Theoretical Computer Science 410 (2009) 2455–2462

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Comparison of simple diversity mechanisms on plateau functions☆

Tobias Friedrich*, Nils Hebbinghaus, Frank Neumann

Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 24 July 2007
Received in revised form 19 June 2008
Accepted 11 August 2008
Communicated by X. Yao

Keywords:
Evolutionary algorithms
Diversity
Runtime analysis

ABSTRACT

It is widely assumed and observed in experiments that the use of diversity mechanisms in evolutionary algorithms may have a great impact on its running time. Up to now there is no rigorous analysis pointing out how different diversity mechanisms influence the runtime behavior. We consider evolutionary algorithms that differ from each other in the way they ensure diversity and point out situations where the right mechanism is crucial for the success of the algorithm. The considered evolutionary algorithms either diversify the population with respect to the search points or with respect to function values. Investigating simple plateau functions, we show that using the "right" diversity strategy makes the difference between an exponential and a polynomial runtime. Later on, we examine how the drawback of the "wrong" diversity mechanism can be compensated by increasing the population size.

1. Introduction

Diversity mechanisms should prevent the optimization process of an evolutionary algorithm (EA) from getting stuck by ensuring that the population consists at each time step of a set of individuals with different properties. It has been observed in numerous experiments [1,19] that the right use of a diversity strategy can play a key role for the success of an EA. As it is important to understand in practice successful algorithms also from a theoretical point of view, it is desirable to strengthen the theoretical understanding of diversity mechanisms. The aim of this paper is to show how the use of different diversity mechanism may influence the behavior of an EA. To achieve this goal we examine some simple EAs using different methods to ensure diversity and analyze them with respect to their runtime behavior.

The rigorous analysis of evolutionary algorithms with respect to their runtime behavior has been started on artificial pseudo-Boolean functions. The main aim of such investigation is to increase the understanding how evolutionary algorithms work. The first result has been obtained by Mühlenbein [12] who proved that the well-known (1+1) EA is able to optimize the function ONEMAX within an expected number of $O(n \log n)$ iterations. Rudolph [15] has shown that the (1+1) EA can solve some Long Path functions in expected polynomial time. Later Droste, Jansen, and Wegener [4] pointed out Long Path functions where the (1+1) EA has an exponential optimization time. The (1+1) EA has been subject of different investigations on pseudo-Boolean functions [10,7] and on some of the best-known combinatorial optimization problems [6,14,21]. It seems to be the "standard" algorithm for investigating the behavior of EAs on new (combinatorial) problems. In the case of crossover-based EAs that work with a population size of at least 2 there are only a few results which show that the use of a crossover operator can speed up computations drastically [11,18,3]. The influence of the population size itself has also been investigated by rigorous runtime analyses [17,20].

☆ A conference version appeared in the Genetic and Evolutionary Computation Conference – GECCO 2007 [T. Friedrich, N. Hebbinghaus, F. Neumann, Rigorous analyses of simple diversity mechanisms, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO'07, ACM Press, 2007, pp. 1219–1225].

* Corresponding author. Tel.: +49 681 9325 620; fax: +49 681 9325 199.
E-mail address: tfried@mpi-inf.mpg.de (T. Friedrich).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.08.021

Considering selection methods with respect to the runtime behavior has been started only recently. Jägerküpper and Storch [8,9] have compared comma and plus strategies and pointed out situations where switching from one strategy to the other may change the runtime from polynomial to exponential and vice versa. Often selection methods consist of a particular mechanism to ensure diversity of the individuals in the population. The aim of this paper is to study the influence of two diversity strategies with respect to the runtime of an evolutionary algorithm. A large diversity of the individuals in the population is needed to explore different regions of the considered search space. Usually, there is a trade-off between exploring different regions of the search space and converging quickly to optimal solutions.

We consider simple mutation-based EAs that use different diversity measures and a constant population size. The first idea is to consider a diversity strategy that tries to avoid duplicates in each generation. Storch [17] has examined the influence of the choice of the population size together with this strategy but did not consider the influence of the strategy itself. We focus on the diversity mechanism rather than on the effect of increasing the population size. In our second algorithm the individuals should differ with respect to their function values which is motivated by the assumption that individuals with the same fitness value have similar properties. In a first step, we point out a situation where ensuring diversity of fitness values leads to an exponential optimization time while the algorithm just avoiding duplicates has a polynomial runtime. Afterwards, we examine situations where the diversity with respect to the fitness values ensures a polynomial runtime. In contrast to this the algorithm just avoiding duplicates has an exponential optimization time. After having obtained these results, we consider the effect of increasing the population size in our algorithms. We are in particular interested in the population size needed to turn the exponential runtimes that are the effect of the "wrong" diversity mechanism into polynomial ones by using larger population sizes.

