# On the Local Performance of Simulated Annealing and the (1+1) Evolutionary Algorithm

Thomas Jansen
Universität Dortmund
FB Informatik, LS 2
44221 Dortmund, Germany
Thomas.Jansen@udo.edu

Ingo Wegener
Universität Dortmund
FB Informatik, LS 2
44221 Dortmund, Germany
Ingo.Wegener@udo.edu

## ABSTRACT

Simulated annealing and the (1+1) EA, a simple evolutionary algorithm, are both general randomized search heuristics that optimize any objective function with probability converging to 1. But they use very different techniques to achieve this global convergence. The (1+1) EA applies global mutations than can reach any point in the search space in one step together with an elitist selection mechanism. Simulated annealing restricts its search to a neighborhood but employs a randomized selection scheme where the probability for accepting a move to a new point in the search space depends on the difference in function values as well as on the current time step. Otherwise, the two algorithms are equal. It is known that the different philosophies of search implemented in the two heuristics can lead to exponential performance gaps between the two algorithms with respect to the expected optimization time. Even for very restricted classes of objective functions where the differences in function values between neighboring points are strictly limited the performance differences can be huge. Here, a more local point of view is taken. Considering obstacles in the fitness landscapes it is proven that the local performance of the two algorithms is remarkably similar in spite of their different search behaviors.

## Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems;
G.3 [**Probability and Statistics**]: Probabilistic Algorithms

## General Terms

Algorithms, Performance, Theory

## Keywords

simulated annealing, evolutionary algorithms, local performance, run time analysis, mutation, selection


*GECCO'06,* July 8–12, 2006, Seattle, Washington, USA.


## 1. INTRODUCTION

General randomized search heuristics are a broad class of algorithms that are often applied to optimization problems when no satisfactory problem-specific algorithm is at hand. Well-known examples for such search heuristics are simulated annealing (SA) and evolutionary algorithms (EAs). While these algorithms are important for the pragmatic solution of real-world problems, there is a growing body of theoretical work concerned with the analysis of the performance such algorithms can deliver and their restrictions. One aspect that routinely is studied is the global convergence of these search heuristics, i. e., the question whether an algorithm finds an optimal solution for any given objective function with a probability that converges to 1 with increasing run time of the algorithm. In particular, such studies have been made for simulated annealing (see for example [2, 5]) as well as for evolutionary algorithms (see for example [15]). Due to the generality of these results they cannot come with any upper bound on the optimization time that is of practical relevance: since they hold for any objective function they also hold for classes of very hard problems where all algorithms need on average exponential optimization time [3]. More relevant with respect to practical applications are studies of the expected optimization time on restricted classes of objective functions. Examples of studies for concrete classes of functions include [11, 16, 17] among many others.

Considering simulated annealing and a very simple evolutionary algorithm, namely the (1+1) EA, it is interesting to see how the two algorithms manage to guarantee global convergence. We consider both algorithms for the maximization of a pseudo-boolean function $f$, i. e., $f\colon \{0,1\}^n \to \mathbb{R}$. Both algorithms search based on one single current point in the search space and create in each round only one new point by randomly changing the current search point. Whereas simulated annealing picks this point from the neighborhood of the current point the (1+1) EA can choose any point in the search space using standard bit mutations. It is this property that is crucial for the global convergence of the (1+1) EA. After the new point is chosen, both algorithms consider the function values of the old and the new point and decide whether they want to accept the new point as new current search point or rather stick to the old one. The (1+1) EA keeps the old point if its function value is larger than that of the new point. In simulated annealing, the same rule is applied. In addition, the new point may still replace the old point (if its function value is smaller) with some probability depending on the difference in function values

and the time elapsed since the start of the algorithm. The function that defines this probability is referred to as cooling schedule and only the choice of an appropriate cooling schedule guarantees global convergence.

In some sense simulated annealing and the (1+1) EA are quite similar: they perform a random search based on a population of size 1 using an offspring population of size 1 and relying on mutation and selection, only. On the other hand, the two algorithms are very different. The (1+1) EA relies on a global mutation and uses strict elitist selection together guaranteeing global convergence. Simulated annealing uses a local mutation but a more complex probabilistic selection leading to global convergence given that an appropriate cooling schedule is used. It is therefore natural to ask if the two randomized search heuristics show similar performance or if reasons for performance differences can be identified. Answers to these questions can help to choose the appropriate search heuristic when characteristics of the objective function are known.

