Journal of Mathematical Modelling and Algorithms **3:** 349–366, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

The Analysis of Evolutionary Algorithms on Sorting and Shortest Paths Problems

JENS SCHARNOW, KARSTEN TINNEFELD⋆ and INGO WEGENER⋆
FB Informatik, LS2, University of Dortmund, 44221 Dortmund, Germany.
e-mail: wegener@ls2.cs.uni-dortmund.de

Abstract. The analysis of evolutionary algorithms is up to now limited to special classes of functions and fitness landscapes. E.g., it is not possible to characterize the set of TSP instances (or another NP-hard combinatorial optimization problem) which are solved by a generic evolutionary algorithm (EA) in an expected time bounded by some given polynomial. As a first step from artificial functions to typical problems from combinatorial optimization, we analyze simple EAs on well-known problems, namely sorting and shortest paths. Although it cannot be expected that EAs outperform the well-known problem specific algorithms on these simple problems, it is interesting to analyze how EAs work on these problems. The following results are obtained:

- Sorting is the maximization of "sortedness" which is measured by one of several well-known measures of presortedness. The different measures of presortedness lead to fitness functions of quite different difficulty for EAs.
- Shortest paths problems are hard for all types of EA, if they are considered as single-objective optimization problems, whereas they are easy as multi-objective optimization problems.

Mathematics Subject Classifications (2000): 68W20, 68W40.

Key words: randomized search heuristics, evolutionary algorithms, analysis of expected run time, sorting, shortest paths.

1. Introduction

Our aim is to contribute to a theory of evolutionary algorithms (EAs) which analyzes the expected optimization time of EAs on important and interesting problems. Nowadays, it is a vision to explain the success of EAs on hard problems by identifying those instances of the problem where the considered EA finds the optimum in expected polynomial time. In order to develop tools for such results EAs have to be analyzed in various situations.

Some interesting classes of fitness functions have been investigated, e.g., separable functions [2], monotone polynomials of small degree [17], long-path functions [8, 15, 3], and royal road functions [11, 9]. However, these are artificial functions and problems.

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

Here we choose the approach to investigate EAs on the most basic and important computer science problems, namely sorting (the maximization of the sortedness) and shortest paths problems. We do not and cannot expect EAs to outperform Quicksort or Dijkstra's algorithm. We are mainly interested in the analysis of EAs for black-box optimization. Such EAs are designed to work on different types of problems. In particular, they apply rather general and, therefore, not problem-specific search operators. Here, we are interested in the way how different fitness functions support the search for the optimum performed by EAs. Note that the considered problems are solvable in short polynomial time by problem-specific algorithms. Using highly specialized search operators would be close to solving the problem directly.

Typically, sorting is not considered as a problem of combinatorial optimization. Sorting algorithms are based on comparisons of two objects. However, each permutation can be considered as an individual which is more or less sorted. The fitness can be measured by one of the well-known measures of presortedness. This turns the sorting problem into a combinatorial optimization problem and it seems to be fundamental to analyze EAs on these problems. In Section 2, the corresponding fitness functions are introduced and mutation operators are discussed. The analysis in Section 3 shows that most measures of presortedness contain enough information to direct the optimization by EAs. However, there is a well-known measure of presortedness where EAs have problems to improve the fitness.

Shortest paths problems are more difficult optimization problems and EAs get stuck for certain instances. In Section 4, we describe the corresponding fitness function and an alternative as multi-objective optimization problem. Moreover, we prove that only the multi-objective optimization problem description directs the search of EAs efficiently. This is the first result of this type for EAs.

This paper is based on the conference paper by Scharnow, Tinnefeld, and Wegener [16]. However, it contains some new and some improved results and all proofs are complete.

This paper has been motivated by the vision that we can analyze EAs in combinatorial optimization in the same way as other types of randomized optimization algorithms. The conference version was the first one with a complete analysis (without any assumption) for practical problems from combinatorial optimization. Afterwards, Giel and Wegener [6] have analyzed EAs on the problem of computing maximum matchings.

2. Optimization Problems Based on Sorting Problems

Given a sequence of n distinct elements from a totally ordered set, sorting is the problem of maximizing the sortedness. By renaming, we can identify the elements with $1, \dots, n$. The aim is to find the unknown optimal permutation π_{opt} such that $(\pi_{\text{opt}}(1), \dots, \pi_{\text{opt}}(n))$ is the sorted sequence with respect to some unknown criterion. The search space is the set of all permutations π on $\{1, \dots, n\}$. The fitness function $f_{\pi_{\text{opt}}}(\pi)$ describes the sortedness of $(\pi(1), \dots, \pi(n))$ with respect to $(\pi_{\text{opt}}(1), \dots, \pi_{\text{opt}}(n))$. Because of the symmetry in the set of all permutations we can simplify our notation by considering the case that $\pi_{\text{opt}} = \text{id}$, i.e., $\pi_{\text{opt}}(i) = i$ for all i. In particular, the fitness function is denoted by $f(\pi)$ instead of $f_{\text{id}}(\pi)$. We stress the fact that this does not change the problem. Each randomized search heuristic which does not use the "names" of the elements works in the same way for $\pi_{\text{opt}} = \text{id}$ as for each other π_{opt}. Now we have to specify an appropriate fitness function, i.e., a measure of presortedness. Such measures have been introduced in the discussion of adaptive sorting algorithms (see, e.g., [14]). The number of these measures is large and one may find measures simplifying the analysis of EAs. Therefore, we have decided to investigate the five most often discussed measures of presortedness.

- INV(π) measures the number of pairs (i, j), $1 \leqslant i < j \leqslant n$, such that $\pi(i) < \pi(j)$ (pairs in correct order),
- HAM(π) measures the number of indices i such that $\pi(i) = i$ (elements at the correct position),
- RUN(π) is by 1 larger than the number of indices i such that $\pi(i+1) < \pi(i)$ (number of maximal sorted blocks called runs), leading to a minimization problem,
- LAS(π) equals the largest k such that $\pi(i_1) < \cdots < \pi(i_k)$ for some $i_1 < \cdots < i_k$ (length of the longest ascending subsequence),
- EXC(π) equals the minimal number of exchanges (of pairs $\pi(i)$ and $\pi(j)$) to sort the sequence, again leading to a minimization problem.

