On the Analysis of Dynamic Restart Strategies for Evolutionary Algorithms

Thomas Jansen

George Mason University, Fairfax, VA 22030, USA
tjansen@gmu.edu

Abstract. Since evolutionary algorithms make heavy use of randomness it is typically the case that they succeed only with some probability. In cases of failure often the algorithm is restarted. Of course, it is desirable that the point of time when the current run is considered to be a failure and therefore the algorithm is stopped and restarted is determined by the algorithm itself rather than by the user. Here, very simple restart strategies that are non-adaptive are compared on a number of examples with different properties. Circumstances under which specific types of dynamic restart strategies should be applied are described and the potential loss by choosing an inadequate restart strategy is estimated.

1 Introduction

Evolutionary algorithms (EAs) are randomized search heuristics applicable to a wide range of problems. We concentrate on optimization here though we do not think that the considerations presented are limited to that case. Due to the random elements employed in EAs one typically observes behavior that varies from run to run. In order to increase the probability of finding a good or even optimal solution of the considered optimization problem, it is common practice to do restarts. Then the choice of the point of time when the EA is stopped and restarted is crucial. In principle, one can distinguish three different classes of ways to choose a restart strategy just like for other parameters in EAs, e. g., mutation probability or population size [1]. The first class is a static setting where a certain number of steps is fixed in advance. Then one can discuss whether it is better to spend a lot of time in one long run or in multiple short runs [10]. Unlike other parameters such a static decision is unusual for restarting. The second class comprises dynamic restart strategies. Here, the point of time for restarts is determined by some fixed schedule that depends on the number of steps counted over all runs or, equivalently, on the number of steps in the current run and the sum of number of steps in all previous runs. In spite of the increased flexibility compared to static choices dynamic restart strategies are seldom used in EAs. A practical example proving the usefulness of dynamic, yet non-adaptive, strategies are cooling schedules used for simulated annealing [9]. What is common practice for determining the point of time for restarts are strategies from the third class, namely adaptive restart strategies. Here, the restart strategy may depend on

J.J. Merelo Guervós et al. (Eds.): PPSN VII, LNCS 2439, pp. 33–43, 2002.

the complete history of the current and all previous runs. There are various different implementations ranging from quite simple ones to quite sophisticated statistical approaches [7]. We note that for EAs it is not necessary to restart the algorithm completely. Partial restarts in the sense of re-initializing parts of the population are possible [11]. Since we concentrate on restart strategies and not on the underlying search algorithm, we will not discuss such mixed strategies.

We consider dynamic restart strategies, although they are not that commonly used. We do this for several good reasons. Obviously, the three classes of strategies form a hierarchy: static choices can be seen as degenerated dynamic ones, dynamic strategies can be described as degenerated adaptive ones. Note that self-adaptive strategies are a special case of adaptive strategies. Considering this hierarchy we believe that it is sensible to prove results on the simpler cases before analyzing the more complex classes. Second, due to their simpler structure such strategies are analyzable and allow for rigorous proofs of non-trivial statements. Third, it will turn out that already simple dynamic restart strategies can increase the performance of an EA enormously. Finally, typical adaptive restart strategies explicitly express some assumptions about the objective function and are therefore relatively easy to fool by constructing functions that do not meet these assumptions. In fact, we will present one example where dynamic restart strategies can be far superior to adaptive strategies that wait until the EA "gets stuck" before restarting it.

Next we discuss briefly the baseline of theoretical analysis of EAs and build our model in accordance to this. In Sect. 3 we present the EA we use to exemplify our restart strategies and the different dynamic restart strategies we analyze. In Sect. 4 we describe our example functions and motivate our choices. In Sect. 5 we derive results on one single run of the EA considered. We use these results for the comparison of the dynamic restart strategies in Sect. 6.

