Swarm Intell (2009) 3: 35–68
DOI 10.1007/s11721-008-0023-3

Analysis of different MMAS ACO algorithms on unimodal functions and plateaus

Frank Neumann · Dirk Sudholt · Carsten Witt

Received: 31 October 2007 / Accepted: 17 November 2008 / Published online: 5 December 2008

Abstract Recently, the first rigorous runtime analyses of ACO algorithms appeared, covering variants of the MAX–MIN ant system and their runtime on pseudo-Boolean functions. Interestingly, a variant called 1-ANT is very sensitive to the evaporation factor while Gutjahr and Sebastiani proved partly opposite results for their variant MMAS_{bs}. These algorithms differ in their pheromone update mechanisms and, moreover, 1-ANT accepts equally fit solutions in contrast to MMAS_{bs}.

By analyzing variants of MMAS_{bs}, we prove that the different behavior of 1-ANT and MMAS_{bs} results from the different pheromone update mechanisms. Building upon results by Gutjahr and Sebastiani, we extend their analyses of MMAS_{bs} to the class of unimodal functions and show improved results for test functions using new and specialized techniques; in particular, we present new lower bounds. Finally, we compare MMAS_{bs} with a variant that also accepts equally fit solutions as this enables the exploration of plateaus. For well-known plateau functions we prove that this drastically reduces the optimization time. Our findings are complemented by experiments that support our asymptotic analyses and yield additional insights.

Keywords Ant colony optimization · MMAS · Runtime analysis · Theory

1 Introduction

Randomized search heuristics have been shown to be good problem solvers with various application domains. Two prominent examples belonging to this class of algorithms are Evolutionary Algorithms (EAs) (Eiben and Smith 2007) and Ant Colony Optimization (ACO)

A conference version appeared in SLS 2007 (Neumann et al. 2007).

F. Neumann (✉)
Algorithms and Complexity, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
e-mail: fne@mpi-inf.mpg.de

D. Sudholt · C. Witt
Fakultät für Informatik, LS 2, Technische Universität Dortmund, 44221 Dortmund, Germany

(Dorigo and Stützle 2004). Especially ACO algorithms have been shown to be very successful for solving problems from combinatorial optimization. Indeed, the first problem to which an ACO algorithm was applied was the Traveling Salesperson Problem (TSP) (Dorigo et al. 1996), which is one of the most studied combinatorial problems in computer science.

Many empirical studies and applications indicate the rapid development of randomized search heuristics, but the theoretical knowledge lags far behind. In the case of ACO algorithms, the analysis of their runtime behavior has been started only recently. The analysis of randomized search heuristics (e.g., Droste et al. 2002) is carried out as in the classical algorithm community and makes use of several strong methods for the analysis of randomized algorithms (Motwani and Raghavan 1995; Mitzenmacher and Upfal 2005).

Concerning evolutionary algorithms, a lot of progress has been made in recent years in analyzing their runtime behavior. Initial studies considered the behavior on artificial pseudo-Boolean functions. This includes results on unimodal (Droste et al. 2002) or plateau functions (Jansen and Wegener 2001) as well as investigations for the class of linear functions. Within these studies, many useful tools have been developed that are at this time *state-of-the-art* when analyzing the computational complexity of evolutionary algorithms. Later on, evolutionary algorithms were successfully analyzed on some well-known combinatorial optimization problems. Problems for which results have been obtained include the single-source shortest path problem (Scharnow et al. 2004), maximum matchings (Giel and Wegener 2003), Eulerian cycles (Neumann 2004; Doerr et al. 2006), different kinds of spanning tree problems (Neumann and Wegener 2006, 2007; Neumann 2007), as well as the NP-hard partition problem (Witt 2005).

1.1 Previous results and our contribution

Regarding ACO, only convergence results (Gutjahr 2002) were known until 2006 and analyzing the runtime of ACO algorithms has been pointed out as a challenging task by Dorigo and Blum (2005). First steps into analyzing the runtime of ACO algorithms have been made by Gutjahr (2008), and, independently, the first runtime analyses of a simple ACO algorithm called 1-ANT were done at the same time by Neumann and Witt (2006). Subsequently this algorithm was further investigated for the optimization of some well-known pseudo-Boolean functions (Doerr et al. 2007). A conclusion of these investigations is that 1-ANT is very sensitive to the choice of the evaporation factor ρ. Decreasing the value of ρ only by a small amount may lead to a phase transition and turn a polynomial runtime into an exponential one. In contrast to this, a simple ACO algorithm called MMAS_{bs}, for which this phase transition does not occur, has been investigated in a recent article by Gutjahr and Sebastiani (2008).

1-ANT and MMAS_{bs} differ in their pheromone update mechanisms as well as in their selection strategies. While MMAS_{bs} reinforces the current best-so-far solution in each iteration, 1-ANT only updates its pheromones in case the best-so-far solution is exchanged. Regarding the selection, MMAS_{bs} only accepts strict improvements while 1-ANT also accepts solutions with equal fitness. Gutjahr (2007, p. 76) conjectures with reference to the above-mentioned phase transition observed for 1-ANT: "*The reason seems to lie in the different update rules for the best-so-far solution*: *exchanging* $\hat{x}$ [*the best-so-far solution*] *also in case of an equally good solution, as it is done in 1-ANT, obviously deteriorates the performance of the algorithm on* ONEMAX. *For* LEADINGONES, *the situation appears to be quite similar as for* ONEMAX." By an investigation of variants of MMAS_{bs}, we will disprove this conjecture that the different behavior of MMAS_{bs} and 1-ANT, i.e., the lack/presence of the phase transition, is due to their different selection strategies. Actually, we show that the different behavior results from the different pheromone update mechanisms.

As already mentioned, 1-ANT needs exponential time to optimize even very simple functions like ONEMAX in case the evaporation factor is set too low (Neumann and Witt 2006; Doerr et al. 2007). Hence, the update mechanism of MMAS_{bs} with pheromone updates in each iteration is to be preferred over the mechanism used by 1-ANT. This motivates us to study MMAS variants with this more persistent update scheme. We consider the MMAS_{bs} algorithm by Gutjahr and Sebastiani (2008), in the present paper referred to as MMAS*, and show both improved and extended results. Due to the update mechanism of MMAS*, it is possible to use the method of fitness-based partitions for the analysis of this algorithm. This simple method has been widely used for the analysis of evolutionary algorithms (see, e.g., Wegener 2002). In Gutjahr and Sebastiani (2008), it is presented in a relatively general and formal way. Within this paper, we state the method more intuitively by extending the presentation for evolutionary algorithms. After having clarified the method of fitness-based partitions in the context of ACO algorithms, we show that it is easy to obtain upper bounds on the simple pseudo-Boolean functions ONEMAX and LEADINGONES. These results for MMAS* have already been obtained by Gutjahr and Sebastiani (2008), but our simplified presentation of fitness-based partitions yields really simple and short proofs which are easily transferable to other functions of similar structure. In particular, we prove general results on the important class of unimodal functions, which are not contained in Gutjahr and Sebastiani (2008).

Since the method of fitness-based partitions contains several pessimistic assumptions on the behavior of MMAS*, it only yields upper bounds. To judge the quality of the method, it is therefore crucial to complement the results by good lower bounds on the optimization time. A major contribution of this paper and, at the same time, an extension to the analysis by Gutjahr and Sebastiani (2008) are the proofs of such lower bounds in the context of ONEMAX, and, in particular, LEADINGONES, where no lower bounds on the runtime of MMAS* were known before. These bounds show that upper bounds obtained before using the method of fitness-based partitions are almost tight. Still, there is room for improvement. For the LEADINGONES example, we present an improved upper bound which is tight with the lower bound for almost all values of ρ. This analysis indicates the limits of the general method. Moreover, the detailed study of MMAS* on LEADINGONES is interesting also from a methodological point of view. The random walk described by pheromone values for "unimportant" bits is investigated in detail and its hitting times for boundary values are bounded using a general study of martingale processes. This technique fosters our understanding of the stochastic process behind random pheromone values.

Finally, we examine the issue of whether to accept equally good solutions or just strict improvements. Accepting equally fit solutions allows MMAS algorithms to explore plateaus of solutions with equal fitness. We compare the runtime behavior of MMAS* with a variant of MMAS that accepts solutions of equal fitness on functions with plateaus of exponential and polynomial size, respectively. The analyses, which again rely on the above-mentioned martingale process behind random pheromone values, reveal that MMAS clearly outperforms MMAS* on both functions.

As our theoretical results are purely asymptotic, they are accompanied by empirical results. A comparison of MMAS and MMAS* on the unimodal functions ONEMAX and LEADINGONES shows that these algorithms have similar performance. However, a closer look reveals that for certain values of ρ MMAS is faster than MMAS*. This difference is statistically significant. With regard to the above-mentioned conjecture by Gutjahr (2007), accepting equally good solutions does not deteriorate the performance of MMAS* on ONEMAX. Instead, the opposite is true for some values of ρ. Our theoretical results on plateau functions predict larger runtimes for MMAS*. Experiments for the investigated plateau

functions show that the observed performance difference between MMAS and MMAS* is huge, even for small problem dimensions.

The remainder of the paper is structured as follows. In Sect. 2, we introduce the above-mentioned algorithms that are the subject of our investigations. In Sect. 3, we analyze MMAS* and MMAS on unimodal functions. Section 4 then compares MMAS* with MMAS on plateau functions. Section 5 contains experimental supplements to a result from Sect. 4 and experiments comparing MMAS with MMAS* as mentioned above. We finish with some conclusions.

2 Algorithms

We consider the runtime behavior of several ACO algorithms that work by a common principle to construct new solutions. Solutions for a given problem, bit strings $x \in \{0,1\}^n$ for pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$, are constructed by a random walk on a so-called construction graph $C = (V, E)$, which is a directed graph with a designated start vertex $s \in V$ and pheromone values $\tau\colon E \to \mathbb{R}$ on the edges. The construction procedure is shown in Fig. 1.

We examine the construction graph given in Fig. 2, which is known in the literature as *Chain* (Gutjahr 2006). For bit strings of length n, the graph has $3n+1$ vertices and $4n$ edges. The decision whether a bit x_i, $1 \le i \le n$, is set to 1 is made at node $v_{3(i-1)}$. If edge $(v_{3(i-1)}, v_{3(i-1)+1})$ (called 1-edge) is chosen, x_i is set to 1 in the constructed solution. Otherwise, the corresponding 0-edge is taken, and $x_i = 0$ holds. After this decision has been made, there is only one single edge which can be traversed in the next step. In the case that $(v_{3(i-1)}, v_{3(i-1)+1})$ has been chosen, the next edge is $(v_{3(i-1)+1}, v_{3i})$, otherwise the edge $(v_{3(i-1)+2}, v_{3i})$ is traversed. Hence, these edges have no influence on the constructed solution and we can assume $\tau_{(v_{3(i-1)},v_{3(i-1)+1})} = \tau_{(v_{3(i-1)+1},v_{3i})}$ and $\tau_{(v_{3(i-1)},v_{3(i-1)+2})} = \tau_{(v_{3(i-1)+2},v_{3i})}$ for $1 \le i \le n$. We ensure that $\sum_{(u,\cdot)\in E} \tau_{(u,\cdot)} = 1$ for $u = v_{3i}$, $0 \le i \le n-1$, and $\sum_{(\cdot,v)} \tau_{(\cdot,v)} = 1$ for $v = v_{3i}$, $1 \le i \le n$. Let $p_i = \mathrm{Prob}(x_i = 1)$ be the probability of setting the bit x_i to 1 in

Construct(C, τ)
1. $v := s$, mark v as visited.
2. Let N_v be the set of non-visited successors of v in C.
   If $N_v \neq \emptyset$ then
   (a) Choose successor $w \in N_v$ with probability $\tau_{(v,w)} / \sum_{(v,u)|u\in N_v} \tau_{(v,u)}$.
   (b) Mark w as visited, set $v := w$ and go to 2.
3. Return the solution x and the path $P(x)$ constructed by this procedure.

Fig. 1 The construction procedure for pseudo-Boolean optimization

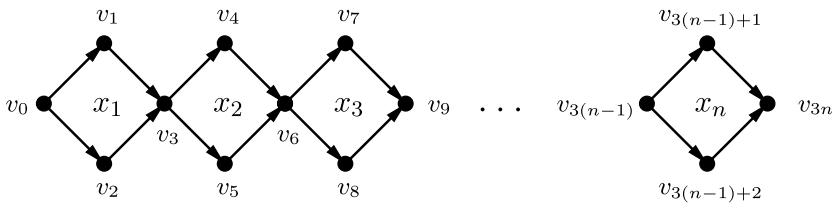

Fig. 2 Construction graph for pseudo-Boolean optimization

the next constructed solution. The probability p_i is often called the *success probability* at a bit. Note that $p_i = \tau_{(3(i-1),3(i-1)+1)}$ and $1 - p_i = \tau_{(3(i-1),3(i-1)+2)}$ holds, implying that for every 1-edge the pheromone value equals the success probability for the corresponding bit. Following the MAX–MIN ant system by Stützle and Hoos (2000), we restrict each $\tau_{(u,v)}$ to the interval $[1/n, 1 - 1/n]$ so that every solution always has a positive probability of being chosen. The pheromone borders $1/n$ and $1 - 1/n$ have already been used in previous studies (Neumann and Witt 2006; Doerr et al. 2007; Gutjahr and Sebastiani 2008). The choice of these values is inspired by standard mutation operators in evolutionary computation where an incorrect bit has a probability of $1/n$ of being corrected.

Depending on whether the edge (u, v) is contained in the path $P(x)$ of the constructed solution x, the pheromone values are updated to τ' in the update procedure as follows:

$$\tau'_{(u,v)} = \begin{cases} \min\{(1-\rho)\cdot\tau_{(u,v)} + \rho,\ 1 - 1/n\}, & \text{if } (u,v) \in P(x), \\ \max\{(1-\rho)\cdot\tau_{(u,v)},\ 1/n\}, & \text{otherwise.} \end{cases}$$

The first algorithm that has been investigated in terms of rigorous runtime analysis is 1-ANT introduced by Neumann and Witt (2006), which is shown in Fig. 3. The attentive reader may notice that in the present paper the pheromone values have been rescaled in comparison to Neumann and Witt (2006). This rescaling was independently proposed by Doerr and Johannnsen (2007).