Without any restriction on the objective function, it is not difficult to come up with example functions that demonstrate an exponential performance gap between the expected optimization time of simulated annealing and the (1+1) EA — in both directions. If an objective function has a unique global optimum that is surrounded by a (with respect to Hamming distance) small valley with enormously small function values, the (1+1) EA will have no difficulties to jump over this valley whereas simulated annealing needs exponentially many steps to accept such a huge decrease in function value. If, on the other hand, an objective function has a unique global optimum that is located at the end of a thin ridge of linear length with function values that on the ridge are almost constant but only decreasing very slightly then simulated annealing can perform an almost unbiased random walk on this ridge and find the optimal solution quickly whereas the (1+1) EA will wait an exponential amount of time for a big mutation across the ridge to the global optimum. In [7], such examples are elaborated and the performance of the two search heuristics is analyzed rigorously. Therefore, restrictions on the objective functions are necessary in order to come to more meaningful results.

Again in [7], a class of functions $f\colon \{0,1\}^n \to \mathbb{Z}$ called *smooth integer* is defined, where the function values of two Hamming neighbors differ by at most 1. Clearly, the examples discussed above are not smooth integer functions. But even on such smooth functions simulated annealing and the (1+1) EA may have expected optimization times that differ by a factor that can be as large as $n^k$ for any constant $k$. Here, we consider such smooth integer functions and describe obstacles in the fitness landscapes the two search heuristics may encounter. We present a rigorous performance analysis of the expected time needed to overcome one such local obstacle. In this sense, we present an analysis of the local performance of simulated annealing and the (1+1) EA. We show that in spite of the very differing global performance the time needed to overcome such local obstacles is remarkably similar for the two algorithms. This is, as far as we know, the first theoretical analysis of such different general search heuristics on a quite large class of functions yielding such a general result.

In the next section, we present formal definitions of both algorithms, the class of functions considered, and the type of obstacles we investigate. In Section 3, we present and prove results on the expected time needed to overcome these obstacles. We conclude with remarks on possible future research in Section 4.

## 2. DEFINITIONS

Kirkpatrick, Gelatt, and Vecchi [12] introduced simulated annealing as a global optimization heuristic that is inspired by the annealing process in metallurgy. The algorithm is a generalization of the well-known Metropolis algorithm [13]. We give a formal definition of SA suitable for maximization of a function $f\colon \{0,1\}^n \to \mathbb{R}$.

**Simulated Annealing (SA)**

0. $t := 1$
1. Choose $x_t \in \{0,1\}^n$ uniformly at random.
2. $y := x_t$;
   Choose $i \in \{1,\dots,n\}$ uniformly at random,
   set the bit $y_i := 1 - y_i$.
3. With probability $\min\{1, \alpha(t)^{f(y)-f(x_t)}\}$ set
   $x_{t+1} := y$, else set $x_{t+1} := x_t$.
4. $t := t + 1$
5. Continue at line 2.

The function $\alpha\colon \mathbb{N} \to [1;\infty($ plays the role of the cooling schedule. Usually one has, in the case of maximization, a probability $\min\left\{1, e^{(f(y)-f(x))/T(t)}\right\}$ for accepting a move from $x$ to $y$ where $T(t)$ is the current temperature. In our notation, we set $\alpha(t) := e^{1/T(t)}$ which implies that $\alpha(t) \geq 1$.

**(1+1) EA**

0. $t := 1$
1. Choose $x_t \in \{0,1\}^n$ uniformly at random.
2. $y := x_t$;
   Independently for each bit $y_i$, with
   probability $p_m(t)$ set $y_i := 1 - y_i$.
3. If $f(y) \geq f(x_t)$, set $x_{t+1} := y$, else set $x_{t+1} := x_t$.
4. $t := t + 1$
5. Continue at line 2.

Usually, the (1+1) EA is used with a fixed mutation probability $p_m$ with $p_m = 1/n$ being the most recommended and usual choice. It is known that situations exist where this standard setting is far from being optimal [8]. Moreover, it is also known that for some problems choosing a time-dependent schedule for the mutation probability can be very beneficial [10]. Therefore, we consider the (1+1) EA with a mutation schedule that may change in every generation. This is in better accordance with a fair comparison with SA, too, since SA uses a time-dependent cooling schedule and not a fixed temperature.

Since both algorithms find a global optimum of any pseudo-boolean function $f$ with probability 1 — at least if the parameters $\alpha$ and $p_m$ are set appropriately — the most interesting question to ask is how long this takes. For both algorithms, we define a random variable that we call $T_{\mathrm{EA}}(f)$ for the (1+1) EA with mutation probability $p_m(t)$ and $T_{\mathrm{SA}}(f)$ for simulated annealing. Both variables are defined by $\min\{t \geq 1 \mid f(x_t) = \max\{f(x') \mid x' \in \{0,1\}^n\}\}$. We call $T_{\mathrm{EA}}(f)$ and $T_{\mathrm{SA}}(f)$ the optimization time and are mostly interested in $\mathrm{E}(T_{\mathrm{EA}}(f))$ and $\mathrm{E}(T_{\mathrm{SA}}(f))$, the expected optimization times.

