Computational Complexity of Ant Colony Optimization and Its Hybridization

Frank Neumann[1], Dirk Sudholt[2⋆], and Carsten Witt[3⋆]

[1] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany,
[2] Informatik 2, Technische Universität Dortmund, 44221 Dortmund, Germany,
[3] DTU Informatics, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

Abstract. The computational complexity of ant colony optimization (ACO) is a new and rapidly growing research area. The finite-time dynamics of ACO algorithms is assessed with mathematical rigor using bounds on the (expected) time until an ACO algorithm finds a global optimum. We review previous results in this area and introduce the reader into common analysis methods. These techniques are then applied to obtain bounds for different ACO algorithms on classes of pseudo-Boolean problems. The resulting runtime bounds are further used to clarify important design issues from a theoretical perspective. We deal with the question whether the current best-so-far solution should be replaced by new solutions with the same quality. Afterwards, we discuss the hybridization of ACO with local search and present examples where introducing local search leads to a tremendous speed-up and to a dramatic loss in performance, respectively.

1 Introduction

The fascinating collective behavior of swarms is a rich source of inspiration for the field of computational intelligence (see, e. g., Kennedy, Eberhart, and Shi, 2001). In particular, the ability of ants to find shortest paths has been transferred to shortest path problems in graphs (Dorigo and Stützle, 2004). The idea is that artificial ants traverse the graph from a start node (nest) to a target node (food). On each edge a certain amount of artificial pheromone is deposited. At each node each ant chooses which edge to take next. This choice is made probabilistically and according to the amount of pheromone placed on the edges. As in real ant colonies, the pheromones evaporate over time. The amount of evaporation is determined by the so-called *evaporation factor* ρ, $0 < \rho < 1$. In every pheromone update on every edge a ρ-fraction of the pheromone evaporates, i. e., if the edge contains pheromone τ, the remaining amount of pheromone is $(1 - \rho) \cdot \tau$ and then eventually new pheromone is added. Intuitively, a large evaporation factor implies that the impact of previously laid pheromones diminishes quickly and new pheromones have a large impact on the system. Small evaporation factors, on the other hand, imply that the system only adapts slowly to new pheromones.

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

In contrast to real ants, however, new pheromones are often not placed immediately after traversing an edge. In order to avoid rewarding cycles or paths leading to dead ends, pheromones are usually placed after the ant has found the target node and only for edges that are not part of a cycle on the ant's trail. Also, in such an artificial system the amount of pheromone placed may depend on the length of the constructed path, so that short paths are rewarded more than longer paths. Such an adaptation is often necessary as the movement of artificial ants is usually synchronized, in contrast to real ants in nature, where the order of ants arriving at a location is essential.

Optimization with artificial ants is known as *ant colony optimization* (ACO). ACO algorithms have been used for shortest path problems, but also for the well-known Traveling Salesman Problem (TSP) and routing problems (Dorigo and Stützle, 2004). However, the use of artificial ants is not limited to graph problems. ACO algorithms can also be used to construct solutions for combinatorial problems, e. g., for pseudo-Boolean functions. In the graph-based ant system (GBAS) by Gutjahr (2000) a solution is constructed by letting an artificial ant traverse a so-called *construction graph* and mapping the path chosen by the ant to a solution for the problem.

The popularity of ACO algorithms has grown rapidly in recent years. The research effort spent to develop, design, analyze, and apply ACO algorithms is demonstrated by a huge number of applications and publications. In order to understand the working principles of ACO algorithms and their dynamic behavior, a solid theoretical foundation is needed. Until 2006 only convergence results were known for ACO algorithms (Gutjahr, 2003) or investigations of simplified models of ACO algorithms (Merkle and Middendorf, 2002). Results on simplified models can give good hints about the real algorithm without simplifying assumptions. However, without bounds on the errors introduced in the simplifications it is usually not possible to draw rigorous conclusions for the real algorithm. Convergence results state that as the time goes to infinity the probability of finding the optimum tends to 1. However, these results do not yield insights into the finite-time behavior of ACO algorithms.

In this chapter we review a new line of research in ACO theory: the computational complexity or runtime analysis of ACO algorithms. The finite-time behavior of ACO algorithms is analyzed using (asymptotic) bounds on the expected time an ACO algorithm needs to find an optimum. These bounds are obtained with mathematical rigor using tools from the analysis of randomized algorithms and they can be used to predict and to judge the performance of ACO algorithms for arbitrary problem sizes.

The analysis of the computational complexity of randomized search heuristics has already been follows for evolutionary algorithms, with great success. The analysis of evolutionary algorithms started for simply structured example functions with interesting properties (see e. g. Droste, Jansen, and Wegener (2002)). Those toy problems encouraged or demanded the development of new methods and tools and revealed important insights into the working principles of evolutionary algorithms. This then allowed the analysis of more sophisticated artificial

problems and, finally, the analysis of problems from combinatorial optimization (see e. g. Giel and Wegener (2003); Neumann and Wegener (2007); Witt (2005)).

