An Analysis of the Behavior of Simplified Evolutionary Algorithms on Trap Functions

Siegfried Nijssen and Thomas Bäck

***Abstract*—Methods are developed to numerically analyze an evolutionary algorithm (EA) that applies mutation and selection on a bit-string representation to find the optimum for a bimodal unitation function called a trap function. This research bridges part of the gap between the existing convergence velocity analysis of strictly unimodal functions and global convergence results assuming the limit of infinite time. As a main result of this analysis, a new so-called $(1:\lambda)$-EA is proposed, which generates offspring using individual mutation rates p_i. While a more traditional EA using only one mutation rate is not able to find the global optimum of the trap function within an acceptable (nonexponential) time, our numerical investigations provide evidence that the new algorithm overcomes these limitations. The analysis tools used for the analysis, based on absorbing Markov chains and the calculation of transition probabilities, are demonstrated to provide an intuitive and useful method for investigating the capabilities of EAs to bridge the gap between a local and a global optimum in bimodal search spaces.**

***Index Terms*—Absorption time, convergence velocity, evolutionary algorithm, genetic algorithm, mutation, trap functions.**

I. Introduction

In the past decade, theoretical research on evolutionary algorithms (EAs) has received significant attention, driven by the insight that their theoretical basis needs to be improved to facilitate most effective usage of these algorithms. Meanwhile, considerable progress has been made especially with respect to the analysis of *convergence velocity* and *convergence reliability* of EAs.

Convergence velocity analysis is a general approach, derived originally [15], [18] and refined subsequently for the analysis of evolution strategies (see [6] for a full picture; [1] for an overview). It has been transferred to EAs with bit-string genotypes in the early 1990s [2], [14].

Convergence velocity is a local measure of progress of the algorithm from one iteration to the next, where progress is defined either in terms of the expected quality gain φ (i.e., objective function value improvement) or in terms of the expected change of distance D to a global optimum:

$$\varphi = E\left[f(\vec{x}_{t+1}) - f(\vec{x}_t)\right], \quad D = E\left[\|\vec{x}^* - \vec{x}_t\| - \|\vec{x}^* - \vec{x}_{t+1}\|\right].$$

Here, $\vec{x}^*$ denotes a global optimum point of the objective function $f\colon M \to \mathbb{R}$, which is defined over a certain domain M. Vector $\vec{x}_t$ denotes the best (or a representative) individual of the population at generation t. A maximization task is assumed in this paper.

Typically, this type of analysis is used to characterize the behavior of EAs for unimodal problems, i.e., their effectiveness as local optimizers. This type of analysis is useful to understand performance relative to other local optimization algorithms, to gain insight into the impact of parameter settings of the algorithm on convergence velocity, and to characterize the final stage of global optimization, when the algorithm ultimately converges to a solution. However, convergence velocity analysis has not yet been generalized to cover the multimodal case of objective functions with more than one optimum, which of course is the interesting case for practical applications of EAs.

At the other extreme, convergence reliability analysis deals with the question of global convergence of an EA, meaning that the algorithm is guaranteed to find a global optimum. Global convergence results are general in the sense that they do not make strong assumptions about the objective function and typically assume unlimited time

$$\lim_{t\to\infty} p\left(\vec{x}^* \in P(t)\right) = 1.$$

Here, $P(t)$ denotes the population maintained by the EA at generation t and $p(A)$ is the probability of the event A. Some of the first global convergence results for EAs were presented for simple (1+1)-evolution strategies [15] and were subsequently refined for population-based strategies, as well as nonelitist strategies [17]. Concerning genetic algorithms, first proofs of global convergence were presented again in the early 1990s [9]. The global convergence type of analysis benefits from the generality of results (i.e., for all possible objective functions), but it is practically not exploitable as no finite expected time results are obtained.

In order to bridge the gap between convergence velocity results and convergence reliability results, it is a natural but difficult step to extend the convergence velocity analysis to multimodal objective functions and to analyze explicitly the time it takes the algorithm to converge to the global optimum rather than a local one. Of course, the results are expected to depend on the starting conditions as well as the specific parameter settings of the EA.

The natural extension from the existing work for unimodal objective functions consists in bimodal problems, where just one local and a distinct global optimum exist in the search space and the regions of attraction of these two optima can be scaled such that it becomes harder or easier to find the global optimum. For real-valued objective functions, this test case was defined and experimentally investigated already more than 15 years ago

Manuscript received August 22, 2001; revised July 9, 2002.

S. Nijssen is with the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, NL-2333 CA Leiden, The Netherlands (e-mail: snijssen@liacs.nl).

T. Bäck is with the LIACS and NuTech Solutions GmbH, D-44227, Dortmund, Germany (e-mail: baeck@nutechsolutions.de).

Digital Object Identifier 10.1109/TEVC.2002.806169

[10], [11] demonstrating the importance of "soft selection" to bridge the gap between the local and global optimum. From a theoretical point of view, however, the first results on specific bimodal test problems have only been published very recently [12], [16]. Approaching the analysis from different perspectives, both papers focus on the advantage of crossover for reducing the time to find the global optimum or to bridge the gap between the local and global optimum.

Here, we explore another piece of the puzzle by analyzing so-called trap functions, which have been designed as scalable bimodal functions to challenge EAs. In contrast to the above-mentioned studies, the analysis concentrates on simplified EAs using only mutation and selection, such as the (1+1)-EA, the (1, λ)-EA, and the (1+λ)-EA. This analysis continues earlier work on a unimodal problem [2], [14], [5] and concludes with the development of a new version of an EA, the $(1:\lambda)$-EA, which generates each offspring with a different mutation rate. The resulting algorithm reduces the time to find the global optimum drastically by increasing the emphasis on exploration, such that the region of attraction of the local optimum can be left at any stage during the search.

In Section II, the general tools for the theoretical analysis are introduced, the trap function is formalized, and the $(1:\lambda)$-EA is defined as a generalization of the (1+λ)- and (1, λ)-EA. Section III presents the numerical evaluation of theoretical results and a comparison to the experimentally observed behavior of the EAs on the trap function. Our conclusions and an outline of further work are given in Section IV.

II. EAs on Trap Functions

A. Prerequisites

Each individual in our EA is represented by a bitstring of length l: $\vec{x} \in \mathbb{B}^l$. The fitness function is a function f: $\mathbb{B}^l \rightarrow \mathbb{R}$ that maps the bitstring to a real number. We restrict ourselves to *unitation functions*, which are functions that depend entirely upon the number of ones in a bitstring and thus not on their position

$$f(\vec{x}) = \hat{f}(u(\vec{x})) = \hat{f}\left(\sum_{k=1}^{l} x_k\right). \tag{1}$$

For any unitation function $\hat{f}$ with a domain $D = \{0, 1, \ldots, l\}$, three subsets can be computed for a given value $u \in D$:

$$I^{+}(u) = \left\{u' \in D \,\middle|\, \hat{f}(u') > \hat{f}(u)\right\} \tag{2}$$

$$I^{=}(u) = \left\{u' \in D \,\middle|\, \hat{f}(u') = \hat{f}(u)\right\} \tag{3}$$

$$I^{-}(u) = \left\{u' \in D \,\middle|\, \hat{f}(u') < \hat{f}(u)\right\}. \tag{4}$$

For every unitation value (the number of ones in a bitstring) there is a set of unitation values (and corresponding bitstrings) that have lower, equal, and higher fitnesses. We do not mention the fitness function in our notation as this function is implicitly the same in all formulas.

In an EA, one bitstring can be transformed into another bitstring using an operator called *mutation*. With probability p_m, this operator flips each of the l bits into its complementary bit. By flipping bits, the unitation value of a bitstring may be changed. The following expression describes the probability that a bitstring of unitation value u_1 is converted into a string of value u_2, when u_2 is higher than u_1 (cf. [5])

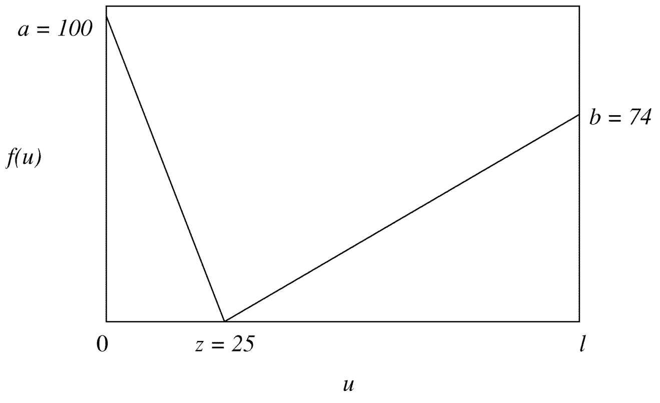


Fig. 1. A basic trap function $\hat{f}(u)$.

$$p_m^{(u_2>u_1)}(u_2|u_1) = \sum_{k=0}^{\min(u_1,\, l-u_2)} \binom{u_1}{k}\binom{l-u_1}{k+u_2-u_1} \times p_m^{u_2-u_1+2k}(1-p_m)^{l-(u_2-u_1+2k)}. \tag{5}$$

From these expressions, $p(u_2|u_1)$ can be calculated for general values

$$p_m(u_2|u_1) = \begin{cases} p_m^{(u_2>u_1)}(u_2|u_1), & \text{if } u_2 > u_1; \\ p_m^{(u_2>u_1)}(l-u_2|l-u_1), & \text{otherwise.} \end{cases} \tag{6}$$

In the second equation, the observation is used that the probability of decreasing the number of one-bits from u_1 to u_2 corresponds to the probability of increasing the number of zero-bits from $l-u_1$ to $l-u_2$. This is due to the equal probability for one-to-zero and zero-to-one bit flips.