Run time analyses of (randomized) algorithms are hardly ever exact to the level of single steps. Typically, worst case analyses presenting asymptotic results with respect to the

size of the inputs $n$ are given [1, 14]. We adopt this approach and present analyses presenting asymptotic results with respect to the dimension of the search space $n$. We give a formal description for the well-known notions for the asymptotic growth of functions for the sake of completeness.

DEFINITION 1. *Let $f, g\colon \mathbb{N} \to \mathbb{R}^+$ be two functions.*

- $f = O(g)$ *iff* $\exists n_0 \in \mathbb{N}, c \in \mathbb{R}^+ \colon \forall n \geq n_0 \colon f(n) \leq c \cdot g(n)$.
- $f = \Omega(g)$ *iff* $g = O(f)$.
- $f = \Theta(g)$ *iff* $f = O(g)$ *and* $f = \Omega(g)$.
- $f = o(g)$ *iff* $\lim_{n\to\infty} f(n)/g(n) = 0$.
- $f = \omega(g)$ *iff* $g = o(f)$.

Since large differences in function values present one main source of performance differences for SA and the (1+1) EA, we rule this out by considering smooth integer functions, only. Moreover, we will restrict ourselves to functions of unitation, i. e., functions where the function value $f(x)$ depends on the number of one-bits in $x$, only. The latter choice makes the description of the functions much easier. Note, however, that for more general smooth integer functions effects may occur that cannot be observed with functions of unitation. In this sense we present a quite specific analysis, here.

DEFINITION 2. *A pseudo-boolean function $f\colon \{0,1\}^n \to \mathbb{R}$ is called* smooth integer (s. i.)*, if $f(x) \in \mathbb{Z}$ for all $x \in \{0,1\}^n$ and if $|f(x) - f(y)| \leq H(x,y)$ for all $x, y \in \{0,1\}^n$, where $H(x,y)$ denotes the Hamming distance between $x$ and $y$.*

DEFINITION 3. *A pseudo-boolean function $f\colon \{0,1\}^n \to \mathbb{R}$ is called a* function of unitation*, if $f(x) = f(y)$ holds for all $x, y \in \{0,1\}^n$ with $H(x,0^n) = H(y,0^n)$. Let $U_n$ denote the set of all such functions. Let $U_n^*$ denote the set of all functions from $U_n$ that have a unique global optimum at the all one bit string $1^n$.*

Clearly, the class $U_n^*$ is a quite restricted class of pseudo-boolean functions. Note, however, that many example functions typically considered when analyzing the performance of evolutionary algorithms belong to this class. Moreover, many other example functions can be changed in such a way that they keep their characteristic properties while becoming members of $U_n^*$.

We are interested in comparing the local performance of SA and the (1+1) EA when meeting "obstacles" in the landscape defined by some objective function from $U_n^*$. Since the unique global optimum is at $1^n$, we measure the average time the heuristics need to reach $1^n$ when considering the optimization time. Assume that the current search point contains $k$ one-bits (with $k < n$) and consequently $n - k$ zero-bits. Clearly, both algorithms need to visit some point with at least $k + d$ one-bits (with $d > 0$) in order to reach the global optimum. More particular, SA needs to visit at least one point with exactly $k+d$ one-bits for each $d < n-k$ before reaching the global optimum. This is due to the local search operator used by SA. The (1+1) EA, on the other hand, may reach $1^n$ via a direct mutation. But this becomes less likely with increasing distance to the optimum. One may believe that this can be counteracted by increasing the mutation probability. But since the number of possible "target points" (i. e., strings with at least $k + d'$ one-bits) decreases with increasing $d'$ rather rapidly, this counteraction is useful only to a certain extent. This reasoning assumes, however, that we have $k \geq n/2$. But we are safe to assume this anyway, since restarts are a common and useful mechanism to increase the performance of any randomized search heuristic [6]. Using an appropriate restart mechanism one can have runs where initially the number of ones in the population is bounded below by $(n/2) + v$ with probability close to 1 for not too large values of $v$. For $v = \Omega(\sqrt{n})$ however, this probability converges to 0 quickly with growing $v$. Thus, we restrict our considerations to obstacles at positions $k$ where $k = (n/2) + \Omega(\sqrt{n})$ holds. Obstacles at positions $k$ with smaller values of $k$ can pose no difficulties that cannot be escaped by restarts alone.

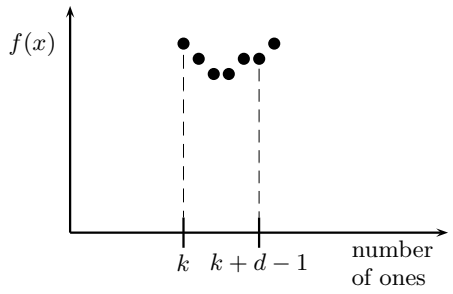


**Figure 1: Illustration of an $(k,d)$-obstacle for the (1+1) EA.**

Since SA and the (1+1) EA employ quite different search strategies, different kinds of landscapes pose difficulties for these two algorithms. We mirror this fact in the following definition of an obstacle for the two algorithms.

