Theoretical Analysis of a Mutation-Based Evolutionary Algorithm for a Tracking Problem in the Lattice

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

Ulf Schellbach
Universität Dortmund
FB Informatik, LS 2
44221 Dortmund, Germany
Ulf.Schellbach@uni-dortmund.de

ABSTRACT

Evolutionary algorithms are often applied for solving optimization problems that are too complex or different from classical problems so that the application of classical methods is difficult. One example are dynamic problems that change with time. An important class of dynamic problems is the class of tracking problems where an algorithm has to find an approximately optimal solution and insure an almost constant quality in spite of the changing problem. For the application of evolutionary algorithms to static optimization problems, the distribution of the optimization time and most often its expected value are most important. Adopting this perspective a simple tracking problem in the lattice is considered and the performance of a mutation-based evolutionary algorithm is evaluated. For the static case, asymptotically tight upper and lower bounds are proven. These results are applied to derive results on the tracking performance for different rates of change.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic Algorithms*; F.2 [**Theory of Computing**]: Analysis of Algorithms and Problem Complexity—*Numerical Algorithms and Problems*

General Terms

Algorithms, Performance, Theory

Keywords

dynamic optimization, tracking problems, mutation operators, offspring population size, run time analysis

GECCO'05, June 25–29, 2005, Washington, DC, USA.

1. INTRODUCTION

Evolutionary algorithms (EAs) are a class of randomized search heuristics and are often used for optimization. They are easy to implement, can be applied to almost any kind of optimization problem and are therefore often the method of choice if no problem-specific optimization method is known and there is no time or expertise to develop one. There are many reports on successful applications. In order to deepen our understanding of evolutionary algorithms it is desirable to have analytical results on their behavior. In the context of optimization, the optimization time is the most important issue to investigate. Different from other algorithms, evolutionary algorithms are not designed in a way that supports their analysis. Thus, EA theory is quite difficult and it makes sense to start with simplified evolutionary algorithms. There are a lot of examples for such research for static optimization problems (see [1] for an overview). For dynamic evolutionary algorithms, only few papers deal with rigorous analyses of the optimization and tracking behavior. For example, Branke [2] concentrates on the development of EA variants for dynamic optimization and presents mostly empirical results. Some work by Droste [4, 5] is an exception: for a dynamic variant of the well-known OneMax-problem the expected first hitting time of the global optimum is analyzed. Since only the case of locating the global optimum exactly is considered, the problem has to be restricted to a slowly changing problem. Moreover, the more interesting tracking behavior is not examined.

Here, we consider a simple problem in the two-dimensional lattice that may be described as a OneMax-variant, too. The original problem was introduced by Weicker [11], but we consider a more general variant. Furthermore, Weicker [11] concentrates on the behavior of an EA in one generation whereas we are interested in the expected time to come sufficiently close to a global optimum for a simple EA and in its tracking behavior.

In the next section, we give precise definitions of the class of problems, the evolutionary algorithm used, and our analytical framework. We begin our analysis with the simple and degenerate case of a static problem that does not change over time (Section 3). For this case we can prove asymptotically tight upper and lower bounds for the expected optimization time. Furthermore, these results deliver tight bounds on the average speed of the evolutionary algorithm that turn out to be very helpful in the dynamic case. In

Section 4, we distinguish slow and fast changing problems and prove results on the tracking behavior. However, our emphasis is on problems which change in some sense more slowly than the EA we consider. We add to our theoretical analysis the results of empirical investigations in Section 5. Since all our results are asymptotic, this provides valuable insights for "small cases." Furthermore, it demonstrates the effects of small changes to the algorithm. This may be seen as a first step towards a more complete theoretical analysis. We conclude with a discussion of our results and some remarks on possible future research in Section 6.

2. DEFINITIONS

We consider dynamic problems in the lattice where the objective function is defined as function $f\colon \mathbb{Z}^2 \times \mathbb{N} \to \mathbb{R}$. The first argument $x \in \mathbb{Z}^2$ is a point in the two-dimensional lattice and the second argument $t \in \mathbb{N}$ gives the time step. Without loss of generality we assume that the aim of optimization is minimization. We consider a very simple problem here that is defined by a single moving target point $a_t \in \mathbb{Z}^2$. For a point $x \in \mathbb{Z}^2$ and a time step $t \in \mathbb{N}$, the function value $f(x,t)$ is given by the distance between x and the current target point a_t. We use the Manhattan metric to measure distances, thus for $x = (x_1, x_2) \in \mathbb{Z}^2$ and $y = (y_1, y_2) \in \mathbb{Z}^2$, the distance between x and y is given by $d(x,y) = |x_1 - y_1| + |x_2 - y_2|$. By $d(x)$ we denote the length of x given by $d((0,0),x)$. We limit the speed of the target by requiring that $d(a_t, a_{t+1}) \leq d_{\max}$ holds for all $t \in \mathbb{N}$ and some fixed $d_{\max} \in \mathbb{N}$. In each generation, the evolutionary algorithm can produce some offspring and compute the fitness values, i. e. the distances to a_t. Since the target may move in the next generation, it makes no sense to ask for hitting the target point a_{t+1} exactly. Instead, we are satisfied if $d(x, a_{t+1}) \leq d_{\max}$ holds for some member x of the population. This can be guaranteed if the algorithm manages to locate a_t and is thus in some sense the best that can be achieved. An even better performance is possible only if some additional assumption on the movement of the target a_t is made. In particular, one may think of the case of a target point that moves uniformly by addition of a fixed movement vector $v \in \mathbb{Z}^2$ with $d(v) \leq d_{\max}$, a target moving randomly according to some bounded probability distribution, or even a target that is moved by some adversary. However, we make no other assumptions about the way the target moves which has the advantage of leaving us with a quite broad class of tracking problems.

The evolutionary algorithm used can be described as a (1+λ) EA: it utilizes a population of size only 1, i. e., a single point x in the search space $\mathbb{Z}^2$. Since we make no assumption about the initial position of the target a_1, the initial population can be chosen in an arbitrary fashion. In each generation, the EA produces λ offspring independently by mutation of x. In the selection step, the parent x is replaced by one of its offspring $y_1, y_2, \ldots, y_\lambda$, if and only if at least for one i the distance to the target is not larger, i. e., $\exists i \in \{1, 2, \ldots, \lambda\}\colon\ d(x, a_t) \geq d(y_i, a_t)$ holds. In this case, an offspring with minimal distance to a_t replaces x. If there is more than one such offspring we choose one of these randomly.

It is well known that the choice of the offspring population size can have great impact on the behavior of the (1+λ) EA [9]. We are interested in the effects of λ and $d_{\max}$ on the expected time needed to reduce the distance to the target to at most $d_{\max}$ for the first time and the subsequent tracking behavior.