These formulas allow us, in principle, to use any unitation function as fitness function, instead of only a basic counting-ones function. In the following, we illustrate this by applying the analysis to so-called trap functions.

B. Trap Functions

We will use two trap functions in our experiments. One is a basic trap function as introduced in [7]. The other is a more complex function that we will use to check the validity of the results obtained for the basic function.

A Basic Trap Function: The definition of a basic trap function is [7]

$$\hat{f}(u) = \begin{cases} \dfrac{a}{z}(z-u), & \text{if } u \le z \\ \dfrac{b}{l-z}(u-z), & \text{otherwise.} \end{cases} \tag{7}$$

Fig. 1 clarifies the meaning of the parameters.

For analyzing our numerical results, it appears useful to compute the $I(u)$ sets of the trap function explicitly. Here, we will show how this can be done using *corresponding points*. A corresponding point of a unitation value u is a different unitation value which has the same $\hat{f}(u)$ value. For a basic trap function, there is at most one such point at the other side of the trap. The corresponding point may be computed using

$$\frac{a}{z}(z-u_1)=\frac{b}{l-z}(u_2-z) \tag{8}$$

which yields the following expressions:

$$u_1=t_1(u_2)=z+\frac{b}{a}\frac{z}{l-z}(z-u_2) \tag{9}$$

$$u_2=t_2(u_1)=z+\frac{a}{b}\frac{l-z}{z}(z-u_1). \tag{10}$$

For a given value u and the resulting corresponding points $u_1(u)$ and $u_2(u)$, the subsets become

$$I^+(u)=\{u'\in D|u'<u_1(u)\vee u'>u_2(u)\} \tag{11}$$

$$I^-(u)=\{u'\in D|u'>u_1(u)\wedge u'<u_2(u)\} \tag{12}$$

$$I^=(u)=D\backslash(I^+(u)\cup I^-(u)). \tag{13}$$

There are many possible choices for the parameters a, b, and z. Given the number of bits l, we stick to the following guidelines:

- $z\approx(1/4)l$;
- $b=l-z-1$;
- $1.5b\le a\le 2b$; a a multiple of z.

For l and a, we prefer values which are a multiple of ten. These values simplify some of the computations without affecting their generality.

A Complex Trap Function: The basic trap function is characterized by two optima that are bit-wise complements of each other. In general, it is difficult for EAs to go from the suboptimal solution to the global optimum in this case. For the trap function, however, a *bit-flip* operator—an operator which reverses all bits—is likely to solve the problem easily. To check the validity of our results, we will also investigate a slightly more complex trap function for which bit-flipping is not always a good solution

$$\hat{f}(u)=\begin{cases}\frac{a}{z}(z_1-u), & \text{if } u\le z_1\\ \frac{b}{l-z_1}(u-z_1), & \text{if } z_1<u\le z_2\\ \frac{b(z_2-z_1)}{l-z_1}\left(1-\frac{1}{l-z_2}(u-z_2)\right), & \text{otherwise.}\end{cases} \tag{14}$$

This trap function is illustrated in Fig. 2.

Note that for $z_2=l$, the complex trap function reduces to the basic trap function. As parameter for z_2, $z_2=l-z_1$ has our interest, as one could expect that bit-flipping performs very badly in that case.

C. A (1 : λ)-EA

We use the mutation operator to obtain the following search algorithm.

```
$u$ := uniformly chosen bitstring
repeat until maximum generation reached
  $S$ := $\{u_1,\dots,u_\lambda\}$; $u_i$ is a mutated child of $u$,
         generated with mutation rate $p_i$.
  $u$ := individual with the highest fitness in
         $S$.
```

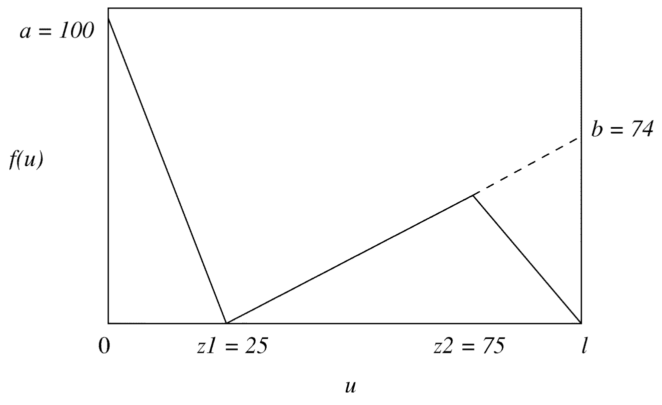


Fig. 2. A complex trap function $\hat{f}(u)$.

This algorithm is a very simplified version of an EA. We use this algorithm to introduce an extension of an ordinary EA: different fixed mutation rates p_i are used to generate children. It is easily shown that this algorithm can be applied to simulate both ordinary (1, λ)- and (1+λ)-EAs:

- (1, λ): $p_1=p_2=\cdots=p_\lambda$;
- (1+λ): use $\lambda'=\lambda+1$, with $p_{\lambda+1}=0$ and $p_1=p_2=\cdots=p_\lambda$. One of the children thus has a mutation rate of zero, which means that the parent is copied.

With $p(u_1\to u_2)$, we will denote the probability that in one iteration of the algorithm the current unitation value of the selected parent changes from u_1 into u_2. To obtain an expression for this probability, we use the following intermediate probabilities:

$$p_m(u_2^-|u_1)=\sum_{k\in I^-(u_1)}p_m(k|u_1) \tag{15}$$

$$p_m(u_2^=|u_1)=\sum_{k\in I^=(u_1)}p_m(k|u_1). \tag{16}$$

These are combined as follows:

$$p(u_1\to u_2)=\prod_{i=0}^{\lambda}\left(p_i(u_2^-|u_1)+p_i(u_2^=|u_1)\right)-\prod_{i=0}^{\lambda}p_i(u_2^-|u_1). \tag{17}$$

The first term is the probability that all offspring are worse than or equal to u_2. From this, the probability is subtracted that all offspring are worse. The resulting probability is the probability that at least one of the offspring has unitation value u_2. The i index goes through all offspring and their corresponding mutation rate. It is here that the "multiple mutation rate" principle is applied.

We will refer to the unitation value of the individual u as the current state of the (1 : λ)-EA. The states can be ordered according to their $\hat{f}(u)$ values. A higher state is a state with a higher $\hat{f}(u)$ value. With $I^+(u)$, we denote the set of all higher states.

D. Measures

The state-transition probabilities according to (17) can be used to compute several quality measures of the evolutionary process. The following measures are short-term performance measures.

Improvement probability for state s

$$p^{+}(s) = \sum_{e \in I^{+}(s)} p(s \rightarrow e).$$

This is the probability of mutating to a better individual and gives a clue how likely it is to make an immediate enhancement when one has an individual with a certain unitation value.

Convergence velocity for state s

$$\varphi(s) = \sum_{e \in I^{+}(s)} (f(e) - f(s))p(s \rightarrow e).$$

This is the enhancement one expects to obtain in one generation, given a certain individual. It is the weighted sum of all possible enhancements by the probability that such an enhancement occurs.

Trap jump probability for state s

$$p'(s) = \sum_{e \in I^{+}(s) \cap C(s)} p(s \rightarrow e).$$

Here, C depends on s as follows: $C(s) = \{u_2(s), \ldots, l\}$ if $s \leq z$; $C(s) = \{0, \ldots, u_1(s)\}$ otherwise. This measure can only be used with the basic trap function; intuitively, it is the probability of going to a better individual at the other side of the trap. Together with the improvement probability, this probability provides an insight into the source of a likely enhancement. A high jump probability indicates that we can easily leave a local path.