1-ANT only performs a pheromone update in case the best-so-far solution x^* is replaced. If ρ is large, such an update has a large effect on the pheromone values. In that case, 1-ANT is able to reproduce the new best-so-far solution x^* quite efficiently and to discover better solutions in the region around x^*. However, if ρ is small, updates are rare and they only have a small effect on the pheromones. This makes it hard for 1-ANT to focus on promising regions of the search space. The analyses by Neumann and Witt (2006); Doerr et al. (2007) have revealed that 1-ANT is unable to keep track of the best-so-far solution. As 1-ANT discovers better and better best-so-far solutions, the hurdles for constructed solutions are set higher and higher. If the update strength is too small, the algorithm fails in storing the knowledge gained by new best-so-far solutions in the pheromones. The probability of reproducing the current best-so-far solution or finding a better one decreases and 1-ANT slowly gets stuck. In fact, even for really simple functions like OneMax, 1-ANT needs exponential time with overwhelming probability if ρ is small, e.g., less than $1/\log n$.

Neumann and Witt (2006) first discovered this so-called phase-transition behavior for 1-ANT on OneMax. The runtime of 1-ANT turns from exponential to polynomial values (with overwhelming probability) with increasing ρ when a certain threshold is passed. The same behavior has also been proved for two other test functions by Doerr et al. (2007). In a recent article, Gutjahr and Sebastiani (2008) investigate a very similar ACO algorithm where, surprisingly, this phase transition behavior does not occur. Therefore, one aim of the present article is to elaborate on the different behavior of these two algorithms. Gutjahr and Sebastiani call their algorithm MMAS_{bs}, we refer to it as MMAS*. This algorithm is shown on the right-hand side in Fig. 3. Compared to 1-ANT, there are two differences. First, MMAS* only accepts strict improvements, i.e., new solutions x with $f(x) = f(x^*)$ are rejected by MMAS* while they are accepted by 1-ANT. A second difference is that MMAS* reinforces the best-so-far solution in every generation, regardless whether it has been replaced or not. Hence, in generations without improvements, the pheromones are changed towards the best-so-far solution in MMAS*, while the pheromones remain unchanged in 1-ANT.

1-ANT
1. Set $\tau_{(u,v)} = 1/2$ for all $(u, v) \in E$.
2. Create x^* using Construct(C, τ).
3. Update pheromones w.r.t. x^*.
4. Repeat
   (a) Create x using Construct(C, τ).
   (b) If $f(x) \geq f(x^*)$ then
      (i) $x^* := x$.
      (ii) Update pheromones w.r.t. x^*.

MMAS
1. Set $\tau_{(u,v)} = 1/2$ for all $(u, v) \in E$.
2. Create x^* using Construct(C, τ).
3. Update pheromones w.r.t. x^*.
4. Repeat
   (a) Create x using Construct(C, τ).
   (b) If $f(x) \geq f(x^*)$ then $x^* := x$.
   (c) Update pheromones w.r.t. x^*.

MMAS*
1. Set $\tau_{(u,v)} = 1/2$ for all $(u, v) \in E$.
2. Create x^* using Construct(C, τ).
3. Update pheromones w.r.t. x^*.
4. Repeat
   (a) Create x using Construct(C, τ).
   (b) If $f(x) > f(x^*)$ then $x^* := x$.
   (c) Update pheromones w.r.t. x^*.

$(1 + 1)$ **EA**
1. Choose x^* uniformly at random.
2. Repeat
   (a) Create x by flipping each bit of x^* independently with probability $1/n$.
   (b) If $f(x) \geq f(x^*)$ then $x^* := x$.

$(1 + 1)$ **EA***
1. Choose x^* uniformly at random.
2. Repeat
   (a) Create x by flipping each bit of x^* independently with probability $1/n$.
   (b) If $f(x) > f(x^*)$ then $x^* := x$.

Fig. 3 The algorithms considered in this work. The starred variants on the right-hand side use a strict selection while the left-hand side algorithms also accept equally fit solutions. 1-ANT only updates pheromones in case the best-so-far solution x^* is replaced. Contrarily, MMAS updates pheromones in every iteration. If ρ is so large that the pheromone borders are hit in a single pheromone update, both 1-ANT and MMAS collapse to $(1 + 1)$ EA and MMAS* collapses to $(1 + 1)$ EA*

As mentioned in the introduction, Gutjahr (2007) conjectures that the different behavior of MMAS_{bs} and 1-ANT depends on the former difference: the fact that MMAS_{bs} only accepts strict improvements. Therefore, we also investigate a variant of MMAS_{bs} that accepts equally fit solutions. We call this algorithm MMAS (see the left-hand side of Fig. 3). In Sect. 3, we will analyze the runtime behavior of MMAS* and MMAS on simple unimodal functions. These results are then compared to known results for 1-ANT. Moreover, we extend and simplify results by Gutjahr and Sebastiani (2008). Following the ideas of these authors, we show how to transfer a popular and effective method for the analysis of evolutionary algorithms to the analysis of MMAS*.

ACO algorithms such as MMAS* are not far away from evolutionary algorithms. If the value of ρ is chosen large enough in MMAS*, the pheromone borders $1/n$ or $1 - 1/n$ are touched for every bit. In this case, MMAS* becomes the same as the algorithm called $(1 + 1)$ EA*, which is known from the analysis of evolutionary algorithms (Jansen and Wegener 2001).

As already pointed out by Jansen and Wegener (2001), $(1+1)$ EA* has difficulties with simple plateaus of constant fitness as no search points of the same fitness as the best so far are accepted. Accepting solutions with equal fitness enables the algorithm to explore plateaus by random walks. Therefore, it seems more natural to replace search points by new solutions that are at least as good. In the case of evolutionary algorithms, this leads to the well-known $(1+1)$ EA algorithm, which differs from $(1+1)$ EA* only in step 2(b) of the algorithm. Both $(1+1)$ EA* and $(1+1)$ EA are also shown in Fig. 3. By essentially the same arguments, we also expect MMAS* to outperform MMAS* on plateau functions. The corresponding runtime analyses are presented in Sect. 4.

For the analysis of an algorithm, we consider the number of solutions that are constructed by the algorithm until an optimal solution has been obtained for the first time. This is called the *optimization time* of the algorithm; it is a well-accepted measure in the analysis of evolutionary algorithms since each point of time corresponds to a fitness evaluation. Often the expectation of this value is considered and called the *expected optimization time*.

3 Unimodal functions, ONEMAX, and LEADINGONES

In the following, we derive upper bounds on the expected optimization time of MMAS* and MMAS on unimodal functions, especially ONEMAX and LEADINGONES. These results can then be compared to previous results for 1-ANT (Neumann and Witt 2006; Doerr et al. 2007). In contrast to 1-ANT, a similarity between MMAS* and evolutionary algorithms can be exploited to obtain good upper bounds. Suppose that during a run there is a phase during which MMAS* never replaces the best-so-far solution x^* in step 4(b) of the algorithm. This implies that the best-so-far solution is reinforced again and again until all pheromone values have reached their upper or lower border corresponding to the setting of the bits in x^*. We can say that x^* has been "frozen in pheromone". The probability to create a 1 for any bit is now either $1/n$ or $1-1/n$. The distribution of constructed solutions equals the distribution of offspring of $(1+1)$ EA* and $(1+1)$ EA with x^* as the current search point. We conclude that, as soon as all pheromone values touch their upper or lower border, MMAS* behaves like $(1+1)$ EA* until a solution with larger fitness is encountered. This similarity between ACO and EAs can be used to transfer a well-known method called the method of fitness-based partitions (also called the method of f-based partitions, see, e.g., Wegener (2002), or, for short, *fitness-level method*) for the runtime analysis from EAs to ACO. In particular, upper bounds for MMAS* will be obtained from bounds for $(1+1)$ EA by adding the so-called freezing time described in the following. This correspondence has already been observed by Gutjahr and Sebastiani (2008). However, as they strive for a higher level of generality, their presentation is rather formal.

We give a simplified and more intuitive presentation of the approach followed by Gutjahr and Sebastiani (2008), adapted to MMAS* with our choice of pheromone borders. In particular, we review some known results from Gutjahr and Sebastiani (2008) and show that specialized results for MMAS* can be proven in an easy and intuitive way. Our presentation follows the presentation of fitness-based partitions for evolutionary algorithms. This enables us to highlight the similarities between EAs and ACO and it reveals a way to directly transfer runtime bounds known for EAs to MMAS*. This is an important step towards a unified theory of EAs and ACO.

3.1 A simplified review of fitness-based partitions for ACO

Suppose the current best-so-far solution is not changed and consider the random time t^* until all pheromones have reached their borders. We will refer to this random time as *freezing*

time. Gutjahr and Sebastiani (2008) bound t^* from above by $-\ln(n-1)/\ln(1-\rho)$. This holds since a pheromone value which is only increased during t steps is at least $\min\{1-1/n, 1-(1-1/n)(1-\rho)^t\}$ after the iterations, pessimistically assuming the worst-case starting value $1/n$. In the present paper, we use $\ln(1-\rho) \leq -\rho$ for $0 \leq \rho \leq 1$ and arrive at the handy upper bound

$$t^* \leq \frac{\ln n}{\rho}. \tag{1}$$

Following Gutjahr and Sebastiani (2008), we now use the freezing time t^* to derive a general upper bound on the expected optimization time of MMAS* by making use of (a restricted version of) the method of fitness-based partitions from the analysis of evolutionary algorithms. Let $f_1 < f_2 < \cdots < f_m$ be an enumeration of all fitness values and let A_i, $1 \leq i \leq m$ contain all search points with fitness f_i. In particular, A_m contains only optimal search points. Now, let s_i, $1 \leq i \leq m-1$, be a lower bound on the probability of $(1+1)$ EA (or, in this case equivalently, $(1+1)$ EA*) to create an offspring in $A_{i+1} \cup \cdots \cup A_m$, provided that the current population belongs to A_i. The expected waiting time until such an offspring is created is at most $1/s_i$ and then the set A_i is left for good. As every set A_i has to be left at most once, the expected optimization time for $(1+1)$ EA and $(1+1)$ EA* is bounded above by

$$\sum_{i=1}^{m-1} \frac{1}{s_i}. \tag{2}$$

Consider t^* steps of MMAS* and assume $x^* \in A_i$. Either the best-so-far fitness increases during this period or all pheromone values are frozen. In the latter case, the probability to create a solution in $A_{i+1} \cup \cdots \cup A_m$ is at least s_i, and the expected time until the best-so-far fitness increases is at most $t^* + 1/s_i$. We arrive at the following upper bound for MMAS*:

$$\sum_{i=1}^{m-1} \left(t^* + \frac{1}{s_i} \right).$$

Note that this is a special case of (13) in Gutjahr and Sebastiani (2008). Since $t^* \leq (\ln n)/\rho$, we obtain the more concrete bound

$$\frac{m \ln n}{\rho} + \sum_{i=1}^{m-1} \frac{1}{s_i}. \tag{3}$$

The right-hand sum is the upper bound obtained for $(1+1)$ EA and $(1+1)$ EA* from (2). Applying the fitness-level method to MMAS*, we obtain upper bounds that are only by an additive term $(m \ln n)/\rho$ larger than the corresponding bounds for $(1+1)$ EA and $(1+1)$ EA*. This additional term results from the (pessimistic) assumption that on all fitness levels MMAS* has to wait until all pheromones are frozen in order to find a better solution. We will see examples where for large ρ this bound is of the same order of growth as the bound for $(1+1)$ EA and $(1+1)$ EA*. However, if ρ is very small, the bound for MMAS* typically grows large. This reflects the long time needed for MMAS* to move away from the initial random search and to focus on promising regions of the search space.

In the article by Gutjahr and Sebastiani (2008), the proposed fitness-level method is applied in the context of the unimodal functions ONEMAX and LEADINGONES. We re-prove specialized results for MMAS*. Our purpose is to demonstrate both power and elegance of

the fitness-level method. Moreover, the following proofs suggest how to apply the method to other problems.

The probably most often studied example function in the literature on evolutionary computation is the function

$$\textsc{OneMax}(x) = x_1 + \cdots + x_n.$$

In the runtime analysis of 1-ANT on OneMax by Neumann and Witt (2006), it is shown that there exists a threshold value for ρ (in our notation basically $\rho = O(1/n^{\epsilon})$ for some small constant $\epsilon > 0$) below which no polynomial runtime is possible. In Gutjahr and Sebastiani (2008), it is shown that such a phase transition does not occur with MMAS*. The following theorem has already been proven as Proposition 5.1 by Gutjahr and Sebastiani (2008) with a more general parametrization for the pheromone borders. We present a proof for MMAS* using our simplified presentation of the fitness-level method.

Theorem 1 *The expected optimization time of MMAS* on* OneMax *is bounded from above by* $O((n \log n)/\rho)$.

Proof The proof is an application of the above-described fitness-level method with respect to the fitness-level sets $A_i = \{x \mid f(x) = i\}$, $0 \le i \le n$. In level A_i, a sufficient condition to increase the fitness is to flip a 0-bit and not to flip the other $n-1$ bits. For a specific 0-bit, this probability is $1/n \cdot (1-1/n)^{n-1} \ge 1/(en)$ with $e = \exp(1)$. As the events for all $n-i$ 0-bits are disjoint, $s_i \ge (n-i)/(en)$ and we obtain the bound

$$\sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^{n} \frac{1}{i} = O(n \log n).$$

Using (3), the upper bound $O((n \log n)/\rho)$ follows. □

Another prominent unimodal example function, proposed by Rudolph (1997), is

$$\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j,$$

whose function value equals the number of leading ones in the considered bit string x. A non-optimal solution may always be improved by appending a single one to the leading ones. LeadingOnes differs from OneMax in the essential way that the assignment of the bits after the leading ones do not contribute to the function value. This implies that bits at the beginning of the bit string have a stronger influence on the function value than bits at the end. Because of this, the methods developed by Neumann and Witt (2006) for the analysis of 1-ANT on OneMax cannot be used for analyzing 1-ANT on LeadingOnes, as these methods make particular use of the fact that all bits contribute equally to the function value. In a follow-up paper by Doerr et al. (2007), 1-ANT is studied on LeadingOnes by different techniques and it is shown that a similar phase transition behavior as on OneMax exists: for $\rho = o(1/\log n)$ (again using the notation of the present paper), the expected optimization time of 1-ANT is superpolynomially large, whereas it is polynomial for $\rho = \Omega(1/\log n)$ and even only $O(n^2)$ for $\rho = \Omega(1)$. Proposition 7.1 by Gutjahr and Sebastiani (2008) proves that this phase transition cannot occur with MMAS* on LeadingOnes. We again present a proof for MMAS* using our simplified presentation of the fitness-level method.

Theorem 2 *The expected optimization time of MMAS* on* LEADINGONES *is bounded from above by* $O(n^2 + (n \log n)/\rho)$.