We remark that these fitness functions are easy to evaluate. By definition, INV(π) can be computed in time $O(n^2)$ and HAM(π) and RUN(π) can be computed in linear time $O(n)$. Orlowski and Pachter [13] describe an algorithm to compute LAS(π) in time $O(n \log n)$. Instead of LAS(π) some authors consider REM(π), the minimal number of elements which have to be removed to obtain an ascending sequence. Obviously, $\text{REM}(\pi) = n - \text{LAS}(\pi)$. It is interesting to note that $\text{LAS}(\pi) = n - \text{JUMP}(\pi)$ where JUMP(π) is the minimal number of jump operations (the formal definition is given later) to sort the sequence. This follows since a jump operation can increase the LAS value by at most 1 and since it is always possible to get this increase. We can take an element which is not in a specific longest ascending subsequence and let it jump to a position where it lengthens this ascending subsequence. Finally, EXC(π) can be computed in linear time. This calculation is based on the cycle structure of permutations. The sorted sequence is the only one with n cycles. In all other cases, it is possible to choose an exchange operation which exchanges an element x at a wrong position with the element sitting at the correct position of x. This increases the number of cycles by one. Moreover, it is easy to see that it is impossible to increase the number of cycles by more than one by a single exchange operation.

In order to specify an EA we have to discuss the considered search operators. We only want to use search operators which have been applied often when manipulating permutations. This again is motivated by the aim to investigate EAs which are not specific for sorting. Most crossover operators for permutations are rather complicated (for an overview see [1]). We are not able to analyze GAs with crossover for sorting problems (although we conjecture that the generic crossover operators for permutations are not useful for our problems). Hence, we investigate only mutation-based EAs.

The most simple local operation is swap(i) which exchanges the elements at the positions i and $i+1$. Swaps are quite local. The minimal number of swaps to sort a random permutation is known (folklore) to be $\Theta(n^2)$. There are three less local operations which generalize swaps:

- exchange(i, j) exchanges the elements at the positions i and j,
- jump(i, j) causes the element at position i to jump to position j while the elements at positions $i+1, \dots, j$ (if $j > i$) or $j, \dots, i-1$ (if $j < i$) are shifted in the appropriate direction,
- reverse(i, j), where $i < j$, reverses the ordering of the elements at the positions $i, \dots, j$.

These are the three local operators which have been applied in many different situations (evolutionary algorithms or the optimization of the variable order for OBDDs (ordered binary decision diagrams), the most often used data structure for Boolean functions). Up to now, we have analyzed only EAs based on exchanges and jumps.

We illustrate these local operations in Figure 1. Later, we will only count the number of fitness evaluations as it is usual in the analysis of EAs. This makes sense only if there are efficient algorithms to compute the fitness efficiently, i.e. in

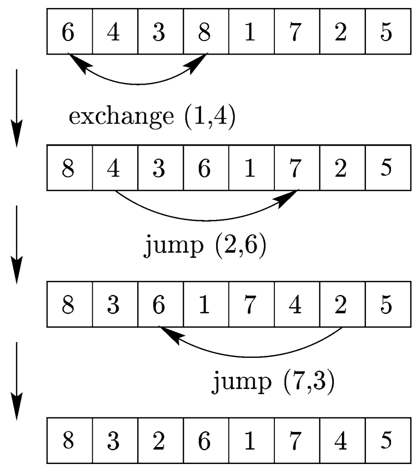

Figure 1. The local operations for the sorting problem.

expected time $O(n)$ or a little more. We have discussed above the time to evaluate the fitness of an individual. The update after a jump step or an exchange step (which can be realized by two jump steps) can be performed more efficiently in many cases. For INV, time $O(|i-j|) = O(n)$ is enough. Using standard data structures, RUN, EXC, and HAM can be updated in time $O(1)$. For LAS, we only have an $O(n \log n)$ bound. Moreover the expected number of local operations is $O(1)$.

As mentioned before, we want to analyze an EA which is not designed for sorting. Hence, we transform the $(1+1)$ EA working on the Boolean search space $\{0,1\}^n$ to work on the search space of all permutations on $\{1,\dots,n\}$. The local operation of the $(1+1)$ EA on $\{0,1\}^n$ is the flip of one bit. Moreover, the $(1+1)$ EA does not accept worsenings. In order not to get stuck forever in a local optimum the $(1+1)$ EA flips each bit independently from the others with probability $1/n$. Hence, the number of local operations is asymptotically Poisson distributed with parameter $\lambda = 1$.

A $(1+1)$ EA on permutations which performs one local operation per step gets stuck in local optima for certain fitness functions. Therefore, we apply a random number of local operations per step. Moreover, we avoid steps doing nothing (this is not essential for the analysis). This leads to the following mutation operator.

– Choose S according to a Poisson distribution with parameter $\lambda = 1$ and perform sequentially $S+1$ exchange or jump steps where for each step (i, j) is chosen uniformly at random among all pairs (k, l), $1 \leqslant k, l \leqslant n$, $k \neq l$, and it is decided by a fair coin flip whether exchange(k, l) or jump(k, l) is performed.

We also may consider only exchange or only jump steps. We analyze the following evolutionary algorithm shortly called $(1+1)$ EA:

– Choose the first search point π uniformly at random.
– Repeat: Produce π' by mutation from π and replace π by π' if π' is not worse than π ($f(\pi') \geqslant f(\pi)$ in the case of a maximization problem and $f(\pi') \leqslant f(\pi)$ otherwise).

In applications, one needs a stopping criterion. Here we consider the infinite stochastic process $(\pi_1, \pi_2, \pi_3, \dots)$ where π_1 is the first chosen permutation and π_t, $t \geqslant 2$, equals the permutation π' in the t-th step of the above algorithm and investigate the random variable that equals the first point of time t when π_t is optimal. This random variable is called optimization time. The optimization problems lead to fitness landscapes since the local operations define a generic graph on the set of all permutations. The permutations π and π' are connected by an edge if π can be obtained from π' by a jump or exchange step. Together with each of the five fitness functions we obtain a fitness landscape. The $(1+1)$ EA does not accept worsenings. This is why the $(1+1)$ EA is often called a hill climber. There is a major difference to (randomized) local search which performs only one local operation per step and does not accept worsenings. The $(1+1)$ EA performs sometimes many local operations and can "jump from one hill to another hill crossing a valley."

3. The Analysis of the $(1+1)$ EA on the Optimization Problems Based on Sorting Problems

Here we analyze the performance of the $(1+1)$ EA on the different fitness functions introduced in Section 2. We start with a simple lower bound.

THEOREM 1. *The expected optimization time of the* $(1+1)$ *EA on each of the fitness functions* INV, HAM, RUN, LAS, *or* EXC *is bounded below by* $\Omega(n^2)$.