Oscillating probability for state s

$$p^{o}(s) = \sum_{e \in I^{+}(s) \cap C(s)} p(s \rightarrow e)p'(e)/p'(s).$$

Intuitively, this is the probability of jumping back to the same side of the trap after two generations, given that we jumped to the other side of the trap in the first generation. This probability gives a better insight into the usefulness of a high trap-jump-probability. A mutation rate that makes trap jumping easy may or may not make it easy to jump back. In the first case, the algorithm may be walking the two sides of the trap in turns, jumping from one side to the other each generation, while in the second case one mutation rate is expected to allow one jump only.

To determine long-term performance measures, several generations have to follow each other. For this purpose the transition probabilities are stored in a transition matrix P. The states are ordered such that for a plus strategy, the matrix is uppertriangular; furthermore, the states that contain the optimum (the so-called set of absorbing states S_a) have the highest indexes. With Q, we denote the submatrix that does not contain absorbing states [8]. This allows to define several measures.

Absorption time for state s

$$T(s) = \sum_{s'} N_{ss'}$$

where $N = (I - Q)^{-1}$; using Markov chain analysis, it can be shown that this formula computes the expected number of generations to reach the global optimum from a certain individual.

TABLE I
PARAMETERS OF THE EMPLOYED TRAP FUNCTIONS

l	z	a	b
10	3	12	6
20	5	20	14
50	10	80	39
75	20	80	54
100	25	100	74

If this number is very high, it is almost impossible for the algorithm to find the optimum.

Number of evaluations until absorption for state s

$$N(s) = T(s) \times \lambda.$$

The previous measure is only reasonable when one compares algorithms that perform the same amount of work every generation. Of course, this is not always the case. Especially on computer architectures that evaluate offspring sequentially, it is much fairer to take into account the number of offspring in order to compare the expected computation time.

Expected absorption time

$$E(T) = \sum_{s} p(s)T(s).$$

Here, $p(s) = \binom{l}{s}/2^l$; the previous measures give the expected computation time when one knows the starting individual. The EA, however, determines its starting individual randomly using a uniform distribution. To determine the performance of the complete algorithm, the expected absorption time has to be determined, thus averaging over all possible starting individuals. With $p(s)$ the probability of a uniformly chosen bitstring with unitation value s is computed.

Expected number of evaluations until absorption

$$E(N) = \sum_{s} p(s)N(s) = E(T) \times \lambda.$$

Using a similar argument as for $N(s)$, it is fairer to take into account the number of offspring.

In the sequel, we will mainly use the metrics that depend upon the number of evaluations. We believe this best reflects the computational effort. To compare algorithm setups, we use the expected number $E(N)$, which averages over the possible starting individuals. To show the influence of the starting individual, we will also report separate experiments on this.

In previous publications [2]–[5], [14], the $p^{+}(s)$ and $\varphi(s)$ measures were used. We will also give results for these measures here. However, not all of the results can be explained intuitively. We will use the jump probability and oscillating probability to provide argumented explanations.

III. NUMERICAL RESULTS AND EXPERIMENTS

We arrange our experiments as follows. First, we analyze the short-term measures of a basic (1+1)-EA. This allows for an easy comparison of our results with the results of earlier publications. Next, we will exploit our observations on these basic cases to analyze the more complex algorithms introduced in this article.

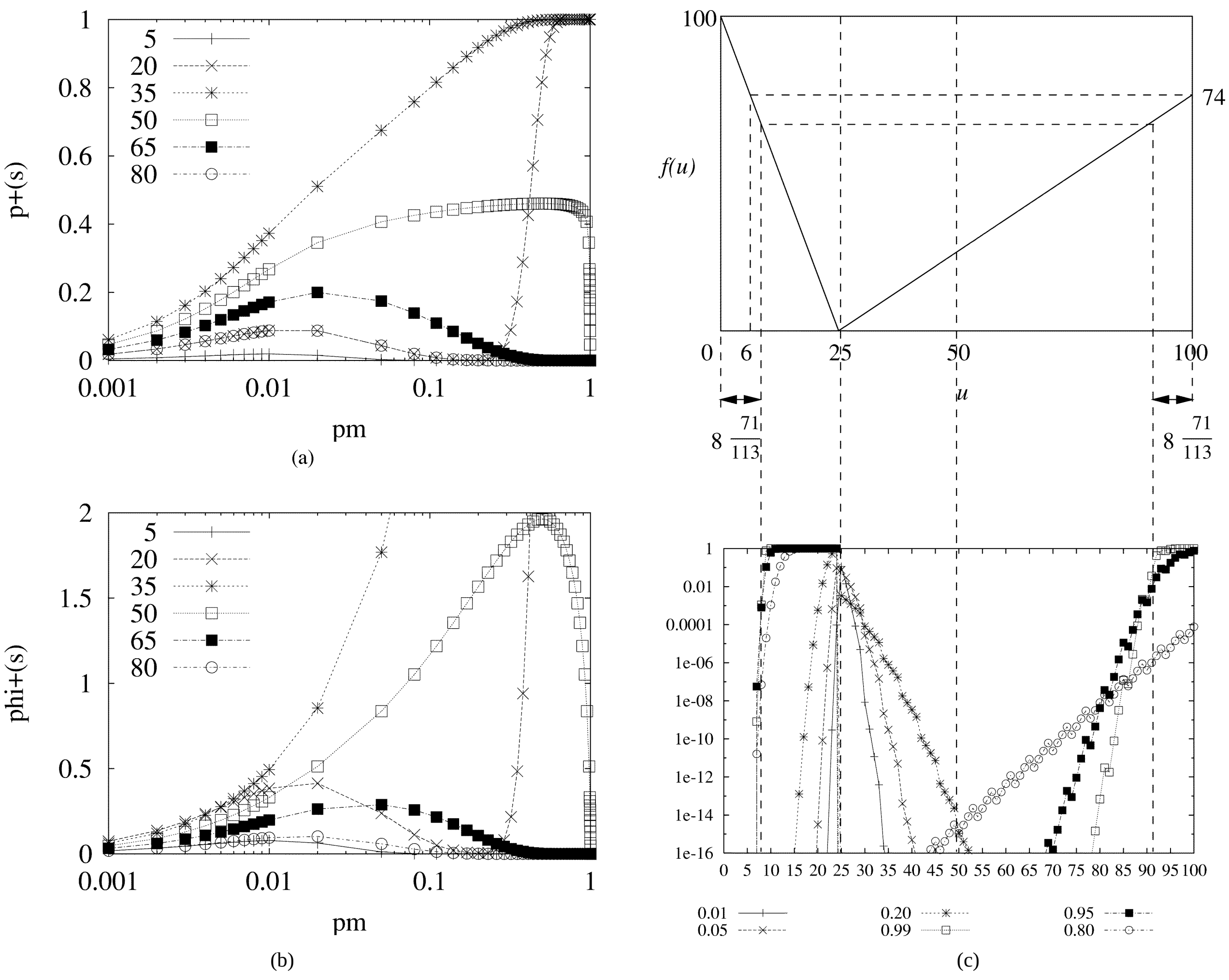


Fig. 3. Probabilistic behavior of the (1+1)-EA for the $l = 100$ trap function. In (a) and (b), which show the improvement probability and the convergence velocity, respectively, the value of p_m is varied horizontally. The lines correspond to $u(\vec{x}) \in \{5, 20, 35, 50, 65, 80\}$. In (c), which shows the trap function (top graph) and the jump probability (bottom graph), the unitation value is displayed horizontally. For several mutation rates, the probability is shown that the next individual is better and on the other side of the trap (which is at $z = 25$).

A. Short-Term Analysis of the Basic Trap Function

Several trap functions are used in our experiments. A summary of all functions can be found in Table I. The parameters are chosen carefully such that $I^{=}(u) = \{u\}$ for all u. This choice of parameters will ease our analysis without affecting their generality. In the sequel, we refer to one of these functions by giving the parameter l.

Fig. 3 displays the improvement probabilities $p^{+}(s)$ and convergence velocities $\varphi(s)$ of a (1+1)-EA in several situations. One coordinate, showing the mutation rate p_m, is drawn in a logarithmic scale to show more details for lower mutation rates. Furthermore, the trap jump probability is shown.

The graphs can be explained by looking at some of the characteristics of the trap function that was used. First, we take a look at the $p_m = 1$ situation, which corresponds to turning a unitation value u into $l - u$. An analysis of the function yields [see upper Fig. 3(c)] that for $u(\vec{x}) \leq 8$ and $50 \leq u(\vec{x}) \leq 91$, such a flip does not result in an improved fitness value. This is reflected in Fig. 3(a): at $p_m = 1$, only the lines for $u(\vec{x}) = 20$ and $u(\vec{x}) = 35$ are 1. In Fig. 3(c) (below), the lines with high mutation rates are 1 in exactly the regions we indicated. The figure also shows that a mutation rate significantly lower than 1 is better in the range $50 \leq u(\vec{x}) \leq 91$ in order to jump to the other side of the trap.

