*, Indianapolis, IN, USA, April 13-16, 1997, pp.19-24.

[82] Whitley L D, Gordon V S, Mathias K E. Lamarckian evolution, the baldwin effect and function optimization. In *Proc. the 3rd Conference on Parallel Problem Solving from Nature*, Jerusalem, Israel, October 9-14, 1994, pp.6-15.

[83] Hinton G E, Nowlan S J. How learning can guide evolution. *Complex Systems*, 1987, 1(3): 495-502.

[84] Holstein D, Moscato P. Memetic algorithms using guided local search: A case study. In *New Ideas in Optimization*, Corne D W, Dorigo M, Glover F, Dasgupta D, Moscato P, Poli R, Price K V (eds.), Maidenhead, England: McGraw-Hill Ltd., 1999, pp.235-244.

[85] Moscato P, Cotta C. A gentle introduction to memetic algorithms. In *Handbook of Metaheuristics*, Glover F, Kochenberger G A (eds.), Norwell, MA, USA: Kluwer Academic Publishers and Springer Netherlands, 2003, pp.105-144.

[86] Radcliffe N J, Surry P D. Formal memetic algorithms. In *Lecture Notes in Computer Science 865*, Fogarty T C (ed.), Springer-Verlag, 1994, pp.1-16.

[87] Mühlenbein H. How genetic algorithms really work — I. mutation and hillclimbing. In *Proc. the Parallel Problem Solving from Nature 2* (*PPSN II*), Brussels, Belgium, September 28-30, 1992, pp.15-26.

[88] Davis L (ed.). Handbook of Genetic Algorithms. Stamford, CT, USA: Thomson Publishing Group, Inc., 1991.

[89] Gruau F, Whitley L D. Adding learning to the cellular development of neural networks: Evolution and the baldwin effect. *Evolutionary Computation*, 1993, 1(3): 213-233.

[90] Liang K, Yao X, Newton C S. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 2000, 4(3): 172-183.

[91] Wang Y, Li B, Weise T. Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. *Information Sciences*

— *Informatics and Computer Science Intelligent Systems Applications*, 2010, 180(12): 2405-2420.

[92] Weise T, Tang K. Evolving distributed algorithms with genetic programming. *IEEE Transactions on Evolutionary Computation*, 2011, 16(2): 242-265.

[93] Weise T. Evolving distributed algorithms with genetic programming [Ph.D. Thesis]. Distributed Systems Group, University of Kassel, 2009.

[94] Goldberg D E. Genetic algorithms and walsh functions: Part i, a gentle introduction. *Complex Systems*, 1989, 3(2): 129-152.

[95] Goldberg D E. Genetic algorithms and walsh functions: Part ii, deception and its analysis. *Complex Systems*, 1989, 3(2): 153-171.

[96] Liepins G E, Vose M D. Deceptiveness and genetic algorithm dynamics. In *Proc. the 1st Workshop on Foundations of Genetic Algorithms* (*FOGA1990*), Bloomington, IN, USA, July 15-18, 1990, pp.36-50.

[97] Weise T, Niemczyk S, Skubch H, Reichle R, Geihs K. A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes. In *Proc. the Genetic and Evolutionary Computation Conference* (*GECCO2008*), Atlanta, GA, USA, July 12-16, 2008, pp.795-802.

[98] Schnier T, Yao X. Using multiple representations in evolutionary algorithms. In *Proc. the IEEE Congress on Evolutionary Computation*, La Jolla, CA, USA, July 16-19, 2000, pp.479-486.

[99] Li X. Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, 2010, 14(1): 150-169.

[100] Hutter M. Fitness uniform selection to preserve genetic diversity. In *Proc. the IEEE Congress on Evolutionary Computation*, Honolulu, HI, USA, May 12-17, 2002, pp.783-788.

[101] Hutter M, Legg S. Fitness uniform optimization. *IEEE Trans. Evolutionary Computation*, 2006, 10(5): 568-589.

[102] Lehman J, Stanley K O. Exploiting open-endedness to solve problems through the search for novelty. In *Proc. the 11th International Conference on Artificial Life*, Winchester, Hampshire, UK, August 5-8, 2008, pp.329-336.