The outline of the paper is as follows. In Section 2, we introduce the algorithms and diversity strategies that are subject to our analysis. In Section 3, we compare the different diversity strategies for our algorithms using populations of constant size. The effect of larger populations is discussed in Section 4. Finally, we finish with some conclusions. A conference version of this paper appeared in [5].

2. Algorithms

Our aim is to study the impact of two simple diversity mechanisms with respect to the runtime behavior. The first algorithm tries to avoid duplicates by diversifying with respect to the search points. The second one is diversifying with respect to the function values. In our opinion, these are the two simplest strategies that can be considered. As this is the first paper considering in particular the use of diversity by rigorous runtime analyses, we start by considering simple algorithms. In particular analyzing the effect of a crossover operator in an evolutionary algorithm seems to be a hard task. Therefore, we investigate simple EAs that produce in each iteration one single offspring by mutation. We hope that the insight we gain by considering these algorithms leads to a better understanding of diversification such that more complicated strategies and algorithms might also be easier to understand.

We examine a simple $(\mu+1)$ evolutionary algorithm called $(\mu+1)$ EA_d (see Algorithm 1) which just tries to avoid duplicates. The algorithm has already been used by Storch [17] to study to impact of the population size on the runtime behavior.

Algorithm 1. $(\mu+1)$ EA_d

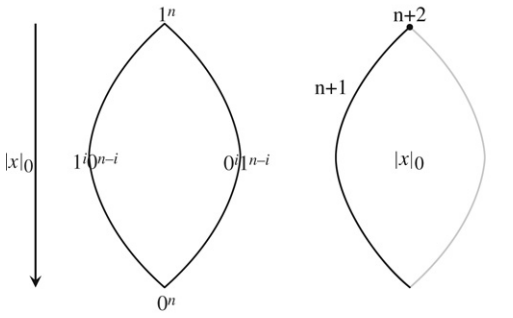
(1) Choose μ individuals $x_i \in \{0,1\}^n$ $(1 \leq i \leq \mu)$ uniformly at random.
(2) $P \leftarrow \{x_1, \ldots, x_\mu\}$.
(3) Repeat
  (a) Choose z from the population P uniformly at random.
  (b) Create z' by flipping each bit of z independently with probability $1/n$.
  (c) If $z' \notin P$, then create the new population P by introducing z' into P and deleting an individual from $P \cup \{z'\}$ with the lowest f-value uniformly at random.

The algorithm starts by choosing μ individuals uniformly at random from the considered search space $\{0,1\}^n$. In each iteration, one of these individuals is selected for mutation. Afterwards, a new parent population is constituted by choosing two individuals from the parents and the offspring. If an individual z' created by mutation is equal to any individual in the population P, P remains unchanged. Otherwise z' is included and an individual with the lowest fitness value is chosen uniformly at random and deleted. Another variant of the $(\mu+1)$-EA has been examined by Witt [20]. This algorithm is similar to the one given above but does not prevent the introduction of duplicates into the population.

We compare the $(\mu+1)$ EA_d with the algorithm called $(\mu+1)$ EA_f that uses a much stronger diversity strategy. The $(\mu+1)$ EA_f (see Algorithm 2) tries to diversify the individuals with respect to their function values. This diversity measure relies on the assumption that individuals with the same function value have some properties in common such that a population consisting of individuals with different function values gives a good sample of the search space.

Algorithm 2. $(\mu+1)$ EA_f

(1) Choose μ individuals $x_i \in \{0,1\}^n$ $(1 \leq i \leq \mu)$ uniformly at random.
(2) $P \leftarrow \{x_1, \ldots, x_\mu\}$.



(a) Search space. (b) Function values.

Fig. 1. An illustration of the explored function *PL*.

(3) Repeat
   (a) Choose z from the population P uniformly at random.
   (b) Create z' by flipping each bit of z independently with probability $1/n$.
   (c) If $f(z') \in \{f(x), x \in P\}$, then create the new population P by deleting an individual with f-value $f(z')$ from P and insert z'; otherwise create the new population P by deleting an individual from $P \cup \{z'\}$ with the lowest f-value uniformly at random.

We have not defined any stopping criteria for our algorithms. We consider the algorithms as infinite stochastic processes. Considering randomized search heuristics with respect to their runtime, a common measure is to count the number of fitness evaluations until an optimal one has been found. This is called the optimization time of the algorithm. Often the expectation of this value is analyzed and called the expected optimization time.