Clearly, the choice of the mutation operator is crucial. For the two-dimensional lattice $\mathbb{Z}^2$ there is no standard mutation operator that is widely used. Clearly, it is a discrete search space. Thus, it seems appropriate to design a mutation operator that coincides at least in spirit with typical mutation operators for other discrete search spaces. We consider standard bit mutation with a fixed mutation probability of $1/l$ to be the most widely used such mutation operator. It is applied in the search space $\{0,1\}^l$. It may be described in the following fashion. A local mutation step consists of the flipping of a single bit. In each mutation, there is a random number k of such local steps where (for the standard mutation probability $1/l$) this number k is approximately distributed according to a Poisson distribution with parameter 1. We mimic this behavior in the following way. One local operation consists of choosing one of the vectors $(1,0)$, $(-1,0)$, $(0,1)$, and $(0,-1)$ uniformly at random, thus choosing any of these vectors with probability $1/4$. One mutation consists of k independent local operations with outcome $m_1, m_2, \ldots, m_k$, and computing the sum $m := \sum_{i=1}^{k} m_i$. The number of local operations k is a random number drawn according to the Poisson distribution with parameter 1, i. e., we have $\text{Prob}(k = r) = 1/(e \cdot r!)$. The offspring $y \in \mathbb{Z}^2$ that results from this mutation applied to some $x \in \mathbb{Z}^2$ is $y := x + m$. For the sake of clarity, we give a formal definition of the complete algorithm.

Algorithm 1. The (1+λ) EA

1. **Initialization**
   $t := 1$
   $x_t := (0,0)$
2. **Mutation**
   For $i := 1$ To λ Do
      $m := (0,0)$
      Choose $k \in \mathbb{N}$ according to a Poisson distribution with parameter 1.
      For $j := 1$ To k Do
         Choose $m' \in \{(1,0), (-1,0), (0,1), (0,-1)\}$ uniformly at random.
         $m := m + m'$
      $y_i := x_t + m$
3. **Selection**
   $f_{\min} := \min\{f(y_1,t), \ldots, f(y_\lambda,t)\}$
   If $f_{\min} \leq f(x_t,t)$ Then
      Choose $y \in \{y' \in \{y_1, \ldots, y_\lambda\} \mid f(y',t) = f_{\min}\}$ uniformly at random.
      $x_{t+1} := y$
   Else $x_{t+1} := x_t$
4. $t := t + 1$
   Continue at line 2.

We measure the computational effort of this (1+λ) EA by counting the number of function evaluations. Let $T_{\lambda,d_{\max}}(n)$ denote the number of function evaluations the (1+λ) EA makes before decreasing the distance to the target point a_t to at most $d_{\max}$ for the first time when n denotes the distance between the initial target point a_1 and the initial population x_1. We are mainly interested in the expected value of this random variable $T_{\lambda,d_{\max}}(n)$. Since the maximal speed $d_{\max}$ and the initial distance n are our only assumptions

about the target and its movement, the expectation is taken over the random choice the (1+λ) EA makes.

We use the well known notions for describing the asymptotic growth of functions, see for example [3]. For the sake of completeness, we repeat the definition here. Using this notation implies that our results are valid if the parameter n that is assumed to be growing is sufficiently large. So for all n that are at least as large as some constant $n_0 \in \mathbb{N}$ our results hold. Note that this is different from an analysis that assumes $n \to \infty$ and is valid in the limit case, only.

Definition 1. For two functions $f\colon \mathbb{N} \to \mathbb{R}$ and $g\colon \mathbb{N} \to \mathbb{R}$, we say that f does not grow faster than g, $f = O(g)$, if there exist $n_0 \in \mathbb{N}$ and $c \in \mathbb{R}^+$, such that for all $n \geq n_0$ we have $f(n) \leq c \cdot g(n)$. We say that f grows at least as fast as g, $f = \Omega(g)$, if $g = O(f)$. We say that f and g have the same order of growth, $f = \Theta(g)$, if $f = O(g)$ and $f = \Omega(g)$. We say that f grows slower than g, $f = o(g)$, if $\lim_{n\to\infty} f(n)/g(n) = 0$ holds. We say that f grows faster than g, $f = \omega(g)$, if $g = o(f)$ holds.

We begin our considerations in the next section with the special case $d_{\max} = 0$. Note that this implies that we are actually investigating the expected first hitting time of the static target point a. Obviously, this first hitting time coincides with the quotient of the initial distance between the (1+λ) EA and the target point and the average speed of the (1+λ) EA. Therefore, results on this static case are useful for the investigation of the dynamic case: it is already intuitively clear that if the maximal speed of the target $d_{\max}$ is below the average speed of the algorithm, the (1+λ) EA will be able to track the target very steadily. Thus, one may read the following section as an investigation of the average speed of the algorithm.

3. THE STATIC CASE

We employ the perspective of asymptotic analysis that is common for the analysis of algorithms. In the classical setting, the (expected) run time is investigated depending on the input size and the results are given asymptotically for growing input sizes. Here, we investigate $T_{\lambda,0}(n)$ depending on the initial distance n and its expectation is given for growing n. Note that the offspring population size λ may depend on this initial distance n, too.

THEOREM 1.

$$E(T_{\lambda,0}(n)) = \Theta\left(\lambda \cdot \left(1 + \frac{n \cdot \log\log\lambda}{\log\lambda}\right)\right)$$

PROOF. In the special case $\lambda = O(1)$ one should read $\Theta(\lambda \cdot (1 + (n \log\log\lambda)/\log\lambda))$ as $\Theta(n)$. All our proof ideas work for this special and very simple case, too. However, most of the calculations presented here hold only for $\lambda = \omega(1)$ which is the more difficult and interesting case. For the simple case $\lambda = O(1)$ it suffices to notice that the average speed of the (1+λ) EA is $\Theta(1)$ and deviations from this expected value are highly unlikely.

In each generation λ offspring are generated and evaluated. Thus, we have to prove that the number of generations that are needed and sufficient in order to hit the target point exactly is $\Theta(1 + (n \log\log\lambda)/\log\lambda)$.