If Fig. 3(a) had been shown on a linear scale, it would appear that the $u(\vec{x}) = 50$ curve is symmetric. This can be explained by considering this construction: a mutated string with mutation rate p_m can also be obtained by first flipping all the bits and by then mutating all bits with mutation rate $1-p_m$. As a bit-flipped string with $u(\vec{x}) = 50$ results in a string with $u(\vec{x}) = 50$, the result of mutating such a string with p_m must be the same as by mutating with $1-p_m$. The same argument also explains that the curves for $p_m = 0.2$ and $p_m = 0.8$ in Fig. 3(c) cross each other in $u(\vec{x}) = 50$.

In Fig. 3(c), the lines for high mutation rates show nonmonotonic behavior when $u(\vec{x}) > 50$: every five unitation values, the trap jump probability decreases slightly before increasing again. This can be explained by looking at corresponding values. In this particular trap function, it appears that four unitation values at the right-hand side of the trap share the same corresponding value at the left-hand side. The highest of these four values has the highest jump probability: in that case, more bits are flipped, which is more likely to occur (remember that we are looking at mutation rates that are more likely to flip bits than to maintain). After four unitation values, the corresponding value gets lower, which reduces the number of unitation values on the other side of the trap that cause an enhancement. This consequently reduces the probability of jumping.

For $u(\vec{x}) = 20$, the curve in Fig. 3(a) has two optima (at approximately 0.01 and approximately 1.0). We saw that the second optimum corresponds to flipping (almost) all bits, resulting in a jump from one side of the trap to the other (better) side. The first local optimum, therefore, corresponds to the mutation rate that maximizes the local improvement probability (which leads the genetic algorithm toward the local optimum $\vec{x} = 100$). Most strikingly, the curves for $u(\vec{x}) = 80$ and $u(\vec{x}) = 20$ (with equal distances to a nearby local optimum) overlap each other for low mutation rates and share their local maximum at $p_m \approx 0.02$.

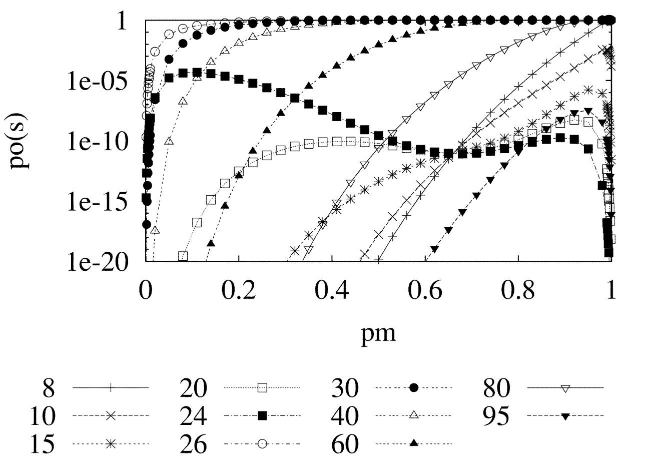
From the graph it can be deduced that the more closely the unitation value approaches a local optimum, the lower the optimal mutation rate for local search becomes, until it reaches $p_m = 0.01$ for $u(\vec{x})$. This is in harmony with the mutation rates derived in [3] $(1/(2(u(\vec{x})+1)-l))$ and [14] $(1/l)$. The intuition of these schedules is as follows: when the local optimum is almost reached, it is most safe to flip one bit in each mutation. If the mutation rate must be constant throughout the whole process, $1/l$ is the best choice, as most of the time is usually spent in fine-tuning the solution.

Graphs similar to those in Fig. 3 for the other values of l also display the mentioned phenomena. For example, also for the other trap functions $1/l$ seems a reasonable mutation rate in order to optimize local search.

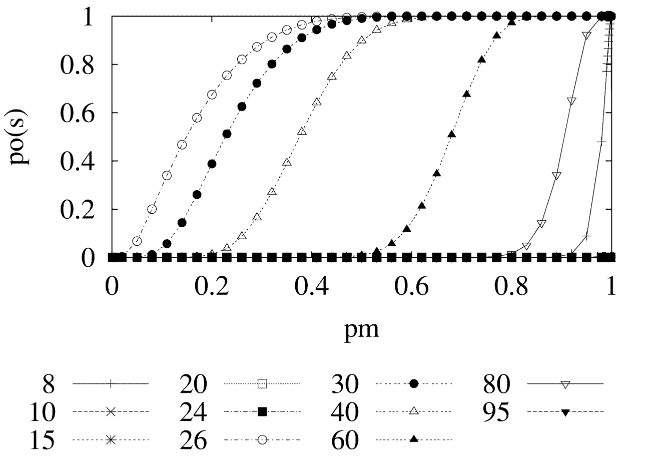
Fig. 3(b) is very similar to Fig. 3(a). As expected, the curves for $u(\vec{x}) = 20$ and $u(\vec{x}) = 80$ do not overlap: as the slope of the curve is much higher at the left-hand side of the trap, the convergence velocity is also higher there.

Some interesting conclusions can be drawn from Fig. 4, which shows the oscillating probability for several unitation values. We will give an analysis for each value.

1) $u(\vec{x}) \leq 6$: No better individual can be found on the other side of the trap. No oscillating will appear for any mutation rate.
2) $6 < u(\vec{x}) \leq 8$: There are better individuals at the other side of the trap. However, applying $p_m = 1$ will not give a better individual at the other side. If a jump to a better individual is made for some high $p_m \neq 1$ (which is very unlikely), then it is very likely that a jump back is made in the next generation (consider $p_m = 1$ for this case). This is reflected in Fig. 4(b).
3) $8 < u(\vec{x}) < 25$: A high mutation rate (for example $p_m = 1$) will most likely result in a jump to the other side of the trap. It is unlikely that a jump will be made back again (reconsider $p_m = 1$), so the oscillating probability is low in any case [Fig. 4(b)]. However, for individuals close to the trap, oscillating is still possible for low mutation rates [Fig. 4(a)]: if a jump to the right-hand side of the trap is made, the new individual will most likely still be close to 25, such that a small mutation could result in a jump back. If the mutation rate gets higher, the first offspring will be further away from 25; an offspring somewhere in the middle of the right-hand side hill is most likely. A moderately high mutation here will result in a new offspring at the right-hand side of the trap, which explains the dip in the graph of $u(\vec{x}) = 24$. If the mutation rate gets even higher, far jumps become more likely again.
4) $25 \leq u(\vec{x}) \leq 92$: Similar arguments can be used as for case 2. It appears here that for individuals close to the trap a low mutation rate suffices to obtain oscillating behavior.
5) $92 < u(\vec{x}) \leq 100$: Similar arguments can be used as for case 3.



(a)

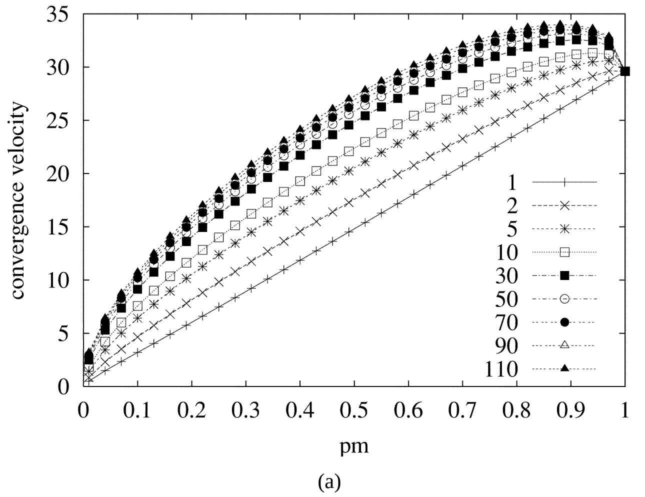


(b)

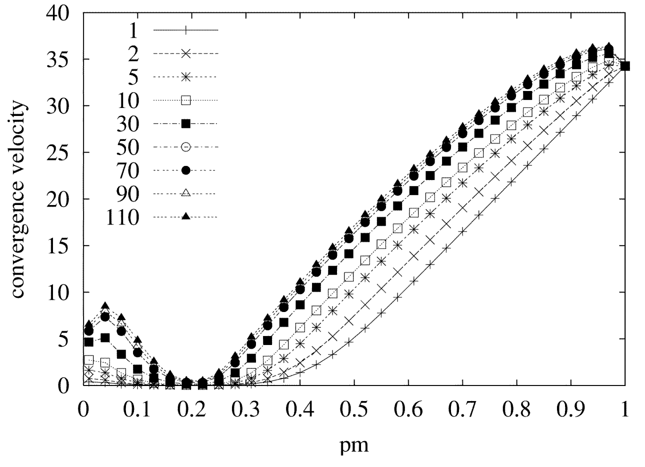
Fig. 4. Oscillating probability for values of $u(\vec{x})$. Mutation rate p_m is shown on the ordinate axis. A (1+1)-EA is applied to the $l = 100$ trap function. (a) Logarithmic scale. (b) Linear scale.