Proof. The probability of starting with the optimal individual equals $1/(n!)$. Otherwise, we investigate the final step producing the optimal individual. It is necessary that the last local operation (exchange or jump) produces the optimal individual. However, for each non-optimal individual there are at most two exchange operations to change it into the optimal one and there are at most two jump operations with this property. The bound two follows from the fact that exchange(i, j) = exchange(j, i) and jump$(i, i+1)$ = jump$(i+1, i)$ and that all other local operations have a different effect. Therefore, the probability of a success is bounded above by $\frac{1}{2} \cdot \frac{2}{n(n-1)} + \frac{1}{2} \cdot \frac{2}{n(n-1)} = 2 \cdot \frac{1}{n(n-1)}$ and the expected waiting time is bounded below by $\frac{1}{2}n(n-1)$. □

Theorem 1 holds for all fitness functions with a unique optimum. Hence, based on the considered local operations quadratic optimization time is necessary for such problems. For four of the five fitness landscapes it is not too difficult to obtain an almost matching upper bound of $O(n^2 \log n)$.

THEOREM 2. *The expected optimization time of the* $(1+1)$ *EA on each of the fitness functions* INV, HAM, LAS, *or* EXC *is bounded above by* $O(n^2 \log n)$.

Proof. First, we consider the fitness function INV. Let π be the current search point, $1 \leqslant i < j \leqslant n$, and $\pi(i) > \pi(j)$, i.e., (i, j) is an incorrect pair. Let a, b, and c be the number of elements at the positions $i+1, \ldots, j-1$ which are smaller than $\pi(j)$, between $\pi(j)$ and $\pi(i)$, and larger than $\pi(i)$, respectively. Then exchange(i, j) increases the fitness by $2b+1$, jump(i, j) changes the fitness by $a+b-c+1$, and jump(j, i) changes the fitness by $-a+b+c+1$. At least one of the values $a+b-c+1$ and $-a+b+c+1$ is positive. Hence, the fitness is increased by at least 1 if we perform exactly one local operation (probability $1/\mathrm{e}$) and the operation is exchange(i, j) or exchange(j, i) (probability $1/(n(n-1))$) or a good one among jump(i, j) and jump(j, i) (probability $1/(2n(n-1))$). If the number of incorrect pairs equals m, the probability of increasing the fitness is at least $3m/(2\mathrm{e}n(n-1))$ and the expected waiting time for such an event is bounded above by $\frac{2}{3}\mathrm{e}n(n-1)/m$. Since $1 \leqslant m \leqslant \binom{n}{2}$, the expected optimization time can be bounded by

$$\frac{2}{3}\mathrm{e}n^2 \sum_{1 \leqslant m \leqslant n(n-1)/2} 1/m = \frac{2}{3}\mathrm{e}n^2 H(N),$$

where $N = n(n-1)/2$ and $H(N)$ is the N-th harmonic number which can be bounded above by $\ln N + 1$.

If $\mathrm{HAM}(\pi) = k$, we have $n - k$ elements at incorrect positions. If i sits at position $j \neq i$, also the element at position i sits at a wrong position. Then exchange(i, j) and exchange(j, i) improve the fitness by at least 1. Hence, there are at least $n-k$ good exchange operations leading to an upper bound of $2en^2 H(n)$ for the expected optimization time of the (1 + 1) EA.

If $\mathrm{LAS}(\pi) = k$, there are $n-k$ elements outside a fixed ascending subsequence of length k. Each of them can jump to at least one position where it fits into the ascending subsequence. Hence, there are at least $n-k$ good jump operations leading again to an upper bound of $2en^2 H(n)$ for the expected optimization time.

If $\mathrm{EXC}(\pi) = k > 0$, the permutation π consists of $n - k$ cycles and at most $n - k - 1$ cycles of length 1. Hence, at least $k + 1$ elements are in cycles of length at least 2. If such an element i is exchanged with the element sitting at position i, we obtain at least one new cycle of length 1 and we increase the fitness by at least one. Hence, there are at least $k + 1$ exchange operations increasing the fitness and we obtain an upper bound of $2en^2 H(n)$ for the expected optimization time. □

The proofs have shown that the constants involved in the O-notation are not very large and that the constants in the Ω-notation are not very small. However, the question is whether the lower or the upper bound is better. Compared to the situation of $\mathrm{ONEMAX}(a) = a_1 + \cdots + a_n$ and the typical (1 + 1) EA flipping each bit with probability $1/n$, our methods lead to the lower bound n and the upper bound $enH(n) \leqslant en \ln n + en$. Droste, Jansen, and Wegener [4] have proved a lower bound of $(1/6)n \ln n$ implying that the upper bound is close to optimal. However, if $\mathrm{ONEMAX}(a) = k$, there are exactly $n - k$ fitness increasing 1-bit mutations. The situation here is much more difficult since there can be more good local operations than considered in the proof of the upper bounds. Moreover, for some fitness functions and search points, one local operation can improve the fitness by more than 1.

First, we describe a lower bound technique which will be applied later to the fitness functions HAM, EXC, and LAS. The method can be applied if the following two conditions are fulfilled. The first one states that one local step can improve the fitness at most by a constant c. Let us investigate whether this is fulfilled in our situation:

- LAS: A jump operation can increase the fitness by at most 1, since $\mathrm{LAS}(\pi) = n - \mathrm{JUMP}(\pi)$ and JUMP equals the minimal number of jump steps. An exchange step can increase the fitness by at most 2 since it can be simulated by two jump steps.
- HAM: An exchange step can improve the fitness by at most 2, since only the positions of two objects are changed. The method does not work for jump steps since $(2, \dots, n, 1)$ with fitness 0 can be optimized by jump $(n, 1)$.
- EXC: an exchange step can improve the fitness by at most 1 (by definition of the fitness function) but $(2, \dots, n, 1)$ has one cycle and, therefore, the worst fitness and can be optimized by jump $(n, 1)$.

The second condition asks for some $k_0 = k_0(n)$ such that the following holds. If the fitness of the current search point is at most $k \leqslant k_0$ away from the optimal fitness, then the probability of a fitness improving step is bounded above by $O(k/n^2)$. Now we consider the variant of the $(1+1)$ EA which performs one local operation in each step. Let c be the constant for the largest possible fitness improvement and let c' be the constant from the $O(k/n^2)$ bound. The expected waiting time to improve a search point whose fitness differs by j from the optimal fitness is at least $n^2/(c'j)$. If we start with a search point whose fitness is at least $k \leqslant k_0$ away from the optimum we may sum some of these waiting times, at least every cth term which is minimized by

$$(n^2/c') \cdot \left(\frac{1}{c} + \frac{1}{2c} + \cdots + \frac{1}{\lfloor k/c \rfloor c}\right) = \Omega(n^2 \log k).$$

If we can apply this method for some $k = \Omega(n^\varepsilon)$, we obtain a lower bound of $\Omega(n^2 \log n)$.