It therefore makes sense to start the investigation of ACO algorithms with simply structured example functions as well. This approach has been explicitly demanded in a survey by Dorigo and Blum (2005) for simple pseudo-Boolean problems. In pseudo-Boolean optimization the goal is to maximize a function $f\colon \{0,1\}^n \to \mathbb{R}$. The following functions are well known from the analysis of evolutionary algorithms and have also been considered in the initial rigorous runtime analyses of ACO algorithms.

$$\begin{aligned} \textsc{OneMax}(x) &= \sum_{i=1}^{n} x_i \\ \textsc{LeadingOnes}(x) &= \sum_{i=1}^{n} \prod_{j=1}^{i} x_j \\ \textsc{BinVal}(x) &= \sum_{i=1}^{n} 2^{n-i} x_i \end{aligned}$$

The function OneMax simply counts the number of bits set to 1. This function may also be rediscovered in a practical problem when a specific target point has to be hit and the objective function gives best possible hints towards this point. LeadingOnes counts the number of leading ones in the bit string and BinVal equals the binary value of the bit string. For these problems the bits have different priorities. For BinVal some local changes have a stronger effect than others. In LeadingOnes the bits to the right do not affect the function value until the bits to the left have been set correctly.

The runtime analysis of ACO algorithms has been started independently by Gutjahr (2008) and Neumann and Witt (2006) for (variations of) the function OneMax. Gutjahr (2008) analyzed a GBAS algorithm with a chain construction graph using a slightly different pheromone update rule on a class of generalized OneMax functions. For a fairly large value of ρ, he proved an upper bound of $O(n \log n)$ on the expected number of function evaluations. Neumann and Witt (2006) independently studied a simple ACO algorithm called 1-ANT. The 1-ANT keeps track of the best ant solution found so far. If an iteration creates a solution that is not worse than the best-so-far solution, a pheromone update with respect to the new solution is performed. Otherwise, all pheromones remain unchanged. Another way of putting it is that every new best-so-far solution is reinforced only once. Interestingly, Neumann and Witt (2006) proved that on the function OneMax the evaporation factor ρ has a tremendous impact on the performance of the 1-ANT. If ρ is larger than a certain threshold, the 1-ANT behaves similar to the (1+1) EA. On the other hand, if ρ is below the threshold, the 1-ANT tends to stagnate. As the best-so-far OneMax-value increases more and more, the 1-ANT is forced to discover solutions with more and more 1-bits. However, the impact of the following pheromone update is so small that 1-bits are not rewarded enough and even rediscovering previously set 1-bits gets

increasingly harder. This leads to stagnation and exponential runtimes. A similar behavior was proven by Doerr, Neumann, Sudholt, and Witt (2007) for the functions LeadingOnes and BinVal. On these functions the phase transition between polynomial and superpolynomial optimization times could be identified more precisely with asymptotically tight bounds. Beside the mentioned results on pseudo-Boolean example functions, there are also results for some classical combinatorial optimization problems. Neumann and Witt (2008) investigated the impact of the construction graph in ACO algorithms for the minimum spanning tree problem and Attiratanasunthron and Fakcharoenphol (2008) analyzed an ACO algorithm for the shortest path problem on directed acyclic graphs.

A lesson learned from the analysis of the 1-ANT is that performing an update only in case a new best-so-far solution is found may, in general, not be a good design choice. In fact, many ACO algorithms used in applications use *best-so-far reinforcement* where in every iteration the current best-so-far solution is reinforced. In other words, in every iteration a pheromone update happens, using either the old or a newly generated best-so-far solution. We show how these algorithms, variants of the MAX-MIN ant system (MMAS) (see Stützle and Hoos, 2000), can be analyzed for various example functions, including the class of unimodal functions (Section 3) and plateau functions (Section 4). Thereby, we follow the presentation from Neumann, Sudholt, and Witt (2009), which extends previous work by Gutjahr and Sebastiani (2008). The latter authors first analyzed MMAS variants on OneMax, LeadingOnes, and functions with plateaus. Their results and our contributions show that the impact of ρ is by far not as drastic as for the 1-ANT. When decreasing ρ, the algorithms become more and more similar to random search and the runtime on simple functions grows with $1/\rho$, but there is no phase transition for polynomially small ρ as for the 1-ANT. We also demonstrate how (a restricted formulation of) the fitness-level method can be adapted to the analysis of ACO algorithms. Finally, we present lower bounds for ACO algorithms: a general lower bound for functions with unique optimum that grows with $1/\rho$ and an almost tight lower bound for LeadingOnes.

Often evolutionary algorithms or ACO algorithms are combined with local search procedures. The effect of combining evolutionary algorithms with local search methods has already been examined by Sudholt (2008, 2009). In a typical ACO algorithm without local search, old and new best-so-far solutions are typically quite close to one another and the distribution of constructed ant solutions follows the most recent best-so-far solutions. When introducing local search, old and new best-so-far solutions might be far apart. We discuss this effect in more detail in Section 5, including possible effects on search dynamics. To exemplify the impact on performance, we present an artificial function where an ant algorithm with local search drastically outperforms its variant without local search and a second function where the effect is reversed.

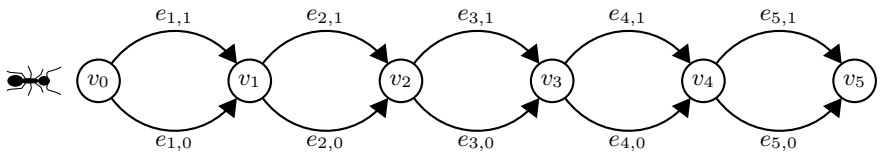

Fig. 1. Construction graph for pseudo-Boolean optimization with $n = 5$ bits.

2 Basic Algorithms

In the following investigations we restrict ourselves to pseudo-Boolean optimization. The goal is to maximize a function $f\colon \{0,1\}^n \to \mathbb{R}$, i. e., the search space consists of bit strings of length n. These bit strings are synonymously called solutions or search points.