Proof For $0 \leq i < n$ the $(1+1)$ EA algorithm adds an $(i+1)$-st bit to the i leading ones of the current solution with probability $s_i = (1-1/n)^i \cdot 1/n \geq 1/(en)$. Using the bound (3) results in the upper bound $((n+1)\ln n)/\rho + en^2 = O(n^2 + (n \log n)/\rho)$. □

In the following, we apply the method to arbitrary unimodal functions. We also extend the method to yield upper bounds for MMAS on unimodal functions. This extension is not immediate as MMAS may switch between solutions of equal fitness, which may prevent the pheromones from freezing. Moreover, we present lower bounds on the expected optimization time of MMAS* on all functions with unique optimum and an improved specialized lower bound for LEADINGONES. On the one hand, these bounds allow us to conclude that the fitness-level method can provide almost tight upper bounds. On the other hand, as can be seen from a more detailed analysis in Sect. 3.4, the method still leaves room for improvement using specialized techniques.

3.2 Upper bounds for unimodal functions

Unimodal functions are an important and well-studied class of fitness functions in the literature on evolutionary computation (e.g., Droste et al. 2002). For the sake of completeness, we repeat the definition of unimodality for pseudo-Boolean functions.

Definition 1 A function $f\colon \{0,1\}^n \to \mathbb{R}$ is called unimodal if there exists for each non-optimal search point x a Hamming neighbor x' where $f(x') > f(x)$.

Unimodal functions are often believed to be easy to optimize. This holds if the set of different fitness values is not too large. On the other hand, Droste et al. (2006) proved the existence of unimodal functions with many fitness values where *every* black-box algorithm needs exponential time to find an optimum.

In the following, we consider unimodal functions attaining D different fitness values for arbitrary $D \in \mathbb{N}$. Such a function is optimized by $(1+1)$ EA and $(1+1)$ EA* in expected time $O(nD)$. This bound is transferred to MMAS* by the following theorem.

Theorem 3 *The expected optimization time of MMAS* on a unimodal function attaining D different fitness values is* $O((n + (\log n)/\rho)D)$.

Proof Because of the unimodality, there is for each current search point x a better Hamming neighbor x' of x in a higher fitness-level set. The probability for $(1+1)$ EA (or, equivalently, MMAS* with all pheromone values at a border) to produce x' in the next step is at least $1/(en)$. By (3), this completes the proof. □

In order to freeze pheromones after t^* steps without an improvement, it is essential that equally good solutions are rejected. The fitness-level argumentation, including the bound from (3), cannot directly be transferred to MMAS as switching between equally fit solutions can prevent the system from freezing. Nevertheless, we are able to prove a similar upper bound on the optimization time of MMAS that is by a factor of n^2 worse than the bound for MMAS* in Theorem 3 if $\rho = O((\log n)/n)$. Despite the factor n^2, Theorem 4 yields a polynomial bound for MMAS if and only if Theorem 3 yields a polynomial bound for MMAS*.

Theorem 4 *The expected optimization time of MMAS on a unimodal function attaining D different fitness values is $O(((n^2 \log n)/\rho)D)$.*

Proof We only need to show that the expected time for an improvement of the best-so-far solution is at most $O((n^2 \log n)/\rho)$. The probability that MMAS produces within $O((\log n)/\rho)$ steps a solution being at least as good as (not necessarily better than) the best-so-far solution x^* is $\Omega(1)$ since after at most $(\ln n)/\rho$ steps without exchanging x^* all pheromone values have touched their borders and then the probability of rediscovering x^* is $(1-1/n)^n = \Omega(1)$. We now show that the conditional probability of an improvement if x^* is replaced is $\Omega(1/n^2)$.

Let $x_1, \ldots, x_m$ be an enumeration of all solutions with fitness values equal to the best-so-far fitness value. Due to unimodality, each x_i, $1 \le i \le m$, has some better Hamming neighbor y_i; however, the y_i need not be disjoint. Let X and Y denote the event of generating some x_i or some y_i, respectively, in the next step. In the worst case, $y_1, \ldots, y_m$ are the only possible improvements, hence the theorem follows if we can show $\text{Prob}(Y \mid X \cup Y) \ge 1/n^2$, which is implied by $\text{Prob}(Y) \ge \text{Prob}(X)/(n^2-1)$.

If $p(x_i)$ is the probability to construct x_i, we have $p(x_i)/p(y_i) \le (1-\frac{1}{n})/\frac{1}{n} = n-1$ as the constructions only differ in one bit. Each y_i may appear up to n times in the sequence $y_1, \ldots, y_m$, hence $\text{Prob}(Y) \ge \frac{1}{n}\sum_{i=1}^{m} p(y_i)$ and

$$\text{Prob}(X) = \sum_{i=1}^{m} p(x_i) \le (n-1) \cdot \sum_{i=1}^{m} p(y_i) \le n(n-1) \cdot \text{Prob}(Y).$$

Therefore, $\text{Prob}(Y) \ge \text{Prob}(X)/(n^2-1)$ follows. □

Theorems 3 and 4 show that the expected optimization times of both MMAS and MMAS* are polynomial for all unimodal functions as long as $D = \text{poly}(n)$ and $\rho = 1/\text{poly}(n)$. The result for MMAS has extensive implications.

Recall that MMAS only differs from MMAS* by accepting equally good solutions. The only difference between MMAS and 1-ANT lies in the pheromone update mechanism: MMAS reinforces the best-so-far solution in every step, while 1-ANT only performs a pheromone update in case the best-so-far solution is exchanged. 1-ANT suffers from a phase transition behavior with exponential runtimes for simple unimodal functions if ρ is a small inverse polynomial (Neumann and Witt 2006; Doerr et al. 2007). However, Theorems 3 and 4 prove that such a phase transition can occur neither with MMAS* nor with MMAS. Therefore, the phase transition behavior of 1-ANT must result from the different pheromone update mechanism, contrary to the conjecture by Gutjahr (2007).

3.3 A general lower bound

The function ONEMAX is probably the simplest function with unique global optimum. The upper bound $O((n \log n)/\rho)$ from Theorem 1 is never better than $\Theta(n \log n)$, which describes the expected runtime of $(1+1)$ EA and $(1+1)$ EA* on ONEMAX. At the moment, we are not able to show a matching lower bound $\Omega(n \log n)$ on the expected optimization time of MMAS*; however, we can show that the expected optimization time is growing with respect to $1/\rho$ as the upper bound suggests. We state our result in a more general framework: as known from the considerations by Droste et al. (2002), the mutation probability $1/n$ of $(1+1)$ EA is optimal for many functions including ONEMAX. One argument is that the probability mass has to be quite concentrated around the best-so-far solution to

allow (1 + 1) EA to rediscover the last accepted solution with good probability. Given a mutation probability of $\alpha(n)$, this probability of rediscovery equals $(1-\alpha(n))^n$, which converges to zero unless $\alpha(n) = O(1/n)$. The following lemma exploits the last observation for a general lower bound on the expected optimization time of both MMAS and MMAS*.

Theorem 5 *Let* $f\colon \{0,1\}^n \to \mathbb{R}$ *be a function with a unique global optimum. Choosing* $\rho = 1/\mathrm{poly}(n)$, *the expected optimization time of MMAS and MMAS* on* f *is* $\Omega((\log n)/\rho - \log n)$.

Proof W.l.o.g., 1^n is the unique optimum. If, for each bit, the success probability (defined as the probability of creating a one) is bounded from above by $1 - 1/\sqrt{n}$, then the solution 1^n is created with only an exponentially small probability of at most $(1-1/\sqrt{n})^n \le e^{-\sqrt{n}}$. Using the uniform initialization and the pheromone update formula of MMAS and MMAS*, the success probability of a bit after t steps is bounded from above by $1-(1-\rho)^t/2$. Hence, all success probabilities are bounded as desired within $t := (1/\rho - 1)\cdot(\ln(n/4)/2) = \Omega((\log n)/\rho - \log n)$ steps since

$$1 - \frac{1}{2}(1-\rho)^t \le 1 - \frac{e^{-(\ln n - \ln 4)/2}}{2} = 1 - \frac{1}{\sqrt{n}}.$$

Since $\rho = 1/\mathrm{poly}(n)$, and therefore $t = \mathrm{poly}(n)$, the total probability of creating the optimum in t steps is still at most $te^{-\sqrt{n}} = e^{-\Omega(\sqrt{n})}$, implying the lower bound on the expected optimization time. □

Hence, the expected optimization time of MMAS* with $\rho = 1 - \Omega(1)$ on ONEMAX is bounded by $\Omega((\log n)/\rho)$. It is an open problem to show matching upper and lower bounds. We conjecture that the lower bound for ONEMAX is far from optimal and that $\Omega(n \log n + n/(\rho \log(2/\rho)))$ holds. A corresponding bound for the LEADINGONES function will be given in the next section.

3.4 Improved upper and lower bounds for LEADINGONES

In this section, we improve the bound of Theorem 2 to $O(n^2 + n/\rho)$ and beyond. Thereby, we show that it is not necessary for MMAS* to freeze the best-so-far solution in pheromone on every fitness level. Moreover, the forthcoming Theorem 6 applies to MMAS as well, where a straight application of the fitness-level method breaks down.

In the remainder of this section, it is crucial to distinguish two states for bits during the optimization process on LEADINGONES (this idea already appears in the analysis of 1-ANT on this function by Doerr et al. 2007). Suppose that the current best-so-far LEADINGONES-value equals k. We call the bits $1, \ldots, k$ *bits in increasing state* and the bits $k+2, \ldots, n$ *bits in random state*. No special attention is paid to the 0-bit at position $k+1$.

Our notions are justified as follows: since, due to the structure of LEADINGONES, all future best-so-far solutions must contain at least k leading ones, the pheromone values of the 1-edges (i.e., the success probabilities) corresponding to the first k bits are monotonically increasing in each iteration until the border $1 - 1/n$ is reached. The bits $k+2, \ldots, n$ have never had an impact on the LEADINGONES-value. Due to the symmetry of the construction procedure, it is intuitively clear that such a bit is unbiasedly set to 1 with probability 1/2 in the next constructed solution, hence the bit is purely random. A formal proof of this will be given later.

We state two improved upper bounds for LEADINGONES. The first bound is better than the bound from Theorem 2 by a factor of order $\log n$ if $\rho = O(1/n)$. The second bound is better than the first one by another factor of order $\log n$ if $\rho \le n^{-(1+\Omega(1))}$.

Theorem 6 *The expected optimization time of MMAS and MMAS* on* LEADINGONES *is bounded by* $O(n^2 + n/\rho)$ *and* $O(n^2 \cdot (1/\rho)^{\varepsilon} + \frac{n/\rho}{\log(1/\rho)})$ *for any constant* $\varepsilon > 0$.

Proof Let ℓ be a positive real value defined later. Divide a run into phases. Phase i, $0 \le i \le n$, ends when the following two conditions are met:

1. The best-so-far LEADINGONES-value is at least i.
2. And for $0 \le j \le i-1$ it holds that bit $i-j$ has been reinforced at least $(j+1)\ell$ times after entering the increasing state.

We now show that the expected time spent in Phase i, $1 \le i < n$, is bounded by $O(\ell + n \cdot e^{5/(\ell\rho)})$. This implies that the expected optimization time is bounded by $O(n\ell + n^2 \cdot e^{5/(\ell\rho)})$. Choosing $\ell := 5/\rho$ yields the first bound, and $\ell := 5/(\varepsilon\rho \ln(1/\rho))$ yields the second bound.

Consider the first generation in Phase i for $1 \le i < n$. As Phase $i-1$ has been finished successfully, the first $i-1$ bits are in increasing state and bit $i-j$, $1 \le j \le i-1$, has been reinforced at least $j\ell$ times. This implies that the success probability for this bit is at least $\min\{1-1/n, 1-(1-\rho)^{j\ell}\} \ge (1-1/n)\cdot(1-(1-\rho)^{j\ell})$. The success probability for bit i is at least $1/n$; hence, the probability to create a solution with at least i leading ones is at least

$$\left(1-\frac{1}{n}\right)^{i-1} \cdot \frac{1}{n} \cdot \prod_{j=1}^{i-1}\left(1-(1-\rho)^{j\ell}\right) \ge \frac{1}{en} \cdot \prod_{j=1}^{\infty}\left(1-(1-\rho)^{j\ell}\right).$$

We estimate the $\prod$-term on the right-hand side and deal with the first factors separately. Define $m := \max\{j \in \mathbb{N}_0 \mid (1-(1-\rho)^{j\ell}) \le 1/2\}$ such that the factors with $j \le m$ are at most $1/2$. We estimate these terms from below by a linear function as follows. By induction, we show that $(1-(1-\rho)^k) \le 1/2$ implies $(1-(1-\rho)^k) \ge k\rho/2$ for $k \in \mathbb{N}_0$. This claim is obvious for $k = 0$. Assume that it holds for $k-1$, then

$$\begin{aligned} 1-(1-\rho)^k &= 1-(1-\rho)^{k-1}(1-\rho) = 1-(1-\rho)^{k-1}+\rho(1-\rho)^{k-1} \\ &\ge 1-(1-\rho)^{k-1}+\frac{\rho}{2} \ge \frac{k\rho}{2}. \end{aligned}$$

Hence, the jth factor, $j \le m$, is bounded from below by $j\ell\rho/2$. Solving the equation $1-(1-\rho)^x = 1/2$ for x yields the bound $m\ell \le -(\ln 2)/\ln(1-\rho) \le (\ln 2)/\rho$. Let $\alpha := (\ln 2)/(\ell\rho) \ge m$ and assume w.l.o.g. that $\alpha \in \mathbb{N}$, then the product of the first m factors is at least

$$\begin{aligned} \prod_{j=1}^{m}\left(1-(1-\rho)^{j\ell}\right) &\ge \prod_{j=1}^{m}\frac{j\ell\rho}{2} \ge \prod_{j=1}^{\alpha}\frac{j\ell\rho}{2} = \alpha! \cdot \left(\frac{\ell\rho}{2}\right)^{\alpha} \\ &\ge \left(\frac{\alpha}{e}\right)^{\alpha} \cdot \left(\frac{\ell\rho}{2}\right)^{\alpha} = \left(\frac{\ln 2}{2e}\right)^{(\ln 2)/(\ell\rho)} \ge e^{-1/(\ell\rho)}. \end{aligned}$$

All factors with index $j > m$ are at least $1/2$ and can be estimated as follows. We exploit $1 + x \le e^x$ for $x \in \mathbb{R}$ and $1 - x \ge e^{-2x}$ for $0 \le x \le 1/2$.

$$\begin{aligned}\prod_{j=m+1}^{\infty} \bigl(1-(1-\rho)^{j\ell}\bigr) &\ge \prod_{j=m+1}^{\infty} \exp\bigl(-2(1-\rho)^{j\ell}\bigr)\\ &\ge \exp\left(-2\sum_{j=0}^{\infty}(1-\rho)^{j\ell}\right) = \exp\left(-\frac{2}{1-(1-\rho)^{\ell}}\right)\\ &\ge \exp\left(-\frac{2}{1-e^{-\ell\rho}}\right).\end{aligned}$$

If $\ell\rho > 1$, the resulting bound is $\Omega(1)$. If $\ell\rho \le 1$ then $1 - e^{-\ell\rho} \ge \ell\rho/2$, and we obtain the bound $\exp(-4/(\ell\rho))$. Together, the probability of creating a solution with at least i leading ones is $\Omega(1/n \cdot e^{-5/(\ell\rho)})$, and the expected waiting time for this event is $O(n \cdot e^{5/(\ell\rho)})$. This fulfills the first goal of Phase i. After an additional waiting time of ℓ steps, the second goal is fulfilled as well as the first i bits are reinforced in every step and bit $i - j$, $1 \le j \le i - 1$, has already been reinforced at least $j\ell$ times before entering Phase i. Therefore, Phase i is completed in expected time $O(\ell + n \cdot e^{5/(\ell\rho)})$ as claimed. □

It is interesting that an almost tight lower bound can be derived. The following theorem shows that the expected optimization time of MMAS* on LEADINGONES is $\Omega(n^2 + n/(\rho\log(2/\rho)))$, hence never better than $\Omega(n^2)$. (We write $\log(2/\rho)$ in the lower bound instead of $\log(1/\rho)$ to make the bound $\Omega(n^2)$ for any constant ρ and to avoid division by 0.) Apart from this technical detail, the lower bound is tight with the upper bounds from Theorem 6 for $\rho = \Omega(1/n)$ and $\rho \le n^{-(1+\Omega(1))}$, hence for almost all ρ. The proof is lengthy; however, for the case of large ρ, one essential idea is easy to grasp: already in the early stages of the optimization process, many (more precisely $\Omega(n)$) success probabilities reach their lower borders $1/n$, and the corresponding bits are set to 0. To "flip" such a bit, events of probability $1/n$ are necessary. This can be transformed into the lower bound $\Omega(n^2)$ on the expected optimization time.

