UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

Perhaps Not a Free Lunch But At Least a Free Appetizer

Stefan Droste, Thomas Jansen, and Ingo Wegener

No. CI-45/98

Technical Report ISSN 1433-3325 September 1998

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

PERHAPS NOT A FREE LUNCH BUT AT LEAST A FREE APPETIZER*

Stefan Droste Thomas Jansen Ingo Wegener
FB Informatik, LS II, Univ. Dortmund, 44221 Dortmund, Germany
droste, jansen, wegener@ls2.cs.uni-dortmund.de

Abstract

It is often claimed that Evolutionary Algorithms are superior to other optimization techniques, in particular, in situations where not much is known about the objective function to be optimized. In contrast to that Wolpert and Macready (1997) proved that all optimization techniques have the same behavior — on average over all $f: X \to Y$ where X and Y are finite sets. This result is called No Free Lunch Theorem. Here different scenarios of optimization are presented. It is argued why the scenario on which the No Free Lunch Theorem is based does not model real life optimization. For more realistic scenarios it is argued why optimization techniques differ in their efficiency. For a small example this claim is proved.

*This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (531).

1 Introduction

The importance of optimization problems in computer science, many other research areas, industrial applications, and even in everyday life is evident. Since a long time a lot of optimization techniques have been presented and applied, among them gradient methods, Lagrangian optimization, greedy methods, divide-and-conquer, dynamic programming, local search, branch-and-bound, simulated annealing, and many variants of Evolutionary Algorithms. Some of these techniques only work under certain assumptions, others are claimed to be robust, i. e., to be useful for all types of problems. The techniques have been compared for certain types of problems but no one was able to prove that one technique is superior to another one in general. Nevertheless, several advocates of Evolutionary Algorithms have stated that Evolutionary Algorithms are superior when averaging over all types of problems. This has been disproved by the No Free Lunch Theorem due to Wolpert and Macready (1995). Before stating the result we precisely describe the underlying scenario.

Scenario 1: (No Free Lunch Scenario) The problem is drawn randomly from $F = \{f : X \to Y\}$ where X and Y are finite sets and Y is completely ordered. The aim is to find some $x \in X$ such that $f(x)$ is maximal (or minimal). The resources used by an algorithm for f are measured by the number of different x such that $f(x)$ is evaluated. The assumption of finite sets X and Y is not really a restriction.

Theorem 1 (No Free Lunch Theorem): With respect to Scenario 1, all optimization techniques (deterministic and even randomized ones) have the same average behavior.

This result is subject to intense and controversial discussions (compare Culberson (1996), Radcliffe and Surry (1995), as well as the No Free Lunch Theorem discussion page at `http://lucy.ipk.fhg.de/~mario/nfl/` or the Yin-Yang page about the panel discussion at the International Conference on Genetic Algorithms 1995 at `http://www.aic.nrl.navy.mil/~spears/yin-yang.html`) that are still going on today (compare recent discussions in `comp.ai.genetic`). In Section 2, we present a list of other scenarios for optimization. Common to all scenarios is that we have a set $F = \{f : X \to Y\}$, where X and Y are finite sets. Furthermore, there is a set $F' \subseteq F$ that contains the functions that may be subject to optimization. We investigate the average performance of optimization algorithms over all $f \in F'$. In the No Free Lunch Scenario we have $F' = F$. We shortly discuss scenarios where some structural knowledge on the function to be optimized is given. Then optimization techniques like Evolutionary Algorithms certainly can have an advantage. But this is no argument in favor of the claim that Evolutionary Algorithms do better than other optimization techniques, e. g., on average over "all" problems. We present a general black box optimization scenario for classes of functions whose complexity or difficulty is restricted in some sense. These scenarios are discussed in detail in Section 3. We give arguments why this scenario fits much better to real life optimization than Scenario 1. Then it is argued how optimization techniques can take advantage from the knowledge that the complexity of the considered function is bounded.

In Section 4, the discussion is focused on the possible advantage of Evolutionary Algorithms in black box optimization. Unfortunately, we cannot prove rigorously that different optimization techniques in general behave differently in these scenarios. Indeed, experiences from complexity theory indicate that such a general proof most probably is not possible. In Section 5, we restrict ourselves to small sets X and Y. Then we can prove that we gain something by choosing a clever optimization technique. Since X and Y are quite small, we cannot gain a lot. Therefore, what we get is "perhaps not a free lunch but at least a free appetizer". Or to state it more seriously: For black box optimization with the only assumption that the functions are not unrealistically complex it is possible to gain something by choosing a good optimization technique.

