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

A New Framework for the Valuation of Algorithms for Black-Box-Optimization

Stefan Droste, Thomas Jansen, and Ingo Wegener

No. CI-118/01

Technical Report ISSN 1433-3325 November 2001

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.

A New Framework for the Valuation of Algorithms for Black-Box Optimization

Stefan Droste[1,3] **Thomas Jansen**[2,3,4] **Karsten Tinnefeld**[1,3] **Ingo Wegener**[1,3]
[1]FB Informatik, LS2, Univ. Dortmund, 44221 Dortmund
droste|tinnefeld|wegener@ls2.cs.uni-dortmund.de
[2]Krasnow Institute, MSN 2A1, George Mason University,
Rock Fish Creek Lane, Fairfax, VA 22030
tjansen@gmu.edu

Abstract

Black-box optimization algorithms cannot use the specific parameters of the problem instance, i.e., of the fitness function f. Their run time is measured as the number of f-evaluations. This implies that the usual algorithmic complexity of a problem cannot be used in the black-box scenario. Therefore, a new framework for the valuation of algorithms for black-box optimization is presented allowing the notion of the black-box complexity of a problem. For several problems upper and lower bounds on their black-box complexity are presented. Moreover, it can can be concluded that randomized search heuristics whose (worst-case) expected optimization time for some problem is close to the black-box complexity of the problem are provably efficient (in the black-box scenario). The new approach is applied to several problems based on typical example functions and further interesting problems. Run times of general EAs for these problems are compared with the black-box complexity of the problem.

1 Introduction

The theory of efficient deterministic algorithms for optimization problems is well developed (see, e.g., Cormen, Leiserson, and Rivest (1990)). The same holds for the class of efficient randomized algorithms (see, e.g., Motwani and Raghavan (1995)). The valuation of an algorithm is based on the expected optimization time for a worst-case input. Although the algorithmic complexity of problems is well defined, we are not able to prove good lower bounds on the algorithmic complexity of important problems. There are only some $\Omega(n \log n)$ lower bounds based on (algebraic) decision trees (sorting and some problems from computational geometry) and there is a powerful complexity theory based on the NP$\neq$P-hypothesis or on the RP$\neq$NP-hypothesis if we are interested in randomized algorithms. All these investigations are concentrated on problem-specific algorithms.

[3]The author was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531).
[4]The author was supported by the German Academic Exchange Service (DAAD).

However, in many applications one is interested in robust optimization algorithms, i.e., algorithms applicable with good success for many different types of problems. Hence, randomized search heuristics like randomized local search, tabu search, simulated annealing, and all types of evolutionary algorithms have been designed. They are applied with good success. It is in principle possible to analyze these search heuristics like other randomized algorithms. However, these algorithms are not designed to support the analysis (like many problem-specific algorithms) and their analysis often is very difficult (see, e.g., Glover and Laguna (1993) for tabu search, Kirkpatrick, Gelatt, and Vecchi (1983) and Sasaki and Hajek (1988) for simulated annealing, and Rabani, Rabinovich, and Sinclair (1998) and Droste, Jansen, and Wegener (2002) for EAs). Moreover, we cannot expect that a general search heuristic beats the best problem-specific randomized algorithm with respect to the worst-case expected optimization time. Nevertheless, we are interested in a fair valuation of such heuristics. For this purpose, we need a scenario where only "instance-independent" algorithms are allowed. The idea is that the algorithm may "know" the problem but it does not "know" the instance which has to be solved. This so-called black-box scenario and the black-box complexity of problems are introduced in Section 2. It is discussed where this scenario is adequate and how the black-box complexity of a problem can differ from its algorithmic complexity. In Section 3, we present some well-known black-box optimization algorithms including one EA which serve as typical algorithms in the later sections. We are faced with another problem, since our focus is on EAs. There are many results on the behavior of certain EAs on specific functions. We show in Section 4 that we have to "embed" such functions f into classes of functions, also called problems, such that all instances of the problem have the "same flavor" as f and such that f is an instance of the problem. In Section 5, methods are presented how to prove lower bounds on the black-box complexity of problems. In order to prove that the new framework is meaningful we present several applications. Our results concern typical example functions, some function classes, as well as two "natural" problems (Section 6). Due to space limitations we give an overview of the results without proofs. A full version is available. In Section 7 we present a general lower bound for the class of unimodal functions.

2 The black-box scenario

In algorithmic complexity theory (see, e.g., Garey and Johnson (1979)) one distinguishes problems from instances of the problem. A problem like TSP (traveling salesman problem) describes the problem, the minimization of the length of a tour. It contains free parameters like the length of an edge. An instance of the problem is the description of concrete values for the free parameters. A problem can also consist of a class of functions, e.g., the pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$ which can be written as polynomials whose degree is bounded by d. Then an instance is a specific function from the given class. These two types of problems can be unified easily. An instance of TSP can be described as a function on the set of tours which assigns to each tour its cost. Therefore, also combinatorial optimization problems correspond to a class of functions. An optimization algorithm has to work for all instances of a problem and it can work with the specific parameters of the given instance. The black-box scenario differs essentially from this classical algorithmic scenario. The algorithm still has to work for all instances of a problem – but without knowing the specific parameters of the given instance. However, a black-box algorithm has the possibility to collect information about the unknown parameters. This can be done by sampling or querying, i.e., the algorithm can produce some search point x contained in the search space S and it will obtain the value $f(x)$. Hence, the general form of a black-box optimization algorithm is the following one.