Theorem 7 *Choosing* $\rho = 1/\mathrm{poly}(n)$, *the expected optimization time of MMAS* on* LEADINGONES *is bounded from below by* $\Omega(n^2 + \frac{n/\rho}{\log(2/\rho)})$.

To prove the preceding theorem, we recall the distinction of bits in increasing and random state. We have already motivated why the latter bits are set to 1 with probability exactly $1/2$ even if the corresponding pheromone values on 1-edges might change from step to step. This is equivalent to a random, time-dependent success probability. The probability of setting the bit to 1 corresponds to the expected success probability.

Lemma 1 *Let the random variable* P_t *denote the probability of setting a bit in random state to* 1 *at time* t. *Then the bit is set to* 1 *at time* t *with probability* $E(P_t)$.

Proof We first observe that the underlying state space is countably infinite. This holds since the initial pheromone value is fixed to $1/2$ and each time step allows at most two new, i.e., previously unvisited, pheromone values.

Conditioned on the fact that the random P_t equals p, the bit is set to 1 at time t with probability p. By the law of total probability, the unconditional probability of setting the bit

to 1 equals

$$\sum_p p \cdot \text{Prob}(P_t = p) = E(P_t).$$

□

If we can prove that $E(P_t) = 1/2$ holds for all t in which the bit is in random state, our analysis of the random-state bits is complete.

Lemma 2 *Let P_t be as defined in Lemma 1. Then* $\text{Prob}(P_t = p) = \text{Prob}(P_t = 1 - p)$ *for* $0 \le p \le 1$, *hence* $E(P_t) = 1/2$. *This even holds if the bit is no longer in random state at time $t + 1$.*

Proof We inductively show for all $t' \le t$ the following stronger statement: for each vector of probabilities $(p_1, \ldots, p_{t'})$ it holds

$$\text{Prob}\big((P_1, \ldots, P_{t'}) = (p_1, \ldots, p_{t'})\big) = \text{Prob}\big((P_1, \ldots, P_{t'}) = (1 - p_1, \ldots, 1 - p_{t'})\big).$$

This symmetry will imply the lemma since

$$\begin{aligned} E(P_t) &= \sum_{0 \le p \le 1} p \cdot \text{Prob}(P_t = p) \\ &= \sum_{0 \le p < 1/2} \big(p + (1 - p)\big) \cdot \text{Prob}(P_t = p) + \frac{1}{2} \cdot \text{Prob}(P_t = 1/2) = \frac{1}{2}. \end{aligned}$$

Due to the pheromone initialization, we have $P_0 = 1/2$ with probability 1. For the induction step, let p be arbitrary but fixed. Conditioned on $P_t = p$, there are two possible pheromone values on the 1-edges of the bit at time $t + 1$, or, equivalently, for the random P_{t+1}: either $d(p) := \max\{1/n, (1 - \rho)p\}$ or $i(p) := \min\{1 - 1/n, (1 - \rho)p + \rho\}$. Depending on whether the best-so-far solution is exchanged in the considered step, the probability of going from p to $d(p)$ (or $i(p)$) takes one of the three values 0, $1 - p'$ and 1 (or 0, p', 1). Actually, given that the best-so-far solution is not exchanged, the probability of increasing or decreasing the value is itself random and depends on the bit's setting in the last best-so-far solution. However, which of the three probabilities is used is independent of the random P_t even if the bit happens to leave the random state in the considered step. The event that a best-so-far solution with k ones is replaced is *equivalent* to the event that the first $k + 1$ bits are all set to 1; hence, the setting of the bits $k + 2, \ldots, n$ is independent of this decision. Note that even in the step where such a bit leaves the random state, it does not affect the decision whether to replace the best-so-far solution.

We are left with two cases. If an exchange of the best-so-far solution happens from time t to $t + 1$ then, given that $P_t = p$, P_{t+1} takes the value $d(p)$ with probability $1 - p$ and the value $i(p)$ with probability p. Conditioned on the event $P_t = 1 - p$, P_{t+1} takes the value $d(1 - p)$ with probability p and the value $i(1 - p)$ with probability $1 - p$. By the induction hypothesis, both events that we condition on are equiprobable. Moreover, $p \to 1 - p$ is a bijection on the probability space for P_t. Noting that $d(1 - p) = 1 - i(p)$ and $i(1 - p) = 1 - d(p)$ and using the law of total probability, we have $\text{Prob}(P_{t+1} = i(p)) = \text{Prob}(P_{t+1} = 1 - i(p))$, and, accordingly, $\text{Prob}(P_{t+1} = d(p)) = \text{Prob}(P_{t+1} = 1 - d(p))$.

If no exchange of the best-so-far solution happens then P_t is increased or decreased with probability P_{t^*} and $1 - P_{t^*}$, respectively, where $t^* < t$ is the time of the last exchange (or 0 for the initial step). We additionally consider an arbitrary fixed p^*. Conditioned on the event

$(P_{t^*} = p^*) \wedge (P_t = p)$, we have $\text{Prob}(P_{t+1} = i(p)) = p^*$ and $\text{Prob}(P_{t+1} = d(p)) = 1 - p^*$. Analogously, if $(P_{t^*} = 1 - p^*) \wedge (P_t = 1 - p)$, we have $\text{Prob}(P_{t+1} = i(1-p)) = 1 - p^*$ and $\text{Prob}(P_{t+1} = d(1-p)) = p^*$. Using the induction hypothesis, both conditions are again equiprobable. Now $(p^*, p) \to (1 - p^*, 1 - p)$ is a bijection on the probability space for (P_{t^*}, P_t). By the law of total probability, we have, also in this case, $\text{Prob}(P_{t+1} = i(p)) = \text{Prob}(P_{t+1} = 1 - i(p))$ and $\text{Prob}(P_{t+1} = d(p)) = \text{Prob}(P_{t+1} = 1 - d(p))$. Altogether, the induction follows. □

We summarize what has been derived so far: the bits in random state are independently set to 1/2 in all constructed solutions until they leave the random state, where the independence follows directly from the construction procedure of MMAS*. This will allow us to treat the random bits in the same way as done in the analysis of (1 + 1) EA on LEADINGONES by Droste et al. (2002). To prove a lower bound on the expected optimization time, we have to take into account the so-called "free-rider" phenomenon. Given that the current LEADINGONES-value increases from k to $k + i$, where $i \geq 2$, the bits at positions $k + 2, \ldots, k + i$ are called free riders since they are collected "for free" in the improvement (contrary to the bit at position $k + 1$, whose setting is necessarily 1). To prove a lower bound, we have to control the number of free riders in improving steps. By the above analysis, the probability of i free riders is at most 2^{-i-1} per improvement since having i free riders is equivalent to setting bits $k + 2, \ldots, k + i$ to 1 and setting bit $k + i + 1$ to 0. Clearly, if $k + i > n$, having i free riders is even impossible. Altogether, the number of free riders in improving steps is now easy to control.

Lemma 3 *After the first $n/12$ improvements the* LEADINGONES*-value of the best-so-far solution is still less than $n/2$ with probability $1 - 2^{-\Omega(n)}$.*

Proof With probability at least $1 - 2^{-n/4}$ the initial LEADINGONES-value is at most $n/4$. Working under this condition, we estimate the number of free riders in improving steps. Modeling the free rider decisions as at most $n/2$ independent trials as in Droste et al. (2002), the number of free riders in k improvements is at most $2k$ with probability at least $1 - e^{-k/16}$. Hence, the probability that $n/12$ improvements increase the LEADINGONES-value by at least $n/4$ is $2^{-\Omega(n)}$. □

The main obstacle in the proof of Theorem 7 is related to the success probabilities of bits that leave the random state to become the leftmost zero-bit. For the lower bound $\Omega(n^2)$ given in the theorem, it would be nice to have many times, at best $\Omega(n)$ times, bits whose random success probabilities equal the lower border $1/n$ when they become the leftmost zero-bit. Then setting the bit to 1, as necessary for an improvement, would require $\Omega(n)$ steps in expectation, and $\Omega(n)$ such situations would take an expected number of $\Omega(n^2)$ steps. With strict selection a weaker condition is sufficient. For MMAS* a lower bound $\Omega(n^2)$ follows if $\Omega(n)$ bits reach success probabilities at the lower border at least once while being in random state.

Lemma 4 *Consider a point of time where the best-so-far* LEADINGONES*-value is less than $n/2$. If there are $\Omega(n)$ bits at the last $n/2$ positions whose success probabilities have reached their lower border $1/n$ at least once up to now then the remaining expected optimization time for MMAS* is $\Omega(n^2)$.*

Proof The last $n/2$ bits are all in random state. Suppose a bit in random state has reached the lower border $1/n$ on the success probability. For each subsequent improvement, it can

become necessary to set such a bit to 1, which has probability $1/n$ provided that the success probability is still at the border. However, the success probability of such a bit might increase again. Therefore, in each of the remaining improvements, we distinguish the events whether setting the bit to 1 is relevant for an improvement or not. If the bit is not relevant since it is still right of the leftmost zero after the improvement, its success probability does not change with probability $1 - 1/n$. Hence, by Chernoff bounds (Motwani and Raghavan 1995), it holds with probability $1 - 2^{-\Omega(n)}$ that after $O(n)$ improvements (note that there are at most n improvements) there are still $\Omega(n)$ success probabilities equal to $1/n$ left. We assume this to happen for some number cn, $c > 0$ a constant, of bits and call these bits *difficult*. Pessimistically ignoring all other bits, an improvement adds an expected number of at most $1 + 1/n < 2$ difficult bits to the leading ones of the best-so-far solution. The expected number of difficult bits gained in $cn/4$ improvements is therefore less than $cn/2$, and this number is less than cn with probability at least $1/2$. Hence, the remaining optimization time is $\Omega(n^2)$ with probability at least $1/2 - 2^{-\Omega(n)}$, and, therefore, also in expectation. □

In order to complete the proof of the lower bound $\Omega(n^2)$ we have to show that the preconditions of Lemma 4 are fulfilled with high probability and many bits in random state reach the lower border for their success probability. However, so far we only know that in expectation, at least half of the bits in random state have a success probability that is bounded from above by $1/2$. The probability, however, might still be very close to $1/2$, which is the actual value in the initial step.

Recall that in generations where the best-so-far solution is not replaced the success probability at every bit is continuously increased or decreased. Intuitively, chances to reach upper or lower borders are high if some time elapses between improving steps. The following lemma bounds the average time between improvements from below. If all success probabilities are close to $1/2$, then MMAS* behaves almost like random search. Hence, to obtain substantial progress in the optimization, it must take some time between two improvements for the pheromone values of the newly gained bits in increasing state to come up and reach their upper border $1 - 1/n$. In the following, we will bound the average time between improving steps by $\Omega(1/(\rho \log(2/\rho)))$, where the average is taken over $\Theta(\log(2/\rho))$ improvements. Thereby, we also prove a lower bound $\Omega(n/(\rho \log(2/\rho)))$ for the expected optimization time on LEADINGONES, corresponding to the second term in the bound from Theorem 7.

Lemma 5 *Let $\rho = 1/\mathrm{poly}(n)$. Then there is some constant $c > 0$ such that after each step improving the* LEADINGONES*-value, the number of steps required by MMAS* for $c \log(2/\rho)$ further improvements is $\Omega(1/\rho)$ with probability $\Omega(1)$. Furthermore, the expected optimization time is $\Omega(n/(\rho \log(2/\rho)))$.*

Proof W.l.o.g., we assume $1/\rho$ to be growing with n since otherwise the first statement of the theorem is trivial and the second one follows from Lemma 3.

We will show that a phase of $s := 1/\rho - 1$ steps following an arbitrary improving step contains an expected number of at most $O(\log(2/\rho))$ further improvements. This will imply the lemma for the following reasons. First, we apply Markov inequality on the random number of improvements in s steps, which is therefore bounded by $O(\log(2/\rho))$ with probability at least $1/2$. This is the first statement of the lemma. Second, we sum up the expected number of improvements in $c'n/\log(2/\rho)$ phases of length s, where $c' > 0$ is another constant. This yields a total number of $c'n/(\rho \log(2/\rho)) - c'n/\log(2/\rho) = \Omega(n/(\rho \log(2/\rho)))$ steps. If c' is small enough, the total expected number of improvements in this number of steps is less than $n/24$ and, by Markov inequality, less than $n/12$ with probability at least $1/2$. By

Lemma 3, with probability $1 - 2^{-\Omega(n)}$, $n/12$ improvements are not enough to reach even a LEADINGONES-value of at least $n/2$. The sum of the failure probabilities is $1/2 + 2^{-\Omega(n)}$. Using the law of total probability, we obtain the second statement of the lemma.

We are left with the claim. Let k be the LEADINGONES-value at the beginning of the phase of length s. For some large enough variable γ to be chosen later, we concentrate on the block of $r := r(\gamma) := \gamma \log(2/\rho)$ bits at positions $k+2, \ldots, k+2+r-1$, i.e., following the leftmost 0-bit at the beginning of the phase. If all r bits are in increasing state by the end of the phase, the phase is called successful. Clearly, if the phase is unsuccessful, it contains, due to strict selection, at most r improvements, which is the event whose probability has to be bounded from above.