We begin with the upper bound. We do not only prove the upper bound on the expectation but show that this bound holds with a probability that is exponentially close to 1. Start with considering some point y with $d(x,y) = d$ where x denotes the current population of the (1+λ) EA. Let p_d denote the probability that at least one of the λ offspring equals y, i. e., $p_d := \text{Prob}(\exists i \in \{1,\dots,\lambda\}\colon y_i = y)$. Since the λ offspring are generated independently, we can start by considering the creation of y_1 by mutation of x. Regardless of the exact position of y, there is at least one sequence of exactly d local operations that leads from x to y. With probability $1/(e \cdot d!)$, y_1 is generated with exactly d local operations. With probability at least 4^{-d}, these d local operations lead exactly to y. Thus, we have $\text{Prob}(y_1 = y) \geq 1/(e \cdot d! \cdot 4^d)$. Using Stirling's formula we get $\text{Prob}(y_1 = y) > e^d/(e \cdot \sqrt{3\pi d} \cdot d^d \cdot 4^d) = \Theta\left(1/(d^{d+1/2} \cdot (4/e)^d)\right)$. With $b := 4/e$ we get $\text{Prob}(y_1 = y) = \Omega\left(b^{-d-(1/2)\log_b(d) - d\log_b(d)}\right)$ and we see that $\text{Prob}(y_1 = y) \geq b^{-\tilde{c}d\log_b(d)}$ holds for any constant $\tilde{c} > 1$ (given that d is sufficiently large).

We consider the case $d \leq (c\log_b\lambda)/\log_b\log_b\lambda$ and observe that $\tilde{c}d\log_b(d) \leq \tilde{c} \cdot (c\log_b\lambda/(\log_b\log_b\lambda)) \cdot (\log_b(c) + \log_b(\log_b\lambda) - \log_b\log_b\log_b\lambda) \leq \tilde{c}\cdot c\cdot\log_b\lambda$ follows. Choosing the constant c in a way that $c \cdot \tilde{c} \leq 1/2$ holds yields $\text{Prob}(y_1 = y) \geq b^{-(1/2)\log_b\lambda}$.

Since the λ offspring are generated independently, we have $p_d \geq 1 - \left(1 - b^{-(1/2)\log_b\lambda}\right)^\lambda = 1 - \left(1 - 1/\sqrt{\lambda}\right)^\lambda \geq 1 - e^{-\sqrt{\lambda}}$. So, for all y with $d(y,x) \leq (\log_b\lambda)/(2\tilde{c}\log_b\log_b\lambda)$ (with $b = 4/e$), $p_d \geq 1 - e^{-\sqrt{\lambda}}$ holds. We can conclude that in each generation the distance between the current population x of the (1+λ) EA and the static target point a is decreased by at least $\lfloor(\log_b\lambda)/(2\tilde{c}\log_b\log_b\lambda)\rfloor$ with probability at least $1 - e^{-\sqrt{\lambda}}$. This decreasement $\lfloor(\log_b\lambda)/(2\tilde{c}\log_b\log_b\lambda)\rfloor$ can only be less than 1 for constant values of λ. Thus, we see that this implies $\text{E}(T_{\lambda,0}(n)) = O\left(\lambda \cdot \left(1 + \frac{n\cdot\log\log\lambda}{\log\lambda}\right)\right)$. It is not difficult to see that this bound even holds with a probability exponentially close to 1. For $\lambda = \Omega(n^2)$, this follows immediately from the bound $e^{-\sqrt{\lambda}}$ on the probability not to have the necessary decreasement in one generation. For smaller values of λ, we can consider the $\Omega(n)$ generations as independent experiments, where in each experiment we have a probability of decreasing the distance as desired that is bounded below by some positive constant. Application of Chernoff bounds (see for example [10]) yields that the probability not to reach the target point is exponentially small.

The proof of the lower bound is slightly more involved. First note that at least one generation is necessary in order to reach the target point. This implies $\text{E}(T_{\lambda,0}(n)) = \Omega(\lambda)$. Thus, it suffices to prove that the (1+λ) EA needs on average $\Omega((n\log\log\lambda)/\log\lambda)$ generations to reach the target point.

Clearly, one offspring can only decrease the distance to the target by d if it is created via a mutation consisting of at least d local operations. Therefore, we have $1/(e \cdot d!)$ as upper bound on the probability that one specific offspring decreases the distance to the target by d. Since in each generation λ offspring are created independently, we are interested in an upper bound of the maximal number of local operations in the λ mutations producing these λ offspring. Since the number of local operations is a Poisson distributed random variable for each offspring, there is no strict upper bound. We begin with bounding the expected maximal number of local operations in one generation from above.

Let k_i denote the number of local operations in the creation of offspring y_i, let $k := \max\{k_i \mid i \in \{1,\dots,\lambda\}\}$ be the maximal number of these local operations in one generation. We want to prove an upper bound on $\mathrm{E}(k)$. By definition of the expectation, $\mathrm{E}(k) = \sum_{r=1}^{\infty} r \cdot \mathrm{Prob}(k = r) = \sum_{r=1}^{\infty} \mathrm{Prob}(k \geq r)$ holds. We have $k \geq r$ if there is at least one offspring with r local operations, i. e., there is some $i \in \{1,\dots,\lambda\}$ with $k_i \geq r$. If we have $k_i < r$ for all i, then $k < r$ holds. Since the λ offspring are generated independently, $\mathrm{Prob}(\forall i \in \{1,\dots,\lambda\}\colon k_i < r) = \prod_{i=1}^{\lambda} \mathrm{Prob}(k_i < r)$ holds. The λ offspring are all generated in the same way, thus the k_i are identically distributed and we have $\mathrm{Prob}(\forall i \in \{1,\dots,\lambda\}\colon k_i < r) = \mathrm{Prob}(k_1 < r)^{\lambda}$. Putting this together yields $\mathrm{E}(k) = \sum_{r=1}^{\infty} \left(1 - (1 - \mathrm{Prob}(k_1 \geq r))^{\lambda}\right)$ and we are interested in an upper bound on $\mathrm{Prob}(k_1 \geq r)$. Since we have

$$\begin{aligned}\mathrm{Prob}(k_1 \geq r) &= \sum_{s=r}^{\infty} \mathrm{Prob}(k_1 = s)\\ &= \sum_{s=r}^{\infty} \frac{1}{e \cdot s!} = \frac{1}{e \cdot r!} \cdot \sum_{s=r}^{\infty} \frac{r!}{s!}\\ &\leq \frac{1}{e \cdot r!} \cdot \sum_{s=0}^{\infty} \left(\frac{1}{2}\right)^s = \frac{2}{e \cdot r!}\end{aligned}$$

we can conclude that

$$\mathrm{E}(k) \leq \sum_{r=1}^{\infty} \left(1 - \left(1 - \frac{2}{e \cdot r!}\right)^{\lambda}\right)$$

holds. Clearly, the term in the sum decreases with increasing values of r. Therefore, it makes sense to consider some threshold value t, use 1 as trivial upper bound on the first t terms and be more accurate for the following terms. This yields $\mathrm{E}(k) \leq t + \sum_{r=t+1}^{\infty} \left(1 - (1 - 2/(e \cdot r!))^{\lambda}\right)$ for all values of $t \in \mathbb{N}$. It is well known that $(1+q)^m \geq 1 + q \cdot m$ holds for all $q > -1$ and all $m \in \mathbb{N}$ (Bernoulli inequality). Since $-2/(e \cdot r!) > -1$ holds for all r, we can apply this with $q := -2/(e \cdot r!)$ and $m := \lambda$. This yields