Algorithm 1 (general black-box algorithm).

1.) Compute w.r.t. some probability distribution $x_1 \in S$. Then $f(x_1)$ is produced.

2.) In Step t compute w.r.t. some probability distribution depending on $(x_1, f(x_1), \ldots, x_{t-1}, f(x_{t-1}))$ some $x_t \in S$. Then $f(x_t)$ is produced.

3.) Stop if some stopping criterion is fulfilled and present a x_i with best f-value as result.

In this paper we consider only finite search spaces, most often $S = \{0,1\}^n$. We analyze black-box algorithms A without stopping criterion and we are interested in the expected optimization time $E(X_{A,f})$, where $X_{A,f}$ describes the first point of time t such that x_t is f-optimal. Also probability distribution, variance and other parameters of $X_{A,f}$ are of interest. However, the focus here is on $E(X_{A,f})$. For a problem described by the class F of functions, the worst-case expected optimization time of A equals $T_{A,F} := \max\{E(X_{A,f}) | f \in F\}$. The class of functions, the random variables, and $T_{A,F}$ implicitly depend on the input length n and we discuss the asymptotic optimization time. In the same way as the algorithmic complexity in the classical algorithmic complexity theory we obtain here the notion of the black-box complexity of a problem or a class of functions. Algorithmic complexity and black-box complexity are incomparable. In the first model more information is given, while in the second model we only count the number of queries, i.e., the number of search points x whose f-values are evaluated . The evaluation itself as well as the computation of the search point are free of cost. Later we present examples where the black-box complexity is exponential while the algorithmic complexity is polynomial. Moreover, NP-hard optimization problems may have a polynomial black-box complexity. The model is nevertheless in many situations fair, since black-box algorithms most often have simple routines to produce x_t and since most optimization problems have the property that the evaluation of the f-value of a search point is easy.

The black-box scenario is adequate in the following situations. Complex technical systems may allow the assignment of values to n free parameters. The problem is to choose parameter values optimizing the behavior of the system. However, the specific function f describing the quality of the machine depending on the parameter assignments may be unknown. Nevertheless, we may know some properties of f and, therefore, we know a class of functions containing f. Randomized search heuristics are also applied to well-defined optimization problems. These heuristics are popular, since they are "robust", i.e., they are easy to implement and often produce good results within a reasonable time. Some of the heuristics are problem-specific like the well-known k-opting algorithm for the TSP. These algorithms fit into our scenario. Other heuristics work on all functions on some search space, e.g., on all functions $f\colon \{0,1\}^n \to \mathbb{R}$. Also these heuristics fit into our scenario, although we cannot expect that they beat the more problem-specific ones. Nevertheless, a comparison between the black-box complexity of a problem and the asymptotic optimization time of a general randomized search strategy leads to a valuation which is more fair and meaningful than the comparison between the algorithmic complexity and the asymptotic run time.

3 Some randomized search heuristics

Here we shortly describe some simple randomized search heuristics which work with one actual search point, i.e., the available knowledge of $x_1, f(x_1), \ldots, x_{t-1}, f(x_{t-1})$ is reduced to some $x \in \{x_1, \ldots, x_{t-1}\}$ and $f(x)$. The additional knowledge is eliminated. All search heuristics choose x_1 according to the uniform distribution and x is set to x_1. In the following we describe the probability distribution how to choose the next search point x'. The new search point x' replaces x iff $f(x') \geq f(x)$. If nothing else is mentioned, the problem is to maximize f.

Algorithm 2 (randomized local search (RLS)). *Choose $i \in \{1, \ldots, n\}$ according to the uniform distribution. Set $x'_i := 1 - x_i$ and $x'_j := x_j$, if $j \neq i$.*

Algorithm 3 (randomized global search for the mutation probability p). *Produce all x'_i independently. Set $x'_i := 1 - x_i$ with probability p and $x'_i := x_i$ otherwise.*

The randomized global search with $p = 1/n$ is the standard $(1+1)$ EA. For $p = 1/2$, x' is chosen independently from x according to the uniform distribution. Then Algorithm 3 is called random search (RS). All algorithms RLS, $(1+1)$ EA, and RS may evaluate x more than once. We may use a dictionary storing the search points x already evaluated together with their f-values. Then we can avoid to evaluate the same search point twice. This leads to the algorithms RLS*, $(1+1)$ EA* and RS*, respectively.

4 Problems in the black-box scenario

Typical combinatorial optimization problems are defined in such a way that they fit directly into the scenario of black-box optimization. Another type of problems can be obtained by considering subclasses of the class of functions $f\colon \{0,1\}^n \to \mathbb{R}$. Such functions can be uniquely represented as a polynomial $f(x) = \sum_{B \in A} w_B \cdot \prod_{i \in B} x_i$ with $A \subseteq \mathcal{P}(\{1, \ldots, n\})$ and $w_B \neq 0$ for all B. We call $N = |A|$ the number of terms and $d = \max\{|B|, B \in A\}$ the degree of the polynomial. So, the following classes of polynomials are of obvious interest.