A natural construction graph for pseudo-Boolean optimization is known in the literature as *Chain* (Gutjahr, 2008). We use a simpler variant as described by Doerr and Johannsen (2007), based on a directed multigraph $C = (V, E)$. In addition to a start node v_0, there is a node v_i for every bit i, $1 \leq i \leq n$. This node can be reached from v_{i-1} by two edges. The edge $e_{i,1}$ corresponds to setting bit i to 1, while $e_{i,0}$ corresponds to setting bit i to 0. The former edge is also called a *1-edge*, the latter is called *0-edge*. An example of a construction graph for $n = 5$ is shown in Figure 1.

In a solution construction process an artificial ant sequentially traverses the nodes $v_0, v_1, \ldots, v_n$. The decision which edge to take is made according to pheromones on the edges. Formally, we denote pheromones by a function $\tau\colon E \to \mathbb{R}_0^+$. From v_{i-1} the edge $e_{i,1}$ is taken with probability $\tau(e_{i,1})/(\tau(e_{i,0}) + \tau(e_{i,1}))$. We give a formal description of this procedure for arbitrary construction graphs that returns a path P taken by the ant. In the case of our construction graph, we also identify P with a binary solution x as described above and denote the path by $P(x)$.

Operator 1 Construct(C, τ)

Let $P := \emptyset$, $v := v_0$, and mark v as visited.
repeat
  Let E_v be the set of edges leading to non-visited successors of v in C.
  if $E_v \neq \emptyset$ **then**
    Choose $e \in E_v$ with probability $\tau(e)/\sum_{e' | e' \in E_v} \tau(e')$.
    Let $e = (v, w)$, mark w as visited, set $v := w$, and append e to P.
until $E_v = \emptyset$.
return the constructed path P.

All ACO algorithms considered in this chapter start with an equal amount of pheromone on all edges: $\tau(e_{i,0}) = \tau(e_{i,1}) = 1/2$. Moreover, we ensure that $\tau(e_{i,0}) + \tau(e_{i,1}) = 1$ holds, i. e., pheromones for one bit always sum up to 1. This implies that the probability of taking a specific edge equals its pheromone value; in other words, pheromones and traversion probabilities coincide. We remark that the solution construction in ACO algorithms is often enhanced by incorporating heuristic information for parts of the solution. The probability of choosing an edge then depends on both the pheromones and the heuristic information.

Given a solution x and a path $P(x)$ of edges that have been chosen in the creation of x, a pheromone update with respect to x is performed as follows. First, a ρ-fraction of all pheromones evaporates and a $(1-\rho)$-fraction remains. Next, some pheromone is added to edges that are part of the path $P(x)$ of x. However, with these simple rules we cannot exclude that pheromone on some edges may converge to 0, so that the probability of choosing this edge becomes nearly 0. In such a setting, the algorithm has practically no chance to revert a wrong decision. To prevent pheromones from dropping to arbitrarily small values, we follow the MAX-MIN ant system by Stützle and Hoos (2000) and restrict all pheromones to a bounded interval. The precise interval is chosen as $[1/n, 1-1/n]$. This choice is inspired by standard mutations in evolutionary computation where for every bit an evolutionary algorithm has a probability of $1/n$ of reverting a wrong decision.

Depending on whether an edge e is contained in the path $P(x)$ of the solution x, the pheromone values τ are updated to τ' as follows:

$$\begin{aligned} \tau'(e) &= \min\left\{(1-\rho)\cdot\tau(e)+\rho,\ 1-\frac{1}{n}\right\} && \text{if } e \in P(x) \text{ and} \\ \tau'(e) &= \max\left\{(1-\rho)\cdot\tau(e),\ \frac{1}{n}\right\} && \text{if } e \notin P(x). \end{aligned} \tag{1}$$

Note that the pheromones on all 1-edges suffices to describe all pheromones. They form a probabilistic model that somehow reflects a collective memory of many previous paths found by artificial ants.

We consider the runtime behavior of several ACO algorithms that all use the above construction procedure to construct new solutions. One such algorithm is called MMAS_{bs} in Gutjahr and Sebastiani (2008); we refer to it as MMAS*. It is shown on the right-hand side in Figure 2. Note that MMAS* only accepts strict improvements when updating the best-so-far solution. We also investigate a variant of MMAS_{bs} that accepts solutions of equal quality as this enables the algorithm to explore plateaus, i. e., regions of the search space with equal function value. We call this algorithm MMAS, it is displayed on the left-hand side of Figure 2. In Section 3, we will analyze the runtime behavior of MMAS* and MMAS on simple unimodal functions.

ACO algorithms like 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).

MMAS	**MMAS***

Set $\tau(e) := 1/2$ for all $e \in E$.
Construct a solution x^*.
Update pheromones w. r. t. x^*.
repeat
  Construct a solution x.
  if $f(x) \geq f(x^*)$ **then** $x^* := x$.
  Update pheromones w. r. t. x^*.
	Set $\tau(e) := 1/2$ for all $e \in E$.
	Construct a solution x^*.
	Update pheromones w. r. t. x^*.
	repeat
	  Construct a solution x.
	  if $f(x) > f(x^*)$ **then** $x^* := x$.
	  Update pheromones w. r. t. x^*.