[103] Lehman J, Stanley K O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 2011, 19(2): 189-223.

[104] Lehman J, Stanley K O. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proc. the Genetic and Evolutionary Computation Conference*, Dublin, Ireland, July 12-16, 2011, pp.211-218.

[105] Wan M, Weise T, Tang K. Novel loop structures and the evolution of mathematical algorithms. In *Proc. the 14th European Conference on Genetic Programming* (*EuroGP2011*), Torino, Italy, April 27-29, 2011, pp.49-60.

[106] Weise T. Evolving distributed algorithms with genetic programming. In *Proc. the 1st ACM/SIGEVO Sunmit on Genetic and Evolutionary Computation*, Shanghai, China, June 12-14, 2009, pp.577-584.

[107] Reidys C M, Stadler P F. Neutrality in fitness landscapes. *Journal of Applied Mathematics and Computation*, 2001, 117(2-3): 321-350.

[108] Barnett L. Ruggedness and neutrality — the NKP family of fitness landscapes. In *Proc. the 6th International Conference on Artificial Life* , Los Angeles, CA, USA, June 27-29, 1998, pp.18-27.

[109] Hu T, Banzhaf W. Evolvability and speed of evolutionary algorithms in the light of recent developments in biology. *Journal of Artificial Evolution and Applications*, 2010, January, Article No.1.

[110] Wagner A. Robustness, evolvability, and neutrality. *FEBS Letters*, 2005, 579(8): 1772-1778.

[111] Kirschner M, Gerhart J. Evolvability. *Proc. the National Academy of Science of the United States of America* (*PNAS*), 1998, 95(15): 8420-8427.

[112] Beyer H. Toward a theory of evolution strategies: The (μ, λ)-theory. *Evolutionary Computation*, 1994, 2(4): 381-407.

[113] Altenberg L. Fitness distance correlation analysis: An instructive counterexample. In *Proc. the 7th International Conference on Genetic Algorithms* (*ICGA1997*), East Lansing, MI, USA, July 19-23, 1997, pp.57-64.

[114] Altenberg L. The schema theorem and price's theorem. In *Proc. the 3rd Workshop on Foundations of Genetic Algorithms*, Estes Park, CO, USA, July 31-August 2, 1994, pp.23-49.

[115] Yu G T. Program evolvability under environmental variations and neutrality. In *Proc. the 9th European Conference on Advances in Artificial Life* (*ECAL2007*), Lisbon, Portugal, September 10-14, 2007, pp.835-844.

[116] Beaudoin W, Vérel S, Collard P, Escazut C. Deceptiveness and neutrality the ND family of fitness landscapes. In *Proc. the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, USA, July 8-12, 2006, pp.507-514.

[117] Kimura M. The Neutral Theory of Molecular Evolution. Cambridge, UK: Cambridge University Press, 1985.

[118] Toussaint M, Igel C. Neutrality: A necessity for self-adaptation. In *Proc. the IEEE Congress on Evolutionary Computation*, Honolulu, USA, May 12-17, 2002, pp.1354-1359.

[119] Gould S J, Eldredge N. Punctuated equilibrium comes of age. *Nature*, 1993, 366(6452): 223-227.

[120] van Nimwegen E, Crutchfield J P, Mitchell M. Statistical dynamics of the royal road genetic algorithm. *Theoretical Computer Science*, 1999, 229(1-2): 41-102.

[121] Cohoon J P, Hegde S U, Martin W N, Richards D. Punctuated equilibria: A parallel genetic algorithm. In *Proc. the 2nd International Conference on Genetic Algorithms and their Applications*, Cambridge, USA, July 28-31, 1987, pp.148-154.

[122] Martin W N, Lienig J, Cohoon J P. Island (migration) models: Evolutionary algorithms based on punctuated equilibria. In *Handbook of Evolutionary Computation*, Bäck T, Fogel D B, Michalewicz Z (eds.), New York, USA: Oxford University Press, Inc., Institute of Physics Publishing Ltd. (IOP), and CRC Press, Inc., 1997, pp.448-463.