We prove that we obtain the same asymptotic lower bound for the $(1 + 1)$ EA. The first claim is that the probability of more than $3t$ local operations within t steps is exponentially small. The number of local operations per step is $X + 1$ where X is Poisson distributed with parameter $\lambda = 1$. Hence, there are t local operations for sure and we have to consider the sum of t independent Poisson distributed random variables (where $\lambda = 1$). Each is the limit of Bernoulli distributions with m trials and a success probability of $1/m$. This are altogether tm Bernoulli trials with a success probability of $1/m$. By Chernoff bounds, the probability of at least $2t$ successes is exponentially small with respect to t. This holds for all m and, therefore, for $m \to \infty$.

A difference between the $(1 + 1)$ EA and the "local" algorithm with one local operation per step is the following. The $(1+1)$ EA performs several local operations in one step and it is decided afterwards whether the new search point is accepted. This is possible even if one of the local operations would decrease the fitness. In general, we cannot use bounds obtained for the "local" algorithm as bounds for the $(1 + 1)$ EA. Here, we are in a special situation. The probability that a local operation improves the fitness is bounded above by $c'k/n^2$ and the fitness improvement is bounded by c. We optimistically assume a probability of $c'k/n^2$ of a fitness improvement which always is assumed to be c. Then the $(1+1)$ EA would get faster if we could ignore the effects of fitness worsenings. Hence, we can apply the asymptotic lower bound for the "local" algorithm. We obtain a factor of $1/3$ since we consider up to $3t$ local operations and a factor of $1 - o(1)$ since there is a tiny probability of having more than $3t$ local operations within t steps.

First, we apply the method to the fitness functions HAM and EXC. By the comments above, we have to restrict the algorithm to a $(1 + 1)$ EA using only exchange operations.

THEOREM 3. *The expected optimization time of the* $(1 + 1)$ *EA using only exchange operations on the fitness function* HAM *equals* $\Theta(n^2 \log n)$.

Proof. The upper bound is contained in the proof of Theorem 2. The HAM value of a random permutation has been investigated intensively in combinatorics (see [7]). The expected fitness is close to 1 and the probability of a fitness larger than εn is exponentially small for each $\varepsilon > 0$. Hence, we can apply our lower bound technique for $k = \Omega(n)$. If the fitness of a search point is $n - m$ (and, therefore, m from the optimal value), an exchange step of the objects at positions i and j can improve the fitness only if the positions i and j belong to the m wrong positions and at least one object obtains its correct position. The probability of choosing at first an object at a wrong position equals m/n and the probability of choosing afterwards an appropriate partner is at most $2/(n-1)$. Altogether, the probability of a fitness improving step is bounded by $O(m/n^2)$. Hence, the lower bound technique leads to the proposed bound. □

It is well known that the fitness measures HAM and EXC are closely related.

PROPOSITION 1. *If* $\mathrm{EXC}(\pi) = k > 0$, *then* $n - 2k \leqslant \mathrm{HAM}(\pi) \leqslant n - k - 1$.

Proof. If $\mathrm{EXC}(\pi) = k > 0$, then permutation π consists of $n - k < n$ cycles. At most $n - k - 1$ cycles can have length 1 implying that $\mathrm{HAM}(\pi) \leqslant n - k - 1$. If $k \geqslant n/2$, the lower bound is trivial. Otherwise, $n - k > n/2$ implying that there are cycles of length 1. We have $n - k$ positive integers (the cycle lengths) whose sum equals n. We get the minimal number of cycles of length 1 if all other cycles have length 2. Then we have k cycles of length 2 and $n - 2k$ cycles of length 1 implying the lower bound on $\mathrm{HAM}(\pi)$. □

THEOREM 4. *The expected optimization time of the* $(1 + 1)$ *EA using only exchange operations on the fitness function* EXC *equals* $\Theta(n^2 \log n)$.

Proof. The upper bound is again contained in the proof of Theorem 2. Since $\mathrm{HAM}(\pi) \leqslant \varepsilon n$ with overwhelming probability for the initial search point π, by Proposition 1, also $\mathrm{EXC}(\pi) \geqslant (n - \varepsilon n)/2 = \Omega(n)$ with overwhelming probability. An algorithm minimizing EXC maximizes HAM. Hence, we may use the potential function HAM to measure the progress of the optimization process. This leads to the same lower bound as obtained in the proof of Theorem 3. □

For the following results we need results on the gambler's ruin problem (see [5]). Alice owns A \$ and Bob B \$. They play a coin-tossing game with a probability of $p \neq 1/2$ that Alice wins a round in this game, i.e., Bob pays 1 \$ to Alice. Let $t := (1 - p)/p$. Then Alice wins, i.e., she has $(A + B)$ \$ before being ruined, with a probability of $(1 - t^A)/(1 - t^{A+B}) = 1 - t^A(1 - t^B)/(1 - t^{A+B})$.

THEOREM 5. *The expected optimization time of the* $(1 + 1)$ *EA on the fitness function* LAS *equals* $\Theta(n^2 \log n)$.

Proof. The upper bound is contained in Theorem 2. For the lower bound, we again apply the lower bound technique discussed above. We choose $k_0 = n^{\varepsilon}$ for some $\varepsilon \in (0, 1/3)$. However, we cannot guarantee that the probability of a fitness increasing step is bounded above by $O(k/n^2)$, the general bound is only $O(k^2/n^2)$ which only leads to the already proved bound of $\Omega(n^2)$. The proof shows that it is likely (which means in this proof a probability of $1 - o(1)$) to produce in short time an accepted search point where the success probability is $O(k/n^2)$ and that it is likely that we only accept search points with this property. This implies the proposed bound.

First, we have to investigate properties of search points π where $\text{LAS}(\pi)$ is large. A permutation π may have many longest ascending subsequences, e.g., $(4, 5, 6, 1, 2, 3, 7, 8, \dots, n)$ has two of them (of length $n - 3$) and $(2, 1, 4, 3, 6, 5, 7, 8, \dots, n)$ has even $2^3 = 8$ of them. However, the following useful fact holds.