2 Performance Analysis of EAs

We consider exact optimization of static pseudo-Boolean functions $f\colon\{0,1\}^n \to \mathbb{R}$. We assume that f is defined for all $n \in \mathbb{N}$ (or at least infinitely many) and are interested in the asymptotic behavior of an EA on f for n growing to infinity. We neglect the choice of a stopping criterion and assume that the EA eventually finds some global optimum with probability 1 within some number of steps as in fact many EAs do [15]. Formally, let $p_{f,t}(n)$ denote the probability that some fixed EA optimizes an objective function $f\colon\{0,1\}^n \to \mathbb{R}$ in at most t steps. For all choices $p_{f,t}(n)$ is monotonic increasing with t. We consider only EAs where $p_{f,t}(n)$ is strongly monotonic increasing and $\lim_{t\to\infty} p_{f,t}(n) = 1$ holds for all f.

One typical measure is the expected optimization time, i. e., the expected number of steps until some global optimum is found. Sometimes a function f is typically efficiently optimized by some EA whereas in very rare cases it needs extremely long to optimize f. The expected optimization time may be exponential in this case, misleadingly indicating that f is difficult for this EA. However, when applying an appropriate restart strategy, this cannot happen: Assume that after

some number of steps T_s the EA has success probability $p_{f,T_s}(n)$ where T_s is not too large and $p_{f,T_s}(n)$ is not too small. Then a restart strategy stopping and restarting the EA after T_s steps leads to an acceptable expected optimization time of $T_s/p_{f,T_s}(n)$.

Note, that in our approach we do not pose some limit on the computational cost that optimization may take. This is fundamentally different from studies where the computational cost is fixed and the goal is to find a restart strategy that maximizes the expected gain while respecting this limit [2,5,10]. Our approach is more similar to the analysis of algorithms, where one is interested in the run time of an algorithm needed to perform a well-defined task without imposing a predefined limit of the time the algorithm may spend.

3 Algorithmic Framework

In principle, any EA can serve as basis for the comparison. In order to allow for an easy comparison it is desirable that the EA itself is well understood and easy to analyze. However, it should exhibit a behavior that can be assumed to be typical for EAs. This rules out extremely simple constructs like pure random search that show no similarity to common EAs. We consider the (1+1) EA to be a good compromise. It uses a population of size 1, bit-wise mutation with mutation probability $1/n$, and a deterministic selection, known as plus-selection from evolution strategies. This algorithm is often studied and well-understood [3,6,14,15]. It is known that it often performs at least comparable to much more complicated and sophisticated EAs [8,12]. We do not claim that the (1+1) EA behaves like more sophisticated EAs that may use crossover. But there are objective functions where more sophisticated EAs behave like the (1+1) EA on the classes of functions we introduce in the next section. Then, the restart strategies we investigate will have the same effects on the EAs as they have on the (1+1) EA. Thus, simplifying the analysis by choosing a simple EA does not restrict the applicability of the obtained results.

Algorithm 1 ((1+1) EA with dynamic restarts)

1. $r := 0$
2. $r := r + 1$; $t_r := 0$ Choose $x \in \{0,1\}^n$ uniformly at random.
3. Create y from x by independently for each bit flipping it with probability $1/n$.
4. If $f(y) \geq f(x)$, then $x := y$.
5. $t_r := t_r + 1$
6. If restart$(r) = t_r$, then continue at 2 else continue at 3.

Let T denote the minimal value of $\sum t_i$ when x is some global maximum. Then, we analyze the expected optimization time $\mathrm{E}(T)$. We concentrate on restart strategies where the length of intervals between two consecutive points of time when a restart is done is increasing, thus each run not being shorter than the preceeding run. This seems reasonable since long runs are a potential waste of time when already short runs have a good success probability. We consider two types of increasing dynamic restart strategies to be of special interest.

Definition 1. *An* additive restart strategy $restart_a\colon \mathbb{N} \to \mathbb{N}$ *is defined by* $s_0, s \in \mathbb{N}$ *and is given by* $restart_a(r) := s_0 + (r-1) \cdot s$. *A* multiplicative restart strategy $restart_m\colon \mathbb{N} \to \mathbb{N}$ *is defined by* $s_0 \in \mathbb{N}$, $s \in \mathbb{R}$ *with* $s > 1$ *and is given by* $restart_m(r) := \lceil s_0 \cdot s^{r-1} \rceil$.