[123] Edelman G M, Gally J A. Degeneracy and complexity in biological systems. *Proc. the National Academy of Science of the United States of America*, 2001, 98(24): 13763-13768.

[124] Whitacre J M, Rohlfshagen P, Bender A, Yao X. The role of degenerate robustness in the evolvability of multi-agent systems in dynamic environments. In *Proc. the 11th International Conference on Parallel Problem Solving From Nature*, Part 1, Kraków, Poland, September 11-15, 2010, pp.284-293.

[125] Smith T, Husbands P, Layzell P, O'Shea M. Fitness landscapes and evolvability. *Evolutionary Computation*, 2002, 10(1): 1-34.

[126] Preuß M, Schönemann L, Emmerich M T. Counteracting genetic drift and disruptive recombination in (μ^{+}, λ)-ea on multimodal fitness landscapes. In *Proc. the Genetic and Evolutionary Computation Conference* (*GECCO2005*), Washington, DC, USA, June 25-27, 2005, pp.865-872.

[127] Weicker K, Weicker N. Burden and benefits of redundancy. In *Proc. the 6th Workshop on Foundations of Genetic Algorithms*, Charlottesville, USA, July 21-23, 2001, pp.313-333.

[128] Shipman R, Shackleton M, Ebner M, Watson R A. Neutral search spaces for artificial evolution: A lesson from life. In *Proc. the 7th International Conference on Artificial Life*, Portland, OR, USA, August 1-2, 2000, pp.162-167.

[129] Shackleton M, Shipman R, Ebner M. An investigation of redundant genotype-phenotype mappings and their role in

evolutionary search. In *Proc. the IEEE Congress on Evolutionary Computation*, La Jolla, USA, July 16-19, 2000, pp.493-500.

[130] Kauffman S A. The Origins of Order: Self-Organization and Selection in Evolution. New York, USA: Oxford University Press, Inc., 1993.

[131] Vassilev V K, Miller J F. The advantages of landscape neutrality in digital circuit evolution. In *Proc. the 3rd International Conference on Evolvable Systems — From Biology to Hardware*, Edinburgh, Scotland, UK, April 17-19, 2000, pp.252-263.

[132] Yu G T, Miller J F. Finding needles in haystacks is not hard with neutrality. In *Proc. the 5th European Conference on Genetic Programming*, Kinsale, Ireland, April 3-5, 2002, pp.46-54.

[133] Goldberg D E. Making genetic algorithm fly: A lesson from the wright brothers. *Advanced Technology for Developers*, 1993, 2: 1-8.

[134] Tschudin C F. Fraglets — A metabolistic execution model for communication protocols. In *Proc. the 2nd Annual Symposium on Autonomous Intelligent Networks and Systems* (*AINS2003*), Menlo Park, CA, USA, June 30-July 1, 2003.

[135] Weise T, Zapf M. Evolving distributed algorithms with genetic programming: Election. In *Proc. the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, Shanghai, China, June 12-14, 2009, pp.577-584.

[136] Ronald S. Robust encodings in genetic algorithms: A survey of encoding issues. In *Proc. the IEEE International Conference on Evolutionary Computation* (*CEC1997*), Indianapolis, IN, USA, April 13-16, 1997, pp.43-48.

[137] Phillips P C. The language of gene interaction. *Genetics*, 1998, 149(3): 1167-1171.

[138] Lush J L. Progeny test and individual performance as indicators of an animal's breeding value. *Journal of Dairy Science*, 1935, 18(1): 1-19.

[139] Davidor Y. Epistasis variance: A viewpoint on GA-hardness. In *Proc. the 1st Workshop on Foundations of Genetic Algorithms*, Bloomington, IN, USA, July 15-18, 1990, pp.23-35.

[140] Altenberg L. Nk fitness landscapes. In *Handbook of Evolutionary Computation*, Bäck T, Fogel D B, Michalewicz Z (eds.), New York, USA: Oxford University Press, Inc., Institute of Physics Publishing Ltd. (IOP), and CRC Press, Inc., 1997, chapter B2.7.2.