Definition 1. *Let $P(n, d, N)$ denote the set of all polynomials on n variables where the degree is bounded by d and the number of terms is bounded by N. For $d = 1$ the functions are called linear, and let LIN denote the set of all such functions. For $N = 1$ the functions are called monomials. Let $MP(n, d, N)$ be the subclass of $P(n, d, N)$ consisting of monotone increasing polynomials, i.e., polynomials where $w_i > 0$ for all i. Let $MP^*(n, d, N)$ be the class of monotone polynomials, i.e., polynomials obtained from polynomials $f \in MP(n, d, N)$ by replacing some x_i with $1 - x_i$.*

Since randomized search heuristics often have difficulties to avoid or to leave locally but not globally optimal search points, the class of unimodal functions is of particular interest.

Definition 2. *A pseudo-Boolean function is called unimodal if only globally optimal search points have no Hamming neighbor with a better f-value (often it is also required that f has only one global optimum, but this is of no importance here).*

Some algorithmic problems are not defined as optimization problems, e.g., the sorting problem. Hence, there are several ways to turn these problems into an optimization problem. We consider the fundamental sorting problem. Each measure of presortedness as discussed in the literature on adaptive sorting algorithms (see, e.g., Petersson and Moffat (1995)) leads to a function on the set of permutations such that the sorted sequence is the unique optimum. We will discuss sorting as an optimization problem using the most prominent measure of presortedness, namely the number of inversions.

Definition 3. *Let $x = (x_1, \ldots, x_n)$ be a vector of n different numbers and let π be a permutation on $\{1, \ldots, n\}$. Then $inv(\pi) := \#\{(i,j) \mid 1 \leq i < j \leq n, x_{\pi(i)} > x_{\pi(j)}\}$. Sorting is the problem of minimizing $inv(\pi)$.*

The literature on EAs contains many results on example functions which are meant to be typical for a class of functions. In almost all cases, these classes are not defined. The black-box complexity of a class containing a single function is always 1, since there exists an algorithm which chooses an f-optimal point as x_1. Therefore, we have to define adequate classes of functions containing f. Moreover, our aim is to present general ideas how to embed functions into an appropriate class of functions.

Definition 4. *1. ONEMAX$(x) := |x| := x_1 + \cdots + x_n$ (number of ones or Hamming distance to 0^n).*

2. BV$(x) := \sum_{1 \le i \le n} x_i 2^{n-i}$ (binary value).

3. LO$(x) := \max\{i | x_1 = \cdots = x_i = 1\}$ (number of leading ones).

4. $N(x) := 1$, if $x = 1^n$, and $N(x) := 0$ otherwise (needle in the haystack).

5. $T(x) :=$ ONEMAX(x), if $x \ne 0^n$, and $T(0^n) := 2n$ (trap).

ONEMAX describes the Hamming distance to 0^n which should be maximized. Obviously, 0^n is chosen only because of its simple description. The function $\text{ONEMAX}_a(x) := (x_1 \oplus a_1) + \cdots + (x_n \oplus a_n)$, where $\oplus$ denotes EXOR, describes the function where the distance to $a \in \{0,1\}^n$ has to be maximized. The class of functions ONEMAX^* contains all ONEMAX_a, $a \in \{0,1\}^n$, and describes a black-box optimization problem which can be considered as "the adequate generalization" of ONEMAX. The idea behind this generalization is that in black-box optimization the roles of 0 and 1 should be interchangeable. In general, in black-box optimization a class of functions should contain with f also all $f_a, a \in \{0,1\}^n$, defined by $f_a(x) := f(x_1 \oplus a_1, \dots, x_n \oplus a_n)$. The class $\text{MP}(n, d, N)$ of monotone increasing polynomials is a simple class for black-box optimization, since the algorithm starting with $x_1 = 1^n$ has a complexity of 1. The class $\text{MP}^*(n, d, N)$ of monotone polynomials is an adequate generalization obtained in the same way as ONEMAX^* from ONEMAX. However, this type of generalization is not always sufficient as the function BV shows. If we consider the class of all BV_a the following black-box algorithm has a complexity of $2 - 2^{-n}$. It starts with $x_1 = 0^n$ and obtains the information that $\text{BV}_a(0^n) = \sum_{i|a_i=1} 2^i = \sum_{1 \le i \le n} a_i 2^i$. Hence, a can be computed easily from $\text{BV}_a(0^n)$ and then we can present the optimal value $x_2 = \overline{a}$ where $\overline{a} = (\overline{a}_1, \dots, \overline{a}_n)$ and $\overline{a}_i = 1 - a_i$. One idea behind BV is that there is a known ordering of the bit positions describing the importance of the position. Let $b, c \in \{0,1\}^n$ and let i be the most important position where $b_i \ne c_i$. The function takes the larger value for b iff b_i is the "right value" for this position. The particular function values are of no importance. Hence, if we consider $h \circ f$ for a strictly increasing function $h \colon \mathbb{R} \to \mathbb{R}$ instead of f, we consider a function with the same ordering of the function values. This preserves many properties of the function. Let BV^{**} be the class of all $h \circ \text{BV}_a, a \in \{0,1\}^n$ and $h \colon \mathbb{R} \to \mathbb{R}$ strictly increasing. (The notation with the superscript ** describes that we have applied both considered generalization concepts. Obviously, we do not obtain more functions by considering the functions $(h \circ \text{BV})_a$.) The behavior of general randomized search heuristics on f and f_a is typically the same. This is not always true for f and $h \circ f$. E.g., fitness-proportional selection works differently on f and $h \circ f$. But, fitness-proportional selection is adequate only if we know something about the relative size of function values.