FACT 1. *If element i is at position j in some longest ascending subsequence* (LAS)*, then it is in each* LAS *containing it at position j.*

Proof. If i can sit at position $j' > j$ of some LAS (similarly for $j' < j$), then we can combine the $j' - 1$ elements smaller than i of this LAS, the element i, and the $\text{LAS}(\pi) - j$ elements larger than i of the first LAS to an ascending subsequence of length $(j' - 1) + 1 + (\text{LAS}(\pi) - j) > \text{LAS}(\pi)$ in contradiction to the definition of a LAS. □

Hence, for $\text{LAS}(\pi) = n - k$, we can define $\text{pos}(i, \pi)$ as the unique position of element i in a LAS where $\text{pos}(i, \pi) = nil$ indicates that i is in no LAS. The function pos takes $n - k$ values different from nil implying that there are at least $n - 2k$ elements with a unique position different from nil. These elements are in each LAS at the same position. Hence, if such an element is jumping away, then the fitness decreases. If the element is part of an exchange step, the fitness cannot increase. Hence, only local operations concerning the at most $2k$ so-called outsiders are of interest.

If an exchange step can improve the fitness, then one of the two jumps resulting in this exchange step increases the fitness. Therefore, it is sufficient to investigate steps where one of the outsiders jumps. The probability of choosing an outsider is $O(k/n)$. Hence, we are in a good position if each of these elements has only $O(1)$ good destinations. Now we apply the fact that we only investigate the last phase of the search where k is small. We consider the $m \geqslant n - 2k$ elements $a_1, \dots, a_m$ contained in each LAS. In the considered search point π there are $m + 1$ so-called spaces, namely the subsequence before a_1, the subsequence behind a_{m+1}, and the subsequences between a_i and a_{i+1}. For each of the $n - m$ outsiders, there is one space where it can increase the fitness. Hence, we are done if it is likely enough that each space contains $O(1)$ elements. This is not necessarily the case for the first considered π. We prove the theorem by proving the following claims.

CLAIM 1. *After a period of $\Theta(n \log^2 n)$ steps, with probability $1 - o(1)$, each space contains at most one outsider and the fitness has not been increased.*

CLAIM 2. *If each space of π contains at most one outsider, then in a period of $\Theta(n^2 \log n)$ steps, with probability $1 - o(1)$, each space contains at most 6 outsiders.*

If the events considered in the claims happen, we can apply our lower bound technique for a period of length $\Theta(n^2 \log n)$ and this proves the theorem. Hence, it is sufficient to prove the claims.

Proof of Claim 1. The coupon collector's theorem (see [12]) implies that, with probability $1 - o(1)$, this period contains for each element one step with a single step where this element jumps. For each of the $n - m$ outsiders, we consider the last of these operations. The probability to jump to a space of length 0 is $1 - O(n^{\varepsilon-1})$. Hence, the probability of at least one element jumping into a space of positive length is $O(n^{2\varepsilon-1})$. These steps are always accepted for outsiders. In order to increase the fitness, one of the $n - m$ outsiders has to move to one of the at most $O(n^{\varepsilon})$ good destinations. The probability of this is $O(n^{2\varepsilon-2})$ and the probability of such a step within the period is $O(n^{2\varepsilon-1} \log^2 n)$. The total error probability can be bounded by $O(n^{3\varepsilon-1})$ which is $o(1)$, since $\varepsilon < 1/3$. □

Proof of Claim 2. First, we investigate only one space. If the space is of size $O(1)$ but not empty, the probability that the space size decreases is $\Omega(1/n)$ (choose an element from the space and let it jump somewhere outside the space). The probability that one local operation increases the space size is $O(n^{\varepsilon-1} \cdot n^{-1}) = O(n^{\varepsilon-2})$ (choose one outsider and move it into the space). We have to take into account that a step can consist of several local operations. The probability of at least two space increasing steps is $O(n^{2\varepsilon-4})$. The probability that this happens for one of the spaces within $O(n^2 \log n)$ steps is $O(n^{3\varepsilon-2} \log n) = o(1)$. Hence, we can exclude this possibility. The conditional probability that a step changing the space size is space increasing is $p := O(n^{\varepsilon-1})$. For each space, we have a gambler's ruin problem and ask for the probability q to obtain space size 6 before space size 0 when starting with space size 1. The parameter t from the gambler's ruin problem equals $t = (1 - p)/p = \Omega(n^{1-\varepsilon})$. Therefore, $q = (t - 1)/(t^6 - 1) = O(n^{5\varepsilon-5})$. The probability that this happens for one space within $O(n^2 \log n)$ steps is bounded by $O(n^{6\varepsilon-5} n^2 \log n) = o(1)$, since $\varepsilon < 1/3$. □

The fitness function INV is the only one where our arguments result in an upper bound of $O(n^2 \log n)$ for the $(1+1)$ EA using only exchange operations and for the $(1 + 1)$ EA using only jump operations. We are not able to prove a corresponding lower bound in any of these cases, although we conjecture that these bounds hold. The difficulty in proving the lower bounds is that a single operation can increase the fitness from $\binom{n}{2} - \Theta(n)$ to the optimal value $\binom{n}{2}$. Again jump$(n, 1)$ optimizes $(2, 3, \ldots, n, 1)$. The vector $(n, 2, \ldots, n - 1, 1)$ is optimized by exchange$(1, n)$.

However, we are only discussing the small differences between the asymptotic run times n^2 and $n^2 \log n$. In any case, we conclude that the $(1 + 1)$ EA solves the sorting problem efficiently if sortedness is measured by INV, HAM, EXC, or LAS. We still have to discuss the fitness function RUN. Let us investigate the individual

$$2\quad 5\quad 6\quad 7\quad 14\quad 15\quad 16\,|\,1\quad 3\quad 4\quad 12\,|\,10\quad 11\quad 13\,|\,8\quad 9$$

with four runs whose borders are visualized. What is the effect of our local operations? For the element 4 only the exchange with one of the elements 5, 10, or 8 is accepted. In any of these cases, not only the number of runs stays the same but also their lengths. However, there are exceptions. The exchange of the elements 12 and 9 reduces the number of runs. The elements of the third run are larger than all but the last element of the second run. If this last element is exchanged with an element larger than the last but one element of the second run and smaller than the first element of the third run, the second and the third run melt together. The number of runs is decreased only if the element 12 fits into the position where it is placed. If n is large and the runs are long, it seems to be very unlikely that two runs melt together. All but one element of a run have to be smaller than all elements of the run that is its right neighbor (or the mirror situation). The fitness function RUN does not have the property of forcing small elements into one run and large elements into another one. Moreover, if such a melting of runs is possible, it can also be realized with jump operations. In our case, the two runs melt together if element 12 jumps to an arbitrary position and the number of runs is decreased if it jumps to one of the positions held by 14, 11, or 9.