In order to further simplify our considerations we investigate one special additive restart strategy r_a and one special multiplicative restart strategy r_m, only. We believe that important properties of additive and multiplicative restart strategies in general are captured by these two examples.

Definition 2. *The additive restart strategy* r_a *is defined by* $s_0 = n \log n$, $s = n \log n$. *The multiplicative restart strategy* r_m *is defined by* $s_0 = n \log n$, $s = 2$.

The length of the first run of the (1+1) EA is $s_0 = n \log n$ for both strategies. This choice is motivated by the fact that optimization of any function that has a unique global optimum takes $\Omega(n \log n)$ steps on average [3]. Therefore, we consider it to be counterproductive to stop the (1+1) EA earlier.

4 Example Functions

We want to compare dynamic restart strategies in different situations. Thus we are interested in functions exemplifying different situations as clearly as possible. This can be done best with artificial, constructed objective functions. We are only interested in functions where at least a perfect restart strategy can achieve polynomial expected optimization time. Note, though, that the choice of an inappropriate restart strategy can cause an overall inefficient optimization.

Definition 3. *For* $n, k \in \mathbb{N}$ *with* $k = O(1)$ *let the functions* $\mathrm{SP}_k\colon \{0,1\}^n \to \mathbb{R}$, $\mathrm{SP2}_k\colon \{0,1\}^n \to \mathbb{R}$, *and* $\mathrm{HSP}_:\{0,1\}^n \to \mathbb{R}$ *be defined by*

$$\mathrm{SP}_k(x) := \begin{cases} n \cdot (i+1) \text{ if } x = 1^i 0^{n-i} \text{ with} \\ \qquad i \in \{0, k, 2k, \ldots, \lceil n/(3k) \rceil\, k\}, \\ n - \sum_{i=1}^{n} x_i \text{ otherwise,} \end{cases}$$

$$\mathrm{SP2}_k(x) := \begin{cases} n \cdot (i+1) \text{ if } \left(x = 1^i 0^{n-i}\right) \vee \left(x = 0^{n-i} 1^i\right) \text{ with} \\ \qquad i \in \{0, k, 2k, \ldots, (\lceil n/(3k) \rceil - 1)\, k\}, \\ n \cdot (n+1) \text{ if } x = 1^i 0^{n-i} \text{ with } i = \lceil n/(3k) \rceil\, k, \\ n - \sum_{i=1}^{n} x_i \text{ otherwise.} \end{cases}$$

$$\mathrm{HSP}_k(x) := \begin{cases} n \cdot (i+1) \text{ if } x = 1^i 0^{n-i} \\ \qquad \text{with } i \in \{0, k, 2k, \ldots, \lceil n/(3k) \rceil\, k\}, \\ n + \sum_{i=1}^{n} x_i \text{ if } x_1 = x_2 = \cdots = x_{\lceil 2n/3 \rceil} = 0, \\ n - \sum_{i=1}^{n} x_i \text{ otherwise.} \end{cases}$$

For SP_k there is a kind of short path starting in 0^n. All points on the path are of the form 1^i0^{n-i} and are formed of $\lceil n/(3k) \rceil$ consecutive blocks of length k. If a block consists only of 1-bits and this holds for all blocks to its left, too, this is rewarded by n. For all points not on the path, each 1-bit in the string reduces the function value by 1 making the first point of the path easy to find. Obviously, the expected optimization time $\mathrm{E}(T)$ will increase with k.

$\mathrm{SP2}_k$ is very similar to SP_k but with two short paths, both starting in 0^n. One of the form 1^i0^{n-i} leading to the unique global optimum, the other of the form 0^i1^{n-i} leading to a local optimum. For symmetry reasons, the (1+1) EA will reach the local optimum in about half of the runs. Then, a direct mutation of at least $(2/3)n - k$ bits is necessary to reach the optimum. Thus, the (1+1) EA is very efficient in about half of the runs and very inefficient in about the other half of the runs and thus very inefficient on average.