Our discussion on "the idea behind BV" was based on the assumption that the ordering of the bit positions by their importance was known. One also may assume that this knowledge is not given. In general, let S_n be the set of all permutations on $\{1, \dots, n\}$

and let $f_\pi(x_1,\dots,x_n) := f(x_{\pi(1)},\dots,x_{\pi(n)})$. Let BV^{***} be the class of all $h \circ (\mathrm{BV}_\pi)_a, a \in \{0,1\}^n, \pi \in S_n, h\colon \mathbb{R} \to \mathbb{R}$ strictly increasing. This third generalization concept does not change the behavior of most general randomized search heuristics. One exception is one-point crossover, it assumes a larger influence of neighbored variables on each other.

Remember that LIN is the class of linear functions. If f is linear, the same holds for $f_{a,\pi}$. Hence, LIN is closed under our first and third generalization concept. However, $h \circ f$ for f linear and $h\colon \mathbb{R} \to \mathbb{R}$ strictly increasing is not necessarily linear and LIN^{**} is a proper superclass of LIN. Note that ONEMAX and BV are merely two special linear functions.

For LO we can consider the classes LO^*, LO^{**}, and LO^{***}. For LO^* the f-value equals the length of the correct prefix, for LO^{**} the f-value is strictly increasing with this length, and for LO^{***} the positions are ordered in some way with respect to their importance and the f-value is strictly increasing with the number of correct "most important" bits.

For the needle-in-the-haystack-function N it is sufficient to apply the first generalization concept and to investigate the class N^* of all N_a such that $N_a(\overline{a}) = 1$ and $N_a(b) = 0$ for $b \neq \overline{a}$. The other generalization concepts do not generalize N^* essentially.

Considering ONEMAX^{**} instead of ONEMAX^* makes only a small difference. The f_a-value for ONEMAX^* describes the Hamming distance to a, while, for ONEMAX^{**}, we only can decide whether a search point x is closer to a than y. Since ONEMAX is a symmetric function, ONEMAX^{***} is equal to ONEMAX^{**}.

Finally, we consider the trap function T. The idea behind this function is that the f-values give wrong hints and that this function is difficult for randomized search heuristics. However, the function is very similar to ONEMAX. Indeed, Whitley (1997) has considered the following variant H' of an arbitrary search heuristic H. If H chooses the search point b, H' chooses b and $\overline{b}$ and uses for the further decisions only b and $f(b)$. Each heuristic which is efficient on one of the ONEMAX-classes has then an efficient variant for the corresponding T-class. Indeed, the expected optimization time for the T-class is at most twice the expected optimization time for the corresponding ONEMAX-class. Here our purely syntactical generalization concepts are not sufficient and we have to consider a semantical generalization concept. The idea behind the trap function is that the T-values give wrong hints. However, they give hints. Having found the local but not global optimum (which is easy for search heuristics) one can easily compute the global optimum. The "real idea" behind the trap function is that the global optimum is somewhere while all hints lead to the local optimum. Therefore, T_{sem}, the semantical generalization of the trap function, contains all functions T^a such that $T^a(x) := \mathrm{ONEMAX}(x)$, if $x \neq a$ and $T^a(a) = 2n$. It makes no big difference to consider also T^*_{sem}, T^{**}_{sem}, and T^{***}_{sem}.

5 Lower bound methods for the black-box complexity of problems

Black-box complexity is defined as the expected optimization time for a worst-case instance. Hence, the expectation is on the algorithm's random bits. In this situation we can apply Yao's minimax principle (Yao (1977), Motwani and Raghavan (1995)). It makes it possible to investigate deterministic algorithms in order to obtain lower bounds for randomized algorithms. Moreover, we are free to choose a probability distribution on the instances in order to make the problem difficult.

Proposition 1. *(Yao's Minimax Principle) If a problem has a finite set of instances of a fixed size and allows a finite set of deterministic algorithms, the minimal worst-case* *instance expected optimization time is lower bounded for each probability distribution on the inputs by the expected optimization time of an optimal deterministic algorithm.*

As mentioned above, the black-box scenario has the drawback that, for certain problems, it is too powerful, since the resources for the computation of the next query are not counted. Thus, NP-hard optimization problems can have a polynomial black-box complexity.

Theorem 1. *The black-box complexity of $P(n, 2, \infty)$ is bounded above by $\binom{n}{2} + n + 2$.*

For NP-hard problems we cannot expect any randomized algorithm to guarantee a polynomial expected optimization time. The conclusion is to bound the time to compute a query by a polynomial. Then hard problems for randomized algorithms are hard in the black-box scenario, too. We have no other results which rely on such time bounds. All well-known randomized search heuristics work with a space-bounded approach. In many cases, only one search point and its f-value (and sometimes the number of f-evaluations) are stored. EAs work with populations and the population size is usually not large. This leads to a space-bounded black-box scenario where at most $s(n)$ search points and their f-values can be stored. The $s(n)$-space-bounded black-box complexity of a problem can be much larger than its black-box complexity. In the case of $P(n, 2, \infty)$ we have no polynomial upper bound on, e.g., its n-space-bounded black-box complexity. Since many randomized search heuristics work with population size 1, the case $s(n) = 1$ is of particular interest. The $s(n)$-space-bounded black-box complexity of a problem can be compared with the unrestricted black-box complexity of the same problem in order to measure the value of additional space. This can be done by competitive analysis as introduced in the theory on online algorithms (see, e.g., Borodin and El-Yaniv (1998)). However, in this paper we investigate only the unrestricted black-box complexity of problems.