DEFINITION 4. *Let $f \in U_n^*$ be a function of unitation and smooth integer and let $v_i$ be its value on inputs with exactly $i$ ones. The function $f$ has an $(k,d)$-obstacle for the (1+1) EA if $v_m < v_k$ if $k < m < k + d$ and $v_{k+d} \geq v_k$. The function has a $(k,d)$-obstacle for SA if $v_k > v_{k+1} > \cdots > v_{k+d-1}$ and $v_{k+d} \geq v_{k+1-1}$.*

Since the (1+1) EA does not accept any point with smaller function value, any region with function values strictly smaller than that of the current search point is an obstacle for the (1+1) EA. The exact function values in that region are of no importance. An illustration of one such $(k,d)$-obstacle for the (1+1) EA can be seen in Figure 1.

Simulated annealing can accept any neighboring point as new search point but the probability for a move decreases with the function value of the new point. Thus, we define an obstacle for SA as a sequence of points with decreasing function values. An illustration of such an obstacle is given in Figure 2.

Note that obstacles for simulated annealing are stronger: each $(k,d)$-obstacle for SA is also a $(k,d')$-obstacle for the (1+1) EA for some $d' \geq d$. The converse is not true.

We use the notion "an algorithm can overcome a $(k,d)$-obstacle" to describe that the expected time the algorithm needs to to reach a level $l$ where $l \geq k+d$ starting from level $k$ is polynomial if steps to levels $l' < k$ are never accepted. In this restricted framework, we are now prepared to compare

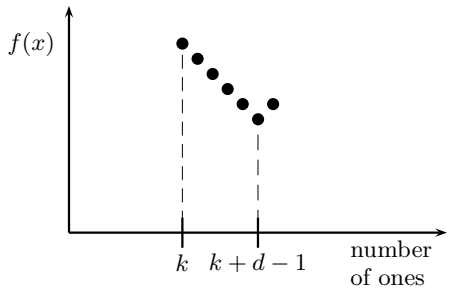


**Figure 2: Illustration of an $(k, d)$-obstacle for SA.**

the performance of simulated annealing and the (1+1) EA in sight of such obstacles. Note that for both algorithms appropriate settings of the parameters can and should be used. Thus, 'SA can overcome an obstacle' means that it can do so using an appropriate annealing schedule. Similarly, 'the (1+1) EA can overcome an obstacle' means it can do so using an appropriate mutation probability.

# 3. ANALYSIS OF THE PERFORMANCE AT OBSTACLES

While the expected optimization time is a global measure we take a more local point of view, here. We determine which obstacles the two heuristics can overcome in expected polynomial time. Note that this is still a much more global point of view than considering the one-step behavior. In particular, it is known that comparing the one-step behavior of search heuristics may have no relevance with respect to the global performance at all [9].

We begin with a result for simulated annealing that gives an asymptotically precise answer to the question which size obstacles may have that still allow for polynomial expected optimization times.

THEOREM 1. *SA can overcome $(k(n), d(n))$-obstacles for SA with $k(n) = (n/2) + \Omega(\sqrt{n})$ iff*

$$d(n) = O\left(1 + \frac{\log(n - k(n))}{\log(n) - \log(n - k(n))}\right)$$

*holds.*

PROOF. Let $k(n) < m < k(n) + d(n)$ and let the current string of SA contain exactly $m$ ones. The probability that the string of the next generation contains $m - 1$ ones equals $m/n$ and the probability that the string of the next generation contains $m+1$ ones equals $(n-m)/(\alpha(t)n)$. Since $\alpha(t) \geq 1$ and SA cannot over-jump levels, we can set $\alpha(t) := 1$.

First, we optimistically replace the probabilities to $k(n)/n$ for going one level down and $(n - k(n))/n$ for going one level up. Then we are in the situation of the gambler's ruin problem [4]. It takes an expected time of $O(n)$ to reach level $k(n) + 1$. Then Alice owns one dollar and Bob owns $d(n) - 1$ dollars. Alice's probability of winning one round equals $(n - k(n))/n < 1/2$. For $t(n) = k(n)/(n - k(n))$, the probability that Alice ruins Bob before being ruined herself equals

$$\frac{t(n) - 1}{t(n)^{d(n)} - 1}.$$

Such a win is necessary to overcome the obstacle. She needs an expected number of trials to ruin Bob once that is polynomially bounded iff the reciprocal

$$\frac{t(n)^{d(n)} - 1}{t(n) - 1}$$

is polynomially bounded. This is equivalent to the condition that $t(n)^{d(n)}$ is polynomially bounded, more precisely

$$\left(\frac{k(n)}{n - k(n)}\right)^{d(n)} = n^{O(1)}.$$

This is equivalent to

$$d(n)\,(\log(n) - \log(n - k(n))) = O(\log n)$$

and

$$d(n) = O\left(1 + \frac{\log(n - k(n))}{\log(n) - \log(n - k(n))}\right).$$

We have made the optimistic assumption that the probability of going one level up does not decrease as it does in reality. It gets its smallest value of $(n - k(n) - d(n) + 1)/n$ one level below the target level. Then the critical term $t(n)$ equals

$$\frac{k(n) + d(n) - 1}{n - k(n) - d(n) + 1}.$$

As long as $d(n)$ fulfills the inequality derived above this value is at most by a constant $c$ larger than the best situation one level above the initial level. Since $d(n) = O(\log n)$, $c^{d(n)}$ is polynomially bounded and we have proved the theorem. □