HSP_k has a short path similar to the function SP_k which is easy to find. However, since once being at the beginning of the path it is much more likely to go away from the path then to follow it, we expect that the success probability $p_{\mathrm{HSP}_k,t}(n)$ is rather small for each $t = n^{O(1)}$.

5 Analysis Without Restarts

First, we investigate the behavior of the (1+1) EA without restarts. Results on the success probability $p_{f,t}(n)$ are useful when analyzing restart strategies. The more precise the upper and lower bounds on the success probability are, the more precise statements on the expected optimization time of the (1+1) EA in combination with different restart strategies can be concluded.

Theorem 1. *There exist two constants $c_1, c_2 \in \mathbb{R}^+$, such that given $n, k \in \mathbb{N}$ with $k = O(1)$ for the (1+1) EA on $\mathrm{SP}_k\colon \{0,1\}^n \to \mathbb{R}$ the following holds:*

$$\forall t \leq c_1 \cdot n^{k+1}\colon p_{\mathrm{SP}_k,t}(n) = e^{-\Omega(n)}$$
$$\forall t \geq c_2 \cdot n^{k+1}\colon p_{\mathrm{SP}_k,t}(n) = 1 - e^{-\Omega(n)}$$

Proof. We consider a run and divide it into two disjoint phases. The first phase starts with the random initialization and ends when the current string x for the first time is of the form 1^i0^{n-i} with $i \in \{0, k, 2k, \ldots, \lceil n/(3k) \rceil\, k\}$. The second phase starts when the first phase ends and ends when the current string x is equal to the unique global optimum. We claim that the probability that the first phase does not end within the first $\lceil 2en^2 \rceil$ steps is bounded above by $e^{-n/4}$. We ignore steps ending the first phase by leading to some 1^i0^{n-i} with $i > 0$. Obviously, this can only increase the probability of not ending the first phase. The current string x has Hamming distance $d(x)$ to the all zero string 0^n with $d(x) > 0$. The probability that the child y has Hamming distance at most $d(x)-1$ to 0^n is bounded below by $(1/n)(1-1/n)^{n-1} \geq 1/(en)$, since it is sufficient to mutate exactly one 0-bit in x. Such a child y has larger function value and will replace x. After at most n such replacements the Hamming distance is reduced to 0 and the first phase ends. Thus we are in the situation that we make random

experiments where each experiment is a success with probability at least $1/(en)$. The expected number of success in $\lceil 2en^2 \rceil$ trials is lower bounded by $2n$. By Chernoff bounds [13] the probability not to have at least n successes in $\lceil 2ne^2 \rceil$ trials is bounded above by $e^{-n/4}$.

We claim that the probability that the second phase does not end within the first $\lceil 2en^k \lceil n/(3k) \rceil k \rceil$ steps of this phase is bounded above by $e^{-n/12}$. In the second phase mutating exactly at most k 0-bits in the first block in x that contains 0-bits yields a child y that replaces x. The probability of such a mutation is bounded below by $(1/n)^k(1-1/n)^{n-k} \geq 1/\left(en^k\right)$. After at most $\lceil n/(3k) \rceil k$ such steps the global optimum is found. We argue as above and see that the probability not to have at least $\lceil n/(3k) \rceil k$ such steps within $2en^k \lceil n/(3k) \rceil k$ steps is bounded above by $e^{-\lceil n/(3k) \rceil k/4}$. Together this proves the second statement.