$$\begin{aligned}\mathrm{E}(k) &\leq t + \sum_{r=t+1}^{\infty} \frac{2\lambda}{e \cdot r!} \leq t + \frac{2\lambda}{e \cdot t!} \cdot \sum_{r=t+1}^{\infty} \frac{t!}{r!}\\ &\leq t + \frac{2\lambda}{e \cdot t!} \cdot \sum_{r=1}^{\infty} \left(\frac{1}{2}\right)^r = t + \frac{2\lambda}{e \cdot t!}\end{aligned}$$

for all $t \in \mathbb{N}$. Using $t := \lfloor (e \cdot \ln\lambda)/\ln\ln\lambda \rfloor$ we get $\mathrm{E}(k) \leq e\ln\lambda/(\ln\ln\lambda) + 1/\lambda$, since for $t = \lfloor (e \cdot \ln\lambda)/\ln\ln\lambda \rfloor$ application of Stirling's formula yields

$$\begin{aligned}\frac{2\lambda}{e \cdot \lfloor (e \cdot \ln\lambda)/\ln\ln\lambda \rfloor!} &< \frac{\lambda}{e^{(e\ln\lambda)/(\ln\ln\lambda)\cdot((\ln\ln\lambda) - \ln\ln\ln\lambda)}}\\ &< \lambda/e^{2\ln\lambda} = 1/\lambda.\end{aligned}$$

Remember that k denotes the maximal number of local operations in the λ mutations in one generation and that k is an upper bound on the decreasement of the distance between x and the target in this generation. Let X_i denote this decreasement in the i-th generation. Since the target is not moving and we are considering a (1+λ) EA, the distance to the target cannot increase. Thus, in the t-th generation this distance equals $n - X_1 - X_2 - \cdots - X_t = n - (X_1 + \cdots + X_t)$. Clearly, each X_i is a random variable and we have $\mathrm{E}(X_i) \leq \mathrm{E}(k)$ for all i. Thus, $\mathrm{E}(X_1 + \cdots + X_t) \leq t \cdot \mathrm{E}(k)$ holds and we have $\mathrm{E}\left(\sum_{i=1}^{t} X_i\right) \leq t\,(e\ln\lambda/(\ln\ln\lambda) + 1/\lambda)$. Observing that

$$t \leq \frac{n \ln\ln\lambda}{6\ln\lambda} \Rightarrow t < \frac{n \cdot \lambda \cdot \ln\ln\lambda}{2e\lambda\ln\lambda + 2\ln\ln\lambda}$$

suffices to see that for $t \leq (n\ln\ln\lambda)/(6\ln\lambda)$ we have $\mathrm{E}\left(\sum_{i=1}^{t} X_i\right) < n/2$. Using Markov's inequality, we get that after $(n\ln\ln\lambda)/(6\ln\lambda)$ generations with probability at least $1/2$ the distance to the target is decreased by less than n and thus still larger than 0. This implies the lower bound on $\mathrm{E}(T_{\lambda,0}(n))$. □

We see that the expected first hitting time is minimal for constant offspring population sizes. This could be expected since for simple fitness functions often a (1+1) EA is the most efficient variant of all (1+λ) EAs. We see that offspring population sizes that even grow very slowly with n are inferior to any static choice of λ. Thus, in some sense the problem considered here is even simpler than OneMax for bit strings as a comparison with the results by Jansen and De Jong [9] shows. Note that for implementations in parallel computing environments setting $\lambda > 1$ still makes sense since it decreases the first hitting time.

The most important observation that stems from Theorem 1 is that the (1+λ) EA achieves on average a "speed" of $\Theta((\log\lambda)/\log\log\lambda)$. If the target does not move at all, the distance to the target is decreased by $\Theta((\log\lambda)/\log\log\lambda)$ on average in each generation. If the target moves, the distance to the old target is on average decreased by this amount but the new target may have a distance of $d_{\max}$ to the old target. In the worst case, this distance $d_{\max}$ adds to the distance to the old target. This suggests a case inspection since we expect very different behavior depending on the speed of the target point. If this speed exceeds the average speed of the (1+λ) EA the target will probably escape very easily while for targets moving slower than the (1+λ) EA we expect stable tracking behavior. We make these ideas concrete in the following section.

4. THE DYNAMIC CASE

In accordance to our findings in the last section we distinguish two main cases. The case of a target point that moves slowly from the case of a target point moving quickly.

Before the tracking behavior can be observed, the target point needs to be found. Remember that this means that the distance to the target point has to be decreased to at most $d_{\max}$. Reusing our results from Section 3, one necessary change is obviously to replace n by $n - d_{\max}$. In fact, Theorem 1 already deals with reducing the distance to the target point to at most $n - d_{\max}$ but for the special case $d_{\max} = 0$. The second component in the proof of Theorem 1 is the decreasement of the distance to the target that can be achieved in each generation. Depending on the actual movement of the target point, this can be very different from the static case. For upper bounds on the expected first "hitting" time (we are actually not interested in hitting the target point but coming within a distance of at most $d_{\max}$), we can assume that the target point moves away from the

(1+λ) EA as fast as possible. For lower bounds, additional assumptions are necessary. Otherwise it may be the case that the target point actually moves towards the current search point of the EA leading to a misleading small time.

We begin with a result on this expected first "hitting" time and continue with two results on the tracking behavior. For results on the tracking behavior, we consider a scenario where the initial distance between the population of the (1+λ) EA and the target point is 0. Then, the process is started and we want to see whether the algorithm is able to keep the distance small with probability close to 1. The result on the expected first "hitting" time is very similar to Theorem 1 and meets intuitive expectations.

THEOREM 2. *Let $b := 4/e$, $n' := n - d_{\max}$, $\tilde{c} > 1$, and $s := \lfloor (\log_b \lambda)/(2\tilde{c} \log_b \log_b \lambda) \rfloor$.*
For $d_{\max} \leq \left(\left(2 - 3e^{-\sqrt{\lambda}}\right)/3 \right) s$

$$\mathit{Prob}\left(T_{\lambda,d_{\max}}(n) = O\left(\lambda \cdot \left(1 + \frac{n' \cdot \log\log\lambda}{\log\lambda}\right)\right)\right) = 1 - 2^{-\Omega(n'/s)}$$

holds.