Each element has exactly one position in each other run where it fits into that run. Such a jump is accepted, but it decreases the number of runs only if two runs melt together or if the element was in a run of length 1. In any case, an accepted jump changes the lengths of the runs. Only with jumps we can hope that a run vanishes because its length is decreased to 0. This is the reason to investigate the $(1+1)$ EA based on jumps only.

THEOREM 6. *Assuming that a run of length at least* $(3/8)n$ *and another run are not melted together, the* $(1+1)$ *EA based on jumps has an exponential expected optimization time and the success probability within less than exponentially many steps is exponentially small.*

Proof. The first search point is a random permutation on $\{1,\ldots,n\}$. The probability of having a run whose length is at least $n^{1/2}$ is exponentially small [10]. Hence, we can assume to start with a large number of short runs.

First, we investigate the $(1+1)$ EA performing one local operation per step. We start our considerations with the first point of time where the length of the longest run is at least $(5/8)n$. Because of our assumption the length of this run is at most $(3/4)n$ since big lengthenings are only possible if runs melt together. Let k be the number of runs ($k \geqslant 2$ until we have reached the optimum) and let $l_1,\ldots,l_k$ be the sorted lengths of the runs, i.e., $l_1 \geqslant l_2 \geqslant \cdots \geqslant l_k$.

Now we analyze a random jump operation. For an accepted step increasing the length of the run with length $l_1 \geqslant (5/8)n$, it is necessary to choose one of the $n-l_1$

elements outside the run and the chosen element has to jump to the unique good position in the considered run. The probability of such a step equals

$$\frac{n-l_1}{n}\cdot\frac{1}{n-1}=\frac{1}{n-1}-\frac{l_1}{n(n-1)}\leqslant(3/8)\cdot\frac{1}{n-1}.$$

For an accepted step decreasing the length of this run it is necessary to choose one of the l_1 elements from the considered run and the element has to jump to one of the $k-1$ good destinations in the $k-1$ other runs. The probability for such a step equals

$$\frac{l_1}{n}\cdot\frac{k-1}{n-1}\geqslant\frac{l_1}{n(n-1)}\geqslant(5/8)\cdot\frac{1}{n-1}.$$

For our analysis, we count only steps changing l_1 since we are proving lower bounds. If $l_1=(3/4)n$, we ask for the probability of reaching an l_1-value of $(7/8)n$ before an l_1-value of $(5/8)n$. In such a phase, the l_1-value is only changed by 1. We overestimate the probability by assuming that l_1 increases with probability $p=3/8$ and decreases with probability $1-p=5/8$. Now we are in the situation of a gambler's ruin problem (see above) where $A=B=n/8$ and $t=(1-p)/p=5/3$. The probability of reaching $(7/8)n$ before $(5/8)n$ equals

$$\frac{(5/3)^{n/8}-1}{(5/3)^{n/4}-1}=(5/3)^{-n/8}(1-\mathrm{o}(1))$$

and the expected number of trials before reaching $(7/8)n$ for the first time is exponentially large. Moreover, the success probability within the first $(5/3)^{n/10}$ trials is exponentially small.

The calculations above cover the essential ideas of the proof, but we have to generalize the result to the $(1+1)$ EA which may perform many jumps within one step. To simplify the calculations we start our investigations with the first search point where $n-2n^{1/2}\leqslant l_1\leqslant n-n^{1/2}$ and investigate phases of length $n^{3/2}$. We prove that, with overwhelming probability, there are steps shortening the long run altogether by $\Omega(n^{1/2})$ and all other steps lengthen the long run by $\mathrm{O}(n^{\varepsilon})$ for each $\varepsilon>0$. For $\varepsilon<1/2$ and n large enough, this implies that the long run never gets full length and is no longer than $n-n^{1/2}$ at the end of the phase. Hence, we can repeat the arguments. We find the optimum only if one phase does not have the properties which occur with overwhelming probability. This proves the theorem.

A step with one jump where an element from the long run jumps to a good position outside this run shortens the long run. The probability of such a step is at least $\Omega(1/n)$ (probability $1/\mathrm{e}$ for a single jump, $\Omega(1)$ for choosing an element from the long run, and at least $1/n$ for a good destination). Hence, by Chernoff bounds, the total shortenings in one phase are $\Omega(n^{1/2})$ with overwhelming probability.

In a step with r jumps, a single jump can cause a lengthening of the long run only if the jumping element does not belong to the long run (probability $\mathrm{O}(n^{-1/2})$) and its destination is at most r positions away from the correct position in the long run (probability $O(r/n)$), since only $r-1$ elements can jump away. The probability of r jumps equals $\Theta(1/(r-1)!)$ and the expected number of jumps in steps with r jumps in one phase is $\Theta(n^{3/2}r/(r-1)!)$. The expected contribution to lengthenings is $\Theta(r^2/(r-1)!)$. The probability of a contribution of $\Omega(n^{\varepsilon/2})$ is exponentially small. This holds for all r but it is sufficient to apply these rough estimates for $r \leqslant n^{\varepsilon/2}$. The probability of one step with more than $n^{\varepsilon/2}$ jumps is exponentially small. This proves the claim discussed above. □

4. The Single Source Shortest Paths Problem

The single source shortest paths problem (SSSP) is a fundamental combinatorial optimization problem. The usual description is the following one. The problem instance is described by a distance matrix $D = (d_{ij})_{1\leqslant i,j\leqslant n}$ where $d_{ij} \in \mathbb{N} \cup \{\infty\}$ is the length of the direct connection from place i to place j. The problem is to compute for the source $s := n$ and each place i a shortest path from s to i. The naive description of all shortest paths may need a storage space of $\Theta(n^2)$. Dijkstra's famous algorithm has a computation time of $\Theta(n^2)$ and computes a description of all shortest paths which needs only storage space $\Theta(n)$. For each place i the place v_i is the direct predecessor on a shortest path from s to i.

In order to consider EAs for the SSSP we use the following model of the problem. The search space consists of all $v = (v_1, \ldots, v_{n-1}) \in \{1, \ldots, n\}^{n-1}$ where $v_i \neq i$. Place v_i is considered as the direct predecessor of place i. Hence, each search point v describes a directed graph on $V = \{1, \ldots, n\}$ where $s = n$ has indegree 0 and all other nodes have indegree 1. However, there are invalid graphs which are not trees rooted at s. Figure 2 shows a valid tree and an invalid graph.