The following examples illustrate the bound of Theorem 1. Let $\gamma(n) := n - k(n)$. Then

- $\gamma(n) = cn$ and $c < 1/2$ implies $d(n) = O(\log n)$.
- $\gamma(n) = n/\log n$ implies $d(n) = O(\log(n)/\log\log n)$.
- $\gamma(n) = O(n^{\varepsilon})$ and $\varepsilon < 1$ implies $d(n) = O(1)$.

The closer the obstacle is located to the unique global optimum $1^n$, the smaller it has to be if simulated annealing is to cope with it on average in polynomial time. This is due to the search bias induced by the local search operator of simulated annealing. The closer the algorithm gets to the unique global optimum the less likely are steps still decreasing the Hamming distance to this point. Clearly, the (1+1) EA with its mutation operator acting globally can behave quite differently. Interestingly, the following result reveals that the performance with respect to the obstacles we consider is nevertheless very similar.

THEOREM 2. *The (1+1) EA can overcome $(k(n), d(n))$-obstacles for the (1+1) EA with $k(n) = (n/2) + \Omega(\sqrt{n})$ iff*

$$d(n) = O\left(1 + \frac{\log(n - k(n))}{\log(n) - \log(n - k(n))}\right)$$

*holds.*

PROOF. The (1+1) EA overcomes a $(k(n), d(n))$-obstacle iff a mutation of the current string containing exactly $k(n)$ ones and $\gamma(n)$ zeros (with $\gamma(n) = n - k(n)$ as above) produces an offspring with at least $k(n) + d(n)$ ones. This happens if exactly $l + i$ out of the $\gamma(n)$ zeros and $i$ out of the $k(n)$ ones flip for some $l \in \{d(n), d(n) + 1, \ldots, \gamma(n)\}$, $i \in \{0, 1, \ldots, \gamma(n) - l\}$. Let $p_1(n, k(n), l, i)$ denote the probability for the mutation of $i$ out of $k(n)$ ones and let $p_0(n,$

$k(n)$, $l$, $i$) denote the probability for the mutation of $l+i$ out of $\gamma(n)$ zeros. Clearly, we have

$$\begin{aligned} p_1(n,k(n),l,i) &= \binom{k(n)}{i} \cdot p_m(t)^i \cdot (1-p_m(t))^{k(n)-i} \\ p_0(n,k(n),l,i) &= \binom{\gamma(n)}{l+i} \cdot p_m(t)^{l+i} (1-p_m(t))^{\gamma(n)-(l+i)} \end{aligned}$$

and

$$\begin{aligned} &q(n,k(n),l,i) \\ &\quad := \sum_{l=d(n)}^{\gamma(n)} \sum_{i=0}^{\gamma(n)-l} p_0(n,k(n),l,i) \cdot p_1(n,k(n),l,i) \end{aligned}$$

as probability to overcome a $(k(n),d(n))$-obstacle. This probability depends on the mutation probability.

We want to determine when $q(n,k(n),l,i)^{-1} = n^{O(1)}$ holds. Considering the definition of $q(n,k(n),l,i)$ we see that for each value of $l$ the inner sum equals the probability to produce an offspring with exactly $k(n)+l$ ones. Since there are $O(n)$ different values $l$ takes, the total probability is bounded below by the reciprocal of some polynomial iff one of the summands is. Since there are $O(n)$ different values $i$ takes, the same holds for the inner sum. Thus, we consider

$$\begin{aligned} &p_1(n,k(n),l,i) \cdot p_0(n,k(n),l,i) \\ &\quad = \binom{\gamma(n)}{l+i} \cdot \binom{k(n)}{i} \cdot p_m(t)^{l+2i} \cdot (1-p_m(t))^{n-l-2i} \end{aligned}$$

in the following. Considering the first derivative we see that this term takes its maximal value for $p_m(t) = (l+2i)/n$. Therefore, we restrict our attention to this choice in the following. We see that

$$(p_1(n,k(n),l,i) \cdot p_0(n,k(n),l,i))^{-1} = n^{O(1)}$$

holds iff $p_1(n,k(n),l,i)^{-1} = n^{O(1)}$ and $p_0(n,k(n),l,i)^{-1} = n^{O(1)}$ both hold. This is due to the fact that $p_1(n,k(n),l,i)$ and $p_0(n,k(n),l,i)$ are both probabilities. Thus, if one of the two probabilities converges to 0 super-polynomially fast the other factor cannot compensate for that since it is bounded above by 1.