We now know that the number of different points in the search space encountered before x becomes some point 1^i0^{n-i} with $i \in \{0, k, 2k, \ldots, \lceil n/(3k) \rceil k\}$ is bounded above by $\lceil 2en^2 \rceil$ with probability $1 - e^{-n/4}$. Thus, the probability that for the first x of such form $i \geq n/12$ holds is bounded above by $2en^2 \cdot (n/4)/\binom{n}{n/12} < e^{-n/12}$. Thus, with probability at least $1 - e^{-n/12}$ at least $n/4$ blocks that are all zero have to become all one. In a mutation that flips one all zero block thereby generating an all one block, other blocks might flip as well. However, the probability that in such a step l additional blocks become all one is bounded above by $(1/n)^{kl}$. Thus, we expect in $n/9$ such steps in total less than $(n/9) + (1/n)^{kl-1}$ blocks to become all one. Due to Chernoff bounds, the probability that in total more then $(2n/9) + 2(1/n)^{kl-1} < n/4$ blocks become all one is bounded above by $e^{-n/25}$. One such step has probability at most $1/n^k$. Thus, we expect to have at most $n/20$ such mutations in $n^{k+1}/20$ steps. Again by Chernoff bounds, the probability to have at least $n/10$ such mutations in $n^{k+1}/20$ steps is bounded above by $e^{-19n/1000}$. This implies that with probability $1 - e^{-9n/500}$ after $n^{k+1}/20$ steps the global optimum is not reached. □

Similarities between SP_k and $\mathrm{SP2}_k$ yield a similar result and allow us to reuse parts of the proofs of Theorem 1 in the proof of the next theorem.

Theorem 2. *There exist two constants $c_1, c_2 \in \mathbb{R}^+$, such that given any $n, k \in \mathbb{N}$ with $k = O(1)$ for the (1+1) EA on $\mathrm{SP2}_k\colon \{0,1\}^n \to \mathbb{R}$ the following holds:*

$$\forall t \leq c_1 \cdot n^{k+1}\colon p_{\mathrm{SP2}_k,t}(n) = e^{-\Omega(n)}$$
$$\forall t \geq c_2 \cdot n^{k+1}\colon p_{\mathrm{SP2}_k,t}(n) = 1/2 - O(1/n)$$
$$\forall t \leq n^{n/4}\colon p_{\mathrm{SP2}_k,t}(n) = 1/2 - \Omega(1/n)$$

Proof. The first statement can be proved similarly to the proof of the first statement in Theorem 1. The proof of the second statement is similar to the proof of the second statement of Theorem 1. We extend the first phase until the first point of time when the (1+1) EA reaches some x with $\mathrm{SP2}_k(x) > \mathrm{SP2}_k(0^n)$. Since the path 1^i0^{n-i} contains one more element then the path 0^i1^{n-i}, the first x on the path belongs to the 1^i0^{n-i} with probability at least $1/2$. We consider one step where x changes and want to estimate the probability that the path

is left in favor of the other path. Assume that $x = 1^i 0^{n-i}$ is the current string. Then at least $2i$ bits have to mutate simultaneously in order to change the path. The probability for such a mutation is bounded above by $1/n^{2i}$. In order to reach the next point on the path it is sufficient to mutate at most k bits in the next block that contains some 0-bits. The probability for such a mutation is bounded below by $(1/n)^k(1-1/n)^{n-k} \geq 1/\left(en^k\right)$. The value of i depends on the current position of the path. It is at least k in the beginning and is increased by at least k after each such step. Thus, the probability to leave the path before reaching the optimum is bounded above by $2e/n$.

The proof of the third statement is a combination of the proof of the second statement in this proof and of the second statement in the proof of Theorem 1. The path $0^i 1^{n-i}$ is entered and not left with probability $(1/2) - O\left(1/n\right)$. Once the local optimum $0^{(\lceil n/(3k)\rceil - 1)k} 1^{n-(\lceil n/(3k)\rceil - 1)k}$ is reached, a mutation of at least $(2/3)n - k$ bits is necessary. Such a mutation has probability at most $n^{-(2/3)n-k}$. Thus, the probability that such a mutations happens within $n^{n/4}$ steps is bounded above by $n^{(n/4)-(2/3)n+k} \leq n^{-n/3}$. □

HSP_k and SP_k are similar, too. But in HSP_k the Hamming distance to the path can be increased by having 1-bits in the right third of x. Thus, it is possible not to reach the path at all. This implies important differences.