PROOF. We consider the current distance from the target point $D_t := d(x_t, a_t)$, so $D_1 = n$ holds. Due to the plus-selection employed $d(x_{t+1}, a_t) \leq D_t$ holds for all t. However, it may be the case that $D_{t+1} > D_t$ holds for some t since the target point a_{t+1} may be different from a_t. Since we have $d(a_t, a_{t+1}) \leq d_{\max}$ for all t, we can conclude that $D_{t+1} \leq D_t + d_{\max}$ holds. For an upper bound we can assume that a_{t+1} is chosen in such a way that $D_{t+1} = d(x_{t+1}, a_t) + d_{\max}$ holds. The sequence $D = (D_t)_{t \geq 1}$ describes a stochastic process on $\mathbb{N}_0$. We are interested in the minimal $t = t_{\min}$ such that $D_t \leq d_{\max}$ holds. Note that t counts generations and we have λ function evaluations in each generation. Thus, the value of $t_{\min} \cdot \lambda$ equals $T_{\lambda,d_{\max}}(n)$. Due to our worst case assumption D can be simplified to a time and space homogeneous Markov chain $D' = (D'_t)_{t \geq 1}$ with state space $\mathbb{N}_0$ which stochastically dominates D, i. e., $\mathrm{Prob}(D_t \geq k) \leq \mathrm{Prob}(D'_t \geq k)$ holds for all $t, k \in \mathbb{N}$. Obviously, an upper bound on $t_{\min} \cdot \lambda$ such that $D'_{t_{\min}} \leq d_{\max}$ holds is an upper bound on $T_{\lambda,d_{\max}}(n)$, too.

We know $\mathrm{Prob}\left(d(x_{t+1}, a_t) = D_t - s\right) \geq 1 - e^{-\sqrt{\lambda}}$ from the proof of Theorem 1. Let $p_{i,j}$ denote the probability for a transition from i to j for the Markov chain D'. We define $p_{i,j}$ in the following way.

$$p_{i,j} := \begin{cases} 1 - e^{-\sqrt{\lambda}} & \text{if } j = i - (s - d_{\max}) \\ e^{-\sqrt{\lambda}} & \text{if } j = i + d_{\max} \\ 0 & \text{otherwise} \end{cases}$$

We know from the proof of Theorem 1 that with probability at least $1 - e^{-\sqrt{\lambda}}$ the distance to the target point is decreased by at least $s_{\mathrm{eff}} := s - d_{\max}$. With the remaining probability of at most $e^{-\sqrt{\lambda}}$ this distance may be increased, but such an increase is bounded above by $d_{\max}$. Thus, D' does indeed stochastically dominate D. We have $D'_1 = n$ and consider the situation after t steps. Application of Chernoff bounds yields that with probability $1 - e^{-\Omega(t)}$ we have at least $3t(1 - e^{-\sqrt{\lambda}})/4$ steps where D' decreases. So we have $\mathrm{Prob}\left(D'_{1+t} \leq n - (3/4)t\left(1 - e^{-\sqrt{\lambda}}\right) s_{\mathrm{eff}} + (t/4) \cdot \left(4 - 3\left(1 - e^{-\sqrt{\lambda}}\right)\right) d_{\max}\right) = 1 - 2^{-\Omega(t)}$ for all values of t. We consider $t = 4n'/s_{\mathrm{eff}}$. Since we have $s_{\mathrm{eff}} = s - d_{\max}$ with $d_{\max} \leq (2/3)s$, $s_{\mathrm{eff}} = \Theta(s)$ holds and $4n'/s_{\mathrm{eff}} = \Theta(n'/s)$ follows. Since in all non-trivial situations at least one generation is needed, it suffices to prove that for $t = 4n'/s_{\mathrm{eff}}$ we have $D'_{1+t} \leq d_{\max}$ with probability $1 - 2^{-\Omega(t)}$.

We have $d_{\max} \leq \left(\left(2 - 3e^{-\sqrt{\lambda}}\right)/3 \right) s$ for all offspring population sizes λ. We can conclude that $\left(2 - 3e^{-\sqrt{\lambda}}\right) s - \left(2 - 3e^{-\sqrt{\lambda}}\right) d_{\max} \geq \left(1 + 3e^{-\sqrt{\lambda}}\right) d_{\max}$ holds. This implies $\left(2 - 3e^{-\sqrt{\lambda}}\right) s_{\mathrm{eff}} \geq \left(1 + 3e^{-\sqrt{\lambda}}\right) d_{\max}$ and we get

$$\frac{4n'}{s_{\mathrm{eff}}} \geq \frac{4n'}{s_{\mathrm{eff}} + \left(2 - 3e^{-\sqrt{\lambda}}\right) s_{\mathrm{eff}} - \left(1 + 3e^{-\sqrt{\lambda}}\right) d_{\max}}$$

and may consider this earlier point of time instead of $t = 4n'/s_{\mathrm{eff}}$. This leads to

$$n - \frac{4n'}{s_{\mathrm{eff}} + \left(2 - 3e^{-\sqrt{\lambda}}\right) s_{\mathrm{eff}} - \left(1 + 3e^{-\sqrt{\lambda}}\right) d_{\max}} \cdot \frac{3\left(1 - e^{-\sqrt{\lambda}}\right) s_{\mathrm{eff}} - \left(4 - 3\left(1 - e^{-\sqrt{\lambda}}\right)\right) d_{\max}}{4} = n - n' = d_{\max}$$

which completes the proof. □

In Theorem 2 we implicitly defined that a target moves slowly if $d_{\max} \leq \left(\left(2 - 3e^{-\sqrt{\lambda}}\right)/3 \right) s$ holds where s with $s = \lfloor (\log_b \lambda)/(2\tilde{c} \log_b \log_b \lambda) \rfloor$ is in some sense a lower bound on the speed of the (1+λ) EA that we can guarantee with probability at least $1 - e^{-\sqrt{\lambda}}$. For such a slow target we expect that tracking it is easy. The next theorem confirms this intuition.