[141] Naudts B, Verschoren A. Epistasis on finite and infinite spaces. In *Proc. the 8th International Conference on Systems Research, Informatics and Cybernetics*, Baden-Baden, Baden-Württemberg, Germany, August 14-18, 1996, pp.19-23.

[142] Tang K, Yao X, Suganthan P N, MacNish C, Chen Y, Chen C, Yang Z. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, 2007.

[143] Tang K, Li X, Suganthan P N, Yang Z, Weise T. Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Technical Report, Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, 2010.

[144] Auger A, Hansen N, Mauny N, Ros R, Schoenauer M. Bio-inspired continuous optimization: The coming of age. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2007*), Singapore, September 25-28, 2007.

[145] Sutton A M, Lunacek M, Whitley L D. Differential evolution and non-separability: Using selective pressure to focus search. In *Proc. the 9th Genetic and Evolutionary Computation Conference*, London, UK, July 7-11, 2007, pp.1428-1435.

[146] Bowers C P. Simulating evolution with a computational model of embryogeny: Obtaining robustness from evolved individuals. In *Proc. the 8th European Conference on Advances in Artificial Life*, Canterbury, Kent, UK, September 5-9, 2005, pp.149-158.

[147] Weise T, Zapf M, Geihs K. Rule-based genetic programming. In *Proc. the 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, Budapest, Hungary, December 10-13, 2007, pp.8-15.

[148] Shinkai M, Aguirre A H, Tanaka K. Mutation strategy improves gas performance on epistatic problems. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2002*), Honolulu, HI, USA, May 12-17, 2002, pp.968-973.

[149] Winter P C, Hickey G I, Fletcher H L. Instant Notes in Genetics (1st edition). Oxford, UK: BIOS Scientific Publishers Ltd., Taylor and Francis LLC, Science Press, 1998.

[150] Munetomo M, Goldberg D E. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 1999, 7(4): 377-398.

[151] Harik G R. Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms [Ph.D. Thesis]. University of Michigan, 1997.

[152] Deb K, Sinha A, Kukkonen S. Multi-objective test problems, linkages, and evolutionary methodologies. In *Proc. the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, USA, July 8-12, 2006, pp.1141-1148.

[153] Goldberg D E, Deb K, Korb B. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 1989, 3(5): 493-530.

[154] Cantú-Paz E, Pelikan M, Goldberg D E. Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation*, 2000, 8(3): 311-340.

[155] Angeline P J, Pollack J B. Evolutionary module acquisition. In *Proc. the 2nd Annual Conf. Evolutionary Programming*, La Jolla, CA, USA, February 25-26, 1993, pp.154-163.

[156] Chen W, Weise T, Yang Z, Tang K. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *Proc. the 11th International Conference on Parallel Problem Solving From Nature*, Part 2, Kraków, Poland, September 11-15, 2010, pp.300-309.

[157] Potter M A, De Jong K A. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 2000, 8(1): 1-29.

[158] Jin Y, Branke J. Evolutionary optimization in uncertain environments — A survey. *IEEE Transactions on Evolutionary Computation*, 2005, 9(3): 303-317.

[159] Yang S, Ong Y, Jin Y (eds.). Evolutionary Computation in Dynamic and Uncertain Environments. Berlin/Heidelberg: Springer-Verlag, 2007.

[160] Miller B L, Goldberg D E. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 1996, 4(2): 113-131.

[161] Lee J Y, Wong P C. The effect of function noise on GP efficiency. In *Lecture Notes in Computer Science 956*, Goos G, Hartmanis J, van Leeuwen (eds.), 1993, pp.1-16.

[162] Fitzpatrick J M, Grefenstette J J. Genetic algorithms in noisy environments. *Machine Learning*, 1998, 3(2-3): 101-120.

[163] Sano Y, Kita H. Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In *Proc. the IEEE Congress on Evolutionary Computation*, Honolulu, USA, May 12-17, 2002, pp.360-365.