Theorem 3. *There exist two constants $c_1, c_2 \in \mathbb{R}^+$, such that given any $n, k \in \mathbb{N}$ with $k = O\left(1\right)$ for the (1+1) EA on $\mathrm{HSP}_k\colon \{0,1\}^n \to \mathbb{R}$ the following holds:*

$$\begin{aligned} \forall t \leq c_1 \cdot n^{k+1}\colon p_{\mathrm{HSP}_k,t}(n) &= e^{-\Omega(n)} \\ \forall t \geq c_2 \cdot n^{k+1}\colon p_{\mathrm{HSP}_k,t}(n) &= \Omega\left(1/n^k\right) \\ \forall t \leq n^{n/4}\colon p_{\mathrm{HSP}_k,t}(n) &= 1 - O\left(1/n^k\right) \end{aligned}$$

Proof. The first statement can be proved similarly to the proof of the first statement in Theorem 1. For the second and third statement it is crucial to give estimations of the probability for reaching the path $1^i 0^{n-i}$. We know from the proof of Theorem 1 that it is unlikely to find the path far from its beginning in 0^n. From the ballot theorem [4] it follows that in fact with probability $\lceil 2n/3 \rceil / n$ the all zero string 0^n is found. Now, let as assume that the current string x is 0^n. Then, the next child that replaces its parent can either be a point $1^i 0^{n-i}$ on the path or some point y different from 0^n with $y_1 = y_2 = \cdots = y_{\lceil 2n/3 \rceil} = 0$. Note, that once a path point is reached all subsequent current strings can only be path points. Obviously, the probability to reach a path point is bounded above by $n^{-k} + n^{-2k} + n^{-3k} + \cdots = O\left(1/n^k\right)$ while we reach some other point with probability at least $(1/3)(1-1/n)^{n-1} = \Omega(1)$. On the other hand, the probability to reach the path is bounded below by $(1/n)^k(1-1/n)^{n-k} = \Omega\left(1/n^k\right)$. Thus, the first point reached is on the path with probability $\Theta\left(1/n^k\right)$. Once on the path the path is never left and we conclude from the proof of Theorem 1 that with probability $1 - e^{-\Omega(n)}$ within $\Theta\left(n^{k+1}\right)$ steps we reach the global optimum.

For the third statement it is sufficient to note that the Hamming distance to points on the path can only increase. In particular, as long as the Hamming

distance to the path is bounded above by $n/6$, the Hamming distance is increased with probability $\Omega(1)$ in each step. Thus with probability $1-O\left(1/n^k\right)$ the point $0^{\lceil 2n/3\rceil}1^{\lfloor n/3\rfloor}$ is reached before reaching any point on the path. Then, a mutation of at least $\lfloor n/3\rfloor$ bits simultaneously is necessary to reach the path and the third statement follows similar to the proof of Theorem 2. □

6 Comparison of Dynamic Restart Strategies

Now we apply the results from the previous section to obtain results for the (1+1) EA with restarts. These results represent the behavior of such dynamic restart strategies for any EA with a success probability converging to 1, converging to some positive constant, and converging to 0 polynomially fast, like the (1+1) EA on SP_k, $\mathrm{SP2}_k$, and HSP_k, respectively.

Theorem 4. *For $n, k \in \mathbb{N}$ with $k = O(1)$ the (1+1) EA with dynamic restart strategy r_m has expected optimization time $E(T) = \Theta\left(n^{k+1}\right)$ on SP_k.*

Proof. The lower bound follows from Theorem 1. For the upper bound we consider the first uninterrupted run of length at least $c_2 n^{k+1}$ where c_2 is the constant from Theorem 1. After $\lceil (k \log n) - (\log\log n) + \log c_2\rceil$ restarts we have one run of at least this length. The expected number of such runs before the global optimum is found is bounded above by 2. Thus, $\mathrm{E}(T)$ is bounded above by $\sum_{i=0}^{\lceil (k\log n)-(\log\log n)+\log c_2\rceil+1} 2^i \cdot n\log n \leq 8c_2 \cdot n^{k+1}$. □

The order of growth of $\mathrm{E}(T)$ with r_m is optimal. Thus, for an easy to optimize function it is good to have a restart strategy quickly increase the length of runs.