In Fig. 5, the effect of using multiple offspring is visualized. The convergence velocity increases in all cases. The impact of using additional offspring however decreases—approximately—exponentially for every new offspring. This is in agreement with findings for (1, λ)-evolution strategies that convergence velocity is $\varphi \sim \sqrt{2 \ln \lambda}$ and thus grows only logarithmically with λ [6].

Fig. 5(b) shows two local maxima for all λ values. This is explained by the two means of improvement: staying on the same side of the trap (for low probabilities) or going to the other side of $u(\vec{x}) = 50$ (for high probabilities). Indeed, Fig. 5(a) has only one local maximum, as both large and small mutation rates will



(a)



(b)

Fig. 5. Convergence velocity of a (1+λ)-EA on the trap function for values of $p_m, \lambda \in \{1, 2, 5, 10, 30, 50, 70, 90, 110\}$. (a) After the trap: $u(\vec{x}) = 35$. (b) Before the trap: $u(\vec{x}) = 20$.

result in offspring at the right-hand side of the trap with most probability.

Keeping in mind our previous discussion, it is instructive in both graphs to see that the convergence velocity decreases for very high mutation rates when $\lambda > 1$. This may be caused by the lack of variety in the pool of offspring for very high mutation rates. With $p_m = 0.5$, the information in an individual is not exploited and the variety is maximal; with $p_m = 1.0$, there is only one possible offspring; the variety is clearly very small in that case. The figure provides a good indication that the optimum is between 0.5 and 1.0, but more close to $p_m = 1.0$. This is an argument for using mutation rates little below 1.0.

B. Long-Term Analysis of the Basic Trap Function

When considering the long-term performance of EAs, the absorption time measure is important. Fig. 6 shows absorption times for a basic (1+1)-EA. As can be seen here, the starting individual (represented by the $u(x)$ axis) has no major influence on the absorption time. One could characterize one optimal mutation rate $p_m^* \approx 0.43$ for the $l = 10$ function. Experiments with other functions showed similar behavior. Table II summarizes the results by only giving the optimal mutation rate and the expected number of generations for that mutation rate. No

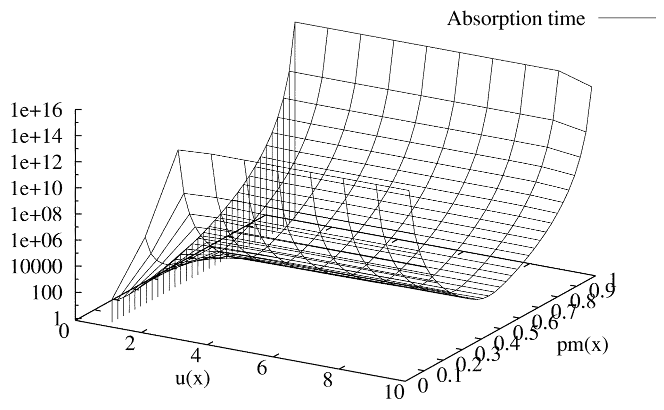


Fig. 6. Absorption times for a (1+1)-EA on a $l = 10$ trap function.

results for $l = 75$ and $l = 100$ are provided, as our computation appeared not to be sufficiently precise with very large numbers.[1]

The number of values of l are insufficient to draw conclusions with respect to the relation between l and the values of p_m^* and $E^*(T)$. The expected absorption time seems to rise exponentially—also taking into account that $E^*(T)$ is very large for $l = 75$ and $l = 100$—but we were not able to derive a straightforward formula. The optimal mutation rate fluctuates somewhere between 0.4 and 0.5. This fluctuation is probably caused by small differences between the trap functions: for $l = 10$, for example, 30% of the unitation values is at the left-hand side of the trap, whereas for $l = 50$, this is 20%.

What is more interesting is the relation between λ and the values of p_m^* and $E^*(T)$. From Table II, it is clear that the optimal mutation rate is independent of the number of offspring. Furthermore, for a known value $E_1^*(T)$ for $\lambda = 1$, values for other numbers of offspring can very well be approximated using $E_\lambda^*(T) = E_1^*(T)/\lambda$. As the expected number of evaluations is defined as $E(N) = E(T) \times \lambda$, we can conclude that adding offspring does not have any influence on the long-term performance when the mutation distribution is kept constant. We will therefore investigate the $(1 : \lambda)$-EA next.

For the λ offspring, we will test the following mutation rates.

- Linear: $p_i = (b_u - b_l)((i-1)/(\lambda - 1)) + b_l$. We will abbreviate this with "$l\ b_l\ b_u$."
- Exponential: $p_i = b_u \rho^{i-1}$, where $\rho = (b_l/b_u)^{1/(\lambda-1)}$. This will be abbreviated with "$e\ b_l\ b_u$."

In all cases, we assume that there is one offspring with mutation rate $p_m = 0$, such that the $(1 : \lambda)$-EA becomes an adapted (1+λ')-EA, with $\lambda' = \lambda - 1$. In the sequel, we will denote this $(1 : \lambda)$ EA with one mutation rate $p_m = 0$ as a (1$\div\lambda'$)-EA (Table III).

We first check the correctness of the formulas we developed. In Fig. 7(a) and (b), experimental expected absorption times and theoretical times are plotted for a (1$\div$10)-EA with $l = 20$. Only those ranges of b_u and b_l are shown, which result in low times. The theory predicts the experimental results accurately and shows the same dependence on the parameters. Fig. 7(d) displays a comparison of numbers. Here, the starting individual is fixed, such that uniform initialization is not taken into account. The experimentally determined numbers (which were av-

[1]A mutation rate of 0.01 would yield an approximate absorption time of 8.39 $\times$ 10^{178} generations.

TABLE II
EXPECTED ABSORPTION TIMES OF OPTIMAL MUTATION RATES FOR A $(1+\lambda)$-EA

	$\lambda=1$		$\lambda=2$		$\lambda=4$		$\lambda=8$	
l	p_m^*	$E^*(T)$	p_m^*	$E^*(T)$	p_m^*	$E^*(T)$	p_m^*	$E^*(T)$
10	0.43	571.181	0.43	285.486	0.43	142.596	0.43	71.1608
20	0.46	446697	0.46	223345	0.46	111669	0.43	55830.3
50	0.42	1.42763×10^{12}	0.42	7.13815×10^{11}	0.42	3.56908×10^{11}	0.43	1.78454×10^{11}

TABLE III
NOTATION OF EVOLUTIONARY ALGORITHMS

Algorithm	parent as offspring	equal mutation rates
$(1,\lambda)$	no	yes
$(1+\lambda)$	yes	yes
$(1:\lambda)$	no	no
$(1\div\lambda)$	yes	no

eraged over 1000 experiments) clearly converge to their theoretical values. The theory, therefore, seems reliable.

An important observation in Fig. 7(d) is that there are no local optima other than the global optimum. We checked several other setups and found this to be true for all test cases. This would mean that any gradient descent algorithm could find the optimal parameters for b_u and b_l. Under the assumption that there is one optimum, we used the following—simple—algorithm to determine approximate optimal values for b_u and b_l as follows:

```
$\epsilon := 1.0/\kappa$, $b_l = 0.05$, $b_u = 0.6$
  **repeat** a fixed number of iterations
    **while** $b_l$ and $b_u$ changed during the last
      assignment
      compute $E(T)$ for $(b_l - \epsilon, b_u - \epsilon)$, $(b_l, b_u - \epsilon)$,
        $(b_l, b_u + \epsilon)$, $(b_l, b_u - \epsilon)$, $(b_l, b_u)$, $(b_l, b_u + \epsilon)$,
        $(b_l + \epsilon, b_u - \epsilon)$, $(b_l + \epsilon, b_u)$, $(b_l + \epsilon, b_u + \epsilon)$
      set $b_l$ and $b_u$ to the best of these values
    $\epsilon := \epsilon/\kappa$
```

Parameter κ defines the coarseness of the search. We use $\kappa = 10$.

We use this algorithm in Fig. 8 to investigate the influence of the λ parameter. The optimal values for b_l and b_u for both linear and exponential mutation rate schemes are shown, as are the number of evaluations for these parameter settingsn.

When $\lambda = 2$, the algorithms reduce to EAs with two mutation rates b_l and b_u. At first sight, the $1/l$ mutation rate guideline is also applicable here: both in linear and exponential schemes, a mutation rate of approximately $1/l$ should be present to obtain the best absorption time. When the number of offspring increases, the optimal lower bound decreases. This is reasonable: it means that it is more advantageous to add some low mutation rates than to add more high mutation rates. When considering the upper bound, it is interesting to recall Figs. 4 and 5. Also, there is a strong relation here between mutation rates that maximize convergence velocity and minimize absorption time.