We consider $p_1(n,k(n),l,i)$ first and begin with the special case $i=0$. We have

$$\begin{aligned} p_1(n,k(n),l,0) = \left(1-\frac{l}{n}\right)^{k(n)} &= \left(1-\frac{l}{n}\right)^{(n/l)\cdot l\cdot(k(n)/n)} \\ &= e^{-\Theta((k(n)/n)\cdot l)} = e^{-\Theta(l)} \end{aligned}$$

and see that $p_1(n,k(n),l,0)^{-1} = n^{O(1)}$ holds iff $l = O(\log n)$.

For $i > 0$, let $M_1$ denote the number of flipping ones. This implies $\text{Prob}(M_1 = i) = p_1(n,k(n),l,i)$. The expected number of flipping ones equals $k(n)\cdot(l+2i)/n$ and we see that

$$\text{E}(M_1) = \frac{k(n)}{n} \cdot (l+2i) > i$$

holds. This implies

$$\text{Prob}(M_1 = i) \leq \text{Prob}(M_1 \leq i) \leq (i+1) \cdot \text{Prob}(M_1 = i)$$

since $\text{Prob}(M_1 = i)$ grows with increasing $i$. Since we have $0 < i < n$ it suffices to consider $\text{Prob}(M_1 \leq i)$. Application of Chernoff [14] bounds yields

$$\begin{aligned} &\text{Prob}(M_1 \leq i) \\ &= \text{Prob}\left(M_1 \leq \left(1-\left(1-\frac{ni}{k(n)(l+2i)}\right)\right) \right. \\ &\qquad \left. \cdot \frac{k(n)(l+2i)}{n}\right) \\ &\leq e^{-\frac{k(n)}{n}\cdot(l+2i)\cdot\left(1-\frac{n}{k(n)}\cdot\frac{i}{l+2i}\right)^2/2} = e^{-\frac{(lk(n)+2ik(n)-ni)^2}{2nk(n)(l+2i)}} \\ &= e^{-\frac{\left(l+2i-\frac{n}{k(n)}\cdot i\right)^2}{2\cdot\frac{n}{k(n)}\cdot(l+2i)}} = e^{-\Theta(l+i)}. \end{aligned}$$

We see that $p_1(n,k(n),l,i)^{-1} = n^{O(1)}$ holds iff $l+i = O(\log n)$. We take this into account when considering $p_0(n,k(n),l,i)$.

In the special case $l+i = \gamma(n)$ we have

$$p_0(n,k(n),l,i) = \left(\frac{l+2i}{n}\right)^{l+i} < \left(\frac{2(l+i)}{n}\right)^{l+i}.$$

We see that $(n/(2(l+i)))^{l+i} = n^{O(1)}$ holds iff $l+i = O(1)$ in this case.

In the general case we use $0 < d(n) \leq l+i < \gamma(n)$ and have

$$\begin{aligned} &p_0(n,k(n),l,i) \\ &= \binom{\gamma(n)}{l+i} \cdot \left(\frac{l+2i}{n}\right)^{l+i} \cdot \left(1-\frac{l+2i}{n}\right)^{\gamma(n)-l-i} \\ &= \frac{\gamma(n)!}{(l+i)!\cdot(\gamma(n)-l-i)!} \cdot \left(\frac{l+2i}{n}\right)^{l+i} \\ &\qquad \cdot \left(1-\frac{l+2i}{n}\right)^{\gamma(n)-l-i} \\ &= \Theta\left(\sqrt{\frac{\gamma(n)}{(l+i)(\gamma(n)-l-i)}} \cdot \left(\frac{\gamma(n)}{\gamma(n)-l-i}\right)^{\gamma(n)-l-i} \right. \\ &\qquad \cdot \left(\frac{\gamma(n)}{n}\right)^{l+i} \cdot \left(\frac{l+2i}{l+i}\right)^{l+i} \cdot \left(\frac{l+2i}{n}\right)^{i} \cdot \\ &\qquad \left. \left(1-\frac{l+2i}{n}\right)^{\gamma(n)-l-i}\right) \end{aligned}$$

by Stirling's formula. Remember that we have $l+i = O(\log n)$ and only want to determine when $p_0(n,\ k(n),\ l,\ i)^{-1} = n^{O(1)}$ holds. We see that

$$\begin{aligned} \sqrt{\frac{(l+i)(\gamma(n)-l-i)}{\gamma(n)}} &= n^{O(1)}, \\ \left(\frac{l+i}{l+2i}\right)^{l+i} &= \left(1+\frac{i}{l+2i}\right)^{\frac{l+2i}{i}\cdot i\cdot\frac{l+i}{l+2i}} \\ &= e^{O(i)} = n^{O(1)}, \end{aligned}$$