THEOREM 3. *Let $b := 4/e$, $n' := n - d_{\max}$, $\tilde{c} > 1$, and $s := \lfloor (\log_b \lambda)/(2\tilde{c} \log_b \log_b \lambda) \rfloor$. For $d_{\max} \leq (2/3)s$ and any $k \in \mathbb{N} \setminus \{1\}$ the expected number of generations until the (1+λ) EA has a distance to the target of at least $k \cdot d_{\max}$ after having a distance of at most $d_{\max}$ is bounded below by $e^{\Omega\left(k\sqrt{\lambda}\right)}$.*

PROOF. We consider the same Markov chain D' as in the proof of Theorem 2. Since $d_{\max} \leq (2/3)s$ holds, we have $s_{\mathrm{eff}} = s - d_{\max} \geq d_{\max}/2$. The probability to decrease the distance to the target by at least $d_{\max}$ in two subsequent generations is bounded below by $\left(1 - e^{-\sqrt{\lambda}}\right)^2$. We can model the situation now in the following way. With probability $p := \left(1 - e^{-\sqrt{\lambda}}\right)^2$ the distance to the target is decreased by $d_{\max}$, with the remaining probability $1 - p$ it is increased by $d_{\max}$. This model obviously can only overestimate the distance to the target. Since we are aiming at a lower bound on the expected number of generations to increase the distance to the target beyond some limit, we can use this model to derive such a lower bound. So, this symmetric model dominates D' stochastically. Since it is symmetric we can apply results on the gambler's ruin problem (see for example [7] for the gambler's ruin problem or [8] for another application of its properties to the analysis of evolutionary algorithms). Using the notation of the gambler's ruin problem and normalizing the step size $d_{\max}$ to 1, we have player A starting

with an amount of $k-1$ (which corresponds to $D_1 \leq d_{\max}$), a probability of winning 1 of p, and a probability of losing 1 of $1-p$. If player A increases her holdings to k, we start a new game, again with an initial amount of $k-1$. If we decrease her holdings to 0, we say that she is ruined. We are interested in an upper bound on the probability of player A's ruin. Observe that the reciprocal of this probability of ruin is a lower bound on the number of generations the (1+λ) EA needs to get a distance to the target point of at least $kd_{\max}$ when starting with a distance of at most $d_{\max}$. Applying the results on the gambler's ruin problem we get

$$\frac{\left(\frac{1-p}{p}\right)^{k-1} - \left(\frac{1-p}{p}\right)^{k}}{1-\left(\frac{1-p}{p}\right)^{k}} \leq (1+\varepsilon)\cdot\left(1-\frac{1-p}{p}\right)\cdot\left(\frac{1-p}{p}\right)^{k-1}$$

$$= (1+\varepsilon)\left(1-\frac{1-\left(1-e^{-\sqrt{\lambda}}\right)^2}{\left(1-e^{-\sqrt{\lambda}}\right)^2}\right)\left(\frac{1-\left(1-e^{-\sqrt{\lambda}}\right)^2}{\left(1-e^{-\sqrt{\lambda}}\right)^2}\right)^{k-1}$$

$$\leq 2\cdot e^{-(k-1)\sqrt{\lambda}}\cdot\left(\frac{2-e^{-\sqrt{\lambda}}}{1-2e^{-\sqrt{\lambda}}+e^{-2\sqrt{\lambda}}}\right)^{k-1}$$

$$= e^{-\Omega\left(k\sqrt{\lambda}\right)}.$$

Note that both inequalities hold since we have $p = p(\lambda) = \left(1-e^{-\sqrt{\lambda}}\right)^2$ implying $\lim_{\lambda\to\infty} p(\lambda) = 1$. Thus, we have $p \geq 1-\varepsilon$ for any constant ε with $0 < \varepsilon < 1$ given that λ is sufficiently large. □

If $d_{\max}$ exceeds the average speed of the (1+λ) EA considerably, we do not expect that the EA will be able to track the target. Of course, this depends on the actual movement of the target. Let us consider a worst case scenario where an adversary determines the movement of the target. Now we can assume that the target moves away from x_t by $d_{\max}$ in each generation. Let us assume that the starting point of the target and the (1+λ) EA are the same, thus $d(x_1, a_1) = 0$ holds. We consider a run of the (1+λ) EA after t generations and are interested in an upper bound on the probability that the algorithm was able to track the target. We formalize "tracking" by asking for an upper bound on the probability that the (1+λ) EA has a distance of at most $k \cdot d_{\max}$ to a_{t+1} for some $k \in \mathbb{N}$. Clearly, it makes no sense to consider $k \geq t$; thus we have to assume $k < t$. However, we assume $k \leq (1-\varepsilon)t$ in the following for some constant ε with $0 < \varepsilon < 1$. Now, it is easy to come up with upper bounds on this probability in two different ways.

We know from Theorem 1 that the expected spatial gain of the (1+λ) EA in one generation is bounded above by $O((\log\lambda)/\log\log\lambda)$. We can conclude that the expected spatial gain in t generations is bounded above by $O\left(t\cdot\log\lambda/(\log\log\lambda)\right)$. It is easy to see that if the spatial gain of the algorithm is bounded above by $td_{\max}(1-k/t)$ the distance $d(x_{1+t}, a_{1+t})$ is bounded below by $k\cdot d_{\max}$. Application of Markov's inequality yields that the probability to have at least such a spatial gain within t generations is bounded above by $O\left(\log\lambda/(d_{\max}\cdot\log\log\lambda)\right)$. Note that this bound does neither depend on t nor k and is particularly weak. This is due to the application of Markov's inequality, a very general but not very powerful tool.

A more direct approach yields an upper bound that is somewhat stronger depending on the choice of t and $d_{\max}$. It is easy to see that with probability at most $(t\lambda)/(m!)$ there is at least one mutation with at least m local operations in t generations. If there is no such mutation, the spatial gain in these t generations is obviously bounded above by $t(m-1)$. Thus, we have $O\left(t\lambda/(\lfloor d_{\max}(1-k/t)\rfloor!)\right)$ as upper bound. While this bound does depend on t, it does so "in the wrong direction." Since we consider the case where the target moves faster than the (1+λ) EA, we expect the bound to decrease with increasing values of t. However, our quite primitive and direct estimation basically estimates the probability to have at least one generation where the (1+λ) EA achieves a speed of at least $d_{\max}(1-k/t)$. Obviously, this probability does increase with t. However, for not too small values of $d_{\max}$ the second upper bound is much better than the one based on Markov's inequality. Consider some fixed offspring population size λ, some number of generations that is polynomially bounded in this offspring population size, i. e., $t = \lambda^{O(1)}$, and a target speed $d_{\max}$ with $d_{\max} = \omega((\log\lambda)/\log\log\lambda)$. For the sake of simplicity, let us consider $d_{\max} = \ln\lambda$. For not too large values of k (say $k \leq t/2$), we have $O(1/\log\log\lambda)$ as upper bound that the (1+λ) EA is within a radius of $kd_{\max}$ after t generations due to application of Markov's inequality. The more direct bound yields $O\left(t/\lambda^{-1+((\ln\ln\lambda)-2)/2}\right)$ which is super-polynomial in λ.

Still, both upper bounds are very weak and far from the truth. The difficulty in proving stronger results lies within the Poisson distribution that is applied in mutation: There is no upper bound on the decreasement of the distance to the target in a single generation.