6 Black-box complexity of different problems

In this section, we present a variety of different results within our framework. We start with classes that are based on single example functions.

Theorem 2.

1. *The black-box complexity of each class of functions $F \subseteq \{f\colon \{0,1\}^n \to \mathbb{R}\}$ is bounded above by $2^{n-1} + 1/2$.*
2. *The black-box complexity of N^* equals $2^{n-1} + 1/2$.*
3. *Random search avoiding duplicates and the $(1+1)$ EA avoiding duplicates are optimal black-box algorithms in the needle-in-the-haystack scenario N^*.*
4. *The black-box complexity of T_{sem} equals $2^{n-1} + 1/2$.*
5. *The black-box complexity of LO* is bounded above by $n/2 + o(n)$ and below by $n/2 - o(n)$.*
6. *The black-box complexity of LO** is bounded above by $n+1$ and below by $n/2 - o(n)$.*
7. *The black-box complexity of LO*** is bounded above by $n \log n + O(n)$ and below by $n/2 - o(n)$.*
8. *The black-box complexity of ONEMAX* is bounded below by $\Omega(n/\log n)$.*
9. *The black-box complexity of BV* equals $2 - 2^{-n}$.*
10. *The black-box complexity of BV** is bounded below by $\Omega(n/\log n)$.*

It is not surprising that RS* is an optimal black-box algorithm for N^*. If we cannot learn anything, it is good to search randomly while avoiding duplicates. The $(1+1)$ EA* also has optimal expected optimization time. Sometimes it is claimed that EAs are bad on needle-in-the-haystack like functions. Theorem 2 shows that an EA is even optimal although slow. This is due to the inherent black-box complexity of N^*. Algorithms avoiding duplicates need a large storage, for N^* on average a storage of $2^{n-1} - 1/2$. The algorithms RLS (randomized local search), RS (randomized search), and the $(1+1)$ EA do not avoid duplicates and store a single search point. Results due to Garnier, Kallel, and Schoenauer (1999) imply that RLS and RS have an expected optimization time of $2^n(1 \pm o(1))$ and pay with a factor of approximately 2 for producing duplicates. A new search point x' that is equal to the current point x is called replica. RLS never produces a replica and RS does this with the tiny probability of 2^{-n}. The (1+1) EA produces a replica with probability $(1 - 1/n)^n \approx e^{-1}$ and thus has expected optimization time $2^n(1 \pm o(1))/(1 - e^{-1}) \approx 3.16 \cdot 2^n(1 \pm o(1))$. Altogether, randomized search heuristics are very efficient in the black-box scenario N^*.

All search strategies avoiding duplicates cannot use more than 2^n queries. Search strategies used in applications do not avoid duplicates because of storage restriction. Droste, Jansen, and Wegener (2002) have shown that the expected optimization time of the $(1+1)$ EA on the classical trap function T is at least $n^n(1 - o(1))$ and it is at most n^n for each function $f : \{0,1\}^n \to \mathbb{R}$. Typical search heuristics need much more time than 2^n in the black-box scenario T_{sem}. Hence, they all are bad in this scenario while random search is optimal. Search heuristics prefer to search close to search points with good f-values. Therefore, it is not surprising that they fail in scenarios where this behavior is wrong.

Droste, Jansen, and Wegener (2002) have proved that the $(1 + 1)$ EA has an expected optimization time of $\Theta(n^2)$ on LO and this result also holds for the classes LO^*, LO^{**}, and LO^{***}. It is even easier to prove the same result for RLS. Both algorithms do not direct the search by the information collected so far. This is possible for black-box search heuristics knowing the considered class of functions. It is an open problem to decrease the difference between the upper and the lower bound on the black-box complexity of LO^{***}. We conjecture that the upper bound is asymptotically tight. However, the difficulty of an improved lower bound is that it is not necessary to learn π for optimizing $\text{LO}_{a,\pi}$.

We believe that the black-box complexity of ONEMAX^* is $\Theta(n)$. Our upper and lower bounds differ by a factor of $\Theta(\log n)$. The reason is that in the lower bound we consider decision trees of degree $n + 1$ while the upper bound uses with the exception of the first query only queries allowing only two answers. We have no idea how to find queries which separate the set of possible a into many large sets. However, the determination of the exact asymptotic size of the black-box complexity of ONEMAX^* remains an open problem.

Now, we consider some classes of functions. Since the optimization of polynomials of degree 2 is NP-hard, only other subclasses of the class of all polynomials are of interest for our analysis. The class of monotone polynomials $\text{MP}^*(n, d, N)$ (see Section 4 for the definition) has several interesting subclasses. We consider the class $M^*(n, d, 1)$ of so-called prefix-monomials $(x_1 \oplus a_1) \cdots (x_k \oplus a_k), k \le d$. Each function takes only two values. Therefore, $M^{**}(n, d, 1)$ is no real generalization and we define $M^{***}(n, d, 1)$ as the set of all monomials $(x_{i(1)} \oplus a_1) \cdots (x_{i(k)} \oplus a_k), 1 \le i(1) < \cdots < i(k) \le n$.