A local operation for this problem is to replace the predecessor v_i of some place $i \leqslant n-1$ by another predecessor $v_i' \in \{1, \ldots, n\} - \{i, v_i\}$. This operation changes the considered paths for all places in the subtree of place i. The number of

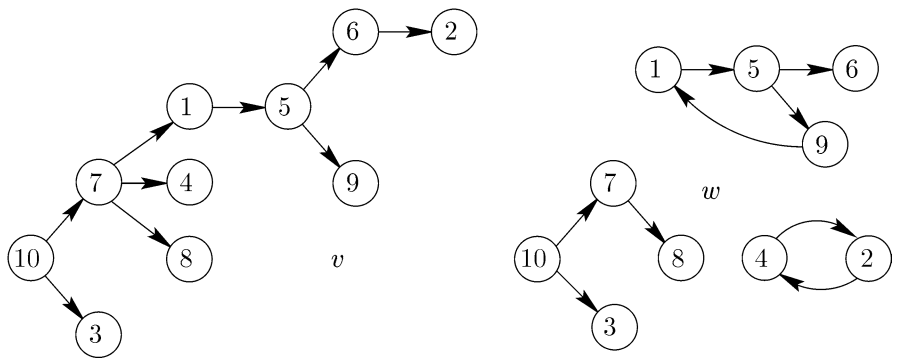

Figure 2. Illustration of the search points $v = (7, 6, 10, 7, 1, 5, 10, 7, 5)$ leading to a tree of s–i-paths and $w = (9, 4, 10, 2, 1, 5, 10, 7, 5)$ leading to an invalid graph.

different local operations equals $(n-1)(n-2)$ and a flip is a randomly chosen local operation. For a mutation step, we choose S according to a Poisson distribution with parameter $\lambda = 1$ and perform sequentially $S + 1$ flips. Again, we cannot analyze crossover-based EAs.

Finally, we have to describe an appropriate fitness function f. The first idea is to define $f(v) = \infty$ for all invalid v and $f(v)$ as the sum of the lengths of the s–i-paths in the tree $T(v)$ described by v. However, this leads to a difficult problem for all randomized search heuristics, at least for certain problem instances. Let $d_{i,i-1} < \infty$ and $d_{ij} = \infty$ if $j \neq i - 1$. Then the search point $v^* = (2, 3, \ldots, n-2, n-1, n)$ is optimal and it is the only search point where $f(v^*) < \infty$. Hence, this optimization problem is equivalent to the well-known scenario named needle in the haystack. There is a unique global optimum and all other search points have the same fitness. Then, nothing is better than random search which takes exponential time with overwhelming probability.

We can hope for better results of randomized search heuristics only if the fitness function provides more information. We may restrict the possible problem instances by considering only distance matrices where $d_{ij} \in \{1, \ldots, d^*\} \cup \{\infty\}$ for some parameter d^* (possibly depending on n). If a search point v describes for j places paths of finite length, then $f(v)$ is defined as the sum of the sum of the lengths of these paths and $(n-1-j)nd^*$. Here "non-paths" and paths of infinite length contribute nd^* to the fitness and, therefore, more than the maximal length of a path of finite length. However, we cannot distinguish between non-paths and paths of infinite length. This can be changed by assigning nd^* to paths of infinite length and n^2d^* to places i for which v does not describe an s–i-path.

We are not able to analyze the (1 + 1) EA for this fitness function. Instead of that we have analyzed a simple EA on a multi-objective fitness function since the core of the SSSP is to minimize the lengths of $n-1$ paths. Let $f(v) = (f_1(v), \ldots, f_{n-1}(v))$ where $f_i(v)$ is the length of the s–i-path if v describes such a path and $f_i(v) = \infty$ otherwise. We define a partial order on $\mathbb{R}^{n-1}$. It is $f(v) \leqslant f(v')$ iff $f_i(v) \leqslant f_i(v')$ for all $i \in \{1, \ldots, n-1\}$. The objective in multi-objective optimization is the computation or approximation of the set of Pareto optimal search points. A search point is called Pareto optimal if it is optimal, in our case minimal, with respect to the partial order described above. The theory on SSSP tells us that there is exactly one Pareto optimal fitness vector $l^* = (l_1^*, \ldots, l_{n-1}^*)$ describing the lengths of all shortest s–i-paths. There can be many search points v such that $f(v)$ is Pareto optimal. We are satisfied if we have computed one optimal search point.

Now, we have a vector-valued fitness function and a partial order on the fitness vectors. The multi-objective (1 + 1) EA chooses a search point v uniformly at random. Then it applies the mutation operator described above and accepts v' iff $f(v') \leqslant f(v)$.

There are SSSP instances with a unique optimal search point (this seems to be a typical case in applications). For these instances we can prove an $\Omega(n^2)$ bound on the expected optimization time of the multi-objective (1 + 1) EA. This can be done by the same arguments as in the proof of Theorem 1 and we do not repeat the arguments.

THEOREM 7. *The expected optimization time of the multi-objective* (1 + 1) *EA on SSSP is bounded above by* $\mathrm{O}(n^3)$.

We prove a more sophisticated bound. Let t_i be the smallest number of edges on a shortest s–i-path, $m_j := \#\{i \mid t_i = j\}$, and $T = \max\{j \mid m_j > 0\}$. Then we prove the upper bound

$$\mathrm{e}n^2 \sum_{1\leqslant j\leqslant T} (\ln m_j + 1).$$

This bound has its maximal value $\Theta(n^3)$ for $m_1 = \cdots = m_{n-1} = 1$. We also obtain the bound $\mathrm{O}(n^2 T \log n)$ which is much better than $\mathrm{O}(n^3)$ in the typical case where T is small.