3. Constant population size

We compare the two diversity mechanisms introduced in Section 2 for a constant population size. All our investigations in this section consider the case $\mu = 2$. We compare the two diversity mechanisms introduced in the previous section on two plateau functions. Plateaus are regions in the search space where all points have the same f-value. Consider a function $f: \{0,1\}^n \to \mathbb{R}$ whose image of the domain $\{0,1\}^n$ is of size $V = |f(\{0,1\}^n)|$. Since often the number of different fitness values V is polynomial, such a function has an exponential number of solutions with the same f-value. However, such functions are optimized easily with a randomized search heuristic if there is a better Hamming neighbor for each non-optimal solution since this implies that an improvement can be reached by flipping a single bit of a non-optimal solution. Otherwise, if this does not hold, the search may get much harder for evolutionary algorithms. One example (from [4]) for this is the degenerated case of the characteristic function NEEDLE, which has fitness value 0 for all solutions apart from a single point x at which it has fitness value 1. The behavior of (1+1) EAs on plateaus of different structures has been studied in [10] by a rigorous runtime analysis. In the case of combinatorial optimization problems, it has been shown that evolutionary algorithms have to cope with plateaus of constant fitness for the Eulerian cycle problem [2,13] and the computation of maximum matchings [6].

3.1. (2+1) EA_d beats (2+1) EA_f

We show that there are functions on which the optimization time of the (2+1) EA_d is a small polynomial while the (2+1) EA_f is exponential with high probability. For this, we consider the following function *PL*, which is similar to the function SPC_n introduced by Jansen and Wegener [10]. Let $|x|_1$ and $|x|_0$ denote the number of ones and the number of zeros in a bitstring x, respectively. The function *PL* is defined as

$$PL(x) := \begin{cases} |x|_0 & : \ x \notin \{1^i0^{n-i}, 0 < i \leq n\} \\ n+1 & : \ x \in \{1^i0^{n-i}, 0 < i < n\} \\ n+2 & : \ x = 1^n. \end{cases}$$

We denote by $SP := \{1^i0^{n-i}, 0 < i < n\}$ the set of search points that constitute the plateau of fitness $n+1$. Fig. 1 shows an illustration of this function. We first prove that the (2+1) EA_d is efficient on *PL*. Our analysis relies on analyzing the random walks that two individuals perform on the plateau instead of one as examined in [10].

Theorem 1. *The expected optimization time of the (2+1) EA_d on PL is $O(n^3)$.*

Proof. The algorithm always keeps two different solutions that have been produced up to now in the population. We first consider the number of steps until both solutions of the population are contained in SP. Note that in every mutation step there is a positive probability to produce a solution that is contained in SP. We neglect this probability that only decreases the expected time until an element of SP is detected. We can assume that $|x|_1 \leq |y|_1$ and that both solutions are not contained in SP. As x is chosen with probability $1/2$ for mutation, the probability of producing a solution z with $|z|_0 > |x|_0$ in the next step is at least $|x|_1/(2en)$. This solution will be introduced into the population P and a solution with a largest number of ones in P will be removed. Summing up over the different number of ones, the expected time to introduce the search point 0^n in P is $O(n \log n)$ as long as no solution of SP has been produced before. A solution of SP can be produced from 0^n by flipping the first bit. This happens after an expected number of $O(n)$ steps. Let x be this solution. The expected waiting time to produce a solution $z \in SP$ with $z \neq x$ is $O(n)$ as only one specific bit in x has to flip. Hence, after an expected number of $O(n \log n)$ steps both solutions of P have fitness value at least $n+1$. As long as the optimal search point 1^n has not been obtained x and y are different and contained in SP.

We now use the backward analysis paradigm for randomized algorithms which has been popularized by Seidel [16] to show that within $O(n^3)$ steps one of the two individuals of the (2+1) EA_d reaches the optimal search point. For every newly generated search point we consider the sequence of all its ancestors. At this point, we interpret step 3.(c) of Algorithm 1 slightly different. Namely, instead of completely ignoring mutations of all z' which are already in the current population P, we now say that we also include such mutations in the population, but remove the old (identical) individual z' to avoid duplicates. This modification behaves exactly as the original algorithm, but now each parent of a search point has chosen an offspring of a certain Hamming distance with exactly the same probability as the offspring in a (1+1) EA is chosen. Therefore, the sequence of all ancestors of a point can also be seen as the sequence of solutions generated (and accepted) by an (1+1) EA. The expected length of this sequence is half of the expected length of such a sequence generated by a (1+1) EA since for every ancestor the probability to be chosen for mutation is $1/2$ in every mutation step. By [10] this shows that after an expected number $O(n^3)$ mutation steps of the (2+1) EA_d, the optimal search point is found. □

In contrast to the (2+1) EA_d, the (2+1) EA_f diversifies with respect to the function values. In this case only one single search point may perform a random walk on the plateau. In the case of the function PL, the second individual in the population has a great effect on the success of this random walk. In particular, we show that the (2+1) EA_f is not efficient on PL. The main reason for this is that the other individual in the population produces faster new solutions of SP that have a small Hamming distance to the search point 0^n than the random walk has reached the optimal solution 1^n.