From the figure, we may conclude that a scheme with a low number of offspring is preferable if one knows the optimal mutation rate. From a practical point of view, it could, however, be difficult to find such optimal mutation rates. It could therefore be interesting to know how the algorithm behaves for mutation rates that are only very rough approximations of the optimal rates. Table IV provides some insight into this. It shows some good examples [e.g., (e 0.01 1) and (e 0.02 9)] of schemes for which a low number of offspring is not beneficial. Fig. 9 may provide an explanation for this. It displays the convergence velocity for some of the algorithms in the table. Although the linear algorithm provides a higher convergence velocity for various unitation values, the exponential scheme is better on difficult values, which are those where the velocity is low. To stress this, the graph is drawn on a logarithmic scale. In particular, a $u(\vec{x}) = 1$ individual is likely to be encountered. The exponential scheme performs best here, as it provides a wider variety of low mutation rates and thus enlarges the probability that the last bit is correctly flipped.

Using Table IV and the graphs of Fig. 7(b) and (c) and Fig. 8, we can compare the linear and the exponential schemes. The exponential mutation scheme performs surprisingly well. For example, a (e 0.02 0.9)-(1÷10)-EA performs significantly better than many (1÷3)-EAs. The surface of the exponential scheme displayed in Fig. 7(c) is also clearly below that of a linear scheme and is less sensitive to the values of the b_u and b_l bounds.

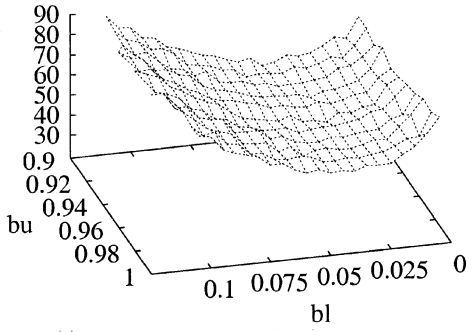
In comparison to constant mutation rates, the number of evaluations of the new algorithm is clearly much smaller. From the current results, however, a disadvantage of the new algorithm can also be extracted. Whereas with constant mutation rates, the addition of new offspring did not make the number of evaluations worse (in case one applies optimal mutation rates), this is not always the case with multiple mutation rates.

C. Long-Term Analysis of the Complex Trap Function

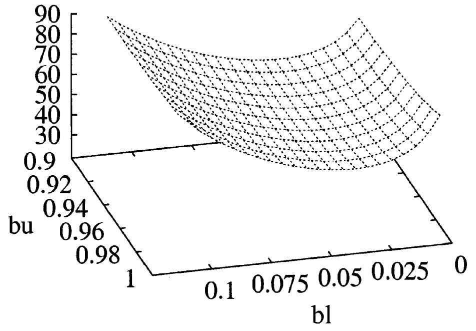
Considering absorption times again, we first have to check whether the one-optimum assumption still holds for this function. As can be seen in Fig. 10 for a particular function, this assumption seems still valid (we checked this also for the details which are invisible in the graph). Note that this figure is plotted slightly different than Fig. 7(c), as here the complete ranges for b_l and b_u are shown. Where b_l is higher than b_u, $E(T)$ for swapped values of b_l and b_u is shown, which may incorrectly give the impression that there are two optima.[2]

Using this assumption, we recompute the optimal upper and lower bounds of multiple mutation rates, for several numbers of offspring (Fig. 11). Many results for the basic trap function are also applicable here: a small number of offspring is better, and two different mutation rates perform better than two identical mutation rates. However, there also major differences: the lower

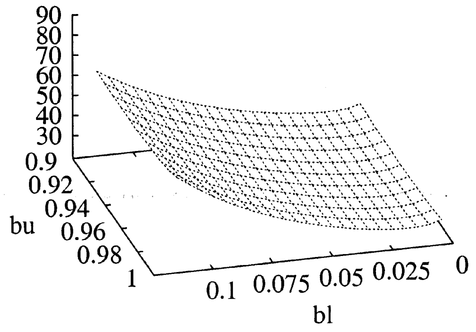
[2]Furthermore, some values which are too large are not plotted. This explains the strange graph for very low mutation rates.



(a)



(b)



(c)

$u(\vec{a})$	theory	experiment
1	48.891	50.126
2	58.526	59.810
3	63.694	62.201
4	68.164	67.678
5	70.970	69.908
6	72.597	71.355
7	73.417	71.101
8	73.893	72.187
9	74.121	74.630
10	74.188	73.230
11	74.120	74.182
12	73.887	73.656
13	73.403	74.239
14	72.560	70.611
15	70.905	70.664
16	68.071	66.359
17	63.694	63.318
18	58.529	59.277
19	49.200	50.478
20	1.000	1.000

(d)

Fig. 7. Absorption times of a (1÷10)-EA for given upper and lower bounds. The experimental values are obtained by averaging over 1000 experiments for each coordinate. The $l = 20$ function is used. (a) Experimental expected absorption time (linear scheme). (b) Theoretical expected absorption time (linear scheme). The mimimum is at $b_u = 0.04$ and $b_l = 0.93$ with a value of 56.0. (c) Theoretical expected absorption time (exponential scheme). The minimum is at $b_u = 0.02$ and $b_l = 0.96$ with a value of 26.94. (d) $T(s)$ for several values of $u(\vec{x})$ with $b_l = 0.01$ and $b_u = 1$ (linear scheme).

bound of the exponential scheme increases instead of decreases, and the exponential scheme performs worse for large numbers of offspring (although much better than a constant mutation rate: for $\lambda = 10$, 76 932 evaluations are needed for the optimal constant mutation rate $p_m^* \approx 0.4$, while ≈2400 evaluations suffice for the exponential scheme).

To explore this difference for high values of λ further, Fig. 12 shows the dependency of the optimal mutation rates on the location of z_2. For $z_2 = 6$, the complex trap function almost reduces to a normal counting ones problem. The optimal lower and upper bounds are almost equal here: one mutation rate performs best. The more the complex trap function turns into a basic trap function, the further the optimal mutation rate boundaries are apart. The $z_2 = 15$ function does not yield as much difficulties as the $z_2 = 14$ problem. As expected, the impact of using several mutation rates is not as large here as for other values, but still considerable. We computed that for an approximate optimal mutation rate of 0.385 still 61 053 evaluations are needed when one mutation rate is used.

For most values of z_2, it appears that the linear scheme performs slightly better, and, what is also important: the lower bound on the mutation rate is much more constant and allows for the application of a rule-of-thumb: $1/l$.

IV. Conclusions

Using trap functions, the results presented here give an idea how previously published EAs scale up to more complex