5. EXPERIMENTAL ANALYSIS

Our analysis of the (1+λ) EA provided us with rigorously proven asymptotical results. However, it remains unclear whether the asymptotic descriptions of the first hitting times are realistic approximations for small initial distances, too. Furthermore, they do not allow us to distinguish between different static choices of λ that differ only by some constant factor. Therefore, it makes sense to supplement our theoretical findings with the results of some experiments. The experiments are carried out using a straight-forward implementation in Java, using the Mersenne Twister from the colt distribution for generation of pseudo-random numbers[1]. For all settings, we did 100 runs each with different random seeds and present the average over these runs together with their 95% confidence intervals.

Adding to Theorem 1, we did experiments for initial distances $n \in \{10, 20, 30, \ldots, 1000\}$ and $\lambda \in \{1, 7, \lceil\sqrt{n}\rceil, n\}$. Theorem 1 predicts that the average first hitting time strictly increases with λ whereas the average number of generations decreases with λ. In Figure 1 the number of function evaluations is displayed together with the theoretical upper bound for the case $\lambda = n$. Clearly, the findings match our expectations. Obviously, the evaluation of the λ offspring can be done in parallel. Therefore, it makes sense to consider the number of generations as a measure for the parallel computation time. In Figure 2 the number of generations for the same experiments is displayed. As could be expected, the number of generations decreases with λ.

For dynamic problems, often a non-elitism comma-selection (sometimes also called truncation selection) is expected to

[1]http://hoschek.home.cern.ch/hoschek/colt/

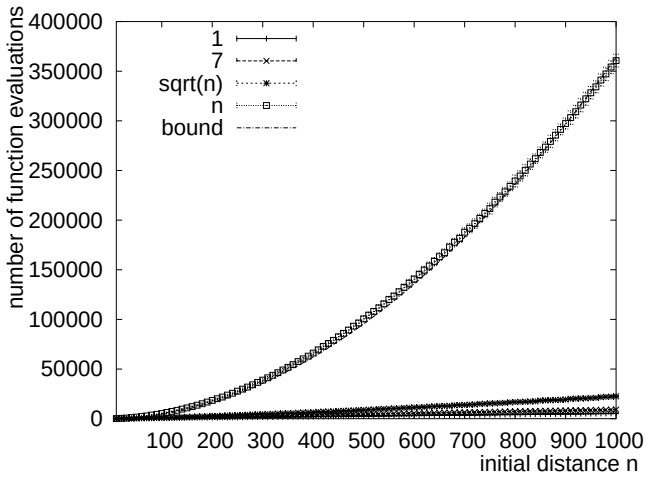

Figure 1: Number of function evaluations for Algorithm 1 for $\lambda \in \{1, 7, \lceil\sqrt{n}\rceil, n\}$, together with the theoretical upper bound for $\lambda = n$.

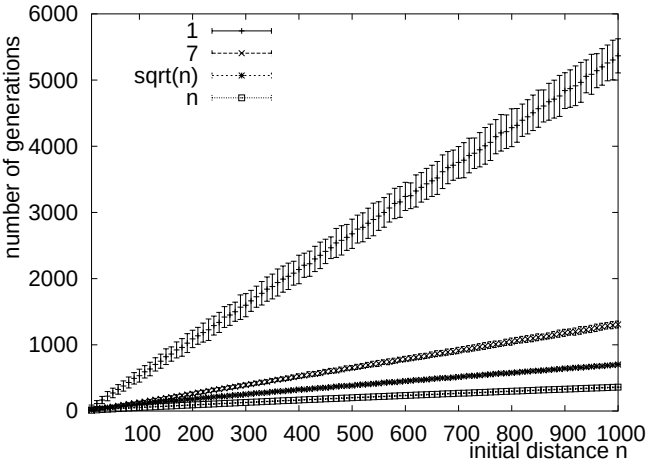

Figure 2: Number of generations for Algorithm 1 for $\lambda \in \{1, 7, \lceil\sqrt{n}\rceil, n\}$.

be superior. Of course, one needs $\lambda > 1$ in this case. Otherwise, the algorithm degenerates to a pure random search. It is important to note that such observations are typically made in the continuous domain. In discrete search spaces, for practical values of λ, plus- and comma-selection are almost identical. They can only differ if all offspring are inferior to their parent. In discrete search spaces with a sufficiently large λ, there is almost certainly at least one offspring identical to its parent. Then it does not matter which selection variant is used. We demonstrate this empirically by displaying the performance of the (1+λ) EA and (1, λ) EA for $\lambda = \lceil\sqrt{n}\rceil$ (in the left graph) and $\lambda = n$ (in the right graph) in Figure 3. Obviously, there is hardly any difference.

Finally, we discuss a very similar mutation operator that we conjecture to be more efficient in Section 6. We support this conjecture by comparing the performance of the (1+λ) EA as defined in Algorithm 1 with the same algorithm using the other mutation operator. We do so for $\lambda = \lceil\sqrt{n}\rceil$ (in the left graph) and $\lambda = n$ (in the right graph) in Figure 4.

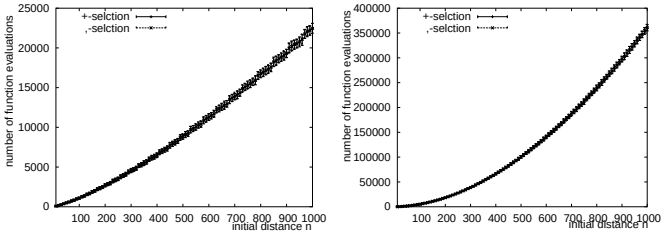

Figure 3: Number of function evaluations for $\lambda \in \{\lceil\sqrt{n}\rceil, n\}$ for Algorithm 1 and the variant with truncation selection.

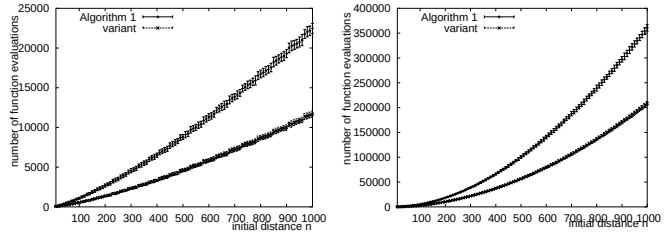

Figure 4: Number of function evaluations for $\lambda \in \{\lceil\sqrt{n}\rceil, n\}$ for Algorithm 1 and the variant with the mutation from Section 6.