(1+1) EA	**(1+1) EA***

Choose x^* uniformly at random.
repeat
  Create x by flipping each bit in x^* independently with prob. $1/n$.
  if $f(x) \geq f(x^*)$ **then** $x^* := x$.
	Choose x^* uniformly at random.
	repeat
	  Create x by flipping each bit in x^* independently with prob. $1/n$.
	  if $f(x) > f(x^*)$ **then** $x^* := x$.

Fig. 2. The algorithms considered in Section 3 and 4. The starred variants on the right-hand side use a strict selection. The left-hand side algorithms also accept solutions of equal quality. If ρ is so large that the pheromone borders are hit in a single pheromone update, MMAS collapses to the (1+1) EA and MMAS* collapses to the (1+1) EA*.

As already pointed out by Jansen and Wegener (2001), the (1+1) EA* has difficulties with simple plateaus of equal quality as no search points of the same function value as the best so far are accepted. Accepting solutions with equal quality 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 (1+1) EA which differs from the (1+1) EA* only in the acceptance criterion. For the sake of completeness, both the (1+1) EA* and the (1+1) EA are also shown in Figure 2. By essentially the same arguments, we also expect MMAS to outperform MMAS* on plateau functions. The corresponding runtime analyses are presented in Section 4. Finally, Section 5 deals with a hybrid of MMAS* and local search. This algorithm is defined in Section 5.

3 On the Analysis of ACO Algorithms

We are interested in the random optimization time or runtime, that is, the number of iterations until a global optimum is sampled first. As both MMAS* and MMAS only evaluate a single new solution in each iteration, this measure equals the number of function evaluations for these algorithms. Often the expected optimization time is considered and we are interested in bounding this expectation from above and/or below. The resulting bounds are asymptotic and stated using

the common notation for asymptotics (see, e. g., Cormen, Leiserson, Rivest, and Stein, 2001). Note that in evolutionary computation an iteration is commonly called a *generation* and the function value is called *fitness*.

The analysis of simple ACO algorithms is more complicated than the analysis of evolutionary algorithms. Typical evolutionary and genetic algorithms can be modelled as Markov chains as the next generation's population only depends on the individuals in the current population. In ACO pheromone traces evaporate only slowly, hence their impact on pheromones is present for a much longer period of time. This means that the next iteration's state typically depends on many previous iterations.

In addition, the state space for the probabilistic model is larger than for evolutionary algorithms. For the simple (1+1) EA there is for each bit a probability of either $1/n$ or $1 - 1/n$ of creating a 1 in the next solution. For an ACO algorithm this probability equals the pheromone on the corresponding 1-edge, which can attain almost arbitrary values in the interval $[1/n,\ 1 - 1/n]$ unless ρ is very large. Note, however, that the number of possible pheromone values for one bit is countably infinite as in every iteration at most two new values can be attained.

The probability of creating a specific ant solution equals the product of probabilities of creating the respective bit value for each bit. This holds since the construction procedure treats all bits independently. At initialization this probability is $(1/2)^n$ for each specific solution as all pheromones equal $1/2$. However, when search becomes more focused and all pheromones reach their borders, the most likely solution has a probability of $(1 - 1/n)^n \approx 1/e$, $e = \exp(1)$, of being (re-)created. In order to understand the dynamic behavior of ACO algorithms it is essential to understand the dynamic behavior of pheromones for specific bits.

Many pseudo-Boolean problems contain bits where a certain bit value is "good" in a sense that a good bit value leads to a better function value than a bad one. We call the creation of such a good value a *success* and speak of a *failure* otherwise. The *success probability* is then the probability of a success, equivalent to the pheromone on the respective edge. We first investigate how quickly the success probability increases if the algorithm only has successes for a while.

Definition 1. *Let p be the current success probability of a specific bit. Let $p^{(t)}$ be its success probability after $t \geq 0$ successes and no failures at the bit.*

The following lemma describes how the rewarded success probability relates to the unrewarded one.

Lemma 1. *For every $t \geq 0$, unless $p^{(t)}$ is capped by pheromone borders,*

$$p^{(t)} \ = \ 1 - (1 - p) \cdot (1 - \rho)^t.$$

Proof. We prove the claim by induction on t. The case $t = 0$ is obvious. For $t \geq 1$, using the induction hypothesis,

$$\begin{aligned} p^{(t)} &= (1-\rho)p^{(t-1)} + \rho \\ &= (1-\rho)(1-(1-p)\cdot(1-\rho)^{t-1}) + \rho \\ &= 1-\rho-(1-p)\cdot(1-\rho)^t + \rho \\ &= 1-(1-p)\cdot(1-\rho)^t. \end{aligned}$$

□

By the preceding lemma, $p \geq q$ implies $p^{(t)} \geq q^{(t)}$ for all $t \geq 0$. Note that this also holds if the upper border for the success probability is hit. This justifies in our forthcoming analyses the places where actual success probabilities are replaced with lower bounds on these probabilities.

3.1 The Fitness-Level Method for the Analysis of ACO

In the following, we demonstrate how to derive upper bounds on the expected optimization time of MMAS* and MMAS on unimodal functions, especially OneMax and LeadingOnes. The techniques from this subsection were first presented by Gutjahr and Sebastiani (2008) in a more general setting, including arbitrary values for the pheromone borders. We stick to the presentation from Neumann et al (2009) where this method was adapted to MMAS* with our precise choice of pheromone bounds. Our presentation follows the presentation of the fitness-level method 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.