and

$$\begin{aligned} &\left(\left(1-\frac{l+2i}{n}\right)^{\gamma(n)-l-i}\right)^{-1} \\ &= \left(\left(1-\frac{l+2i}{n}\right)^{\frac{n}{l+2i}\cdot\frac{\gamma(n)-l-i}{n}\cdot(l+2i)}\right)^{-1} \\ &= e^{O(l+i)} = n^{O(1)} \end{aligned}$$

hold. Thus, it suffices to determine when

$$\left(\frac{n}{\gamma(n)}\right)^{l+i} \cdot \left(\frac{n}{l+2i}\right)^{i} \cdot \left(\frac{\gamma(n)-l-i}{\gamma(n)}\right)^{\gamma(n)-l-i}$$
$$= \left(\frac{n}{\gamma(n)-l-i}\right)^{l+i} \cdot \left(\frac{n}{l+2i}\right)^{i} \cdot \left(\frac{\gamma(n)-l-i)}{\gamma(n)}\right)^{\gamma(n)}$$
$$= n^{O(1)}$$

holds. Since

$$\left(\frac{\gamma(n)-l-i)}{\gamma(n)}\right)^{\gamma(n)} = \left(1-\frac{l+i}{\gamma(n)}\right)^{\frac{\gamma(n)}{l+i}\cdot(l+i)}$$
$$= e^{\Theta(l+i)} = n^{O(1)}$$

holds, we only have to determine when

$$\left(\frac{n}{\gamma(n)-l-i}\right)^{l+i} \cdot \left(\frac{n}{l+2i}\right)^{i} = n^{O(1)}$$

holds. We consider $(n/(l+2i))^i$ and remember that we can assume $l+i = O(\log n)$. This implies $i = O(1)$ is necessary for $(n/(l+2i))^i = n^{O(1)}$. Remembering that we have $l \geq d$ we see that $p_0(n,k(n),l,i)^{-1} = n^{O(1)}$ holds iff

$$d \ln\left(\frac{n}{\gamma(n)-d}\right) = O(\log n)$$

holds. Clearly,

$$d = O\left(1+\frac{\log(\gamma(n))}{\log(n)-\log(\gamma(n))}\right)$$

suffices since

$$\left(1+\frac{\log(\gamma(n))}{\log(n)-\log(\gamma(n))}\right) = \frac{\log(n)}{\log(n/\gamma(n))}$$

implies

$$d \ln\left(\frac{n}{\gamma(n)-d}\right)$$
$$= O\left(\frac{\log(n)}{\log(n/\gamma(n))} \cdot \log\left(\frac{n}{\gamma(n)-\frac{\log(\gamma(n))}{\log(n)-\log(\gamma(n))}}\right)\right)$$
$$= O(\log n).$$

Furthermore,

$$d = \omega\left(1+\frac{\log(\gamma(n))}{\log(n)-\log(\gamma(n))}\right)$$

implies

$$d \ln\left(\frac{n}{\gamma(n)-d}\right) = \omega(\log n)$$

in the same way. Using $\gamma(n) = n - k(n)$ completes the proof. □

Theorem 2 reveals an interesting fact about the performance of the (1+1) EA on functions from $f \in U_n^*$ that are smooth integer. Due to our definition, there are many ways how the (1+1) EA can overcome a $(k(n),d(n))$-obstacle: reaching any point with at least $k(n)+d(n)$ ones suffices. This is in some sense optimistic. When optimizing such a function $f$ it depends on the function values of search points with at least $k(n)+d(n)$ ones if the $(k(n),d(n))$-obstacle can really be overcome this way. From the calculations in the proof of Theorem 2 we learn that all $(k(n),d(n))$-obstacles are of equal difficulty. Reaching any level $k(n)+d(n)+i$ with $i > 0$ is with respect to polynomial expected waiting time not easier than reaching the level $k(n)+d(n)$.

Due to the differences in the search operator and the different ways of accepting new search points, obstacles for SA and the (1+1) EA are different. But Theorems 1 and 2 reveal that SA and the (1+1) EA can overcome obstacles of the same size and location in expected polynomial time. Note, however, that nothing is said about the concrete expected number of steps needed by both algorithms. We know that the expected time needed to overcome an obstacle is polynomial for both algorithms (given that size and location of the obstacle is given as described in Theorems 1 and 2) but we should not expect to see polynomials of the same degree.

# 4. CONCLUSIONS

Simulated annealing and the (1+1) EA are two examples for general randomized search heuristics, a large class of algorithms that are often used in real-world optimization problems when no problem-specific algorithm can be used. The theoretical analysis of such search heuristic has the potential to deliver insights in the behavior of the algorithms that can lead to guidelines describing what specific search heuristic should be used given some characteristics of the optimization problem at hand.