Theorem 2. *The optimization time of the (2+1) EA_f on PL is $2^{\Omega(n^{1/24})}$ with probability $1 - e^{-\Omega(n^{1/24})}$.*

Proof. The population consists at each time step t of 2 individuals with the first and the second best fitness values that have been produced during the first t steps. Our aim is to show that a population including solutions with fitness values $n+1$ and n are constructed before the optimal search point 1^n is reached.

The initial solutions x and y consist with probability $1 - e^{-\Omega(n)}$ of at most $2n/3$ ones using Chernoff bounds. If no solution of SP has been obtained before, only solutions with at most $\max\{|x|_1, |y|_1\}$ ones are accepted. Hence, as long as no solution of SP has been obtained the probability to produce the optimal search point is upper bounded by $n^{-n/3}$. This implies that the number of iterations needed to produce an optimal search point without creating a solution of SP before is $n^{\Omega(n)}$ with probability $1 - n^{-\Omega(n)}$. In the following, we assume that a solution of SP has been obtained within a phase of 2^{cn} steps, where $c > 0$ is an appropriate small constant. The first solution of SP obtained within this phase has with probability $1 - e^{-\Omega(n)}$ at most $3n/4$ ones as the probability of flipping at least $n/12$ bits in a single mutation step is $e^{-\Omega(n)}$.

We now consider a phase of $n^{3/2}$ steps of the algorithm after for the first time a solution in SP has been produced. Roughly speaking, we will show that within such a phase the random walk of the solution $y \in SP$ reaches the optimal search point 1^n only with a small probability while at the same time the other solution x quickly becomes $x = 0^n$. Such a solution x produces with high probability a descendant on SP by a single bit flip within this phase which implies that the random walk has to start again from a solution that has a large (Hamming) distance to the optimal search point.

Let $y = 1^i 0^{n-i}$ be the solution on SP. We call a step relevant if and only if it produces a solution $z \in SP$ with $z \neq y$. To achieve this the bit y_i or y_{i+1} has to flip. Therefore, the probability of not having a relevant step is at least $1 - 2/n$ and the expected number of non-relevant steps during this phase is at least $(1 - 2/n)\, n^{3/2} = n^{3/2} - 2n^{1/2}$. There are at least

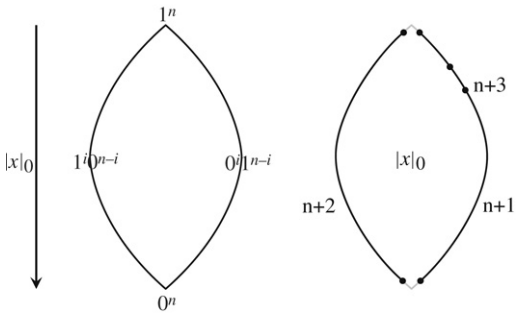
$$(1 - n^{-2/3}) \cdot (n^{3/2} - 2n^{1/2}) \geq n^{3/2} - 2n^{5/6}$$

non-relevant steps with probability

$$1 - e^{\left(-n^{3/2} \cdot \frac{n^{-4/3}}{2}\right)} = 1 - e^{-\Omega(n^{1/6})}$$

using Chernoff bounds.

The probability that at least $n^{1/12}$ bits flip in a single accepted mutation step is at most $n^{-n^{1/12}}$. Such an event happens in the phase of $n^{3/2}$ steps only with probability at most $n^{3/2-n^{1/12}} = n^{-\Omega(n^{1/12})}$. Therefore, within this phase the Hamming distance to the optimal search point decreases by at most $2n^{5/6}n^{1/12} = 2n^{11/12}$ and an optimal search point has not been obtained with probability $1 - e^{-\Omega(n^{1/12})}$.



(a) Search space. (b) Function values.

Fig. 2. An illustration of the explored function *PLS*.

In the following we show that after $n^{3/2}$ steps, the solution 0^n is inserted into the population. We consider in each step the solution x with the largest number of zeros in the population P. As an optimal search point will not be produced within $n^{3/2}$ steps with probability $1 - e^{-\Omega(n^{1/12})}$ such a solution will never be removed from P in this phase. Assume $|x|_1 = k$. Then the probability of producing in the next step a solution z with $|z|_0 > |x|_0$ is at least $k/(2en)$. Summing up over the different values of k, the search point 0^n is included into P after an expected number of at most $en \log n$ steps. After an expected number of $O(n)$ steps a solution with fitness value $n + 1$ is included afterwards. Hence, after an expected number of $2e\,n \log n$ steps $P = \{x, y\}$ where $f(x) = n$ and $f(y) = n+1$, i. e., after $4e\,n \log n$ steps this happens with probability at least $1/2$. The probability of not having obtained these solutions within $n^{3/2}$ steps is upper bounded by $2^{-(n^{1/2}/(4e \log n))} = e^{-\Omega(n^{1/4})}$ considering $n^{1/2}/(4e \log n)$ phases of length $4e\,n \log n$.