There is a similarity between MMAS* and evolutionary algorithms that 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^*. This implies that the best-so-far solution is reinforced again and again until all pheromone values have reached their upper or lower borders corresponding to the setting of the bits in x^*. We can say that x^* has been "frozen in pheromone." The probability of creating a 1 for every bit is now either $1/n$ or $1-1/n$. The distribution of constructed solutions equals the distribution of offspring of the (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 borders, MMAS* behaves like the (1+1) EA* until a solution with larger fitness is encountered. This similarity between ACO and EAs can be used to transfer the fitness-level method to ACO. In particular, upper bounds for MMAS* will be obtained from bounds for the (1+1) EA by adding the so-called freezing time described in the following.

Suppose the current best-so-far solution x^* is not changed and consider the random time t^* until all pheromones reach their borders corresponding to the bit values in x^*. We will refer to this random time as *freezing time*. Plugging the pheromone border $1-1/n$ into Lemma 1, along with the worst-case initial

pheromone value $1/n$, and solving the resulting equation for t, we have that t^* is bounded from above by $-\ln(n-1)/\ln(1-\rho)$. We use $\ln(1-\rho) \leq -\rho$ for $0 \leq \rho \leq 1$ and arrive at the handy upper bound

$$t^* \leq \frac{\ln n}{\rho}. \tag{2}$$

We now use the freezing time t^* to derive a general upper bound on the expected optimization time of MMAS* by making use of the following restricted formulation of the fitness-level method. 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 the (1+1) EA (or, in this case equivalently, the (1+1) EA*) of creating an offspring in $A_{i+1} \cup \cdots \cup A_m$, provided 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 the (1+1) EA and the (1+1) EA* is bounded above by

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

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 of creating 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*:

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

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

$$\frac{m \ln n}{\rho} + \sum_{i=1}^{m-1} \frac{1}{s_i}. \tag{4}$$

The right-hand sum is the upper bound obtained for the (1+1) EA and (1+1) EA* from (3). 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* cannot find a better solution until all pheromones are frozen. 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.

The following theorem has already been proven in 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 $A_i = \{x \mid f(x) = i\}$, $0 \leq i \leq n$. On 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 $1/n \cdot (1-1/n)^{n-1} \geq 1/(en)$. As the events for all $n - i$ 0-bits are disjoint, $s_i \geq (n-i)/(en)$ 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 (4), the upper bound $O((n \log n)/\rho)$ follows. □

The function LeadingOnes counts the number of leading ones in the considered bit string. A non-optimal solution may always be improved by appending a single one to the leading ones. We analyze MMAS* on LeadingOnes 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 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 (4) results in the upper bound $((n+1) \ln n)/\rho + en^2 = O(n^2 + (n \log n)/\rho)$. □

In the remainder of this section we review original results from Neumann et al (2009). First, we apply the fitness-level method to MMAS* on 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 function value, 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 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 Section 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. A function is called *unimodal* if every non-optimal search point has a Hamming neighbor with strictly larger function value. Unimodal functions are often believed to be easy to optimize. This holds if the set of different function values is not too large. On the other hand, Droste, Jansen, and Wegener (2006) proved for classes of unimodal functions with many function values that *every* black-box algorithm needs exponential time on average.

We consider unimodal functions attaining d different fitness values for arbitrary $d \in \mathbb{N}$. Such a function is optimized by the (1+1) EA and (1+1) EA* in expected time $O(nd)$ (see Droste et al (2002)). 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 function 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 the (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 (4), 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 (4), cannot directly be transferred to MMAS as switching between solutions of equal quality 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. Because of the unimodality, each x_i, $1 \leq i \leq m$, has some better Hamming neighbor y_i; however, the y_i need not be disjoint. Let X

and Y denote the event to generate 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 $\text{Prob}(Y \mid X \cup Y) \geq 1/n^2$, which is implied by $\text{Prob}(Y) \geq \text{Prob}(X)/(n^2-1)$.

If $p(x_i)$ is the probability of constructing x_i then $p(x_i)/p(y_i) \leq (1-\frac{1}{n})/\frac{1}{n} = n-1$ 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 $\text{Prob}(Y) \geq \frac{1}{n}\sum_{i=1}^{m} p(y_i)$ and

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

Therefore, $\text{Prob}(Y) \geq \text{Prob}(X)/(n^2-1)$ 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 $\rho = 1/\text{poly}(n)$.

3.3 A General Lower Bound

In order to judge the quality of upper bounds on expected optimization times it is helpful to aim at lower bounds. We present a lower bound that is weak but very general; it holds for both MMAS and MMAS* on all functions where the global optimum is unique.

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

Proof. As 0-edges and 1-edges are treated symmetrically, we can assume w. l. o. g. that 1^n is the unique optimum. If, for each bit, the success probability (defined as the probability of creating a 1) 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 $(1-1/\sqrt{n})^n \leq e^{-\sqrt{n}}$. Using the uniform initialization and Lemma 1, 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

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

Since $\rho = 1/\text{poly}(n)$ and, therefore $t = \text{poly}(n)$, 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. □

There is often a trade-off between quality and generality of runtime bounds. The bounds presented so far are general, but they often do not represent the best possible bounds when considering specific functions. The following section shows exemplarily that stronger results can be obtained by more specialized investigations.

3.4 Specialized Analyses

The fitness-level method is a quite powerful and general method to derive upper bounds on the expected optimization time. However, as mentioned above, it relies on the pessimistic assumption that pheromones have to be frozen after each improvement of the function value. This provably leads to an overestimation of the expected runtime on the function LeadingOnes. Neumann et al (2009) proved the following bound which is much more precise.