In the beginning of the phase, all bits of the block are in random state. We apply Lemma 2 to bound the number of block bits whose success probability is at most $1/2$ in the beginning of a phase. The probability is at least $1/2$ for a single bit, and by Chernoff bounds, with a failure probability of most $2^{-r/12}$, less than $r/4$ success probabilities are at most $1/2$. We assume $r/4$ bits with this property and estimate the probability of setting all these bits to 1 simultaneously in at least one improving step by the end of the phase (which is necessary for the phase to be successful). The success probability of a bit with initial pheromone value $1/2$ is still at most $1 - (1-\rho)^t/2$ if it has been only in increasing state for t steps. The total number of iterations in the phase is $1/\rho - 1$. Hence, by the end of the phase, all considered success probabilities are at most $1 - (1-\rho)^{1/\rho - 1}/2 \leq 1 - e^{-1}/2 < 0.82$. The probability of a single improving step setting the $r/4$ bits to 1 is therefore at most $(0.82)^{r/4} \leq 2^{-r/14}$. By the union bound, the probability of this happening within s steps is at most $s2^{-r/14} = 2^{\log(2/\rho) - \gamma \log(2/\rho)} \leq 2^{-r/15}$ if γ is large enough. More precisely, there is a constant γ_0 such that the statement holds for all $\gamma \geq \gamma_0$. Additionally taking into account the above failure probability, the probability of the phase being successful is at most $2^{-r/12} + 2^{-r/15} \leq 2^{-r/20}$ if γ is large enough.

Using the last observation, we finally bound the expected number of improvements per phase. Let I denote the random number of improvements. Then $E(I) = \sum_{v \geq 1} \text{Prob}(I \geq v) \leq \sum_{i=0}^{\infty} r_0 \, \text{Prob}(I \geq i r_0)$, where now $r_0 := \gamma_0 \log(2/\rho)$ with γ_0 as in the last paragraph. In the definition of r, we can choose γ arbitrarily large. Applying the result of the last paragraph for $\gamma := i\gamma_0$, we obtain that $\text{Prob}(I \geq i\gamma_0 \log(2/\rho))$ is at most $2^{-i\gamma_0 \log(2/\rho)/20}$ for all $i \geq 1$. Since $2^{-\gamma_0 \log(2/\rho)/20} = 1 - \Omega(1)$, we have $\sum_{i=1}^{\infty} 2^{-i\gamma_0 \log(2/\rho)/20} = O(1)$, and, altogether, $E(I) \leq r \, \text{Prob}(I \geq 0) + r \cdot O(1) = O(r) = O(\log(2/\rho))$. □

We remark that Lemma 5 does not contain a statement on the expected time between two improvements. Rather, an amortized consideration over $\Omega(\log(2/\rho))$ improvements is given. The reason is that the very first improvements are easy to obtain even for random search.

Now we are left with the proof that $\Omega(n)$ bits in random state reach their lower border on the success probabilities during the first $n/12$ improvements. To this end, we consider a bit in random state and again denote by P_t its random success probability at time t. All the following estimations hold until the bit leaves the random state.

We are interested in the stopping time $T_{\min} := \min\{t \geq 0 \mid P_t = 1/n\}$. Before this time, the stochastic process describing the P_t-value equals the same stochastic process with the lower border $1/n$ on the success probability removed (i.e., $P_{t+1} = \min\{1 - 1/n, (1-\rho)P_t + \rho\}$ if P_t is increased, and $P_{t+1} = (1-\rho)P_t$ otherwise. In the following, we only consider this modified process, however, still denote it by P_t, $t \geq 0$.

The P_t-process is not necessarily Markovian. Recalling the reasoning in the proof of Lemma 2, P_{t+1} is obtained either by making a binary decision with probability P_t or by

increasing or decreasing the value according to the result of an earlier decision when the best-so-far solution is not exchanged while going from time t to time $t+1$. Let the random times T_i, $i \geq 0$, describe the points of time in which the best-so-far solution is exchanged; in particular, $T_0 = 0$ denotes the initialization of the best-so-far solution. Since the bit describing the P-process is assumed to stay in random state, the lengths $L_i := T_{i+1} - T_i$ of the time spans between exchanges are independent of the P_t-values. We consider the process $M_i := P_{T_i}$, $i \geq 0$, which is just a macroscopic view of the P-process at the exchange points. Since exactly the behavior at the exchange points is relevant for the decision whether to increase or decrease the pheromone, the M-process is Markovian. Furthermore, it can be characterized as follows.

Lemma 6 *The M_i form a supermartingale, i.e., $E(M_{i+1} \mid M_0, \ldots, M_i) \leq M_i$.*

Proof Since the process is Markovian, it suffices to condition on M_i. Given $M_i = p$, the value is increased L_i times with probability p and otherwise decreased L_i times. Hence, by the definition of the M-process,

$$\begin{aligned} &E(M_{i+1} \mid M_i = p) \\ &\quad = (1-p) \cdot \left(M_i \cdot (1-\rho)^{L_i}\right) + p \cdot \min\left\{1 - \frac{1}{n}, 1 - (1-M_i) \cdot (1-\rho)^{L_i}\right\} \\ &\quad \leq (1-p) \cdot p \cdot (1-\rho)^{L_i} + p - (1-p) \cdot p \cdot (1-\rho)^{L_i} = p = M_i. \end{aligned}$$

□

We are still looking for a bound on $T_{\min}$. This will be obtained from the stopping time $T_{\text{M-min}} := \min\{i \geq 0 \mid M_i \leq 1/n\}$ by incorporating our knowledge on the average time between two improvements from Lemma 5. The following lemma, which is adapted from Jansen (2008), states bounds on stopping times for martingale processes in a quite general setting.

Lemma 7 *Consider a stochastic process $\{X_t\}_{t \geq 0}$ on a bounded subset of $\mathbb{R}_0^+$. Let $\mathfrak{F}_t$ denote $X_0, \ldots, X_t$, $E_{t+1} := E(X_{t+1} \mid \mathfrak{F}_t)$ and*

$$\Delta_{t+1} = E\left((X_{t+1} - E_{t+1})^2 \cdot \mathbb{1}\{X_{t+1} < E_{t+1}\} \mid \mathfrak{F}_t\right)$$

with $\mathbb{1}\{F\}$ being the indicator function of an event F.

Given $\alpha \in \mathbb{R}$, define $T := \min\{t : |X_t - X_0| \geq \alpha \mid X_0\}$. If

1. *$\{X_t\}_{t \geq 0}$ is a supermartingale (i.e., $E(X_{t+1} \mid \mathfrak{F}_t) \leq X_t$) and*
2. *There exist $\delta > 0$ and $\ell \geq 1$ such that $\sum_{k=t}^{t+\ell} \Delta_k / \ell \geq \delta$ for all $1 \leq t \leq T - \ell$*

then

- $E(T) \leq \ell + (2X_0 + \alpha) \cdot \alpha / \delta$ *and*
- $\text{Prob}(X_T < X_0) \geq \alpha / (\alpha + E(X_0 - X_T \mid X_T < X_0))$.

The proof of this lemma is deferred to the Appendix. Note that the second condition of the lemma is implied if we have a lower bound on the single-step one-sided variance according to $\Delta_t \geq \delta$ for all $1 \leq t \leq T$. We state the more general condition since, in the forthcoming analysis, Lemma 5 yields only a statement on the amortized one-sided variance in $\Theta(\log(2/\rho))$ steps.

We now study the M-process to bound $T_{\text{M-min}}$. Note that $M_0 = P_0 = 1/2$ since our consideration starts with the initialization of MMAS*.

Lemma 8 *Let $\epsilon > 0$ be an arbitrarily small constant. Then $T_{\text{M-min}} = O(n^{\epsilon} \log^2(2/\rho))$ with probability $\Omega(1)$.*

Proof To apply Lemma 7, we need a bound on the variance of the M-process. However, this variance heavily depends on the current M-value since the process is not homogeneous with respect to place. The closer the value is to one of its borders 0 and $1 - 1/n$, the smaller the expected displacement from the current position becomes. We therefore split the run of the process into different phases.

Phase i, $i \geq 1$, starts at the first point of time (counting the steps of the M-process) where the current M-value is in the interval $(\ell_i, u_i] := (1/n^{(i+1)\gamma}, 1/n^{i\gamma}]$, where $\gamma = \epsilon/2$. Phase 0 is different and starts when the value is in the interval $(\ell_0, u_0] := (1/n^{\gamma}, 1/2]$ for the first time. Let $\alpha_i := u_i - \ell_i$, $i \geq 0$, denote the length of the interval. We additionally introduce an infinitely long error phase, which is entered from Phase i when the current M-value exceeds $u_i + \alpha_i$. Note that, due to $M_0 = 1/2$, we start with a non-error phase of positive index. Obviously, if we can prove that Phase $1/\gamma$ starts after $O(n^{\epsilon} \log^2(2/\rho))$ steps with probability $\Omega(1)$, the lemma follows.

The proof strategy is now to show that with probability $\Omega(1)$, the M-process passes through phases with increasing index without falling into the error phase. Depending on the size of ρ, some phases might be empty. We therefore estimate the number of steps until Phase at least $i + 1$ begins when Phase i has started. Let X_{0_i} be the M-value at the start of Phase i. In terms of Lemma 7, we are interested in the first point of time T_{i+1} in which the distance α_i from X_{0_i} has been reached and, in particular, in the probability $\text{Prob}(X_{T_{i+1}} \leq X_{0_i})$. Since for $E^- := E(X_{0_i} - X_{T_{i+1}} \mid X_{T_{i+1}} < X_{0_i})$ we have $E^- \leq X_{0_i} \leq u_i \leq 2\alpha_i$ (for n large enough), it follows $\alpha/(E^- + \alpha) \geq 1/3$. Hence, by the lemma, $X_{T_{i+1}} \leq X_{0_i} - \alpha_i$ holds with probability at least $1/3$. The latter is sufficient to start Phase at least $i + 1$ without falling into the error phase. Assuming $X_{T_{i+1}} \leq X_{0_i} - \alpha_i$ and repeating this argument at most $1/\gamma$ times, we conclude that Phase $1/\gamma$ starts after an expected number of at most $E^* := \sum_{i=1}^{1/\gamma} E(T_i)$ steps with probability at least $(1/3)^{1/\gamma}$. Additionally using Markov inequality, the total number of steps to start Phase $1/\gamma$ is at most $2E^*$ with probability at least $(1/3)^{1/\gamma+1} = \Omega(1)$ since γ is constant. Hence, it suffices to prove a bound on E^* using Lemma 7. This will be done by bounding Δ-values and, thereby, $E(T_i)$, separately for all phases. Each Δ-value can be bounded from below by identifying a possible decrease of the current state and estimating the probability and the decrease from below.

Let us pessimistically ignore that Phase i may be left in favor of a higher-index phase before time T_{i+1}. As the M-value is always at most $1/2$, the M-process has probability at least $1/2$ of decreasing its value. Recall that the M-process records the P-value of a random-state bit only at exchanges of the best-so-far solution. By Lemma 5, a sequence of $\ell = \Theta(\log(2/\rho))$ steps of the M-process takes $\Omega(1/\rho)$ steps of MMAS* with probability at least $\Omega(1)$. If all these steps of the M-process were decreasing, this would decrease the value at the beginning of the sequence by a factor of $f := (1 - \rho)^{\Omega(1/\rho)} = 1 - \Omega(1)$ using the pheromone update formula. Actually, some steps of the sequence will probably be increasing. However, according to Chernoff bounds, since each single step is decreasing with probability $\Omega(1)$, we have $\Omega(\ell)$ decreasing steps within ℓ steps with probability $\Omega(1)$. Given a current value of v before the first decreasing step, the sum of the amounts of decrease in decreasing steps (not counting increasing steps in between since we are aiming at a bound of the one-sided variance Δ) is then at least $v - vf = \Omega(1) \cdot v$, hence at least $v \cdot \Omega(1/\ell)$ on average over ℓ steps. Again using the definition of the phases, we have $v \geq \ell_i$, hence this

decrease is at least

$$\frac{1}{n^{(i+1)\gamma}} \cdot \Omega\left(\frac{1}{\ell}\right) = \Omega\left(\frac{1}{n^{(i+1)\gamma}\log(2/\rho)}\right).$$

In order to compute Δ, the last value still has to be multiplied by the probability $\Omega(1)$ of the event and taken to the square. Applying Jensen's inequality, the average of the squares is at least as large as the square of the average. Altogether, the average Δ-value in $\Theta(\log(2/\rho))$ steps before time T_{i+1} is at least $\delta_i = \Omega(1/(n^{2(i+1)\gamma}\log^2(2/\rho)))$.

We apply Lemma 7. Estimating $(2X_{0_i} + \alpha_i) \leq 5\alpha_i$, we get $E(T_i) \leq 5\alpha_i^2/\delta_i + \ell = O(n^{2(i+1)\gamma-2i\gamma}\log^2(2/\rho) + \log(2/\rho)) = O(n^{2\gamma}\log^2(2/\rho))$. By the definition of γ, we have shown $E(T_i) = O(n^\epsilon \log^2(2/\rho))$, and, using $1/\gamma = O(1)$, also $E^* = O(n^\epsilon \log^2(2/\rho))$. This completes the proof. □

Now all arguments for the proof of Theorem 7 are ready.

Proof of Theorem 7 A lower bound $\Omega(n/(\rho\log(2/\rho)))$ follows from Lemma 5, hence we only need to show a lower bound $\Omega(n^2)$. Consider a time phase containing the first $n/12$ improvements. Lemma 3 implies that at the end of the phase the current LEADINGONES-value is still less than $n/2$ with probability $1 - 2^{-\Omega(n)}$, which we assume to happen.

Consider the last $n/2$ bits in random state and note that every improvement corresponds to one step of the M-process. Choosing $0 < \epsilon < 1$, along with $\rho = 1/\text{poly}(n)$ and n large enough, the time bound for the M-process from Lemma 8 is $cn^\epsilon \log^2(2/\rho) \leq n/12$ for some constant $c > 0$. This implies that during the phase each of the last $n/2$ bits reaches the lower border for its success probability, independent of the other bits, at least once with probability $\Omega(1)$. By Chernoff bounds, we have $\Omega(n)$ such bits with probability $1 - 2^{-\Omega(n)}$. Assuming this to happen, Lemma 4 is in force and the outstanding lower bound $\Omega(n^2)$ follows by the law of total probability. □

In the lengthy proof, we had to carefully look at the random bits and to study the structure of the optimization process. It seems to be even harder to prove a corresponding lower bound for MMAS since accepting equally good solutions implies that more than n exchanges of the best-so-far solution can happen. Also additional ideas are required to transfer the proof of Theorem 7 and to obtain an improved lower bound for MMAS* on ONEMAX. Nevertheless, our analysis for LEADINGONES is important since it contains important proof techniques for lower bounds on the runtime of MMAS algorithms. Moreover, it rigorously shows that the upper bounds derived by the method of fitness-based partitions described in Sect. 3 can be very good and almost asymptotically tight. Still, they leave room for improvement using specialized techniques as presented in Theorem 6 since it is not necessary for the process to freeze the best-so-far solution in pheromone for every fitness level.