The probability to produce from x a search point z with fitness $n+1$ is at least $1/(en)$ as this can be achieved by flipping the first bit of x. The probability to select z in the next mutation step is $1/2$. Using Markov's inequality the probability that such a z has not been produced during $4en$ steps is bounded above by $1/2$ and the probability that this has not happened during r $(4en)$ steps is $(1/2)^r$. Choosing $r = n^{1/2}$ such a z is produced within $n^{3/2}$ steps with probability $1 - 2^{-\Omega(n^{1/2})}$. We already know that, with probability $1 - e^{-\Omega(n^{1/12})}$ a phase of $n^{3/2}$ steps does not lead to an optimal solution. Considering $2^{\Omega(n^{1/24})}$ steps the probability of obtaining an optimal solution is still upper bounded by $e^{-\Omega(n^{1/24})}$ which proves the theorem. □

Our analyses for the function *PL* have pointed out how to analyze the random walk of more than one individual on a plateau by a backward analysis. The success of a random walk of a single search point on such a plateau may be prevented by producing individuals at the "wrong" part of a plateau. This is in particular the case for the (2+1) EA_f which diversifies the population with respect to their function values.

3.2. (2+1) EA_f beats (2+1) EA_d

We now consider an example where diversifying with respect to the function values reduces the runtime significantly compared with a mechanism that just avoids duplicates. The idea is to construct a function with two plateaus of different fitnesses.

We investigate the function *PLS* defined as

$$PLS(x) := \begin{cases} n+1 & : \ x \in \{0^i 1^{n-i},\ 1 < i < n-1\} \setminus \{0^{3n/4}1^{n/4}\} \\ n+2 & : \ x \in \{1^i 0^{n-i},\ 1 < i < n-1\} \\ n+3 & : \ x = 0^{3n/4}1^{n/4} \\ |x|_0 & : \ \text{otherwise} \end{cases}$$

and define

$$SP_1 := \{1^i 0^{n-i},\ 1 < i < n-1\},$$
$$SP_2 := \{0^i 1^{n-i},\ 1 < i < n-1\} \setminus \{0^{3n/4}1^{n/4}\}.$$

SP_1 where all search points have fitness $n + 2$ may mislead the search. All search points in SP_1 have a large Hamming distance of at least $n^{1/4}$ to the optimal one $0^{3n/4}1^{n/4}$. Therefore, it is unlikely that steps mutating individuals of SP_1 produce optimal solutions. In addition, SP_2 serves as a royal road for the fitness diversifying algorithm (2+1) EA_f as it always keeps two individuals of different fitnesses during the optimization process. Fig. 2 shows an illustration of the function *PLS*.

Theorem 3. *The optimization time of the (2+1) EA_d on PLS is at least $n^{n/4}$ with probability $1/2 - o(1)$.*

Proof. In the initialization step of the (2+1) EA_d, two solutions $x, y \in \{0,1\}^n$ are produced that fulfill $\max\{|x|_1, |y|_1\} \leq \frac{2}{3}n$ with probability $1 - e^{-\Omega(n)}$ as already observed in Theorem 2. Again only solutions with at most $\max\{|x|_1, |y|_1\}$ ones are accepted.

To reach the optimal search point $0^{3n/4}1^{n/4}$, at least $n/12$ bits have to swap at once. The probability for this to happen is $e^{-\Omega(n)}$ as steps flipping $\Theta(n)$ bits are exponentially unlikely. The expected number of steps until the first individual reaches the search point 0^n (neglecting the possibility of reaching one of the plateaus with f-value greater than n before this) is $O(n \log n)$ as shown in the proof of Theorem 2. After the search point 0^n is found by the (2+1) EA_d, the expected time until a search point from one of the plateaus with f-value greater than n is inserted into the population is $O(n^2)$ because for this event the hamming distance of 2 to the plateaus has to be covered (with probability at least $1/(en^2)$). The probability that the first search point $z \in SP_1 \cup SP_2$ that has been produced is in SP_1 is $\text{Prob}(z \in SP_1) = \text{Prob}(z \in SP_2) = 1/2$.