Proof. The proof is based on the following simple observation. Whenever $f_i(v) = l_i^*$, we only accept search points v' where $f_i(v') = l_i^*$. Hence, we do not forget the length of shortest paths which we have found (although we may switch to another shortest path). Now we assume that we have a search point v where $f_i(v) = l_i^*$ for all i where $t_i < t$. Then we wait until this property holds for all i where $t_i \leqslant t$. For each place i where $t_i = t$ and $f_i(v) > l_i^*$ there exists a place j such that $t_j = t - 1$, j is the predecessor of i on a shortest s–i-path using t edges, and $f_j(v) = l_j^*$. Then a mutation flipping only v_i into j is accepted and leads to a search point v' where $f_i(v') = l_i^*$. The probability of such a mutation equals $1/(\mathrm{e}(n-1)(n-2))$ ($1/\mathrm{e}$ the probability of flipping exactly one position, $1/(n-1)$ the probability of flipping the correct position, and $1/(n-2)$ the probability of flipping it to the right value). If we have r such places, the success probability is at least $r/(\mathrm{e}n^2)$ and the expected waiting time is bounded above by $\mathrm{e}n^2/r$. The largest value for r is m_t and we have to consider each of the values $m_t, \ldots, 1$ at most once. Hence, the total expected time of this phase is bounded above by $\mathrm{e}n^2(1 + \frac{1}{2} + \cdots + \frac{1}{m_t}) \leqslant \mathrm{e}n^2(\ln m_t + 1)$. Since t can take the values $1, \ldots, T$ we have proved the claimed bound. □

The upper bound of Theorem 7 holds even in the case where we allow infinite distance values. Let us consider the special case where $d_{i,i-1} = 1$ and $d_{ij} = \infty$ otherwise. This is the needle-in-the-haystack scenario for the single-objective optimization problem. Theorem 7 implies an $\mathrm{O}(n^3)$ bound of the (1 + 1) EA in the multi-objective optimization problem. This bound is tight for this problem instance. As long as $v_{n-1} \neq n$ we have $f(v) = (\infty, \ldots, \infty)$. The probability of starting with $v_{n-1} = n$ equals $1/(n-1)$. In the negative case, we have to wait for a mutation where v_{n-1} is mutated into n. The probability that a local operation does this change is $1/(n-1)(n-2)$. The expected number of local changes per step equals 2. Hence, the expected time until $v_{n-1} = n$ equals $\Theta(n^2)$. Until $v_{n-1} = n$, the value of v_{n-2} does not influence the fitness vector. Therefore, we can repeat the arguments for $v_{n-2}, \dots, v_1$ and obtain an expected optimization time of $\Theta(n^3)$.

Altogether, the multi-objective (1 + 1) EA on SSSP has an expected optimization time of $O(n^3)$ and also $\Omega(n^2)$ if the solution is unique. For typical problem instances, the more sophisticated bound which follows from the proof of Theorem 7 is “much closer” to n^2 than to n^3. Hence, the multi-objective (1 + 1) EA is an efficient heuristic to solve SSSP (without beating Dijkstra’s algorithm). Our results on SSSP also show that multi-objective problems should not be transformed artificially into single-objective problems.

5. Conclusion

Robust problem solvers should also solve well-known simple optimization problems efficiently. This has been investigated for the sorting problem (maximizing the sortedness based on some measure of presortedness) and the single-source-shortest-paths problem. For four out of five fitness functions described by the best-known measures of presortedness simple EAs work very efficiently with the mutation operators used. However, the fifth measure of presortedness leads to an optimization problem which is difficult for the simple EAs investigated here.

There are instances of the SSSP problem which are difficult for single-objective optimization. The modeling of the SSSP as a multi-objective optimization problem reflects the structure of the problem and the fitness vector reveals enough information to direct the search of a simple EA. Usually, multi-objective optimization is only applied if no single-objective optimization problem covers the whole structure of the problem. Here it has been shown that a multi-objective problem model may lead to a simpler problem.

References

1. Bäck, T., Fogel, D. B. and Michalewicz, Z. (eds): *Handbook of Evolutionary Computation*, Oxford University Press, 1997.
2. Droste, S., Jansen, T. and Wegener, I.: A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with Boolean inputs, *Evolutionary Computation* **6** (1998), 185–196.
3. Droste, S., Jansen, T. and Wegener, I.: On the optimization of unimodal functions with the (1 + 1) evolutionary algorithm, in *Parallel Problem Solving from Nature – PPSN V*, Lecture Notes in Comput. Sci. 1498, 1998, pp. 13–22.
4. Droste, S., Jansen, T. and Wegener, I.: On the analysis of the (1 + 1) evolutionary algorithm, *Theoret. Comput. Sci.* **276** (2002), 51–81.
5. Feller, W.: *An Introduction to Probability Theory and its Applications*, Wiley, New York, 1971.
6. Giel, O. and Wegener, I.: Evolutionary algorithms and the maximum matching problem, in *Proc. of 20th Symp. of Theoretical Aspects of Computer Science – STACS*, Lecture Notes in Comput. Sci. 2607, 2003, pp. 415–426.
7. Graham, R. L., Knuth, D. E. and Patashnik, O.: *Concrete Mathematics*, Addison-Wesley, 1994.

8. Horn, J., Goldberg, D. E. and Deb, K.: Long path problems, in *Parallel Problem Solving from Nature – PPSN III*, Lecture Notes in Comput. Sci. 866, 1994, pp. 149–158.
9. Jansen, T. and Wegener, I.: Real royal road functions – where crossover provably is essential, in *Genetic and Evolutionary Computation Conf. – GECCO*, 2001, pp. 375–382.
10. Knuth, D. E.: *The Art of Computer Programming. Vol. 3: Searching and Sorting*, Addison-Wesley, 1973.
11. Mitchell, M., Holland, J. H. and Forrest, S.: When will a genetic algorithm outperform hill climbing, in J. Cowan, G. Tesauro and J. Alspector (eds), *Advances in Neural Information Processing Systems*, Morgan Kaufman, 1994, pp. 51–58.
12. Motwani, R. and Raghavan, P.: *Randomized Algorithms*, Cambridge University Press, 1995.
13. Orlowski, M. and Pachter, M.: An algorithm for the determination of a longest increasing subsequence in a sequence, *Comput. Math. Appl.* **17** (1989), 1073–1075.
14. Petersson, O. and Moffat, A.: A framework for adaptive sorting, *Discrete Appl. Math.* **59** (1995), 153–179.
15. Rudolph, G.: How mutations and selection solve long path problems in polynomial expected time, *Evolutionary Computation* **4** (1997), 195–205.
16. Scharnow, J., Tinnefeld, K. and Wegener, I.: Fitness landscapes based on sorting and shortest paths problems, in *Parallel Problem Solving from Nature – PPSN VII* (best paper award), Lecture Notes in Comput. Sci. 2439, 2002, pp. 54–63.
17. Wegener, I.: Theoretical aspects of evolutionary algorithms, in *Int. Colloq. on Automata, Languages, and Programming – ICALP*, Lecture Notes in Comput. Sci. 2076, 2001, pp. 64–78.