4 Plateau functions

The general upper bounds from Theorems 3 and 4 for unimodal functions yield a gap of only polynomial size between MMAS and MMAS*. In addition, we have proven the same upper bounds on LEADINGONES for both MMAS and MMAS*. This may give the impression that MMAS and MMAS* behave similarly on all functions. However, this only holds for functions with a certain gradient towards better solutions. On plateaus MMAS and MMAS* can have a totally different behavior.

Plateaus are regions in the search space where all search points have the same fitness. Consider a function $f\colon \{0,1\}^n \to \mathbb{R}$ and assume that the number of different objective values for that function is D. Then there are at least $2^n/D$ search points with the same objective value. Often, the number of different objective values for a given function is polynomially bounded. This implies an exponential number of solutions with the same objective value. In the extreme case, we end up with the function NEEDLE where only one single solution has objective value 1 and the remaining ones get value 0. The function is defined as

$$\text{NEEDLE}(x) := \begin{cases} 1 & \text{if } x = x_{\text{OPT}}, \\ 0 & \text{otherwise,} \end{cases}$$

where x_{OPT} is the unique global optimum. Gutjahr and Sebastiani (2008) compare MMAS* and $(1+1)$ EA* w.r.t. their runtime behavior. For suitable values of ρ that are exponentially small in n, MMAS* has expected optimization time $O(c^n)$, $c \geq 2$ an appropriate constant, and beats $(1+1)$ EA*. The reason is that MMAS* behaves nearly as random search on the search space, while the initial solution of $(1+1)$ EA* has Hamming distance n to the optimal one with probability 2^{-n}. To obtain from such a solution an optimal one, all n bits have to flip, which has expected waiting time n^n, leading in summary to an expected optimization time $\Omega((n/2)^n)$. In the following, we show a similar result for MMAS* if ρ decreases only polynomially with the problem dimension n.

Theorem 8 *Choosing* $\rho = 1/\text{poly}(n)$, *the optimization time of MMAS* on* NEEDLE *is at least* $(n/6)^n$ *with probability* $1 - e^{-\Omega(n)}$.

Proof Let x be the first solution constructed by MMAS* and denote by x_{OPT} the optimal one. As it is chosen uniformly at random from the search space, the expected number of positions where x and x_{OPT} differ is $n/2$ and there are at least $n/3$ such positions with probability $1 - e^{-\Omega(n)}$ using Chernoff bounds. At these positions of x the "wrong" edges of the construction graph are reinforced as long as the optimal solution has not been obtained. This implies that the probability to obtain the optimal solution in the next step is at most $2^{-n/3}$. After at most $t^* \leq (\ln n)/\rho$ (see inequality (1)) iterations, the pheromone values of x have touched their borders provided x_{OPT} has not been obtained. The probability of having obtained x_{OPT} within a phase of t^* steps is at most $t^* \cdot 2^{-n/3} = e^{-\Omega(n)}$. Hence, the probability to produce a solution that touches its pheromone borders and differs from x_{OPT} in at least $n/3$ positions before producing x_{OPT} is $1 - e^{-\Omega(n)}$. In this case, the expected number of steps to produce x_{OPT} is $(n/3)^n$ and the probability of having reached this goal within $(n/6)^n$ steps is at most 2^{-n}. □

The probability to choose an initial solution x that differs from x_{OPT} by n positions is 2^{-n}, and in this case, after all n bits have reached their corresponding pheromone borders, the probability to create x_{OPT} equals n^{-n}. Using the ideas of Theorem 8, the following corollary can be proved which asymptotically matches the lower bound for $(1+1)$ EA* given by Gutjahr and Sebastiani (2008).

Corollary 1 *Choosing* $\rho = 1/\text{poly}(n)$, *the expected optimization time of MMAS* on* NEEDLE *is* $\Omega((n/2)^n)$.

It is well known that the $(1+1)$ EA algorithm that accepts each new solution has expected optimization time $O(2^n)$ on NEEDLE (see Garnier et al. 1999; Wegener and Witt 2005)

even though it samples with high probability in the Hamming neighborhood of the latest solution. On the other hand, MMAS* will have a much larger optimization time unless ρ is superpolynomially small (Theorem 8). Our aim is to show in the forthcoming Theorem 9 that MMAS is more efficient than MMAS* and almost competitive with $(1+1)$ EA.

As in Sect. 3, the heart of our analysis will be the random process describing the pheromone values if the fitness function does not give any information. The random walk on the success probabilities of a bit in random state, which was captured by the M-process, reappears. The situation now is easier: as long as the needle has not been found, MMAS exchanges the best-so-far solution in each step. Hence, in terms of the M-process (which, however, has no lower bound on success probabilities) we would have the special case of $L_i = 1$ for $i \geq 0$. Our setting collapses to a Markovian random walk P_t, $t \geq 0$, on the state space $[1/n, 1-1/n]$ that, according to the pheromone update formula, increases or decreases its current state with probability P_t and $1-P_t$, respectively. We call this random walk the *P-process*. Note that we assume each step to exchange the solution; hence, the process does not freeze after the needle has been found. Until the first hitting time of the needle, the P-process and the random success probability at a bit coincide.

Many analyses conducted in Sect. 3 carry over in a simplified shape. In particular, Lemma 2 is in force. In conjunction with Lemma 1, it implies the unconditional success probability at time t to be purely random, hence the underlying bit is set to 1 with probability $1/2$ at each point of time. We will, however, have to inspect success probabilities at later points of time, assuming that the needle has not been found yet. This condition biases the current success probability towards decreasing values.

In the following, we will basically redo the random walk analysis from Lemma 8 to bound the time until a random success probability, starting at an arbitrary point in $[1, n, 1-1/n]$, exceeds $1/2$. Intuitively, then the expected success probability will stay at least $1/2$ for all following points of time. This is made precise as follows.2

Lemma 9 *If $P_t \geq 1/2$ then for any $t' \geq t$, $E(P_{t'} \mid P_t) \geq 1/2$.*

Proof We inductively show a stronger statement. If at some time t the so-called super-symmetry

$$\text{Prob}(P_t \geq p) \geq \text{Prob}(P_t \leq 1-p) \quad \text{for all } p \geq 1/2$$

holds, then the super-symmetry holds at all times $t' \geq t$. Using the same arguments as in the proof of Lemma 2, the super-symmetry at time t implies $E(P_t) \geq 1/2$.

To prove the statement, assume the super-symmetry to hold at time t. For the induction step, we exploit the following monotonicity of the formula $i(p) = (1-\rho)p + \rho = 1 - (1-\rho)(1-p)$ for the pheromone increase: $p' \geq p \Rightarrow i(p') \geq i(p)$. Moreover, the probability of increasing a pheromone value (i.e., success probability) is obviously monotone in the probability itself.

We exploit these observations to iteratively replace the distribution behind the random variable P_t by a simpler distribution whose successor P_{t+1} is easier to estimate; actually, we will apply Lemma 2 in the end. Let $\mu_t(p) := \text{Prob}(P_t = p)$ be the probability mass of point p at time t. Given two points $p_1 > p_2$, we set $\mu'_t(p_1) := \mu_t(p_1) - a$ for some $0 \leq a \leq \mu_t(p_1)$, $\mu'_t(p_2) := \mu_t(p_2) + a$, and $\mu'_t(p) := \mu_t(p)$ for all $p \notin \{p_1, p_2\}$. In other words, we conduct a *probability shift* to decrease the mass of a point corresponding to a higher success probability by some amount and increase the mass of a lower point by the same amount. Let P'_t be the random variable corresponding to μ'_t. As a

consequence, $\text{Prob}(P_t \geq p) \geq \text{Prob}(P'_t \geq p)$ for all $0 \leq p \leq 1$. Now, due to the above-mentioned monotonicities, $\text{Prob}(P_{t+1} \geq p) \geq \text{Prob}(P'_{t+1} \geq p)$ and $\text{Prob}(P_{t+1} \leq 1-p) \leq \text{Prob}(P'_{t+1} \leq 1-p)$, i.e., also the distributions at the following step stochastically dominate each other as needed to show super-symmetry at time $t+1$.

Starting from P_t, we carry out a sequence of probability shifts as follows. Let $\ell_0 := -\infty$ and let $\ell_1 < \cdots < \ell_r < 1/2$ be all points in the lower half of the pheromone scale with positive mass, sorted in ascending order. Note that there are only finitely many such points after finitely many steps. Let μ denote the distribution which is iteratively modified. We start from $\mu := \mu_t$. Then, for $1 \leq i \leq r$, we shift probability $\text{Prob}(1-\ell_{i-1} > P_t \geq 1-\ell_i) \geq \mu(\ell_i)$ from all points in the interval $(1-\ell_{i-1}, 1-\ell_i]$ to the point $1-\ell_i$. If $\mu(1-\ell_i) > \mu(\ell_i)$ after this shift, we additionally shift the difference $\mu(1-\ell_i) - \mu(\ell_i)$ from $1-\ell_i$ to $1-\ell_{i+1}$, where $\ell_{r+1} := 1/2$. Inductively over i, it follows from the super-symmetry at time t that we do not shift more probability than allowed. Moreover, by the procedure, the super-symmetry is still valid for the distribution P'_t obtained after the k iterations since even the perfect symmetry $\text{Prob}(P'_t = p) = \text{Prob}(P'_t = 1-p)$ then holds. By Lemma 2, its successor P'_{t+1} is perfectly symmetrical, too. Hence, the actual distribution P_{t+1} stochastically dominates a symmetrical distribution, and, therefore, has the super-symmetry. This completes the induction from t to $t+1$. □

We now describe the analysis of the random walk as it will be needed in the proof of the theorem. Still concentrating on the random success probabilities P_t, $t \geq 0$, at a single bit, let η_a denote the first point of time $t \geq 0$ such that $P_t \geq 1/2$, given that for the initial pheromone value it holds $P_0 = a < 1/2$ (different from the initialization in the MMAS algorithm).

Lemma 10 *For any $a < 1/2$, $E(\eta_a) = O(n^2/\rho^2)$.*

Proof We start with a useful transformation. Since we would like to reuse Lemma 7, which was previously applied to bound the first hitting time of the state $1/n$ for the M-process, we swap the meaning of ones and zeros and consider the process $X_t := 1 - P_t$, $t \geq 0$, instead. We are now interested in the first hitting time for the states $\{x \mid x \leq 1/2\}$ of the process X_t given $X_0 = 1 - a > 1/2$. To this end, we are allowed to remove the lower border $1/n$ on the success probabilities. As a consequence, the X-process equals the M-process given that $L_i = 1$ for all i.

By Lemma 6, the X-process is a Markovian supermartingale. Hence, we can apply the general Lemma 7. To this end, we need a lower bound on Δ_{t+1}, which is obtained by identifying a single decreasing step and the amount and probability of decrease. We can assume $X_t > 1/2$.

For $X_{t+1} \leq X_t$ to happen, an event of probability $1 - X_t \geq 1/n$ is sufficient (note that the upper bounds on success probabilities have not been removed). In this case, $X_{t+1} = (1-\rho)X_t$. Moreover, due to $X_t \geq 1/2$, we have $X_{t+1} \leq X_t - \rho/2$ then. Altogether, $\Delta_{t+1} \geq ((1-X_t)(\rho/2))^2 \geq \rho^2/(4n^2)$. We invoke Lemma 7 with $\alpha = 1/2$, $\delta := \rho^2/(4n^2)$, and $\ell = 1$. As $X_0 \leq 1$, for the resulting stopping time T we have $E(T) = O(n^2/\rho^2)$. Moreover, $X_0 > 1/2$, $X_T \leq 1$, and $|X_T - X_0| \geq 1/2$ implies $X_T \leq 1/2$. Hence, $P_T \geq 1/2$ and $E(\eta_a) = O(n^2/\rho^2)$ as claimed. □

The following theorem shows that the expected optimization time of MMAS on NEEDLE is at most by a polynomial factor larger than the one of $(1+1)$ EA unless ρ is superpolynomially small.

Theorem 9 *The expected optimization time of MMAS on* NEEDLE *is bounded from above by* $O((n/\rho)^2(\log n)2^n)$.

Proof By the symmetry of the construction procedure and uniform initialization, we w.l.o.g. assume that the needle x_{OPT} equals the all-ones string 1^n. As in Wegener and Witt (2005), we study the process on the constant function $f(x) = 0$. The first hitting times for the needle are the same on NEEDLE and the constant function while the constant function is easier to study as MMAS accepts each new search point forever for this function.

The proof idea is to study a kind of "mixing time" $t(n)$ after which each bit is independently set to 1 with a probability of at least $1/2$ regardless of its initial success probability (recall that this means the probability of setting the bit to 1). Since bits are treated independently, this implies that the probability of creating the needle is at least 2^{-n} in some step after at most $t(n)$ iterations. We successively consider independent phases of (random) length $t(n)$ until the needle is sampled. The number of phases required follows a geometric distribution with parameter at least 2^{-n}, hence, the expected number of phases required to sample the needle is at most 2^n. By the linearity of expectation, the expected time until 2^n phases have elapsed is bounded by $E(t(n)) \cdot 2^n$. The theorem follows if we can show that $E(t(n)) = O((n/\rho)^2 \log n)$.