We consider a phase consisting of the next $n^{3/2}$ steps. Let us assume (as a worst case) that the second best individual (the one that not entered SP_1 as first individual) reaches the plateau SP_2 immediately after the first search point of SP_1 has been found by the (2+1) EA_d. Following the line of argument in the proof of Theorem 2 (for an individual on the plateau SP), the second best individual does not reach the optimal search point $0^{3n/4}1^{n/4}$ with probability $1 - e^{-\Omega(n^{1/12})}$. The probability that in $n^{3/2}$ steps the individual in SP_1 has produced another individual is at least $(1 - \frac{1}{n})^{n^{3/2}} = 1 - e^{-\Omega(n^{1/2})}$. This individual solution will be accepted since the fitness value $n+2$ is greater than the so far second best fitness value. This shows that with probability $1/2 - o(1)$ the (2+1) EA_d produces a population where both individuals are contained in SP_1 before obtaining the optimum.

Afterwards, only search points of $SP_1 \cup \{0^{3n/4}1^{n/4}\}$ are accepted. Each search point of SP_1 has Hamming distance at least $n/4$ to the optimal search point. Hence, the expected time to produce from a solution in SP_1 the optimal one is lower bounded by $n^{n/4}$. This implies that the optimization time of the (2+1) EA_d is at least $n^{n/4}$ with probability $1/2 - o(1)$. □

The (2+1) EA_f cannot end up in the "dead-end" of having both individuals on the same plateau $n + 2$. For *PLS* it indeed outperforms significantly the (2+1) EA_d as shown in the following theorem.

Theorem 4. *The expected optimization time of the (2+1) EA_f on PLS is $O(n^3)$.*

Proof. Analogue to the proof of Theorem 3, $|x|_1, |y|_1 \leq \frac{2}{3}n$ holds with probability $1 - e^{-\Omega(n)}$ for the initial population $\{x, y\}$ of the (2+1) EA_f. Afterwards, no solution z with $|z|_1 \geq \max\{|x|_1, |y|_1\}$ is accepted until a solution of $SP_1 \cup SP_2$ has been produced. Assuming that the population does not consist of individuals with fitness values greater than n, the search point 0^n is included in the population after an expected number of $O(n \log n)$ steps as shown in the proof of Theorem 2. The expected time to produce from this search point a solution of SP_1 (or SP_2) is $O(n^2)$ as this can be achieved in both cases by flipping two specific bits. Therefore, the population consists after an expected number of $O(n^2)$ steps of two individuals x and y where $x \in SP_1$ and $y \in SP_2$.

The probability to produce from a solution $x \in SP_1$ a solution $y \in SP_2$ in the next mutation step is $O(1/n^4)$ as the Hamming distance between such solutions is at least 4. Using Markov's inequality this does not happen within a phase of $O(n^3)$ steps with probability $1 - O(1/n)$. Hence, the random walks on the two plateaus are completely independent of each other with probability $1 - O(1/n)$. Using the arguments of Jansen and Wegener [10] for the function *PL* after a phase of cn^3 steps, c an appropriate constant, the (2+1) EA_f has produced a solution $z = 0^j1^{n-j}$ where $3n/4 \leq j < n$ with probability bounded below by a constant α_1. Consider the first time such a z has been obtained. The probability that it has been obtained by a mutation step flipping a single bit is at least $(1 - 1/n)^{n-1} \geq 1/e$. Hence, the probability that the optimal solution has been produced during the considered cn^3 steps is at least $(1 - O(1/n)) \cdot \alpha_1/e$. The arguments can be repeated by considering $c'n^3$ steps, c' an appropriate constant, to produce from a solution $y = 0^j1^{n-j}$, $3n/4 < j < n$, a solution $z = 0^k1^{n-k}$, $1 < k \leq 3n/4$ with probability α_2. Similarly the probability of having obtained the optimum is at least $(1 - O(1/n)) \cdot \alpha_2/e$ within this phase . This implies that the expected time to produce the optimal solution is upper bounded by

$$\frac{1}{1 - O(1/n)} \cdot \frac{\max\{c, c'\}}{\min\{\alpha_1, \alpha_2\}} \cdot en^3 = O(n^3). \quad \square$$

Our analyses for the function *PLS* show that diversifying the population with respect to the function values may prevent an evolutionary algorithm from getting stuck in a local optimum. We assume that the (μ+1) EA_d is also not able to optimize *PLS* efficiently for larger values of μ not greater than $n - 3$ as the defined local optimal plateau consists of $n - 3$ different search points that all have a large Hamming distance to the optimal one.

Similar results given for the (μ+1) EA_d in this section also hold for the variant of the ($\mu + 1$)-EA examined by Witt [20]. For *PL* the results given in [20] can be transferred and for the lower bound on *PLS* it is not too hard to adapt our ideas. This implies that using the diversity mechanism of the (μ+1) EA_d does not change the runtime behavior for the examined problems compared with an algorithm not using any diversity mechanism.