6. CONCLUSIONS

We have presented a rigorous complexity analysis of a (1+λ) EA on a simple, OneMax-like dynamic problem in the lattice. We gave asymptotically tight upper and lower bounds on the expected first hitting time in the case that the target point does not move in Section 3. Starting with an initial distance of n, the first hitting time of the target is $\Theta(\lambda \cdot (1 + n \cdot (\log\log\lambda)/\log\lambda))$. Since in each generation λ function evaluations are made, this reveals that the average speed of the (1+λ) EA is $\Theta((\log\lambda)/\log\log\lambda)$. If in the dynamic case a target moves clearly slower than this, it will be located by the (1+λ) EA and tracked very steadily. We did not discuss other cases in such detail. In particular, the behavior of the (1+λ) EA when target point and algorithm have asymptotically the same speed is an interesting subject for future research.

We motivated the design of the mutation operator used by its similarities to standard bit mutation for bit strings. Considering this mutation operator independent of this motivation may make it appear somewhat silly. There is a number of k local operations that are chosen uniformly at random independently. Thus, some local operations may be in the exactly opposite direction of previous local operations in the very same mutation. In particular, due to its symmetry the expected distance traveled in one mutation equals 0. Thus, considering one mutation as step-wise process, one sees a random walk that has the tendency to return to its origin. This is not a very wise strategy if one wants to travel at high speed. It seems to be more sensible to replace the mutation operator in the (1+λ) EA by the following.

2. **Mutation**
   For $i := 1$ To λ Do
      Choose $k_1 \in \mathbb{N}$ according to a Poisson distribution with parameter 1.
      Choose $m_1 \in \{(1,0), (-1,0)\}$ uniformly at random

Choose $k_2 \in \mathbb{N}$ according to a Poisson distribution with parameter 1.
Choose $m_2 \in \{(0,1),(0,-1)\}$ uniformly at random
$y_i := x_t + k_1 \cdot m_1 + k_2 \cdot m_2$

This mutation operator decides on the number of steps in the directions of both axes in the lattice independently and chooses only one direction on each of the two axes. This way the distance traveled equals $k_1 + k_2$ where k_1 and k_2 are both chosen independently according to a Poisson distribution with parameter 1. It is interesting to note that this change in the mutation operator does not affect our results at all. The reason for this perhaps surprising observation can be found in the proof of Theorem 1 in Section 3. For the lower bound on $\mathrm{E}(T_{\lambda,0}(n))$ we estimated the distance traveled by the number of local operations and ignored the fact that some local operations may cancel each other. Thus, we only have to take into account that with the new mutation operator we double the number of local operations. This does not change any of our asymptotic results. For the upper bound, the reason is slightly more subtle. We estimate the probability to hit some target point y with a direct mutation using $\mathrm{Prob}(y_1 = y) \geq 1/(e \cdot d! \cdot 4^d)$. Considering the denominator, the important factors are $d!$ and 4^d. The factor $d!$ stems from the Poisson distribution and is not changed by the change of the mutation operator. The factor 4^d however comes from the choice of the local operations and can be changed to 4 for the new mutation operator. However, subsequent calculations reveal that the factor d^d dominates the order of growth of this probability so that changing 4^d to 4 does not imply any change of our asymptotic bound. Note, however, that in experiments the second mutation operator would appear to be clearly superior. However, as we have discussed this advantage is bounded above by some constant factor that disappears in O-notation.

An interesting area for future research is the investigation of different mutation operators for such tracking problems in the lattice. This is similar in spirit to the work of Weicker [11] who found biased and asymmetric mutations with a bias coinciding with the movement of the target point clearly superior. However, we considered a symmetric mutation operator, i. e., for all $x = (x_1, x_2)$, $y_a = (y_{a_1}, y_{a_2})$, and $y_b = (y_{b_1}, y_{b_2})$ with

$$\{|x_1 - y_{a_1}|, |x_2 - y_{a_2}|\} = \{|x_1 - y_{b_1}|, |x_2 - y_{b_2}|\}$$

$\mathrm{Prob}(m(x) = y_a) = \mathrm{Prob}(m(x) = y_b)$ holds. Note, however, that this kind of symmetry does not imply that a mutation operator is unbiased as defined by Droste and Wiesmann [6] in their design rules for metric-based EAs. They call a mutation operator m unbiased if for all $x, y_1, y_2 \in \mathbb{Z}^2$

$$d(x, y_1) = d(x, y_2) \Rightarrow \mathrm{Prob}(m(x) = y_1) = \mathrm{Prob}(m(x) = y_2)$$

holds. Obviously, unbiased operators are always symmetric.

Clearly, unbiased or at least symmetric operators are to prefer unless some problem-specific knowledge suggests otherwise. Such operators can be designed like the mutation operator defined in this section, possibly using different probability distributions for k_1 and k_2. We conjecture that with different probability distributions considerably higher speeds can be achieved without losing the good tracking behavior.

In some sense the class of tracking problems in the lattice considered here is quite general. We make no very specific assumption about the way the target point moves in the lattice. The only restriction we impose is the upper bound $d_{\max}$ on the speed of the target. Our results reveal that such an upper bound is necessary: if the target point may move arbitrarily far in one generation, nothing can be achieved. Our general setting leaves several more specific questions as subject of future research. It makes sense to consider the same problem class with more specific assumptions on the movement of the target point. Two cases appear to be of special interest. First, the target point may move by adding a fixed movement vector m with $d(m) = d_{\max}$ in each step, i. e., $a_t := a_1 + (t-1) \cdot m$. Together with a suitable initial position of a_1 this constitutes the worst case. Note that all our bounds hold in this situation, too. A different interesting scenario works with a target point that moves randomly according to some distribution that guarantees $d(a_t, a_{t+1}) \leq d_{\max}$ for all t. Depending on the distribution employed better bounds may be possible in this case.

In some sense, however, the class of tracking problems in the lattice considered here is very specific. We assume that the fitness value equals the distance to the current target point. This is equivalent to the situation for OneMax, where there is a direct correspondence between the (Hamming) distance to the optimum and the fitness value, too. Therefore, the results presented here are merely a first step into the direction of rigorous analyses of tracking problems.

7. REFERENCES

[1] H.-G. Beyer, H.-P. Schwefel, and I. Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287:101–130, 2002.

[2] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[4] S. Droste. Analysis of the (1+1) EA for a dynamically changing onemax-variant. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 55–60, 2002.

[5] S. Droste. Analysis of the (1+1) EA for a dynamically bitwise changing onemax. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 909–921, 2003.

[6] S. Droste and D. Wiesmann. On the design of problem-specific evolutionary algorithms. In *Advances in Evolutionary Computing*, pages 153–173. Springer, 2003.

[7] W. Feller. *An Introduction to Probability Theory and Its Applications. Volume I.* Wiley, 1968.

[8] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[9] T. Jansen and K. De Jong. An analysis of the role of offspring population size in EAs. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 238–246, 2002.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[11] K. Weicker. Analysis of local operators applied to discrete tracking problems. *Softcomputing Journal*, 2005. To appear.