2 Scenarios for Optimization

The No Free Lunch Theorem holds in Scenario 1 where all functions $f : X \to Y$ are considered. In a specific situation one has to optimize one specific function.

Scenario 2: (One Shot Scenario). A function $f : X \to Y$ (X and Y finite, Y completely ordered) has to be optimized.

It makes no sense to compare optimization techniques in this scenario. One may be lucky and start with some optimal $x^* \in X$. Therefore, a problem (from a scientific point of view) has to have a lot of problem instances.

Scenario 3: (Fixed Problem Scenario). We are concerned with a fixed type of problem, e. g., sorting, linear programming, or the traveling salesman problem TSP.

In this scenario we have some kind of semantic knowledge about the set F' of functions that can be subject to optimization. This is the classical situation attacked by methods from the research area called Efficient Algorithms. We guess that there are no doubts that specific algorithms like quicksort or heapsort for sorting or the simplex method for linear programming can be superior to general techniques. For other problems like TSP the situation is more involved. Specific algorithms are very successful but only recently Michalewicz (1998) has presented a problem specific crossover operator (inver-over) and has obtained some impressive results with Evolutionary Algorithms. It seems to be necessary to use problem specific components for Evolutionary Algorithms in order to compete with other problem specific algorithms.

Scenario 4: (Fixed Function Type Scenario). It is known that the function f is chosen from some class of functions sharing some structural property like separability, unimodality or being a polynomial of small degree.

Here, we have some kind of syntactic knowledge about the set F' of possible objective functions. For simple classes of functions, e. g., separable functions or linear functions, simple optimization techniques are so fast that no general technique can compete with them. In more complex situations general techniques, among them variants of Evolutionary Algorithms, are very successful. There is a lot of literature discussing these successes, we refer for Evolutionary Strategies to Rechenberg (1994) and Schwefel (1995), for Evolutionary Programming to Fogel (1995), for Genetic Algorithms to Goldberg (1989) and Holland (1975), and for Genetic Programming to Koza (1992). But all these scenarios are far from the No Free Lunch Scenario and, moreover, successes of Evolutionary Algorithms in one of these scenarios do not support the claim that Evolutionary Algorithms adapt to (almost) each type of problem. For such a discussion we suggest the following scenario.

Scenario 5: (Restricted Black Box Optimization). This scenario is the same as the No Free Lunch Scenario with the only exception that $F = \{f : X \to Y\}$ is replaced by some subset F' of functions $f : X \to Y$ whose complexity (in a sense to be specified) is not too large.

In Section 3, we discuss some realistic restrictions. Here we motivate this scenario with some general remarks. In order to apply optimization techniques which evaluate the considered function f at a lot of points x it is necessary that the computation of $y = f(x)$ can be done efficiently. Hence, the claim that Evolutionary Algorithms are superior to other optimization techniques is meaningful only in some type of restricted black box scenario. The No Free Lunch Theorem holds in the no free lunch scenario which also may be called unrestricted black box optimization scenario. The question is whether the No Free Lunch Theorem can be generalized to *restricted* black box optimization scenarios.

3 Some Realistic Black Box Optimization Scenarios

We look for scenarios of restricted black box optimization which do not exclude some interesting problem from consideration. The first idea is to consider only functions from the complexity class P, i. e., functions where one single evaluation is computable in polynomial time. This complexity class usually is defined with respect to Turing machines which leads people to deny its practical significance. But the definition is robust, i. e., we may replace Turing machines by random access machines or by our favorite computer (with a finite number of processors), given that the memory is sufficiently large for the task at hand. The typical NP-hard optimization problems have objective functions which can be evaluated in polynomial time, often in linear time and sometimes even in sublinear time. E. g., the distance matrix of a TSP instance with n cities has n^2 entries but the cost of a tour can be computed in $O(n)$ time. This indicates that it is meaningful to restrict the resource bounds for the evaluation of the objective function to bounds like $O(n^3)$, $O(n^2)$ or $O(n)$. Then the computation model has to be fixed and it would be stupid to consider Turing machines. Random access machines (see Garey and Johnson (1979) and Papadimitriou (1994)) are widely accepted as suitable model but the reader should feel free to choose his or her own model.