Theorem 3. *1. The black-box complexity of LIN^{**} is bounded above by $n + 1$.*

2. The black-box complexity of $M^*(n, d, 1)$ and of $M^{***}(n, d, 1)$ is of size $\Theta(2^d)$.

3. The black-box complexity of $MP^*(n, d, N)$ is bounded above by $O(2^d \log n + n^2)$.

Droste, Jansen, and Wegener (2002) have proved that the expected optimization time of the $(1+1)$ EA for all linear functions is $O(n \log n)$ and even $\Theta(n \log n)$, if all $w_i \neq 0$. Their proof also works for the functions in LIN^{**}. The same bounds follow much more easily for RLS. Our results show that the usual randomized search heuristics are at least close to optimal on linear functions and generalizations. The $(1+1)$ EA and RLS have an expected optimization time of $\Theta(2^d n/d)$ on $M^*(n, d, 1)$ (Wegener (2001) based on the results of Garnier, Kallel, and Schoenauer (1999)). These algorithms waste the factor $\Theta(n/d)$, since the expected time before a step changes one of the essential bits equals $\Theta(n/d)$. It is easy to show that the expected optimization time of the $(1+1)$ EA with a mutation probability of $1/d$ also equals $\Theta(2^d)$ on $M^{***}(n, d, 1)$. Hence, simple randomized search heuristics are asymptotically optimal on arbitrary monomials. Wegener (2001) has proved that the expected optimization time of the (1+1) EA on $\mathrm{MP}^*(n, d, N)$ is bounded above by $O(2^d \cdot \frac{n}{d} \cdot N)$. It is conjectured that this bound can be improved to $O(2^d \cdot \frac{n}{d} \cdot \log \frac{n}{d})$, since overlapping monomials should support the optimization. In particular, the number of terms N should not have a large influence. This can be proved for the black-box complexity. The upper bound of $O(2^d \log n + n^2)$ for $\mathrm{MP}^*(n, d, N)$ can perhaps be improved by considering overlapping monomials more carefully.

Sorting is the best-investigated algorithmic problem. Its algorithmic complexity equals $\Theta(n \log n)$. Here we consider sorting as the problem of minimizing $\mathrm{inv}(\pi)$. Our main motivation is our interest in the black-box complexity of the fundamental sorting problem. Scharnow (2001) has described a variant of the $(1+1)$ EA working on permutations. This EA minimizes the number of inversions in an expected number of $O(n^2 \log n)$ steps. Finally, we mention the well-known shortest-s-t-path problem in order to show that a fundamental problem with a small algorithmic complexity, namely $O(n^2)$, can lead to a difficult problem in the black-box scenario. Each instance is described by a distance matrix $d(i, j), 1 \leq i < j \leq n$, where $d(i, j) \geq 0$ describes the length of the edge (i, j). The value $d(i, j) = \infty$ indicates that this edge does not exist. The search space consists of all paths from s to t. The cost of such a p is the sum of all $d(i, j)$ where the edge (i, j) lies on p.

Theorem 4. 1. *The black-box complexity of minimizing the number of inversions is bounded above by $n+1$ and bounded below by $n/2 - o(n)$.*

2. *The black-box complexity of the shortest-s-t-path problem is bounded below by $\Omega((n-2)!) = 2^{\Omega(n \log n)}$.*

7 Unimodal functions

Unimodal functions (see Definition 2) are those functions which can be optimized by local search algorithms with probability 1. In our case, the local neighborhood of $a \in \{0,1\}^n$ consists of its n Hamming neighbors. It is not difficult to describe unimodal functions where all (randomized) local search algorithms need an exponential expected optimization time (Horn, Goldberg, and Deb (1994)). Some of these functions are easy for EAs (Rudolph (1997)). Droste, Jansen, and Wegener (1998) have proved for a unimodal function defined by Rudolph (1997) that the expected optimization time of mutation-based EAs grows exponentially. Here we investigate the black-box complexity of the class of unimodal functions. Llewellyn, Tovey, and Trick (1989) have proved that this class has an exponential complexity for deterministic black-box algorithms. We apply Yao's minimax principle to generalize this result to randomized black-box algorithms.

First, we choose a probability distribution on the set of unimodal functions $f \colon \{0,1\}^n \to \mathbb{R}$. A random path R starts at 1^n and has a length of $l(n) := 2^{n^{1-\varepsilon}}, \varepsilon > 0$ a constant. The successor of a point on the path is chosen randomly among all its Hamming neighbors. We observe that $R = (p_0, \ldots, p_l)$ is typically not a simple path. The corresponding random path function RP is defined as follows:

$$\mathrm{RP}(x) := \begin{cases} \mathrm{ONEMAX}(x) - n & \text{if } x \neq p_i \text{ for all } i \\ i & \text{if } x = p_i \text{ and } x \neq p_j \text{ for all } j > i. \end{cases}$$

For x on R let i be the largest index such that $x = p_i$. This implies $\mathrm{RP}(x) = \mathrm{RP}(p_i) < RP(p_{i+1})$. Each point x outside R has a Hamming neighbor y with more ones and in any case $\mathrm{RP}(x) < \mathrm{RP}(y)$. Hence, RP is unimodal with a unique global maximum at p_l. In the following, we investigate deterministic black-box algorithms against RP which describes a probability distribution on the set of all unimodal pseudo-Boolean functions.

The random path R has with high probability some cycles. However, we believe that there are no long cycles. Moreover, we even believe that $H(p_i, p_j)$ is large if $j - i$ is not small. After having left p^* and having walked for a while the path is almost always far from p^*.