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

The first bound is better than the one from Theorem 2 by a factor of $\Theta(\log n)$ if $\rho = O(1/n)$. In addition, the second bound basically saves another factor of $\Theta(\log n)$ if $\rho \leq 1/n^{1+\Omega(1)}$ holds, which, e. g., is the case if $\rho \leq 1/n^2$.

The proof of the improved bounds makes use of the following observation: As soon as a bit contributes to the LeadingOnes-value, i. e., the bit is part of the block of leading ones, its pheromone value will only increase in the following. (To decrease the pheromone, a solution with less leading ones would have to be accepted, which is ruled out by the selection.) Hence, if bits enter the block of leading ones one after the other with a certain delay, earlier gained bits will have had time to reach the maximal pheromone value. The freshly gained bits are likely to have some intermediate pheromone value, but if the delay between improvements is large enough, there is still a good probability of rediscovering all bits of the block of leading ones with a good probability. Basically, the analysis for the first bound from Theorem 6 shows that an average delay of $\Theta(1/\rho)$ steps between subsequent improvements is enough. Then there is always only a small "window" consisting of $O(\log n)$ bits in the block of leading ones where the pheromone values have not yet reached the upper borders. The bits in the window can be ranked from the left to the right, which means that the bits to the left have received more increases of their pheromone. Taking also this into consideration, a technical calculation proves the first bound from Theorem 6. Repeating these arguments for a smaller average delay of $\Theta(1/(\varepsilon\rho\ln(1/\rho))$ yields the second bound.

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 $\rho \leq n^{-(1+\Omega(1))}$, hence for almost all ρ.

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

The proof of this lower bound is the most demanding analysis in the paper by Neumann et al (2009). It makes use of tailored probabilistic tools for the analysis of a so-called martingale process. Such processes arise at the bits that have never contributed to the LeadingOnes-value so far. For this reason, their pheromone values are completely random, and, unless the bits become part of the block of leading ones, their expected values remain at their initial value of $1/2$. This is exactly the property of a martingale. However, there are random fluctuations (a variance of the random change of pheromone) at these bits which drive their pheromone values away from the "middle" $1/2$ by means of an unbiased random walk. During the time needed to gather the first $n/2$ leading ones, the pheromones of the rightmost $n/2$ bits therefore tend to reach one of their borders $1/n$ and $1 - 1/n$. Due to symmetry, on average half of these bits touch their lower border and have a good chance to remain there until the respective bit becomes the first 0-bit. Then an event of probability $1/n$ is necessary to "flip" the bit. As this situation is likely to occur a linear number of times, the bound $\Omega(n^2)$ follows. Finally, the bound $\frac{n/\rho}{\log(2/\rho)}$ goes back to an analysis on the average time between two improvements. Similarly as above, the "window" of bits with an intermediate pheromone value among the leading ones is studied. If the average time between two improvements is too small, the windows becomes too large and the probability of rediscovering the leading ones is too low to sustain the average time between two improvements.

Without going into the proof details, the summaries from this subsection should emphasize that specialized techniques for the analysis of ACO algorithms can lead to much improved results. However, much more involved proofs are required and the methods are not easy to transfer to different scenarios.

4 How ACO Algorithms Deal with Plateaus

Rigorous runtime analyses can be used to estimate the expected optimization time and to make precise predictions about the practical performance of ACO algorithms. This can be used to clarify important design issues from a rigorous perspective. One such issue already discussed in Gutjahr (2007); Gutjahr and Sebastiani (2008) is the question whether the current best-so-far solution should be replaced by a new solution with the same function value. This section again reviews results from Neumann et al (2009).

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.

We consider the function Needle where only one single solution has objective value 1 and the remaining ones get value 0. In its general form, the function is

defined as

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

where x_{OPT} is the unique global optimum. Gutjahr and Sebastiani (2008) compare MMAS* and (1+1) EA* with respect to 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 the (1+1) EA*. The reason is that MMAS* behaves nearly as random search on the search space while the initial solution of the (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 of obtaining the optimal solution in the next step is at most $2^{-n/3}$. After at most $t^* \leq (\ln n)/\rho$ (see Inequality (2)) 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 of producing 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 of choosing 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 of creating x_{OPT} equals n^{-n}. Using the ideas of Theorem 8, the following corollary can be proved which asymptotically matches the lower bound for the (1+1) EA* given in 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 that accepts each new solution has expected optimization time $\Theta(2^n)$ on NEEDLE (see Garnier, Kallel, and Schoenauer, 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 prove that MMAS is more efficient than MMAS* and almost competitive to the (1+1) EA. The following theorem even shows that the expected optimization time of MMAS on NEEDLE is at most by a polynomial factor larger than the one of the (1+1) EA unless ρ is superpolynomially small. We sketch the proof ideas and refer the reader to Neumann et al (2009) for the full proof.

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

Sketch of 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 $\mathrm{E}(t(n)) \cdot 2^n$. The theorem follows if we can show that $\mathrm{E}(t(n)) = O((n^2 \log n)/\rho^2)$.