4. Larger populations

Theorems 2 and 3 showed exponential runtimes of the (2+1) EA_f and the (2+1) EA_d on *PL* and *PLS*, respectively. In this section, we examine larger population sizes and prove that the algorithms achieve polynomial runtimes for sufficiently large

populations μ. This complements similar results by Storch [17] who has investigated the choice of μ for an algorithm just avoiding duplicates.

Theorem 5. *The expected optimization time of the (μ+1) EA$_f$ with $\mu = n - k$ ($0 \leq k < n$) on PL is $O(\mu n^{k+2})$.*

Proof. In the initialization step it cannot be guaranteed that all individuals have pairwise different fitness values. Therefore, we show that after a phase of $O(\mu n \log n)$ mutation steps there are μ different fitness values present in the current population. We assume the worst case that all μ individuals produced in the initialization step share the same fitness value. After d different fitness values are included in the current population, there is at least one individual x in the population such that $m(x) := \min\{|x|_0, |x|_1\} \geq (n-d)/2$ and there is no individual in the population with $m(x) - 1$ 0–bits or 1–bits. Thus, the probability that an individual with this fitness value is produced in the next mutation step is at least $(n-d)/(2\mu n)$. The factor μ in the denominator is due to the selection of the individual in the population that is chosen for the mutation step. Hence, the expected time until all μ individuals in the current population have pairwise different fitness values is at most

$$\sum_{d=1}^{\mu-1} \mu \tfrac{2n}{n-k} \leq 2\mu n \sum_{d=1}^{n-1} \tfrac{1}{k} = 2\mu n \log n.$$

Since there are $n+2$ different fitness values and the $n+2$ is the optimal fitness value, the least fitness value in the population is at most $n+2-\mu = k+2$. Thus, there is an individual x in the current population with $|x|_0 \leq k+2$. The probability that this individual is mutated in the next step into the optimal search point 1^n is at least $\frac{1}{\mu}\frac{1}{n}^{k+2}(1-\frac{1}{n})^{n-k-2} \geq 1/(e\mu n^{k+2})$. Therefore, the expected time (from the moment where all fitness values in the current population are pairwise different) until the optimal search point is determined is $O(\mu n^{k+2})$. This proves the theorem. □

Theorem 6. *The expected optimization time of the (μ+1) EA$_d$ with $\mu = 2n - k$ ($6 \leq k < n$) on PLS is $O(\mu n^{\max(2, \lceil k/2 \rceil - 3)})$.*

Proof. As a first step, we examine the phase until the first search point on one of the plateaus is found. The reasoning for this phase is similar to the one in the proof of Theorem 3, but here we have to deal with a larger population size. The expected number of steps till the first individual reaches the search point 0^n (neglecting the possibility of reaching one of the plateaus with f-value greater than n before this) is $O(\mu n^2)$ since the probability to choose from the population the individual with the largest $|x|_0$-value is $1/\mu$, the probability that its mutation increases the f-value is at least $1/(en)$, and there are $O(n)$ such steps needed. After the search point 0^n is detected by the (μ+1) EA$_d$ the expected time until a search point from one of the plateaus with f-value greater than n is determined is $O(\mu n^2)$ because for this event the search point 0^n has to be chosen for mutation (with probability $1/\mu$) and the hamming distance of 2 to the plateaus has to be covered (with probability at least $1/(en^2)$).

We now assume that $f(x) > n$ holds for at least one individual x and that there is another individual x' with $f(x') \leq n$. With probability $1/\mu$ we choose a solution x on the plateaus whose adjacent neighbor z (i. e., Hamming distance 1) on the plateau is not yet in the population. The mutation of x yields z with probability at least $1/(en)$, which will be accepted since $f(z) > n$. After $O(\mu n^2)$ steps, there are no individuals left with f-value smaller or equal n since there are only $|\{x : f(x) > n\}| = 2n \leq \mu$ elements on the two plateaus.

If the optimal search point has not been found yet, we now know that there is at least one individual on the plateau $n+1$ with Hamming distance $\lceil k/2 \rceil - 3$ to the optima. This individual is chosen with probability $1/\mu$ and jumps directly to the optimum with probability $\frac{1}{e}n^{-(\lceil k/2 \rceil - 3)}$. Therefore, after an expected number of $O(\mu n^{\lceil k/2 \rceil - 3})$ steps, the optimal search point is found. □

5. Conclusions

Ensuring diversity in the population of an evolutionary algorithm is important to prevent the algorithm from getting stuck in the optimization process. For the first time the use of simple diversity mechanisms used in selection procedures have been analyzed with respect to their influence on the runtime behavior. The strategies considered to ensure diversity are really simple ones and differ from each other by either diversifying with respect to the decision space or objective space. By investigating two plateau functions, we have shown that this may make the difference between a polynomial and an exponential optimization time.