Lemma 1. *For each $\beta > 0$ there exists some $\alpha = \alpha(\beta) > 0$ such that the probability that $H(p_i, p_{i+j}) \leq \alpha n$ for some i and some $j \geq \beta n$ is bounded above by $2^{-\Omega(n)}$.*

Proof. Since $0 \leq i \leq i + j \leq 2^{n^{1-\varepsilon}}$ and because of symmetry, it is sufficient to prove the lemma for $i = 0$ and some fixed $j \geq \beta n$. Let H_t denote the Hamming distance between $p_0 = 1^n$ and p_t. We prove $\mathrm{Prob}(H_j \leq \alpha \cdot n) = 2^{-\Omega(n)}$ for some constant α depending on β, only. Obviously, $\mathrm{Prob}(H_{t+1} = H_t + 1) = 1 - H_t/n$ holds for each t, since the Hamming distance is increased by 1, iff one of the $n - H_t$ bits that do not differ from p_0 is flipped.

We set $\gamma := \min\{j/n, 1/10\}$ and remark that γ is bounded below by a positive constant. We divide the j first steps of the path construction into phases of length $\gamma \cdot n$. If $j/(\gamma n) \notin \mathbb{N}$, the first phase is shorter. Note, that the final phase has the length $\gamma \cdot n$ due to the choice of γ. We prove that at the end of this final phase H_t is bounded below by $(\gamma/5) \cdot n$ with probability $1 - 2^{-\Omega(n)}$. Then, setting $\alpha := \min\{\beta/5, 1/50\}$ completes the proof.

First, assume that at the beginning of the final phase we have $H_t \geq 2\gamma \cdot n$. Then, at the end of the phase the Hamming distance is bounded below by $\gamma \cdot n$ with probability 1.

Now, assume that at the beginning of the final phase we have $H_t < 2\gamma \cdot n$. Obviously, we have $H_t < 3\gamma \cdot n$ during the whole phase. Thus, the probability to increase the Hamming distance in one step is bounded below by $1 - (3\gamma \cdot n)/n \geq 7/10$. By Chernoff bounds the probability to have less than $(3/5)\gamma \cdot n$ such increasements in the $\gamma \cdot n$ steps of this phase is bounded above by $e^{-\Omega(n)}$. Thereby, at the end of the final phase we have Hamming distance at least $(3/5)\gamma \cdot n - (2/5)\gamma \cdot n = (\gamma/5) \cdot n$ with probability $1 - 2^{-\Omega(n)}$. $\square$

Until now we have investigated the random path R of length $l(n) = 2^{n^{1-\varepsilon}}$. However, an adversary may save the loops by always looking for the Hamming neighbor with the largest RP-value. Hence, we investigate the so-called true path T which chooses always the Hamming neighbor with the largest RP-value. The following is a direct consequence.

Corollary 1. *The probability that the length of T is smaller than $l(n)/n$ or that some point x has more than $(1 - \alpha(1))n$ Hamming neighbors on R is bounded above by $2^{-\Omega(n)}$.*

In order to investigate black-box algorithms which try to optimize RP, we make the situation for the algorithm easier. First, $\mathrm{RP}(1^n)$ and all points x on R with $\mathrm{RP}(x) < \mathrm{RP}(1^n)$ and their RP-values are given to the algorithm. Later, the knowledge of the algorithm is described by N, the set of points that do not lie not on R, and j, implying that the algorithm knows $p_0, \ldots, p_j$ and $\mathrm{RP}(p_i) \leq \mathrm{RP}(p_j) = j$ for all $i \leq j$. Based on this information the algorithm chooses a query point x^*. Let y be the nth successor of p_j on T. The algorithm may stop its search if x^* is optimal or $\mathrm{RP}(x^*) > RP(y)$. Otherwise, the knowledge of the algorithm is updated as follows. In any case, j is replaced by $\mathrm{RP}(y)$. If x^* does not lie on R, x^* is added to N. Lower bounds on the expected stopping time of black-box algorithms are obviously also lower bounds on their expected optimization time on RP. The following theorem implies that all (typically informal) statements in the literature that unimodal functions are easy for randomized search heuristics are not correct in a strong sense.

Theorem 5. *Each randomized black-box search heuristic has on the class of unimodal pseudo-Boolean functions a worst-case expected optimization time of at least* $2^{n^{1-\delta}}, \delta > 0$. *Moreover, the worst-case success probability after* $2^{n^{1-\delta}}$ *steps is bounded above by* $2^{-\Omega(n)}$.

Proof. It is sufficient to prove that in the scenario discussed above the black-box algorithm wins only with a probability of $2^{-\Omega(n)}$ before j is so large that p_j belongs to the n last points of T. This gives a lower bound of $(2^{n^{1-\delta}} - n)/n^2$ steps. However, since $\delta > 0$ can be chosen arbitrarily, this proves the theorem. Altogether, it is sufficient to prove that the success probability of the black-box algorithm within each step is bounded by $2^{-\Omega(n)}$.