In order to bound $t(n)$, we study a 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 $O(n^2/\rho^2)$ (see Neumann et al (2009)). Due to Markov's 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 $O(n^2/\rho^2 \cdot \log n)$ 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, $\mathrm{E}(T^*) = O((n^2 \log n)/\rho^2)$ follows. By definition, all success probabilities have been at least 1/2 at least once after T^* steps. One can show that then the probability of creating bit value 1 for such a bit remains at least 1/2. Hence, 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 random-

ized search heuristics have a good chance to leave them within a polynomial number of steps. Gutjahr and Sebastiani (2008) consider the function NH-OneMax that consists of the Needle-function on $k = \log n$ bits and the function OneMax on $n - k$ bits, which 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. *If $\rho = 1/\text{polylog}(n)$, 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 $\mathrm{E}(t(n)) = O((n^2 \log n)/\rho^2)$ 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 $O((n^2 \log n)/\rho^2 \cdot n)$ steps. After this goal has been achieved, the unimodal function OneMax, which contains at most $n+1$ different function 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 The Effect of Hybridizing ACO with Local Search

In this section we now turn to the hybridization of ACO with local search and follow investigations carried out by Neumann, Sudholt, and Witt (2008). Combining ACO with local search methods is quite common (see, e. g., Dorigo and Stützle, 2004; Hoos and Stützle, 2004; Levine and Ducatelle, 2004). Experimental investigations show that the combination of ACO with a local search procedure improves the performance significantly. On the other hand, there are examples where local search cannot help to improve the search process or even mislead the search process (Balaprakash, Birattari, Stützle, and Dorigo, 2006). Therefore, it is interesting to figure out how the incorporation of local search into ACO algorithms can significantly influence the optimization process.

Our aim is to point out situations where the effect of local search becomes visible in a way that can be tackled by rigorous arguments. Therefore we present functions where MMAS variants with and without local search show a strongly different runtime behavior. On one function, MMAS with local search outperforms MMAS without local search, while on a different function the effect is reversed. The differences are so drastic that the question of whether to use local search or not decides between polynomial and exponential runtimes.

We enhance MMAS* with local search and call the result MMAS-LS*. In the following, LocalSearch(x) is a procedure that, starting from x, repeatedly replaces the current solution by a Hamming neighbor with a strictly larger function value until a local optimum is found. We do not specify a pivot rule, hence we implicitly deal with a class of algorithms.

Algorithm 2 MMAS-LS*

Set $\tau(e) := 1/2$ for all $e \in E$.
Construct a solution x^*.
Set $x^* :=$ LocalSearch(x^*).
Update pheromones with respect to x^*.
repeat
    Construct a solution x.
    Set $z :=$ LocalSearch(x).
    if $f(z) > f(x^*)$ **then** $x^* := z$.
    Update pheromones with respect to x^*.

When taking the number of function evaluations as performance measure, the computational effort of local search has to be accounted for. The objective functions considered in the following only have a linear number of function values, hence the number of function evaluations in one local search call is bounded by $O(n)$. Depending on the pivot rule, the number of evaluations needed to find a better Hamming neighbor may vary; however, it is trivially bounded by n. Hence, the number of function evaluations is at most by a factor $O(n^2)$ larger than the number of iterations.

Hybridization with local search has become very popular for various other optimization paradigms such as estimation-of-distribution algorithms (Aickelin, Burke, and Li, 2007) and evolutionary algorithms. Evolutionary algorithms using local search are known as *memetic (evolutionary) algorithms*. They have been successfully applied to many combinatorial problems, see the book by Hart, Krasnogor, and Smith (2004) and the survey by Krasnogor and Smith (2005). One particular memetic algorithm is known *iterated local search* (Lourenço, Martin, and Stützle, 2002) where local search is used in every generation to turn new search points into local optima. The first rigorous runtime analyses for memetic algorithms were presented by Sudholt (2009). The author highlights the importance of a good balance between local search and evolutionary search. He presents examples where small changes to the parametrization have a huge impact on the performance of simple memetic algorithms. A further study for problems from combinatorial optimization (Sudholt, 2008) demonstrates that an iterated local search algorithm with a sophisticated local search can outperform several common heuristics on simply structured problem instances.

In the rest of this chapter, we want to examine the effect of combining ACO algorithms with local search methods. Neumann et al (2008) have pointed out that the effect of using local search with ACO algorithms is manifold. Firstly, local search can help to find good solutions more quickly as it increases the "greediness" within the algorithm. As argued in Sudholt (2009) for memetic algorithms, local search can also be used to discover the real "potential" of a solution as it can turn a bad looking solution into a good local optimum. Moreover, the pivot rule used in local search may guide the algorithm towards certain regions of the search space. For example, first ascent pays more attention to the first bits in the bit string, which may induce a search bias. However, we will not deal with this effect here. In particular, our functions are designed such that the pivot rule is not essential.

There is another effect that has been investigated more closely in Neumann et al (2008). The pheromone values induce a sampling distribution over the search space. On typical problems, once the best-so-far solution has reached a certain quality, sampling new solutions with a high variance becomes inefficient and the current best-so-far solution x^* is maintained for some time. The analyses from Section 3 have shown that then the pheromones quickly reach the upper and lower borders corresponding to x^*. This means that the algorithm turns to sampling close to x^*. In other words, MMAS variants typically reach a situation where the "center of gravity" of the sampling distribution follows the current best-so-far solution and the variance of the sampling distribution is low.

When introducing local search into an MMAS algorithm, this may not be true. Local search is able to find local optima that are far away from the current best-so-far solution. In this case the "center of gravity" of the sampling distribution is far away from the best-so-far solution.