[164] Bäck T, Hammel U. Evolution strategies applied to perturbed objective functions. In *Proc. the 1st IEEE Conference on Evolutionary Computation*, Orlando, USA, June 27-29, 1994, pp.40-45.

[165] Hammel U, Bäck T. Evolution strategies on noisy functions: How to improve convergence properties. In *Proc. the 3rd Conference on Parallel Problem Solving from Nature*, Jerusalem, Israel, October 9-14, 1994, pp.159-168.

[166] Pan H, Wang L, Liu B. Particle swarm optimization for function optimization in noisy environment. *Journal of Applied Mathematics and Computation*, 2006, 181(2): 908-919.

[167] Branke J. Creating robust solutions by means of evolutionary algorithms. In *Proc. the 5th International Conference on Parallel Problem Solving from Nature*, Amsterdam, The Netherlands, September 27-30, 1998, pp.119-128.

[168] Taguchi G. Introduction to Quality Engineering: Designing Quality into Products and Processes. Chiyoda-ku, Tokyo, Japan: Asian Productivity Organization (APO) and Kraus International Publications, 1986.

[169] Greiner H. Robust optical coating design with evolutionary strategies. *Applied Optics*, 1996, 35(28): 5477-5483.

[170] Wiesmann D, Hammel U, Bäck T. Robust design of multilayer optical coatings by means of evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 1998, 2(4): 162-167.

[171] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 2000, 8(2): 173-195.

[172] Coello Coello C A. Evolutionary multiobjective optimization. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2011, 1(5): 444-447.

[173] Rudolph G. On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *Proc. the 1998 IEEE International Conference on Evolutionary Computation*, Anchorage, AK, USA, May 4-9, 1998, pp.511-515.

[174] Khare V, Yao X, Deb K. Performance scaling of multi-objective evolutionary algorithms. In *Proc. the 2nd International Conference on Evolutionary Multi-Criterion Optimization*, Faro, Portugal, April 8-11, 2003, pp.367-390.

[175] Coello Coello C A, Lamont G B, van Veldhuizen D A. Evolutionary Algorithms for Solving Multi-Objective Problems. Boston, MA, USA: Springer US and Kluwer Academic Publishers, 2002.

[176] Khare V. Performance scaling of multi-objective evolutionary algorithms [Master Thesis]. School of Computer Science, University of Birmingham, 2002.

[177] López Jaimes A, Santana-Quintero L V, Coello Coello C A. Ranking methods in many-objective evolutionary algorithms. In *Nature-Inspired Algorithms for Optimisation*, *Studies in Computational Intelligence 193/2009*, Chiong R (ed.), Berlin/Heidelberg: Springer-Verlag, 2009, pp.413-434.

[178] Purshouse R C. On the evolutionary optimisation of many objectives [Ph.D. Thesis]. Department of Automatic Control and Systems Engineering, University of Sheffield, 2003.

[179] Farina M, Amato P. On the optimal solution definition for many-criteria optimization problems. In *Proc. the Annual Meeting of the North American Fuzzy Information Processing Society*, New Orleans, USA, June 27-29, 2002, pp.233-238.

[180] Deb K, Pratab A, Agrawal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 2002, 6(2): 182-197.

[181] Corne D W, Knowles J D, Oates M J. The pareto envelope-based selection algorithm for multiobjective optimization. In *Proc. the 6th International Conference on Parallel Problem Solving from Nature*, Paris, France, September 18-20, 2000, pp.839-848.

[182] Hughes E J. Evolutionary many-objective optimisation: Many once or one many? In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2005*), Edinburgh, Scotland, UK, September 2-5, 2005, pp.222-227.

[183] Kang Z, Kang L, Zou X, Liu M, Li C, Yang M, Li Y, Chen Y, Zeng S. A new evolutionary decision theory for many-objective optimization problems. In *Proc. the 2nd International Symposium on Advances in Computation and Intelligence*, Wuhan, Hubei, China, September 21-23, 2007, pp.1-11.

[184] Ishibuchi H, Nojima Y, Doi T. Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2006*), Vancouver, BC, Canada, July 16-21, 2006, pp.3959-3966.