The problem with this approach is that we implicitly consider functions $f : \Sigma^* \to \mathbb{N}$, i. e., functions with an infinite support. This excludes by definition results like the No Free

Lunch Theorem. We restrict ourselves to finite functions, e. g., $f : \{0,1\}^n \to \{0,1\}^m$ for fixed n and m.

Scenario 5.1: (Time Restricted Black Box Optimization). The restriction is given by a time bound T on the number of steps to evaluate f.

Having fixed $X = \{0,1\}^n$ and $Y = \{0,1\}^m$ we have switched from uniform computation models (the input length is not fixed) to so-called nonuniform computation models (see Garey and Johnson (1979) for a thorough discussion of these terms). The most popular nonuniform computation model is the Boolean circuit with AND-, OR-, and NOT-gates. Although circuits are a hardware model, they can be used as appropriate model here. Time restricted computations lead to size restricted circuit representations where the size bound only is by a logarithmic factor larger than the given time bound (see Garey and Johnson (1979) and Wegener (1987)).

Scenario 5.2: (Size Restricted Black Box Optimization). The restriction is given by a bound s on the size of a representation of f, e. g., the circuit size.

The last restriction we like to introduce is a bound on the Kolmogoroff complexity (see the monograph of Li and Vitányi (1993)). This complexity measure is defined in a rather abstract way. The Kolmogoroff complexity of a sequence s of zeros and ones is, for some universal Turing machine, the length of the shortest program producing s. The essential Invariance Theorem says that the Kolmogoroff complexity changes only by a constant additive term if we replace one universal Turing machine by another. This theory seems to be too strange to have applications. But the theory is very robust and we may use it for our purposes. A function $f : \{0,1\}^n \to \{0,1\}^m$ can be described as list of all $f(x), x \in \{0,1\}^n$, and, therefore, as 0-1-sequence. Then we may replace the universal Turing machine by a programming language. Loosely speaking the Kolmogoroff complexity of f is the minimal length of a standard C++-program which computes the value table of f. Problems in applications have a small Kolmogoroff complexity, since we may write a program executing a loop which evaluates and lists all $f(x), x \in X$.

Scenario 5.3: (Kolmogoroff Complexity Restricted Black Box Optimization). The restriction is given by a bound b on the Kolmogoroff complexity of f.

Summarizing we claim that situations where Evolutionary Algorithms can or will be applied fall into the classes covered by the different restricted black box optimization scenarios. Hence, comparisons of optimization techniques should be performed within such scenarios.

4 Evolutionary Algorithms and Restricted Black Box Optimization

First, we recall the most important argument in the proof of the No Free Lunch Theorem. If all $f(x), x \in X' \subseteq X$, are known, then $g : X - X' \to Y$ defined by $g(x) = f(x)$ on the restricted input set is still a random function on $X - X'$. By evaluating f on some inputs x we learn the corresponding values $f(x)$ but nothing else. In restricted black box optimization we are not confronted with all functions $f : X \to Y$. Hence, even if $f(x)$ and also $f(x')$ takes each value from Y with equal probability, the information that $f(x) = y$ may lead to the conclusion that a small or large value of $f(x')$ is less ore more likely. Such information can be used by optimization techniques in restricted black box optimization scenarios.

Indeed, all explanations of the success of Evolutionary Algorithms are based on the fact that we can deduce something on unknown function values from known ones. Mutations lead in most cases to small changes of the considered x. It is believed that we may follow some path to better and better inputs. Optimal inputs are not expected to be surrounded by bad ones only. More involved arguments lead to the hypothesis of *gradient diffusion*. Also the *building block* hypothesis is based on the assumption that the fitness values of different x are correlated.

Here we have to argue why such correlations are possible in restricted and not in unrestricted black box optimization. We have already argued that a time restriction implies a size restriction for circuits. Since the evaluation of a circuit can be described by a program of finite length, a size restriction for circuits leads to quite small restrictions of the Kolmogoroff complexity. There are 2^{m2^n} functions $f : \{0,1\}^n \to \{0,1\}^m$. The theory of Kolmogoroff complexity implies that almost all of them have a Kolmogoroff complexity of almost $m2^n$ and only an insignificant (more exactly exponentially small) fraction of these functions has a Kolmogoroff complexity bounded above by $m2^{n/2}$. Let us consider a small numerical example where $n = 100$ and $m = 20$. Programs which generate the function table typically contain much less than $m2^{n/2} = 20 \cdot 2^{50}$ bits. The whole function table with $m2^n$ bits is described (implicitly) by perhaps a few thousand bits. On average, each bit in the program describes a huge number of bits in the function table. Such short descriptions cannot create independent function values.