Simulated annealing and the (1+1) EA bear some resemblance but employ very different search mechanisms. It does therefore not come as a surprise that they can exhibit very different search behavior. Concentrating on a very restricted class of objective functions, pseudo-boolean functions that are functions of unitation with unique global optimum at $1^n$ and in addition smooth-integer, we considered the performance of the two algorithms when confronted with an obstacle in the landscape. Since the two algorithms employ different variation operators and different selection strategies, we defined two different types of obstacles, one for each search heuristic under consideration. Keeping in mind that we call algorithms efficient if they provide us with an polynomially bounded expected optimization time, we say that an algorithm is able to overcome such an obstacle if it is able to come beyond this obstacle (closer to the unique global optimum) on average in polynomial time. Our analysis is local in the sense that we assume that the algorithm will either overcome the current obstacle or stay at its beginning — this neglects the possibility that the algorithm may wonder off in some other region of the search space. Adopting this local perspective we are able to prove that SA and the (1+1) EA are able to overcome obstacles of the same size and location. For both algorithms, the maximal allowable size depends on the location of the obstacle: the closer the obstacle is to the global optimum $1^n$ the smaller it needs to be.

The definitions of local obstacles imply that any obstacle for simulated annealing is also an obstacle of at least the same size for the (1+1) EA. The converse, however, does not hold in general. This may lead to the belief that simulated annealing is a more robust search heuristic. But such a conclusion ignores the assumptions made. In particular, assuming that the algorithm does not leave the beginning of the obstacle and cannot "get lost" in the search space enabled us to use $\alpha(t) = 1$ (corresponding to an infinite

temperature) in the performance of SA. Clearly, this choice lets simulated annealing degenerate to a pure random walk, not a very useful general search heuristic. One would have to come up with complete analyses of the expected optimization time in order to show that simulated annealing can benefit from this seemingly advantage when optimizing some objective function.

Clearly, an analysis concentrating on the local performance leaves interesting questions open. Can the similarities that are revealed here be carried over to the analysis of the expected optimization time of some function classes? Can any relevant class of functions be identified where simulated annealing and the (1+1) EA behave similarly? Can the two search heuristics have very different expected optimization times on smooth-integer functions $f \in U_n^*$ where only local obstacles are present where the two algorithms behave similarly?

Answering such questions adds to our understanding of the search mechanisms of simulated annealing and the (1+1) EA. Clearly, such an understanding can lead to a better understanding of other (but similar) randomized search heuristics, in particular to a better understanding of more complex evolutionary algorithms. Moreover, it may lead to practical guidelines for the application of randomized search heuristics to real-world problems.

## 5. ACKNOWLEDGMENTS

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

## 6. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2002.

[2] J. R. Cruz and C. C. Y. Dorea. Simple conditions for the convergence of simulated annealing type algorithms. *Journal of Applied Probability*, 35(4):885–892, 1998.

[3] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 2006. To appear. Available online via Springer Online First. http://www.springerlink.com/index/10.1007/s00224-004-1177-z.

[4] W. Feller. *An Introduction to Probability Theory and its Applications I*. Wiley, 1968.

[5] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, 1988.

[6] T. Jansen. On the analysis of dynamic restart strategies for evolutionary algorithms. In J. Merelo-Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors, *Proceedings of the 7th International Conference on Parallel Problem Solving From Nature (PPSN VII)*, pages 33–43. Springer, 2002. LNCS 2439.

[7] T. Jansen. A comparison of simulated annealing with a simple evolutionary algorithm. In A. H. Wright, M. D. Vose, and K. A. De Jong, editors, *Foundation of Genetic Algorithms 8 (FOGA 2005)*, pages 37–57. Springer, 2005. LNCS 3469.

[8] T. Jansen and I. Wegener. On the choice of the mutation probability for the (1+1) EA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo-Guervos, and H.-P. Schwefel, editors, *Proceedings of the 6th International Conference on Parallel Problem Solving From Nature (PPSN VI)*, pages 89–98. Springer, 2000. LNCS 1917.

[9] T. Jansen and I. Wegener. Evolutionary algorithms — how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599, 2002.

[10] T. Jansen and I. Wegener. On the analysis of a dynamic evolutionary algorithm. *Journal of Discrete Algorithms*, 2006. To appear. Available online via Science Direct. http://dx.doi.org/10.1016/j.jda.2005.01.002.

[11] M. Jerrum and G. Sorkin. Simulated annealing for graph bisection. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 1993)*, pages 94–103. IEEE Press, 1993.

[12] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[14] R. Motwani and P. Rhagavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[15] G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Kovac, 1997.

[16] I. Wegener. Simulated annealing beats metropolis in combinatorial optimization. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*, pages 589–601. Springer, 2005. LNCS 3580.

[17] I. Wegener and C. Witt. On the optimization of monotone polynomials by randomized search heuristics. *Combinatorics, Probability and Computing*, 14:225–247, 2005.