[185] Ikeda K, Kita H, Kobayashi S. Failure of pareto-based moeas: Does non-dominated really mean near to optimal? In *Proc. the IEEE Congress on Evolutionary Computation*, Gangnam-gu, Seoul, Korea, May 27-30, 2001, pp.957-962.

[186] Deb K, Thiele L, Laumanns M, Zitzler E. Scalable multi-objective optimization test problems. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2002*), Honolulu, HI, USA, May 12-17, 2002, pp.825-830.

[187] Bouyssou D. Building criteria: A prerequisite for MCDA. In *Selected Readings from the 3rd International Summer School on Multicriteria Decision Aid: Methods, Applications, and Software* (*MCDA1990*), Monte Estoril, Lisbon, Portugal, July 23-27, 1990, pp.58-80.

[188] Miller G A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 1956, 63(2): 81-97.

[189] Ishibuchi H, Tsukamoto N, Nojima Y. Evolutionary many-objective optimization: A short review. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2008*), Hong Kong, China, June 1-6, 2008, pp.2424-2431.

[190] Deb K. Multi-Objective Optimization Using Evolutionary Algorithms. New York, USA: John Wiley & Sons Ltd., 2001.

[191] López Jaimes A, Coello Coello C A. Some techniques to deal with many-objective problems. In *Proc. the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference* (*GECCO2009*), Montréal, QC, Canada, July 8-12, 2009, pp.2693-2696.

[192] Praditwong K, Yao X. A new multi-objective evolutionary optimisation algorithm: The two-archive algorithm. In *Proc. the 2006 International Conference on Computational Intelligence and Security* (*CIS2006*), Guangzhou, China, November 3-6, 2006, pp.286-291.

[193] Praditwong K, Harman M, Yao X. Software module clustering as a multi-objective search problem. *IEEE Transactions on Software Engineering*, 2011, 37(2): 264-282.

[194] Sato H, Aguirre A H, Tanaka K. Controlling dominance area of solutions and its impact on the performance of moeas. In *Proc. the 4th International Conference on Evolutionary Multi-Criterion Optimization* (*EMO2007*), Matsushima, Sendai, Japan, March 5-8, 2007, pp.5-20.

[195] Drechsler N, Drechsler R, Becker B. Multi-objective optimisation based on relation *favour*. In *Proc. the 1st International Conference on Evolutionary Multi-Criterion Optimization* (*EMO2001*), Zürich, Switzerland, March 7-9, 2001, pp.154-166.

[196] Köppen M, Yoshida K. Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In *Proc. the 4th International Conference on Evolutionary Multi-Criterion Optimization* (*EMO2007*), Matsushima, Sendai, Japan, March 5-8, 2007, pp.727-741.

[197] Corne D W, Knowles J D. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *Proc. the 9th Genetic and Evolutionary Computation Conference*, London, UK, July 7-11, 2007, pp.773-780.

[198] Kukkonen S, Lampinen J A. Ranking-dominance and many-objective optimization. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2007*), Singapore, September 25-28, 2007, pp.3983-3990.

[199] Wagner T, Beume N, Naujoks B. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Proc. the 4th International Conference on Evolutionary Multi-Criterion Optimization* (*EMO2007*), Matsushima, Sendai, Japan, March 5-8, 2007, pp.742-756.

[200] Ishibuchi H, Tsukamoto N, Nojima Y. Iterative approach to indicator-based multiobjective optimization. In *Proc the IEEE Congress on Evolutionary Computation* (*CEC2007*), Singapore, September 25-28, 2007, pp.3967-3974.

[201] Bringmann K, Friedrich T. The maximum hypervolume set yields near-optimal approximation. In *Proc. the Genetic and Evolutionary Computation Conference* (*GECCO2010*), Portland, OR, USA, July 7-11, 2010, pp.511-518.

[202] Ishibuchi H, Doi T, Nojima Y. Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. In *Proc. the 9th International Conference on Parallel Problem Solving from Nature*, Reykjavik, Iceland, September 9-13, 2006, pp.493-502.