It seems to be hard (or even impossible) to use these dependencies for a clever systematic and deterministic optimization technique which, moreover, leads to a fast algorithm. Randomness can use these dependencies in a much simpler way (see Motwani and Raghavan (1995)). So one may hope that the random modules of Evolutionary Algorithms implicitly benefit from the dependencies implied by the bounded complexity of the considered functions.

The claim that one optimization technique is in one of the restricted black box optimization scenarios superior to another can, at least in principle, be proved or disproved. Such results do not exist for sets X and Y of reasonable size, since it is difficult to prove precise results on classes of random functions with some complexity bound. This difficulty has been established for a lot of similar problems in complexity theory.

5 A Free Appetizer

Since we cannot analyze optimization techniques in some restricted black box optimization scenario for reasonable n and m, we restrict ourselves to the toy example $X = \{0,1\}^3$ and $Y = \{0,1\}^2$. The class of functions $F = \{f : X \to Y\}$ contains $4^8 = 65536$ functions which may be handled by case inspection. We interpret the binary strings as binary representation of numbers which leads to an ordering of Y. The aim of our investigations is to show how optimization techniques may take advantage from the structure given by the restriction that f does not belong to the most complex ones.

We consider 10 classes of functions. The circuit class C consists of all $f : \{0,1\}^3 \to \{0,1\}^2$ representable by circuits whose size is bounded by 3. To consider more classes it is more convenient to investigate representations where it is easier to compute the minimal size of some function f. Such representations are OBDDs (ordered binary decision diagrams) introduced by Bryant (1986). This representation is nowadays the state-of-the-art representation of Boolean functions with applications in CAD tools and, in particular, for verification purposes. In the recent years, OBDDs have turned out to be also a quite powerful representation in Genetic Programming. An OBDD (see Figure 1 for an example) on the variable set $X_n = \{x_1, \ldots, x_n\}$ is a directed acyclic graph. Sinks are labelled by Boolean constants from $\{0,1\}$ and inner nodes by Boolean variables from X_n. Each inner node has two outgoing edges one labelled by 0 and the other by 1. Furthermore, there is an ordering of the variables such that on each path the labelling of the inner nodes obeys this ordering. An OBDD represents $f : \{0,1\}^n \to \{0,1\}^m$ if each output bit f_j is represented at some OBDD node v_j. In order to evaluate the function f_v represented at v on input $a \in \{0,1\}^n$ we start at v. At a node with label x_i we follow the edge with label a_i, then $f_v(a)$ is the label of the sink finally reached. The size of an OBDD is the number of its inner nodes. The OBDD size of f is the size of the smallest OBDD (minimized over all $n!$ variable orderings) representing f. In our example we have only six variable orderings. For a fixed variable ordering it is easy to determine the minimal size of an OBDD representing f. Indeed we have used one of the available OBDD packages.

Let $F_i \subseteq F$ denote the class of all functions $f : \{0,1\}^3 \to \{0,1\}^2$ whose OBDD size is bounded by i. Then $F = F_8$. Is it possible to use the information that $f \in F_i$, $i < 8$, for an optimization algorithm?

We investigate three types of optimization algorithms. The first one consists of nonadaptive algorithms which sample the search space in a predefined order, i. e., a nonadaptive strategy does not react to the information already gathered. A nonadaptive strategy is a permutation of the search space. We consider all $8! = 40320$ nonadaptive algorithms. Evolutionary Algorithms are adaptive. We investigate the $(1+1)$ Evolutionary Algorithm $\text{EA}_{\geq}$ and a slight modification $\text{EA}_{>}$. These Evolutionary Algorithms follow general optimization techniques. The last two algorithms denoted by MaxOpt and MinMax are specialized algorithms using some knowledge about the class of considered functions.

We describe $\text{EA}_{\geq}$, $\text{EA}_{>}$, MaxOpt, and MinMax in more detail. According to our scenario, there is no stopping criterion. The algorithms stop as soon as an optimal x is sampled.