There are a lot of open problems related to the topic investigated in this paper. In particular, other more complicated diversity mechanisms should be analyzed where the decision whether an individual is included into the population is not strict but has a probability that depends on the individual itself and its relation to other individuals in the population. Another interesting task is to investigate the use of diversifying methods for combinatorial optimization problems. Up to now most of the analyses on such problems consider the (1+1) EA for pseudo-Boolean functions, but there are no analyses related to the use of diversity for such problems.

References

[1] N. Chaiyaratana, T. Piroonratana, N. Sangkawelert, Effects of diversity control in single-objective and multi-objective genetic algorithms, Journal of Heuristics 13 (1) (2007) 1–34.
[2] B. Doerr, N. Hebbinghaus, F. Neumann, Speeding up evolutionary algorithms through unsymmetric mutation operators, Evolutionary Computation 15 (4) (2007) 401–410.
[3] B. Doerr, E. Happ, C. Klein, Crossover can provably be useful in evolutionary computation, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO'08, ACM Press, 2008, pp. 539–546.
[4] S. Droste, T. Jansen, I. Wegener, On the analysis of the $(1 + 1)$ evolutionary algorithm, Theoretical Computer Science 276 (2002) 51–81.
[5] T. Friedrich, N. Hebbinghaus, F. Neumann, Rigorous analyses of simple diversity mechanisms, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO'07, ACM Press, 2007, pp. 1219–1225.
[6] O. Giel, I. Wegener, Evolutionary algorithms and the maximum matching problem, in: Proc. of Symposium on Theoretical Aspects of Computer Science, STACS'03, in: LNCS, vol. 2607, 2003, pp. 415–426.
[7] J. He, X. Yao, Drift analysis and average time complexity of evolutionary algorithms, Artificial Intelligence 127 (2001) 57–85.
[8] J. Jägersküpper, T. Storch, How comma selection helps with the escape from local optima, in: Proc. of Parallel Problem Solving from Nature, PPSN'06, in: LNCS, vol. 4193, 2006, pp. 52–61.
[9] J. Jägersküpper, T. Storch, When the plus strategy outperforms the comma strategy — and when not, in: Proc. of Symposium on Foundations of Computational Intelligence, FOCI'07, IEEE Press, 2007, pp. 25–32.
[10] T. Jansen, I. Wegener, Evolutionary algorithms — How to cope with plateaus of constant fitness and when to reject strings of the same fitness, IEEE Trans. Evolutionary Computation 5 (6) (2001) 589–599.
[11] T. Jansen, I. Wegener, The analysis of evolutionary algorithms — A proof that crossover really can help, Algorithmica 34 (1) (2002) 47–66.
[12] H. Mühlenbein, How genetic algorithms really work: Mutation and hillclimbing, in: Proc. of Parallel Problem Solving from Nature, PPSN'92, Elsevier, 1992, pp. 15–26.
[13] F. Neumann, Expected runtimes of evolutionary algorithms for the Eulerian cycle problem, Computers and Operations Research 35 (9) (2008) 2750–2759.
[14] F. Neumann, I. Wegener, Randomized local search, evolutionary algorithms, and the minimum spanning tree problem, Theoretical Computer Science 378 (1) (2007) 32–40.
[15] G. Rudolph, How mutation and selection solve long path problems in polynomial expected time, Evolutionary Computation 4 (2) (1996) 195–205.
[16] R. Seidel, Backward analysis of randomized geometric algorithms, in: J. Pach (Ed.), New Trends in Computational Geometry, Springer-Verlag, Berlin, 1993, pp. 37–67.
[17] T. Storch, On the choice of the population size, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO'04, in: LNCS, vol. 3102, 2004, pp. 748–760.
[18] D. Sudholt, Crossover is provably essential for the Ising model on trees, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO'05, ACM Press, 2005, pp. 1161–1167.
[19] R.K. Ursem, Diversity-guided evolutionary algorithms, in: Proc. of Parallel Problem Solving from Nature, PPSN'02, in: LNCS, vol. 2439, 2002, pp. 462–474.
[20] C. Witt, An analysis of the $(\mu + 1)$ EA on simple pseudo-boolean functions, Evolutionary Computation 14 (1) (2006) 65–86.
[21] C. Witt, Worst-case and average-case approximations by simple randomized search heuristics, in: Proc. of Symposium on Theoretical Aspects of Computer Science, STACS'05, in: LNCS, vol. 3404, 2005, pp. 44–56.