We recall the above-described random walk on the success probabilities. Consider the independent success probabilities of the n bits for any initial distribution. We call a success probability *good* at a certain time t if is has been bounded from below by $1/2$ at least once in the t steps after initialization, and *bad* otherwise. We are interested in the time T^* until all n success probabilities have become good. For each single success probability, the expected time until becoming good is $O(n^2/\rho^2)$ according to Lemma 10. Due to Markov inequality, the time is $O(n^2/\rho^2)$ with probability at least $1/2$. Repeating $2 \log n$ independent such phases, this implies that, after $O((n/\rho)^2(\log n))$ steps, each success probability is bad with probability at most $1/(2n)$. Hence, by the union bound, the probability is only at most $1/2$ that there is a bad success probability left after this number of steps. Repeating this argument an expected number of at most 2 times, $E(T^*) = O((n/\rho)^2 \log n)$ follows. By definition, all success probabilities have been at least $1/2$ at least once after T^* steps. Using Lemma 9, we obtain that T^* is an upper bound on $t(n)$, and the theorem follows. □

The function NEEDLE requires an exponential optimization time for each algorithm that has been considered. Often plateaus are much smaller, and randomized search heuristics have a good chance to leave them within a polynomial number of steps. Gutjahr and Sebastiani (2008) consider the function NH-ONEMAX, which consists of the NEEDLE-function on $k = \log n$ bits and the function ONEMAX on $n - k$ bits. The latter can only be optimized if the needle has been found on the needle part. The function is defined as

$$\text{NH-ONEMAX}(x) = \left(\prod_{i=1}^{k} x_i\right)\left(\sum_{i=k+1}^{n} x_i\right).$$

Taking into account the logarithmic size of the NEEDLE-function of NH-ONEMAX, MMAS* with polylogarithmically small ρ cannot optimize the needle part within an expected polynomial number of steps. The proof ideas are similar to those used in the proof of Theorem 8. After initialization, the expected Hamming distance of the needle part to the needle is $(\log n)/2$, and it is at least $(\log n)/3$ with probability $1 - o(1)$. Working under this condition, this means that the probability of sampling the needle is at most $2^{-(\log n)/3} = n^{-1/3}$ in each step before the needle is found. As $\rho = 1/\text{polylog}(n)$ holds, the

lower pheromone borders of the $(\log n)/3$ "wrong" bits from the initial solution are reached in at most $t^* \leq (\ln n)/\rho = \text{polylog}(n)$ steps. Hence, the needle is found before this situation has been reached with probability at most $\text{polylog}(n)/n^{1/3} = o(1)$. Afterwards, the probability of sampling the needle is at most $n^{-(\log n)/3} = 2^{-\Omega(\log^2 n)}$. This proves the following superpolynomial lower bound.

Theorem 10 *Choosing* $\rho = 1/\text{polylog}(n)$, *the expected optimization time of MMAS* on* NH-ONEMAX *is* $2^{\Omega(\log^2 n)}$.

Also the proof of Theorem 9 carries over to a major extent. The random walk arguments leading to $E(t(n)) = O((n/\rho)^2 \log n)$ still hold since random bits are considered independently and the borders for the pheromone values have not been changed. What has been changed is the size of the needle. As now the needle part only consists of $\log n$ bits, the probability of creating it is at least $2^{-\log n} = 1/n$ after $t(n)$ steps. Hence, MMAS can find the needle after an expected number of $O((n/\rho)^2 \cdot (\log n) \cdot n)$ steps. After this goal has been achieved, the unimodal function ONEMAX, which contains at most $n+1$ different fitness values, has to be optimized. We conclude from Theorem 4, the general bound on unimodal functions, that MMAS optimizes ONEMAX in an expected number of $O((n^3 \log n)/\rho)$ steps. Putting the two bounds together, the following result has been proved.

Theorem 11 *The expected optimization time of MMAS on* NH-ONEMAX *is at most* $O((n^3 \log n)/\rho^2)$.

This bound is polynomial if $\rho = 1/\text{poly}(n)$, in contrast to the superpolynomial bound for MMAS* from Theorem 10. Hence, MMAS is superior to MMAS* on NH-ONEMAX as well.

5 Experiments

We supplement our theoretical investigations by experiments. The time bounds presented in the previous sections are asymptotic; hence, they do not reveal whether certain effects predicted for large n also occur for small problem dimensions. Moreover, experimental data can reveal details that are not captured by our theorems.

5.1 The performance of MMAS on NEEDLE

First, we take a closer look at MMAS on NEEDLE for several values of ρ. Theorem 9 gives an upper bound of order $O((n/\rho)^2(\log n)2^n)$, thus growing with $1/\rho$. However, we conjecture that for some values of ρ the upper bound is not tight. If ρ is extremely large such that MMAS equals $(1+1)$ EA, the best-so-far solution performs a random walk through the search space and the expected time until the needle is found is about $e/(e-1) \cdot 2^n \approx 1.58 \cdot 2^n$ due to Garnier et al. (1999). In addition, if ρ is extremely small, MMAS degenerates to almost random search and the expected time to find the needle is then close to 2^n.

We are tempted to believe that the optimization time is always of order 2^n and hence independent of ρ. The upper bound $O((n/\rho)^2(\log n)2^n)$ would then be far too pessimistic for small values of ρ. This is true for extremely small ρ, but we do not know how MMAS behaves with intermediate ρ-values. Therefore, we perform experiments for NEEDLE with $n \in \{8, 16\}$ and exponentially decreasing values for ρ, $\rho = 2^{-1}, 2^{-2}, 2^{-3}, \ldots$ and measure

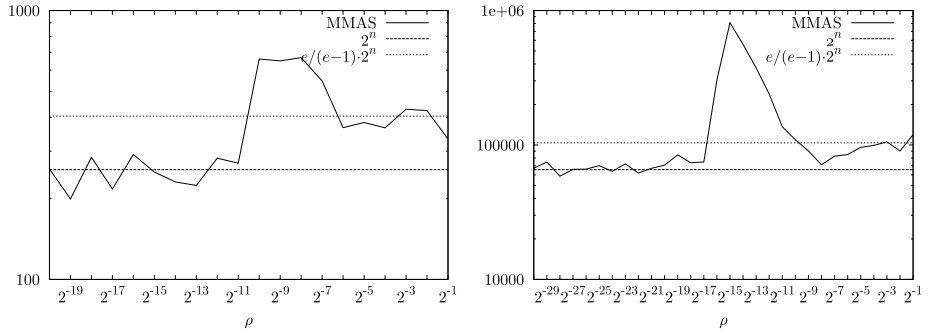

Fig. 4 Average runtime for MMAS on NEEDLE with $n = 8$ (*left*) and $n = 16$ (*right*) and exponentially decreasing values of ρ. The data is averaged over 100 runs for each setting

the average runtime of MMAS during 100 independent runs for each setting. The results are shown in Fig. 4.

Consider the plot for $n = 16$. First, observe that the average runtime slightly decreases when decreasing ρ from 2^{-1} to 2^{-8}. A possible explanation is that the $(1 + 1)$ EA algorithm tends to resample the current solution, which happens with probability approximately $1/e$. Hence, the expected waiting time until a new solution is sampled is about $1/(1 - 1/e) = e/(e - 1)$, which explains the bound of about $e/(e - 1) \cdot 2^n$ by Garnier et al. (1999). This bound also holds for MMAS with very large ρ. When decreasing ρ, the probability of resampling the current solution decreases and so does the average runtime.

When further decreasing ρ, the average runtime increases very fast. This can be explained since the "mixing time" (cf. proof of Theorem 9) increases with a function of $1/\rho$. Due to the slow mixing, MMAS tends to spend a lot of time sampling the same region of the search space over and over again. If we are unlucky, MMAS focuses on regions that are far away from the needle where many (say $\Omega(n)$) pheromones hit the "wrong" border. In this case the probability of generating the needle is far below 2^{-n}, yielding an overly large average optimization time. This observation shows that it is necessary for the upper bound from Theorem 9 to grow with some function of $1/\rho$ for these ρ-values.

Lastly, at some point on the ρ-scale, MMAS suddenly turns into almost random search. Here, ρ is so small that typically the needle is found by random search before the pheromones get the chance to move towards a border. A comparison with the lower horizontal line indicates that the average runtime is highly concentrated around 2^n.

For $n = 8$, MMAS shows a similar behavior, although the effects are less pronounced. We conclude that, surprisingly, MMAS shows a phase transition behavior. Although MMAS is competitive with $(1 + 1)$ EA for very large and very small values of ρ, it is much more inefficient for intermediate values of ρ.

5.2 Comparing MMAS and MMAS* on unimodal functions

We now turn to a comparison of MMAS and MMAS*. Regarding unimodal functions, the upper bound for MMAS from Theorem 4 can be by a factor of n^2 larger than the upper bound for MMAS* from Theorem 3. We believe that the larger bound for MMAS is too pessimistic for many unimodal functions. In this bound, we had to account for situations where pheromones in MMAS cannot freeze to a single solution on the current fitness level. However, the only situation in which many bits will not freeze properly occurs when the current

best-so-far solution consistently switches between different solutions with the same fitness and large Hamming distance. Such a situation appears to be very atypical for most unimodal functions. We expect MMAS to either freeze towards a small set of solutions that are close in Hamming space or to find an improvement beforehand. This behavior is very close to MMAS*. Another reason why we believe that MMAS and MMAS* behave similarly is that the upper bounds for LEADINGONES from Theorem 6 are equal for the two algorithms.

We perform experiments for the two unimodal functions investigated in Sect. 3, ONEMAX and LEADINGONES. The experimental setting is taken over from the previous subsection. That is, we use exponentially decreasing values for ρ to cover the range from almost pure random search to an algorithm close to $(1+1)$ EA. Preliminary experiments indicated a small but noticeable difference between MMAS and MMAS* for a specific range of ρ-values. Therefore, we repeated the experiments with an increased number of 1000 runs per setting. The results are shown in Fig. 5.

The observed average runtimes for MMAS and MMAS* are nearly equal for almost all values of ρ, except for a range of ρ-values, roughly around $\rho = 2^{-9}$, where MMAS is more efficient than MMAS*. We applied a statistical test to see for each ρ-value whether the differences between MMAS and MMAS* are significant. A non-parametric Mann-Whitney test revealed highly significant results ($p \leq 0.001$) in favor of MMAS for ONEMAX with $n = 8$ and $\rho \in \{2^{-9}, \ldots, 2^{-5}\}$, ONEMAX with $n = 16$, $\rho \in \{2^{-14}, \ldots, 2^{-5}\}$ and $\rho = 2^{-16}$, and finally for LEADINGONES with $n = 16$ and $\rho \in \{2^{-13}, \ldots, 2^{-7}\}$.

The differences on ONEMAX are surprising in the light of the conjecture by Gutjahr (2007) that accepting equally good solutions deteriorates the performance of MMAS* on ONEMAX. Our statistical tests confirm that for some ρ-values the opposite holds and that accepting equally good solutions can lead to a significant speed-up.

It seems that for ONEMAX jumping between equally fit solutions is beneficial. For a possible explanation we identify bits with their corresponding 1-edges. Imagine two bits, one with low pheromone and one with high pheromone. If only the low-pheromone bit is reinforced, the increase in pheromone is larger compared to the case where only the high-pheromone bit is reinforced. Also the probability to set both bits to 1 simultaneously increases more if only the low-pheromone bit is reinforced. The difference is even stronger if the high-pheromone bit already hit the upper pheromone border. This observation might explain why on ONEMAX it is better to switch between different bits to reinforce than to reinforce the same bits over and over again.

5.3 Comparing MMAS and MMAS* on plateaus

Now we compare the performance of MMAS* and MMAS on the plateau functions NEEDLE and NH-ONEMAX, averaged over 100 runs. The results from Sect. 4 predict much larger runtimes for MMAS*.

First, consider NEEDLE. After the random freezing time t^*, all pheromones in MMAS* are frozen until the end of the run. If k bits are frozen towards the "wrong" bit value, the probability to find the needle with the next constructed solution is exactly $(1/n)^k \cdot (1 - 1/n)^{n-k}$. This probability is extremely small. For, say, $n = 16$ and $k = 12$, it is less than $(1/16)^{12} = 2^{-64}$, and the expected time needed to find the needle is larger than 2^{64}. As these values are far too large to allow a complete simulation, the optimization times of MMAS* with $n = 16$ are estimated by adding the expected time to find the needle to the observed freezing time. (In rare cases where the needle was found before all pheromones were frozen, the real optimization time was recorded.) The resulting values still contain randomness as k is a random variable. On the other hand, the estimations yield a reduction of variance, which

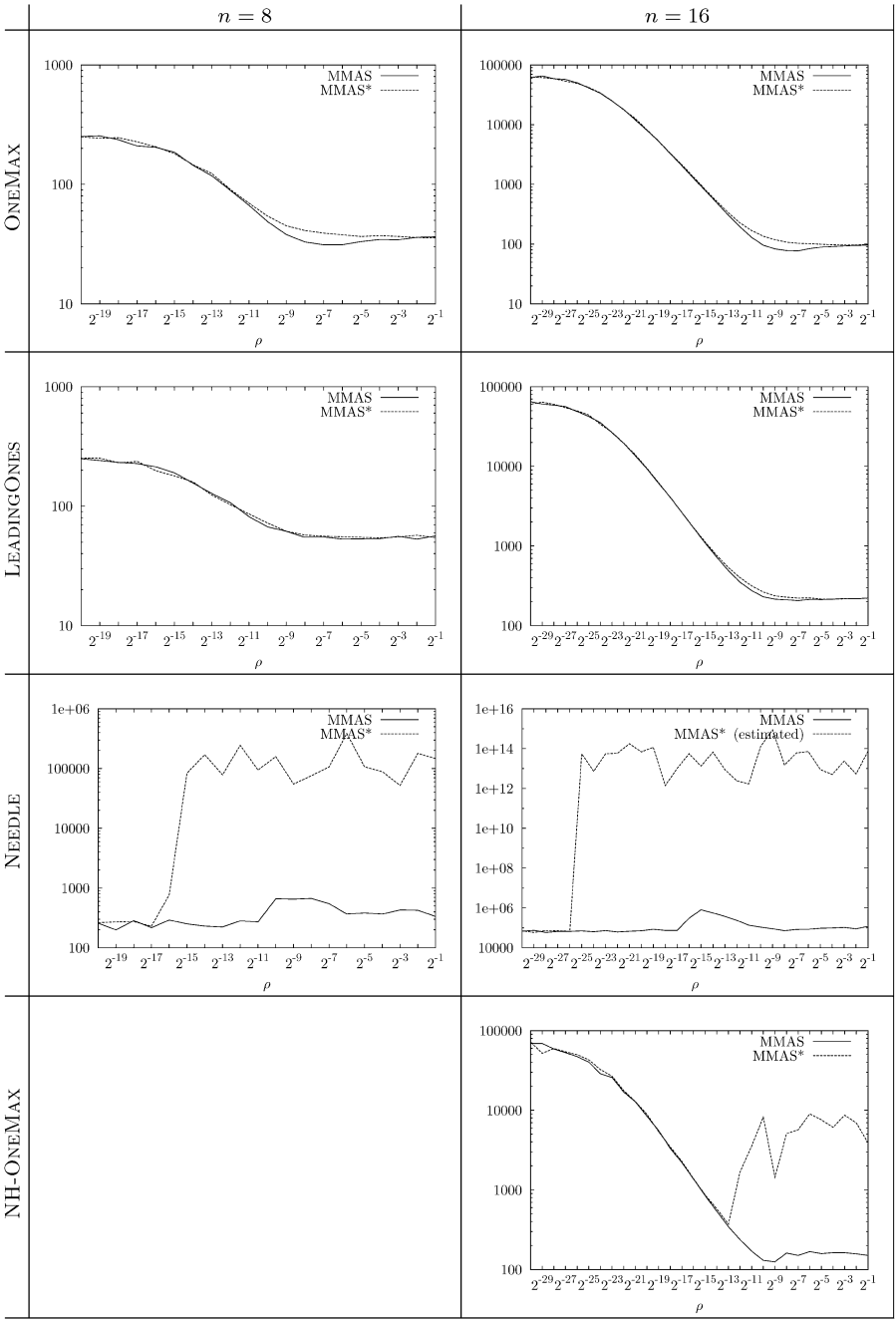

Fig. 5 Average runtime for MMAS and MMAS* on OneMax, LeadingOnes, Needle, and NH-OneMax with exponentially decreasing values of ρ for $n = 8$ and $n = 16$. The data is averaged over 1000 runs for OneMax and LeadingOnes and 100 runs for Needle and NH-OneMax. The runtime of MMAS* on Needle with $n = 16$ is estimated by the (conditional) expected remaining optimization time once all pheromones are frozen. The case of NH-OneMax with $n = 8$ is omitted due to the small size of the needle

gives us an even better picture of the average performance than a complete simulation. The optimization times for $n = 8$ were obtained by real simulations, though.