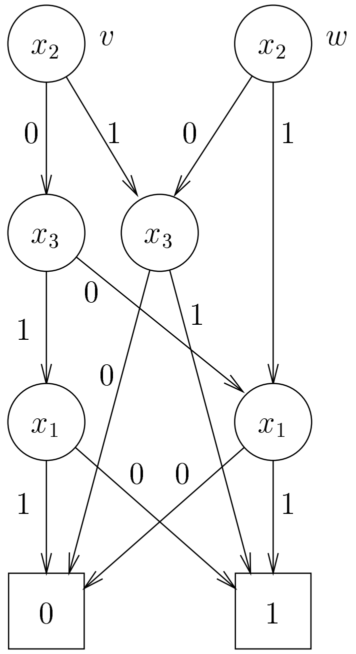

Figure 1: An OBDD for the variable ordering x_2, x_3, x_1 representing at (v, w) the function $f = (f_1, f_0)$ such that $f_1(x) = 1$ for $x = x_1x_2x_3 \in \{001, 011, 100, 111\}$ and $f_0(x) = 1$ for $x = x_1x_2x_3 \in \{001, 101, 110, 111\}$.
The OBDD contains six inner nodes, i. e. its size is 6.

The two $(1+1)$ EAs are possibly the most simple variants of an Evolutionary Algorithm and have been object of various studies (Rudolph (1997)).

Algorithm 1: Evolutionary Algorithm EA$_{\geq}$

1. Choose $x \in \{0,1\}^3$ uniformly at random.
2. For i := 1 To 3
   With probability 1/3 set $y_i := 1 - x_i$
   Else set $y_i := x_i$.
3. If $f(y) \geq f(x)$ then $x := y$.
4. Continue at line 2.

Algorithm 2: Evolutionary Algorithm EA$_{>}$
This algorithm is almost the same as EA$_{\geq}$, only line 3 reads

3. If $f(y) > f(x)$ then $x := y$.

MaxOpt and MinMax are simple greedy algorithms based on an extensive study of the situation. MaxOpt considers all possible functions and chooses that search point x which is optimal for the largest number of functions. Either x is optimal and we stop or we investigate one of the functions where x is not optimal. MinMax also considers the class of possible functions and looks for a search point x which leads to the largest decrease of the number of possible functions if x is not optimal. Both algorithms need a lot of time to compute the search points but our complexity measure is the number of different sampled points. If an algorithm considers some point more than once, this is not counted.

Algorithm 3: MaxOpt

```
1. Set G := F_i.
2. For each x ∈ {0,1}^3 not yet sampled
     Set m_x := number of g ∈ G such that x is optimal for g
3. Sample some x where m_x is maximal.
4. Remove all g ∈ G that are maximal at x or differ from f(x) at x.
5. Continue at line 2.
```

Algorithm 4: MinMax

```
1. Set G := F_i.
2. For each x ∈ {0,1}^3 not yet sampled
     For each y ∈ {0,1}^2
       m_{x,y} := size of G after removing all
       functions g where g(x) is maximal or g(x) ≠ y.
3. For each x ∈ {0,1}^3
     m_x := max{m_{x,y}}.
4. Sample some x where m_x is minimal.
5. Remove all g ∈ G that are maximal at x or differ from f(x) at x.
6. Continue at line 2.
```

In contrast to many other papers we have not performed experiments. In our toy example it is possible to compute exactly the behavior of each of the algorithms on each of the functions. For the Evolutionary Algorithms $\mathrm{EA}_{\geq}$ and $\mathrm{EA}_{>}$ we have investigated the Markov chain describing the search process and have computed the expected number of different sample points. All these computations have been done exactly without any rounding. Only at the very end the numbers are rounded and presented in Table 1. Therefore, different table entries imply that it is rigorously proven that the algorithms belonging to these entries have different behavior.

The first column contains the name of the considered class of functions. We remember that $F_8 = \{f : \{0,1\}^3 \to \{0,1\}^2\}$ describes the no free lunch scenario, the other sets describe size restricted black box optimization scenarios where for F_i the OBDD size is restricted and for C the circuit size. The next columns describe the size of the sets and the (expected) number of different sample points of the different algorithms. The notion n-a avg. stands for the average over all nonadaptive strategies, n-a best and n-a worst for the best resp. worst nonadaptive strategy.