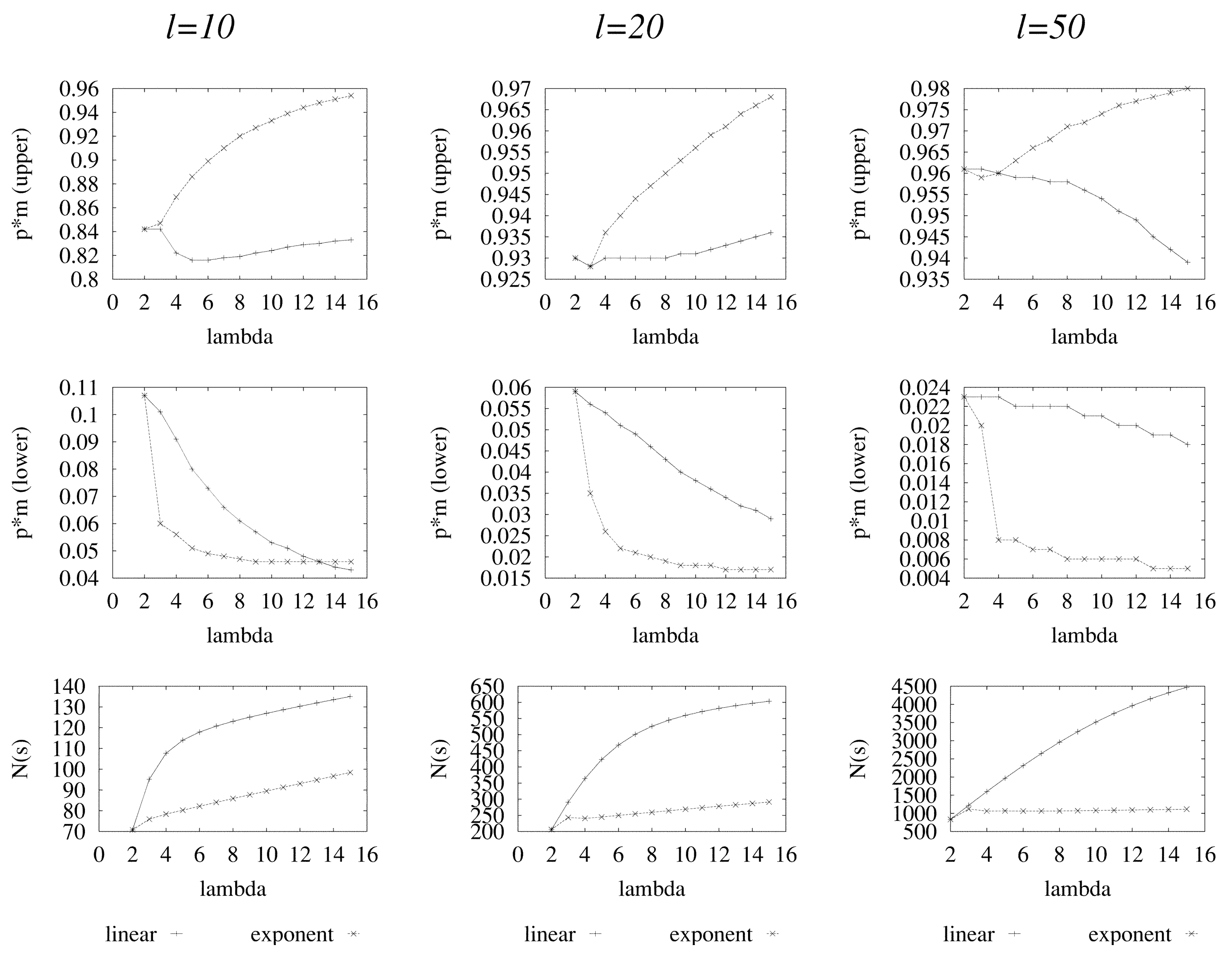


Fig. 8. Optimal upper and lower bounds for the mutation rates of a (1 ÷ λ) EA, with the resulting number of evaluations.

TABLE IV
EXPECTED NUMBER OF EVALUATIONS FOR SEVERAL ALGORITHMS

	$l = 10$				$l = 20$				$l = 50$ $(\times 10^{11})$			
λ	2	3	5	10	2	3	5	10	2	3	5	10
l 0.01 1	435	367	207	155	648	848	933	737	49	71	112	186
l 0.05 1	121	137	138	140	253	359	495	610	54	80	132	244
l 0.1 1	91	111	129	148	294	428	612	830	266	399	661	1270
l 0.02 0.9	150	184	151	136	299	398	533	596	**37**	**55**	**87**	**154**
l 0.05 0.9	88	114	123	**130**	**211**	**295**	**428**	**576**	53	80	130	243
l 0.1 0.9	**72**	**96**	**118**	137	244	353	536	789	261	392	650	1260
e 0.01 1	435	115	106	105	648	303	289	291	49	55	**47**	**48**
e 0.05 1	121	91	88	92	253	297	306	331	54	78	110	139
e 0.1 1	91	94	95	101	294	401	483	545	266	398	633	956
e 0.02 0.9	150	84	86	94	299	**249**	**251**	**284**	**37**	**49**	53	60
e 0.05 0.9	88	**77**	**80**	**90**	**211**	251	276	324	53	78	109	139
e 0.1 0.9	**72**	81	87	98	244	333	416	511	261	390	619	928

problems. More specifically, we have shown using numerical experiments that the $1/l$ guideline for optimal mutation rates [2], [14] optimizes local search (exploitation), but is incapable of providing sufficient diversity in the search (exploration). After having shown that an EA with one mutation rate and no crossover has many difficulties finding an optimum if there is an attracting local optimum, we investigated the possibility of using mutation to perform exploration.

For a trap function, we found that a mutation rate approximating 1, but not exactly equal to it, maximizes the probability that a new hill is climbed every generation, and thus enhances exploration. We used these observations to construct a new al-

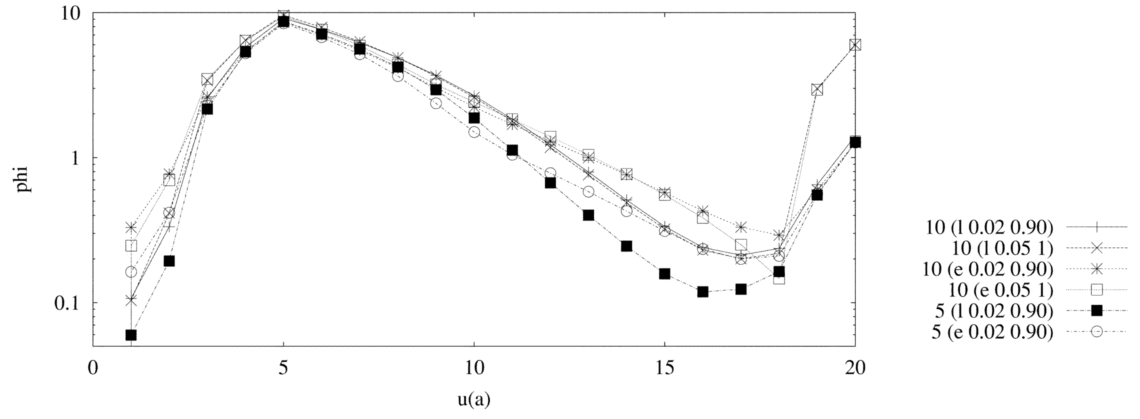
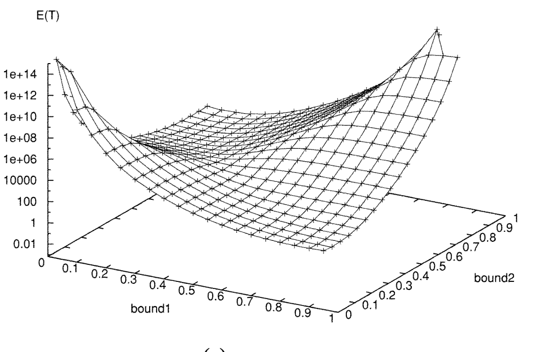


Fig. 9. Convergence velocity for several mutation schemes. Note that one scale is logarithmic. The $l = 20$ problem was used.



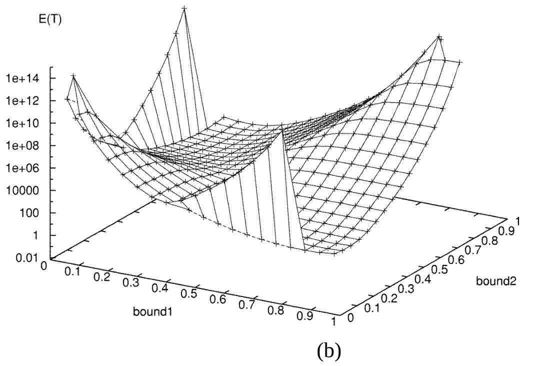


Fig. 10. Expected absorption times for a $(1 \div 10)$-EA with multiple mutation rates. The complex $l = 20$ trap function is used with $z_2 = 15$. (a) Linear mutation rates. (b) Exponential mutation rates.

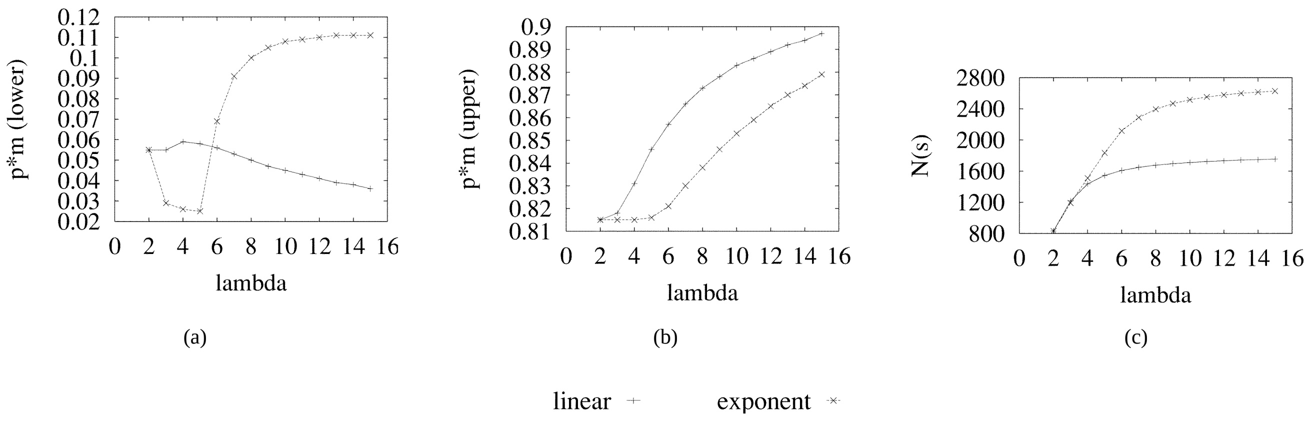


Fig. 11. Optimal upper and lower bounds of mutation rates, and the corresponding expected number of evaluations for the $l = 20$ complex trap function with $z_2 = 15$.

gorithm which provides several mutation rates every generation. Using numerical experiments, we found that indeed two mutation rates $1/l$ and ≈ 1 solved the trap problem efficiently. The experiments confirmed that both exploration and exploitation can be obtained by the mutation operator as long as there is variation in the available mutation rates.