[203] Hughes E J. MSOPS-II: A general-purpose many-objective optimiser. In *Proc. the IEEE Congress on Evolutionary Computation*, Singapore, September 25-28, 2007, pp.3944-3951.

[204] Fleming P J, Purshouse R C, Lygoe R J. Many-objective optimization: An engineering design perspective. In *Proc. the 3rd International Conference on Evolutionary Multi-Criterion Optimization*, Guanajuato, México, March 9-11, 2005, pp.14-32.

[205] Deb K, Sundar J. Reference point based multi-objective optimization using evolutionary algorithms. In *Proc. the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, USA, July 8-12, 2006, pp.635-642.

[206] Brockhoff D, Zitzler E. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Proc. the 9th International Conference on Parallel Problem Solving from Nature*, Reykjavik, Iceland, September 9-13, 2006, pp.533-542.

[207] Saxena D K, Deb K. Dimensionality reduction of objectives and constraints in multi-objective optimization problems: A system design perspective. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2008*), Hong Kong, China, June 1-6, 2008, pp.3204-3211.

[208] Brockhoff D, Zitzler E. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *Proc. the IEEE Congress on Evolutionary Computation* (*CEC2007*), Singapore, September 25-28, 2007, pp.2086-2093.

[209] Furuhashi T, Yoshikawa T. Visualization techniques for mining of solutions. In *Proc. the 8th International Symposium on Advanced Intelligent Systems* (*ISIS2007*), Sokcho, Korea, September 5-8, 2007, pp.68-71.

[210] Köppen M, Yoshida K. Visualization of pareto-sets in evolutionary multi-objective optimization. In *Proc. the 7th International Conference on Hybrid Intelligent Systems*, Kaiserslautern, Germany, September 17-19, 2007, pp.156-161.

[211] Bellman R E. Dynamic Programming. Princeton, NJ, USA: Princeton University Press, 1957.

[212] Bellman R E. Adaptive Control Processes: A Guided Tour. Princeton, NJ, USA: Princeton University Press, 1961.

[213] Sabharwal A. Combinatorial problems i: Finding solutions. In *the 2nd Asian-Pacific School on Statistical Physics and Interdisciplinary Applications*, Beijing, China, March 3-14, 2008.

[214] Amdahl G M. Validity of the single processor approach to achieving large-scale computing capabilities. In *Proc. the Spring Joint Computer Conference* (*AFIPS*), Atlantic City, NJ, USA, April 18-20, 1967, pp.483-485.

[215] Liu P, Lau F C M, Lewis M J, Wang C. A new asynchronous parallel evolutionary algorithm for function optimization. In *Proc. the 7th International Conference on Parallel Problem Solving from Nature*, Granada, Spain, September 7-11, 2002, pp.401-410.

[216] Cantú-Paz E. A survey of parallel genetic algorithms. *Calculateurs Parallèles, Réseaux et Systèmes Répartis*, 1998, 10(2): 141-171.

[217] Alba Torres E, Troya J M. A survey of parallel distributed genetic algorithms. *Complexity*, 1999, 4(4): 31-52.

[218] Alba Torres E, Tomassini M. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2002, 6(5): 443-462.

[219] Weise T, Geihs K. DGPF — An adaptable framework for distributed multi-objective search algorithms applied to the genetic programming of sensor networks. In *Proc. the 2nd International Conference on Bioinspired Optimization Methods and their Applications* (*BIOMA2006*), Ljubljana, Slovenia, October 9-10, 2006, pp.157-166.

[220] Hauser R, Männer R. Implementation of standard genetic algorithm on mimd machines. In *Proc. the 3rd Conference on Parallel Problem Solving from Natur*, Jerusalem, Israel, October 9-14, 1994, pp.504-513.

[221] Langdon W B, Banzhaf W. A SIMD interpreter for genetic programming on GPU graphics cards. In *Proc. the 11th European Conference on Genetic Programming* (*EuroGP2008*), Naples, Italy, March 26-28, 2008, pp.73-85.