First of all, for $F_8 = F$ all algorithms perform equal as the No Free Lunch Theorem states. The same holds for F_0, since this class contains only the four constant functions,

set	|set|	$\mathrm{EA}_{>}$	$\mathrm{EA}_{\geq}$	n-a avg.	n-a best	n-a worst	MaxOpt	MinMax
F_0	4	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
F_1	34	1.79288	1.79594	1.70588	1.44118	2.02941	1.44118	1.44118
F_2	280	2.29274	2.29443	2.24408	1.96071	2.52143	1.94643	2.00714
F_3	2166	2.50852	2.50887	2.48673	2.26362	2.69575	2.24331	2.23223
F_4	8908	2.87136	2.87085	2.86225	2.69185	3.00707	2.64111	2.64042
F_5	25694	2.98338	2.98287	2.97920	2.86974	3.05943	2.83518	2.84600
F_6	51520	3.02621	3.02609	3.02528	2.98453	3.04926	2.96555	2.97977
F_7	65144	3.06392	3.06387	3.06392	3.06126	3.06499	3.05928	3.06219
F_8	65536	3.06369	3.06369	3.06369	3.06369	3.06369	3.06369	3.06369
C	1290	2.76581	2.76497	2.74710	2.26744	3.55814	2.13953	2.15736

Table 1: Average number of distinct function evaluations.

and every algorithm finds an optimal point by sampling the first point. For the other classes there are differences.

We have proved that size restricted black box optimization allows different behavior of optimization techniques. The small size of the example allows only a small profit. Therefore, we do not speak of a free lunch but only of a free appetizer. We list some more results for our example drawn from Table 1.

- $\mathrm{EA}_{>}$ and $\mathrm{EA}_{\geq}$ perform very similar,
- $\mathrm{EA}_{>}$ and $\mathrm{EA}_{\geq}$ are worse than the best nonadaptive strategy and even worse (except for F_7) than the average nonadaptive strategy
- the greedy algorithms outperform the other algorithms, in most cases MaxOpt is the winner,
- it seems to be harder to optimize functions with large complexity than functions with small complexity.

The fact that we obtain similar results for restrictions on the OBDD and the circuit size leads to the conjecture that our results are not biased by the chosen representation.

6 Conclusions

The No Free Lunch Theorem is the correct answer to statements that an optimization technique is on the set of all functions $f : X \to Y$ superior to another one. To resolve the apparent contradiction between the No Free Lunch Theorem and the observed differences in the behavior of optimization techniques one has to describe clearly the scenario before one starts discussions on the behavior of optimization techniques. For several scenarios specific techniques are superior to general ones. One main statement is that nobody in applications is concerned with the unrestricted black box optimization scenario which is the base of the No Free Lunch Theorem. Taking randomly a function f of this class we have with large probability not enough time to evaluate f at only one sample point. We suggest restricted black box optimization scenarios to compare the general behavior of optimization techniques. For a small example we have proved that then a small free lunch or a free appetizer is possible.

References

Bryant, R. E. (1986): Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* C-35, 677-691.

Culberson, J. C. (1996): On the futility of blind search. Technical Report TR96-18, University of Alberta.

Fogel, D. B. (1995): *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press.

Garey, M. R. and Johnson, D. S. (1979): *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman Company.

Goldberg, D. E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Holland, J. H. (1975): *Adaption in Natural and Artificial Systems.* The University of Michigan Press.

Koza, J. R. (1992): *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Li, M. and Vitányi, P. (1993): *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag.

Michalewicz, Z. (1998): *An evolutionary algorithm for the ETSP.* PPSN'98, to appear.

Motwani, R. and Raghavan, P. (1995): *Randomized Algorithms.* Cambridge University Press.

Papadimitriou, C. H. (1994): *Computational Complexity.* Addison-Wesley.

Radcliffe, N. J. and Surry, P. D. (1995): Fundamental limitations on search algorithms: Evolutionary Computing in perspective. In: J. van Leeuvwen (Ed.): *Computer Science Today: Recent Trends and Developments.*, LNCS 1000, Springer-Verlag, 275–291.

Rechenberg, I. (1994): *Evolutionsstrategie '94.* Frommann-Holzboog.

Rudolph, G. (1997): *Convergence Properties of Evolutionary Algorithms.* Ph. D. Thesis, Verlag Dr. Kovač.

Schwefel, H.-P. (1995): *Evolution and Optimum Seeking.* Wiley.

Wegener, I. (1987): *The Complexity of Boolean Functions.* Wiley.

Wegener, I. (1994): Efficient data structures for Boolean functions. *Discrete Mathematics* 136, 347–372.

Wolpert, D. H. and Macready, W. G. (1995): No Free Lunch Theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

Wolpert, D. H. and Macready, W. G. (1997): No Free Lunch Theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1(1), 67–82.