To get a better insight in the importance of the multiple mutation rates, we applied the idea of multiple mutation rates on another trap function. This made clear that using several rates is beneficial in many difficult situations, also if one does not know the optimal mutation rate. The height of the optimal upper bound however depends very much on the characteristics of the function. On a basic counting ones problem, using additional mutation rates only increases the number of evaluations.

In this paper, we focused on one particular function with two local optima. We found that one high mutation rate is sufficient to obtain population diversity. Further investigations with more complex functions could provide better insight into how many high mutation rates are necessary. For such functions, it would also be interesting to compare the impact of high mutation rates with the effect of a crossover operator.

The use of multiple mutation rates to obtain diversity may not only be advantageous in the static problem we investigated currently; it could also be useful in dynamic problems. Using

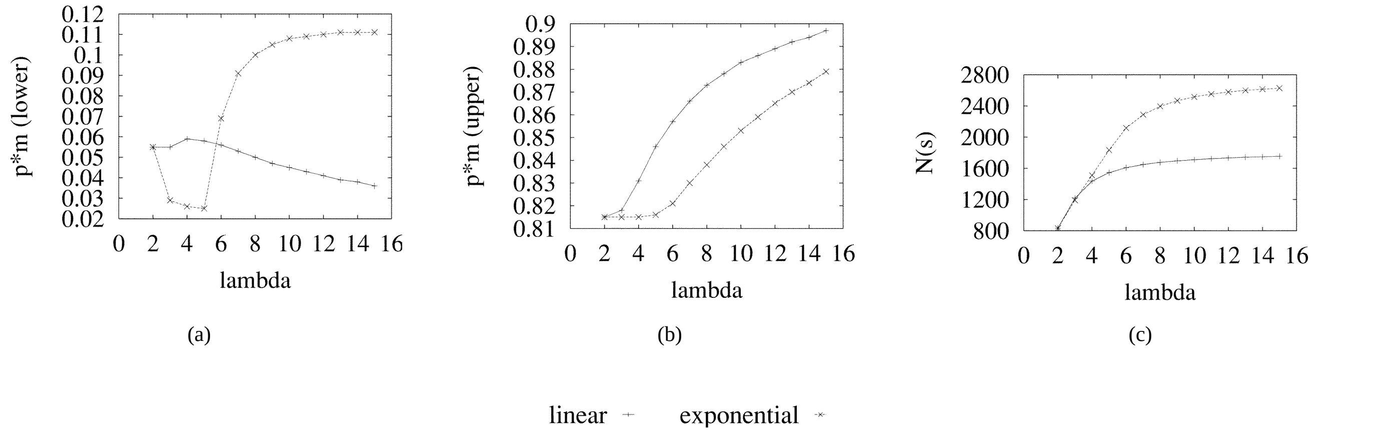


Fig. 12. Optimal upper and lower bounds of mutation rates, and the corresponding expected number of evaluations, for the $l = 20$ complex trap function and $\lambda = 10$.

some of the measures in this paper, we plan to investigate the applicability of such an EA on dynamic problems.

References

[1] H.-G. Beyer and D. V. Arnold, "Theory of evolution strategies—A tutorial," in *Theoretical Aspects of Evolutionary Computing*. ser. Natural Computing Series, L. Kallel *et al.*, Eds. Berlin, Germany: Springer-Verlag, 2001, pp. 109–133.

[2] Th. Bäck, "The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm," in *Parallel Problem Solving From Nature 2*, R. Männer and B. Manderick, Eds. Amsterdam, The Netherlands: Elsevier, 1992, pp. 85–94.

[3] ——, "Optimal mutation rates in genetic search," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 2–8.

[4] ——, "Order statistics for convergence velocity analysis in simplified evolutionary algorithms," in *Foundations of Genetic Algorithms 3*, D. Whitley and M. Vose, Eds. San Mateo, CA: Morgan Kaufmann, 1995, pp. 91–102.

[5] ——, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford Univ. Press, 1996.

[6] H.-G. Beyer, *The Theory of Evolution Strategies*, ser. Natural Computing. Berlin, Germany: Springer, 2001.

[7] K. Deb and D. E. Goldberg, "Analyzing deception in trap functions," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 93–108.

[8] W. Dörfler, *Mathematik für Informatiker*. München, Germany: Carl Hanser, 1978, vol. 1.

[9] A. E. Eiben, E. H. L. Aarts, and K. M. Van Hee, "Global convergence of genetic algorithms: An infinite Markov chain analysis," in *Parallel Problem Solving From Nature—Proc. 1st Workshop PPSN 1*. ser. Lecture Notes in Computer Science, H.-P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer, 1991, vol. 496, pp. 4–12.

[10] R. Galar, "Handicapped individual in evolutionary processes," *Biol. Cybern.*, vol. 53, pp. 1–9, 1985.

[11] ——, "Evolutionary search with soft selection," *Biol. Cybern.*, vol. 60, pp. 357–364, 1989.

[12] T. Jansen and I. Wegener, "On the analysis of evolutionary algorithms—A proof that crossover really can help," in *Proc. 7th Annu. Eur. Symp. Algorithms (ESA '99)*. ser. Lecture Notes in Computer Science, J. Nesetril, Ed. Berlin, Germany: Springer, 1999, vol. 1643, pp. 184–193.

[13] L. Kallel, B. Naudts, and A. Rogers, Eds., *Theoretical Aspects of Evolutionary Computing*. ser. Natural Computing Series. Berlin, Germany: Springer-Verlag, 2001.

[14] H. Mühlenbein, "How genetic algorithms really work: I. Mutation and hillclimbing," in *Parallel Problem Solving From Nature 2*, R. Männer and B. Manderick, Eds. Amsterdam, The Netherlands: Elsevier, 1992, pp. 15–25.

[15] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart, Germany: Frommann-Holzboog, 1973.

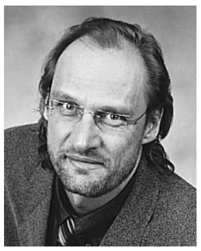
[16] A. Rogers and A. Prügel-Bennett, "A solvable model of a hard optimization problem," in *Theoretical Aspects of Evolutionary Computing*. ser. Natural Computing Series, L. Kallel , B. Naudts, and A. Rogers, Eds. Berlin, Germany: Springer–Verlag, 2001, pp. 207–221.

[17] G. Rudolph, *Covergence Properties of Evolutionary Algorithms*. Hamburg, Germany: Verlag Dr. Kovač, 1997.

[18] H.-P. Schwefel, "Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie," in *Interdisciplinary Systems Research*. Basel, Germany: Birkhäuser, 1977, vol. 26.

Siegfried Nijssen received the Master's degree in computer science in 2000 from Leiden University, Leiden, The Netherlands, where he is currently working toward the Ph.D. degree.

His research interests include evolutionary optimization and artificial intelligence.



Thomas Bäck received the Diploma degree in computer science in 1990 and the Ph.D. degree in computer science in 1994, both from the University of Dortmund, Dortmund, Germany.

Since 1996, he has been an Associate Professor of Computer Science at the Leiden Institute for Advanced Computer Science (LIACS), Leiden, The Netherlands. Since early 2000, he has been Managing Director of NuTech Solutions GmbH, Dortmund, Germany, and CTO of NuTech Solutions, Inc., Charlotte, NC. He is the author of the book *Evolutionary Algorithms and Theory and Practice* (Oxford, U.K.: Oxford Univ. Press, 1996), co-editor of *Handbook of Evolutionary Computation* (Bristol, U.K.: Institute of Physics, 1997 and Oxford, U.K.: Oxford Univ. Press, 1997), and Associate Editor of the *Natural Computing* book series and the journals *Evolutionary Computation* and *Natural Computing*. He has authored or co-authored more than 100 publications on evolutionary computation and related fields. His current research interests include dynamic optimization, evolutionary algorithms for industrial problems, knowledge extraction from large databases, and bioinformatics applications of natural computing.

Dr. Bäck is a Program Committee member of many major conferences on evolutionary computation. In 1995, he received the Best Dissertation Award from the German Association for Computer Science (GI) for his Ph.D. dissertation on evolutionary algorithms.