Theorem 5. *For $n, k \in \mathbb{N}$ with $k = O(1)$ the (1+1) EA with dynamic restart strategy r_a has expected optimization time $E(T) = \Theta\left(n^{2k+1}/\log n\right)$ on SP_k.*

Proof. We can proof the upper bound similar to the proof of Theorem 4. After $\left\lceil c_2 n^k/(\log n)\right\rceil$ restarts, we have a run of length at least $c_2 n^{k+1}$. Thus, $\mathrm{E}(T)$ is bounded above by $\sum_{i=1}^{\lceil c_2 n^k/(\log n)\rceil+1} i \cdot n\log n < 2c_2^2 \cdot n^{2k+1}\log n$.

For the lower bound we consider the first $\left\lfloor c_1 n^k/(\log n)\right\rfloor$ runs of the (1+1) EA. All runs have length at most $\left\lfloor c_1 n^k/(\log n)\right\rfloor \cdot n\log n \leq c_1 n^{k+1}$. Thus, all runs have success probability $e^{-\Omega(n)}$. The total length of theses runs is at least $\sum_{i=1}^{\lfloor c_1 n^k/(\log n)\rfloor} i \cdot n\log n = \Omega\left(\frac{n^{2k+1}}{\log n}\right)$. □

In comparison with r_m the restart strategy r_a performs poorly. This is due to the long time that is needed to have runs of sufficient length.

Theorem 6. *For $n, k \in \mathbb{N}$ with $k = O(1)$ the (1+1) EA with dynamic restart strategy r_m has expected optimization time $E(T) = \Theta\left(n^{k+1}\right)$ on $\mathrm{SP2}_k$.*

Proof. The proof can be done in the same way as the proof of Theorem 4. □

Theorem 7. *For $n, k \in \mathbb{N}$ with $k = O(1)$ the (1+1) EA with dynamic restart strategy r_a has expected optimization time $E(T) = \Theta\left(n^{2k+1}/\log n\right)$ on $\mathrm{SP2}_k$.*

Proof. The proof can be done in the same way as the proof of Theorem 6. □

Using additive or multiplicate restart strategies, it makes no difference for the order of growth of $\mathrm{E}(T)$ whether the success probability converges to 1 or to some positive constant.

Theorem 8. *For $n, k \in \mathbb{N}$ with $k = O(1)$ the (1+1) EA with dynamic restart strategy r_m has expected optimization time $E(T) = 2^{\Omega(n/\log n)}$ on HSP_k.*

Proof. Due to Theorem 3, on average $\Theta\left(n^k\right)$ runs of length $> c_1 n^{k+1}$ are needed to optimize HSP_k. After $n/\log n$ restarts the last run has length $2^{n/\log n} n \log n$. Since this is $o\left(n^{n/4}\right)$, the success probability for each run is still $o\left(1/n^k\right)$. So, the optimum is not found after these restarts with probability $1 - o(1)$. □

Theorem 9. *For $n, k \in \mathbb{N}$ with $k = O(1)$ the (1+1) EA with dynamic restart strategy r_a has expected optimization time $E(T) = \Theta\left(n^{2k} \log n\right)$ on HSP_k.*

Proof. The proof of the upper bound follows roughly the same lines as the proof of the upper bound of Theorem 5. After $\left\lceil c_2 n^k/(\log n)\right\rceil$ restarts the length of the run (and all following runs) is bounded below by $c^2 n^{k+1}$. Thus, this run and all following runs have success probability $\Theta\left(1/n^k\right)$ according to Theorem 3. Thus, the expected number of runs that is needed on average is bounded above by $O\left(n^k + n^k/(\log n)\right) = O\left(n^k\right)$. Then the total length of all runs is $O\left(n^{2k} \log n\right)$.

The lower bound can be proved in the same way. What is needed additionally is the third statement from Theorem 3. As in the proof of Theorem 8 we see from there that the steps taken after the $\left(c_2 n^{k+1}\right)$-step in all runs do not add something significant to the success probability since we are considering only a polynomial number of runs and a polynomial number of steps. □

Compared to r_a the strategy r_m performs very poorly. This is due to the fact that the length of the runs increases so quickly. Thus, for difficult functions where the probability of finding an optimum at all is very small even for quite long runs additive restart strategies are preferred.