[222] Zhu W. Nonlinear optimization with a massively parallel evolution strategy-pattern search algorithm on graphics hardware. *Applied Soft Computing*, 2011, 11(2): 1770-1781.

[223] Dubreuil M, Gagné C, Parizeau M. Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics*, 2006, 36(1): 229-235.

[224] Tongchim S, Yao X. Parallel evolutionary programming. In *Proc. the IEEE Congress on Evolutionary Computation*, Portland, OR, USA, June 20-23, 2004, pp.1362-1367.

[225] O'Neill M, Ryan C. Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. New York, USA: Springer Science+Business Media, Inc., 2003.

[226] Devert A. Building processes optimization: Toward an artificial ontogeny based approach [Ph.D. Thesis]. Centre de Recherche Saclay — Île-de-France, Ecole Doctorale d'Informatique and Institut National de Recherche en Informatique et en Automatique, Université Paris-Sud, 2009.

[227] Husbands P, Mill F. Simulated co-evolution as the mechanism for emergent planning and scheduling. In *Proc. the 4th International Conference on Genetic Algorithms* (*ICGA 1991*), San Diego, CA, USA, July 13-16, 1991, pp.264-270.

[228] Yang Z, Tang K, Yao X. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences — Informatics and Computer Science Intelligent Systems Applications*, 2008, 178(15).

[229] LaTorre A, Peña J M, Muelas S, Zaforas M. Hybrid evolutionary algorithms for large scale continuous problems. In *Proc. the 11th Annual Conference on Genetic and Evolutionary Computation* (*GECCO2009*), Montréal, QC, Canada, July 8-12, 2009, pp.1863-1864.

[230] Peng F, Tang K, Chen G, Yao X. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 2010, 14(5): 782-800.

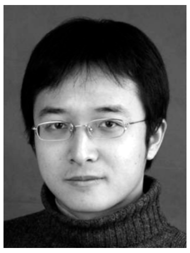
[231] Wolpert D H, Macready W G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 67-82.

[232] Auger A, Teytaud O. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 2010, 57(1): 121-146.

[233] Radcliffe N J. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 1994, 10(4): 339-384.

Thomas Weise received the Diplom Informatiker (equivalent to M.Sc.) degree from the Department of Computer Science, Chemnitz University of Technology, Germany, in 2005, and the Ph.D. degree at the Distributed Systems Group of the Fachbereich Elektrotechnik and Informatik, University of Kassel, Germany in 2009. Since 2009, he is with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China. His major research interests include evolutionary computation, genetic programming (GP), and real-world applications of optimization algorithms. His experience ranges from applying GP to distributed systems and multi-agent systems, efficient web service composition for Service Oriented Architectures, to solving large-scale real-world vehicle routing problems for multimodal logistics and transportation. Besides being the author/co-author of over 60 refereed publications, Dr. Weise also authors the electronic book Global Optimization Algorithms – Theory and Application which is freely available at his website http://www.it-weise.de/.

Raymond Chiong is with the Faculty of Higher Education Lilydale, Swinburne University of Technology, Australia. He has been lecturing in computer science/information systems for many years. His teaching has focused on programming and databases. Besides teaching, he has been actively pursuing research in computational and artificial intelligence, with a particular interest in the applications of nature-inspired computational methodologies to various optimisation, scheduling and planning problems. He is the Editor-in-Chief of the Interdisciplinary Journal of Information, Knowledge, and Management (IJIKM), and Editor of the journal Engineering Applications of Artificial Intelligence (EAAI). He also serves on the review board/program committee of several international journals and conferences. To date, he has more than 70 refereed publications in books, journals and conference proceedings.



Ke Tang received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002 and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007. In 2007, he joined the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, where he was promoted to Professor in 2011. He is the author/co-author of more than 60 refereed publications. His major research interests include evolutionary computation, machine learning, and their real-world applications. Dr. Tang is an associate editor of IEEE Computational Intelligence Magazine and the Chair of the IEEE Task Force on Collaborative Learning and Optimization. He served as a program co-chair of 2010 IEEE Congress on Evolutionary Computation, held in Barcelona.