We start with the simplest situation where $N = \emptyset$ and $j = 0$. Lemma 1 implies that a query x^* where $H(1^n, x^*) \le \alpha(1)n$ has a success probability of $2^{-\Omega(n)}$. We still have to consider a query x^* where $H(1^n, x^*) > \alpha(1)n$. In order to reach x^* on R after $t \ge n$ steps we have $H(1^n, x^*)$ specified positions which have flipped for an odd number of times while the other $n - H(1^n, x^*)$ specified positions have flipped for an even number of times. This has a probability of $2^{-\Omega(n)}$. In the same way as Lemma 1 it can be proved that even the probability of reaching after $t \ge n$ steps a point with a Hamming distance of at most $\alpha(1)n$ to x^* is bounded above by $2^{-\Omega(n)}$. This will be important when we know more. Then, all the points in N and all the points x on R such that $\mathrm{RP}(x) < j$ will not lie on the part of R starting at p_j. Let M_{far} be the subset of these points x with $H(y, x) > \alpha(1)n$ and M_{close} the set of the remaining points. Lemma 1 and the discussion above prove that $|M_{\text{close}}| \le n$ with a probability of $1 - 2^{-\Omega(n)}$. First, we assume that we only know that the points in M_{far} are forbidden. We can investigate a random path starting at x_j. Let $p(x^*)$ be the probability that this path reaches x^* after t steps where $t \in \{n, \dots, l(n)\}$ and let E be the event that this path does not reach any point in M_{far}. Then $\mathrm{Prob}(x^*|E) = \mathrm{Prob}(x^* \wedge E)/\mathrm{Prob}(E) \le \mathrm{Prob}(x^*)/\mathrm{Prob}(E) \le \mathrm{Prob}(x^*)/(1 - 2^{-\Omega(n)}) = \mathrm{Prob}(x^*)(1 + 2^{-\Omega(n)})$. Hence, the success probability of any query has been increased only by a factor of $1 + 2^{-\Omega(n)}$ by our knowledge that the points in M_{far} are forbidden. Hence, the probability of each query is still $2^{-\Omega(n)}$. In a last step we have to take into account the at most n forbidden points in M_{close}. We investigate the first $n/2$ steps on R starting from p_j. Applying Lemma 1 for $\beta = 1/2$ we are after these $n/2$ steps with an overwhelming probability at a distance of at least $\alpha(1/2)n$ from each of the forbidden points. Hence, we can afterwards repeat our above argument, since now all forbidden points are far, if far means now a Hamming distance of at least $\alpha(1/2)n$. This still implies a success probability of each query of $2^{-\Omega(n)}$. □

8 Conclusion

The classical complexity theory contains (relative) results on the hardness of problems with respect to deterministic and randomized problem-specific algorithms. General randomized search heuristics not working with the specific parameters of a problem instance have to solve a somewhat more difficult problem. Black-box complexity is more adequate in this scenario. This concept and adequate problem classes have been introduced and many upper and lower bounds on their black-box complexity presented. Some simple results show that search heuristics like the $(1+1)$ EA are almost optimal on needle-in the-haystack functions (although slow), but they are inefficient on trap functions. For this class of functions one has to reveal the real idea behind the classical trap function in order to find the right class of functions. Often, Yao's minimax principle leads to good lower bounds. However, in scenarios where queries may have infinitely many answers, one has to show that the information "really" contained in the answer is finite, before Yao's minimax principle can be applied. Perhaps the most interesting bounds concern the class of monotone polynomials of bounded degree, where the upper and lower bounds almost match, and the class of unimodal functions, where it is proved that no randomized search heuristic can optimize these functions in subexponential expected time.

References

A. Borodin and R. El-Yaniv (1998). *Online Computation and Competitive Analysis.* Cambridge University Press.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest (1990). *Introduction to Algorithms.* MIT Press.

S. Droste, T. Jansen, and I. Wegener (1998). On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In *Parallel Problem Solving from Nature* (*PPSN V*), LNCS 1498, 47–56.

S. Droste, T. Jansen, and I. Wegener (2002). On the analysis of the $(1+1)$ evolutionary algorithm. *To appear in: Theoretical Computer Science*.

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

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

F. Glover and M. Laguna (1993). Tabu search. In C. R. Reeves (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, 75–150. Blackwell Oxford.

J. Horn, D. E. Goldberg, and K. Deb (1994). Long path problems. In Y. Davidor, H.-P. Schwefel, and R. Männer (Eds.), *Parallel Problem Solving From Nature* (*PPSN III*), LNCS 866, Berin, Germany, 149–158. Springer.

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

D. C. Llewellyn, C. Tovey, and M. Trick (1989). Local optimization on graphs. *Discrete Applied Mathematics 23*, 157–178.

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

O. Petersson and A. Moffat (1995). A framework for adaptive sorting. *Discrete Applied Mathematics 59*, 153–179.

Y. Rabani, Y. Rabinovich, and A. Sinclair (1998). A computational view of population genetics. *Random Structures and Algorithms 12*, 314–330.

G. Rudolph (1997). How mutation and selection solve long-path problems in polynomial expected time. *Evolutionary Computation 4*, 195–205.

G. Sasaki and B. Hajek (1988). The time complexity of maximum matching by simulated annealing. *Journal of the ACM 35*, 387–403.

J. Scharnow (2001). Evolutionäre Algorithmen für Sortierprobleme. Master's thesis, Univ. Dortmund, Dortmund, Germany. In preparation.

I. Wegener (2001). Theoretical aspects of evolutionary algorithms. In *28th International Colloquium of Automata, Languages and Programming* (*ICALP 2001*), LNCS 2076, 64–78.

D. L. Whitley (1997). Permutations. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, C1.4:1–8. Institute of Physics Publishing.

A. C. Yao (1977). Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 17. Annual IEEE Symposium on the Foundations of Computer Science* (*FOCS '77*), 222–227.