Figure 5 shows the resulting average optimization times. The data for MMAS is taken over from Fig. 4, but the large peaks from Fig. 4 now clearly pale in comparison with the average runtime of MMAS*. One can see that MMAS and MMAS* behave almost identically for very small values of ρ, since then both algorithms degenerate to pure random search. Focussing on the remaining ρ-values, there is no doubt that MMAS outperforms MMAS*. For $n = 8$, MMAS* is by a factor of more than 100 slower than MMAS. For $n = 16$, the differences are even more drastic: the average optimization time of MMAS* is often larger than 10^{13} while MMAS never exceeds 10^6.

For NEEDLE, the best strategy is to choose ρ so small that MMAS and MMAS* degenerate to a random search. This is different for NH-ONEMAX where, due to the underlying ONEMAX-problem, we can expect MMAS and MMAS* to outperform pure random search with a sufficiently large value of ρ. We investigate NH-ONEMAX with $n = 16$, i.e., a needle consisting of 4 bits. The size of the needle may look small, but finding needles is more challenging than for NEEDLE. The problem dimension is much larger than the size of the needle, and so the pheromone borders are more extreme compared to NEEDLE when considering equally sized needles. As a consequence, it is harder to find the needle in the case some bits have pheromones at the "wrong" border.

Figure 5 shows the average optimization times for MMAS and MMAS*. The results show that it is necessary to increase the value of ρ to deal with the ONEMAX-part of the function until $\rho \geq 2^{-13}$. Larger values of ρ increase the optimization time of MMAS* as this algorithm is not able to perform a random walk on the NEEDLE-part of the function. In contrast to this, MMAS also performs well for large values of ρ as the search for the needle is achieved by replacing equally good solutions.

6 Conclusions

The rigorous runtime analysis of ACO algorithms is a challenging task where the first results have been obtained only recently. We have compared previous results for an MMAS variant called 1-ANT with known results for a similar MMAS algorithm called MMAS*. 1-ANT is extremely inefficient on ONEMAX and LEADINGONES if the evaporation factor is small and exhibits a phase transition from polynomial to exponential runtime, in sharp contrast to MMAS*. Gutjahr (2007) conjectured that this difference is due to the different selection mechanisms, i.e., the question whether solutions of equal fitness are accepted or not. Therefore, we investigated a variant of MMAS*, called MMAS, that accepts equally fit solutions, so MMAS and 1-ANT only differ in their pheromone update mechanisms. Our analyses from Sect. 3 revealed that on ONEMAX and LEADINGONES, there is no phase transition for MMAS* and MMAS. Hence, the phase transition of 1-ANT must result from the pheromone update mechanism.

Already Gutjahr and Sebastiani (2008) have shown how to transfer fitness-level arguments, a powerful and well-known tool from the analysis of evolutionary algorithms, to the analysis of ACO algorithms. Here, we have presented these previous results in a simplified framework and extended them to the broad class of all unimodal functions. Moreover, the upper bounds obtainable by this method have been improved using specialized techniques for the LEADINGONES function, and the first lower bounds for MMAS* on ONEMAX and LEADINGONES have been given. The analyses cover random walks of pheromone values, which were not well understood before. In particular, we have estimated first hitting times

for pheromone borders in a general framework for random processes that are not homogeneous with respect to place. As an important insight, it has been shown that any pheromone value quickly reaches one of the borders even if the corresponding bit does not have any significant contribution to the fitness. The random-walk analysis is considered valuable also from a methodological point of view.

Another goal of this paper was to elaborate on the difference between MMAS and MMAS*. Regarding unimodal functions, our theoretical bounds and experiments for ONEMAX and LEADINGONES showed similar performance for both algorithms. However, for a range of evaporation factors, surprisingly, MMAS empirically outperforms MMAS*. The observed differences were confirmed by statistical tests. Unlike MMAS*, accepting equally fit solutions allows MMAS to explore plateaus of equal fitness by a random walk of the best-so-far solution. For well-known plateau functions such as NEEDLE, replacing equally fit solutions is essential. Both our asymptotical theoretical results and additional experiments clearly show that the performance gap between MMAS and MMAS* on plateau functions is huge.

Acknowledgements Dirk Sudholt and Carsten Witt were supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531). The authors thank two anonymous reviewers for helpful suggestions.

Appendix

Proof of Lemma 7 We first replace $\{X_t\}_{t\geq 0}$ by a simpler process $\{Y_t\}_{t\geq 0}$ defined by

$$Y_t := \min\{X_0 + \alpha, X_t\}.$$

Hence, the Y-process is capped at $X_0+\alpha$ when the X-process reaches a point at least $X_0+\alpha$ for the first time and is identical to the X-process before. By definition, the stopping time T in terms of the Y-process does not change. Second, $\{Y_t\}_{t\geq 0}$ clearly is a supermartingale, too. Third, by the definition of Δ_{t+1}, the second condition of the lemma holds for $\{Y_t\}_{t\geq 0}$ as well. We consider

$$V(t+1) := \mathrm{Var}(Y_{t+1} \mid \mathfrak{F}_t) = E\big(\big(Y_{t+1} - E(Y_{t+1} \mid \mathfrak{F}_t)\big)^2 \mid \mathfrak{F}_t\big)$$

and obtain

$$V(t+1) \geq \Delta_{t+1}$$

for all $t \leq T-1$.

We define a third process $\{Z_t\}_{t\geq 0}$ by $Z_t := -(Y_t)^2 + \sum_{k=1}^{t} V(k)$ and consider

$$E(Z_{t+1} \mid \mathfrak{F}_t) = -E\big((Y_{t+1})^2 \mid \mathfrak{F}_t\big) + \sum_{k=1}^{t+1} E\big(V(k) \mid \mathfrak{F}_t\big).$$

Regarding the $\sum$-term, the summand for $k = t+1$ by definition of $V(k)$ equals

$$E\big(V(k) \mid \mathfrak{F}_t\big) = V(k),$$

and for $k \le t$ we have

$$E\big(V(k) \mid \mathfrak{F}_t\big) = V(k)$$

since the right-hand side is $\mathfrak{F}_t$-measurable. Secondly, by the formula $E(A^2) = (E(A))^2 + E((A - E(A))^2)$ we have

$$\begin{aligned} E\big((Y_{t+1})^2 \mid \mathfrak{F}_t\big) &= \big(E(Y_{t+1} \mid \mathfrak{F}_t)\big)^2 + E\Big(\big(Y_{t+1} - E(Y_{t+1} \mid \mathfrak{F}_t)\big)^2 \mid \mathfrak{F}_t\Big) \\ &\le (Y_t)^2 + V(t+1), \end{aligned}$$

where the last inequality follows from $\{Y_t\}_{t\ge 0}$ being a supermartingale on the space $\mathbb{R}_0^+$. Together,

$$\begin{aligned} E(Z_{t+1} \mid \mathfrak{F}_t) &\ge -(Y_t)^2 - V(t+1) + \sum_{k=1}^{t+1} V(k) \\ &= -(Y_t)^2 + \sum_{k=1}^{t} V(k) \\ &= Z_t. \end{aligned}$$

Thus, $\{Z_t\}_{t\ge 0}$ is a submartingale with respect to $\{Y_t\}_{t\ge 0}$.

$\sum_{k=t}^{t+\ell} \Delta_k/\ell \ge \delta$ implies, along with the supermartingale property, that we have a non-zero probability that $Y_{t+1} < Y_t$ at least every ℓ steps, hence $T < \infty$ follows. Since X_t is bounded and positive by assumption, we have $|Y_t| < \infty$ and $V(t) < \infty$, hence

$$|Z_t| = \left| -Y_t^2 + \sum_{k=1}^{t} V(k) \right| < \infty$$

and we can apply the optional stopping theorem (Brémaud 1998). On the one hand, this yields $E(Z_T) \ge E(Z_0) = (Y_0)^2 = (X_0)^2$. On the other hand, along with $\sum_{k=1}^{T} V(k) \ge \sum_{k=1}^{T} \Delta_k \ge \sum_{k=1}^{\lfloor T/\ell \rfloor \cdot \ell} \Delta_k \ge \lfloor T/\ell \rfloor \ell\delta \ge (T-\ell)\delta$,

$$E(Z_T) = E\big(Y_T^2\big) - E\left(\sum_{k=1}^{T} \Delta_k\right) \le E\big(Y_T^2\big) - \delta \cdot E(T-\ell),$$

which implies

$$E(T) \le \frac{E(Y_T^2) - (X_0)^2}{\delta} + \ell \le \frac{(X_0+\alpha)^2 - (X_0)^2}{\delta} + \ell = \frac{(2X_0+\alpha)\cdot\alpha}{\delta} + \ell,$$

the first statement of the lemma.

The second statement $\mathrm{Prob}(X_T < X_0) \ge \alpha/(\alpha + E(X_0 - X_T \mid X_T < X_0))$ follows, again using the optional stopping theorem for supermartingales, from

$$\begin{aligned} 0 \ge E(X_T) - X_0 = &- \mathrm{Prob}(X_T < X_0) \cdot E(X_0 - X_T \mid X_T < X_0) \\ &+ \big(1 - \mathrm{Prob}(X_T < X_0)\big) \cdot E(X_T - X_0 \mid X_T > X_0) \end{aligned}$$

by rearranging terms and estimating $E(X_T - X_0 \mid X_T > X_0) \ge \alpha$. □

References

Brémaud, P. (1998). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. New York: Springer.

Doerr, B., & Johannsen, D. (2007). Refined runtime analysis of a basic ant colony optimization algorithm. In *Proceedings of the congress on evolutionary computation (CEC '07)* (pp. 501–507). New York: IEEE Press.

Doerr, B., Hebbinghaus, N., & Neumann, F. (2006). Speeding up evolutionary algorithms through restricted mutation operators. In *LNCS: Vol. 4193. Parallel problem solving from nature (PPSN '06)* (pp. 978–987). Berlin: Springer.

Doerr, B., Neumann, F., Sudholt, D., & Witt, C. (2007). On the runtime analysis of the 1-ANT ACO algorithm. In *Proceedings of the genetic and evolutionary computation conference (GECCO '07)* (pp. 33–40). New York: ACM.

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, *344*, 243–278.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, *26*(1), 29–41.

Droste, S., Jansen, T., & Wegener, I. (2002). On the analysis of the $(1+1)$ evolutionary algorithm. *Theoretical Computer Science*, *276*, 51–81.

Droste, S., Jansen, T., & Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, *39*(4), 525–544.

Eiben, A., & Smith, J. (2007). *Introduction to evolutionary computing* (2nd ed.). Berlin: Springer.

Garnier, J., Kallel, L., & Schoenauer, M. (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation*, *7*(2), 173–203.

Giel, O., & Wegener, I. (2003). Evolutionary algorithms and the maximum matching problem. In *LNCS: Vol. 2607. Proceedings of the 20th annual symposium on theoretical aspects of computer science (STACS '03)* (pp. 415–426). Berlin: Springer.

Gutjahr, W. J. (2002). ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, *82*(3), 145–153.

Gutjahr, W. J. (2006). On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability*, *8*(1), 105–133.

Gutjahr, W. J. (2007). Mathematical runtime analysis of ACO algorithms: Survey on an emerging issue. *Swarm Intelligence*, *1*, 59–79.

Gutjahr, W. J. (2008). First steps to the runtime complexity analysis of ant colony optimization. *Computers and Operations Research*, *35*(9), 2711–2727.

Gutjahr, W. J., & Sebastiani, G. (2008). Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability*, *10*, 409–433.

Jansen, T. Sudholt, D. (2009, to appear). Analysis of an asymmetric mutation operator. *Evolutionary Computation*.

Jansen, T., & Wegener, I. (2001). Evolutionary algorithms—how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, *5*(6), 589–599.

Mitzenmacher, M., & Upfal, E. (2005). *Probability and computing—randomized algorithms and probabilistic analysis*. Cambridge: Cambridge University Press.

Motwani, R., & Raghavan, P. (1995). *Randomized algorithms*. Cambridge: Cambridge University Press.

Neumann, F. (2004). Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. In *Proceedings of the congress on evolutionary computation (CEC '04)* (Vol. 1, pp. 904–910). New York: IEEE Press.

Neumann, F. (2007). Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, *181*(3), 1620–1629.

Neumann, F., & Wegener, I. (2006). Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, *5*(3), 305–319.

Neumann, F., & Wegener, I. (2007). Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, *378*(1), 32–40.

Neumann, F., & Witt, C. (2006). Runtime analysis of a simple ant colony optimization algorithm. In *LNCS: Vol. 4288. Proceedings of the 17th international symposium on algorithms and computation (ISAAC '06)* (pp. 618–627). Berlin: Springer. Extended version to appear in *Algorithmica*.

Neumann, F., Sudholt, D., & Witt, C. (2007). Comparing variants of MMAS ACO algorithms on pseudo-Boolean functions. In *LNCS: Vol. 4638. Proceedings of engineering stochastic local search algorithms (SLS '07)* (pp. 61–75). Berlin: Springer.

Rudolph, G. (1997). *Convergence properties of evolutionary algorithms*. Hamburg: Kovač.

Scharnow, J., Tinnefeld, K., & Wegener, I. (2004). The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, *3*(4), 349–366.

Stützle, T., & Hoos, H. H. (2000). MAX-MIN ant system. *Future Generations Computer Systems*, *16*, 889–914.

Wegener, I. (2002). Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In R. Sarker, X. Yao, & M. Mohammadian (Eds.), *Evolutionary optimization* (pp. 349–369). Dordrecht: Kluwer Academic.

Wegener, I., & Witt, C. (2005). On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability and Computing*, *14*, 225–247.

Witt, C. (2005). Worst-case and average-case approximations by simple randomized search heuristics. In *LNCS: Vol. 3404*. *Proceedings of the 22nd symposium on theoretical aspects of computer science (STACS '05)* (pp. 44–56). Berlin: Springer.