7 Conclusions

We presented a framework for the evaluation of restart strategies different from other settings: We compared the expected optimization time instead of fixing the computational cost in advance. We restricted ourselves to dynamic restart strategies, which are more complex than fixing some point of time for a restart but less complex than adaptive restart strategies. We know that in applications adaptive restart strategies are "state of the art." But we believe that it makes sense to begin theoretical investigations with a simpler and still interesting class.

We described two classes of dynamic restart strategies and investigated one additive strategy and one multiplicative strategy in detail. We employed the simple (1+1) EA as the underlying search algorithm and presented three classes of example functions with very different success probabilities. The time that is typically spent optimizing these examples can be controlled via a parameter k.

We saw that it made almost no difference whether the success probability within a polynomial number of steps is near to 1 or some other positive constant. In both cases r_m, which quickly increases the length of each run, is superior. However, if the success probability stays close to 0 very long, r_a is by far superior. For r_a the expected optimization is polynomial, for r_m it is exponential. It is easy to see that all functions that can be optimized with multiplicative restart strategies in expected polynomial time can be optimized with additive restart strategies in expected polynomial time, but the degree of the polynomial can be larger. Here the quotient was of order $\Theta\left(n^k / \log n\right)$, with k any positive integer.

We saw that considering expected optimization time is reasonable when comparing restart strategies. The investigation of adaptive restart strategies within this framework is subject to future research. Also open is the empirical comparison of r_a and r_m not only on our examples but also in practical settings.

Acknowledgments

The author thanks Paul Wiegand for helpful discussions. The author was supported by a fellowship within the post-doctoral program of the German Academic Exchange Service (DAAD).

References

1. Th. Bäck. An overview of parameter control methods by self-adaptation in evolutionary algorithms. *Fundamenta Informaticae*, 35:51–66, 1998.
2. E. Cantú-Paz. Single vs. multiple runs under constant computation cost. In *Proc. of he Genetic and Evolutionary Computation Conf.* (*GECCO 2001*), page 754. Morgan Kaufmann, 2001.
3. S. Droste, Th. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. CI 21/98, SFB 531, Univ. Dortmund, 1998. To appear in: TCS.
4. W. Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 1968.
5. A. S. Fukunaga. Restart scheduling for genetic algorithms. In *Parallel Problem Solving from Nature* (*PPSN V*), LNCS 1498, pages 357–366. Springer, 1998.
6. J. Garnier, L. Kallel, and M. Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7(2):173–203, 1999.
7. M. Hulin. An optimal stop criterion for genetic algorithms: A Bayesian approach. In *Proc. of the Seventh International Conf. on Genetic Algorithms* (*ICGA '97*), pages 135–143. Morgan Kaufmann, 1997.
8. A. Juels and M. Wattenberg. Hillclimbing as a baseline method for the evaluation of stochastic optimization algorithms. In *Advances in Neural Information Processing Systems 8*, pages 430–436. MIT Press, 1995.
9. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

10. S. Luke. When short runs beat long runs. In *Proc. of he Genetic and Evolutionary Computation Conf.* (*GECCO 2001*), pages 74–80. Morgan Kaufmann, 2001.
11. J. Maresky, Y. Davidor, D. Gitler, Gad A., and A. Barak. Selectively destructive restart. In *Proc. of the Sixth International Conf. on Genetic Algorithms* (*ICGA '95*), pages 144–150. Morgan Kaufmann, 1995.
12. M. Mitchell, J. H. Holland, and S. Forrest. When will a genetic algorithm outperform hill climbing? In *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1994.
13. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
14. H. Mühlenbein. How genetic algorithms really work. Mutation and hillclimbing. In *Proc. of the 2nd Parallel Problem Solving from Nature* (*PPSN II*), pages 15–25. North-Holland, 1992.
15. G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Dr. Kovač, 1997.