Assume there is a path of Hamming neighbors with increasing function value leading to a local optimum. Assume further that all points close to the path have lower quality. Then for MMAS* it is likely that the sampling distribution closely

follows the path. The path of increasing function value need not be straight. In fact, it can make large bends through the search space until a local optimum is reached. On the other hand, MMAS-LS*, when starting with the same setting, will reach the local optimum within a single iteration of local search. Then the local optimum becomes the new best-so-far solution x^*, while the sampling distribution is still concentrated around the starting point. In the following iterations, as long as the best-so-far solution is not exchanged, the pheromone values on all bits synchronously move towards their respective borders in x^*. This implies for the sampling distribution that the "center of gravity" takes a (sort of) direct route towards the local optimum, irrespective of the bent path taken by local search. An illustration is given in Figure 3.

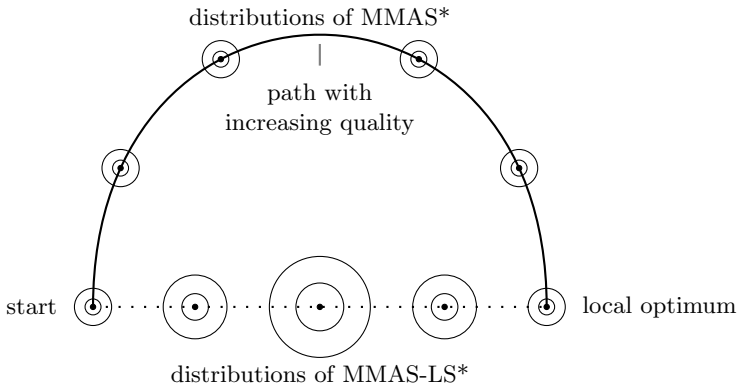

Fig. 3. A sketch of the search space showing the behavior of MMAS* and MMAS-LS*. The dots and circles indicate the sampling distributions of MMAS* and MMAS-LS*, respectively, at different points of time. While the distribution of MMAS* tends to follow the path of increasing function value from left to right, the distribution of MMAS-LS* takes a direct route towards the local optimum.

Consequences are that different parts of the search space are sampled by MMAS* and MMAS-LS*, respectively. Moreover, with MMAS* the variance in the solution construction is always quite low as the sampling distribution is concentrated on certain points on the path. But when the best-so-far solution with local search suddenly moves a long distance, the variance in the solution construction may be very high as the bits differing between the starting point and x^* may have pheromones close to 1/2. These bits are assigned almost randomly. This strongly resembles a uniform crossover operation well known in evolutionary computation. There, every bit in the offspring receives a bit value from a parent that is chosen uniformly at random and anew for each bit, which implies that bits differing in the two parents are assigned randomly. MMAS-LS* in this

setting therefore simulates a uniform crossover of the starting point and the local optimum x^*.

Our aim in the following is to create functions where MMAS* and MMAS-LS* have a different runtime behavior. Moreover, we want the performance difference to be drastic in order to show how deep the impact of local search can possibly be. To this end, we exploit that the sampling distributions can follow different routes through the search space. For one function we place a target region with many global optima on the straight line between starting point and local optimum and turn the local optimum into a trap that is hard to overcome. In such a setting, we expect MMAS-LS* to drastically outperform MMAS*. Contrarily, if the region of global optima is made a trap region and a new global optimum is placed close to the former trap, we expect MMAS-LS* to get trapped and MMAS* to find the global optimum effectively.

5.1 A Function where Local Search is Beneficial

We now formally define a function where local search is beneficial according to the ideas described above. It is named SP-Target (short path with target). The path with increasing function value is given by the set $\mathrm{SP} = \{1^i 0^{n-i} \mid 0 \leq i \leq n\}$. The path ends with the local optimum 1^n. Let $|x|_1$ denote the number of 1-bits in x and $|x|_0$ denote the number of 0-bits. A large target area containing all global optima is specified by $\mathrm{OPT} = \{x \mid |x|_1 \geq (3/4) \cdot n \wedge H(x, \mathrm{SP}) \geq n/(\gamma \log n)\}$, where $H(x, \mathrm{SP})$ denotes the Hamming distance of x to the closest search point of SP and $\gamma \geq 1$ is a constant to be chosen later. For all remaining search points, the function SP-Target gives hints to reach 0^n, the start of the path SP.

$$\text{SP-Target}(x) := \begin{cases} |x|_0 & \text{if } x \notin (\mathrm{SP} \cup \mathrm{OPT}), \\ n + i & \text{if } x = 1^i 0^{n-i} \in \mathrm{SP}, \\ 3n & \text{if } x \in \mathrm{OPT}. \end{cases}$$

The function SP-Target is sketched in Figure 4. Note that we have actually defined a class of functions dependent on γ. All following results will hold for arbitrary constant $\gamma \geq 1$ unless stated otherwise.

The following theorem shows that MMAS* without local search is not successful. We restrict ourselves to polynomially large $1/\rho$ here and also in the following as otherwise the ACO component would be too close to random search.

Theorem 12. *Choosing $\rho = 1/\mathrm{poly}(n)$, the optimization time of MMAS* on* SP-Target *is at least $2^{cn^{2/9}}$ with probability $1 - 2^{\Omega(n^{2/9})}$ for some constant $c > 0$.*

To prove the preceding theorem, we have to take into account situations where the pheromone values of MMAS* have not yet reached their borders and the construction procedure samples with high variance. This is the case in particular after initialization. The following lemma will be used to check the probability of finding the optimum in the early steps of MMAS* on SP-Target.

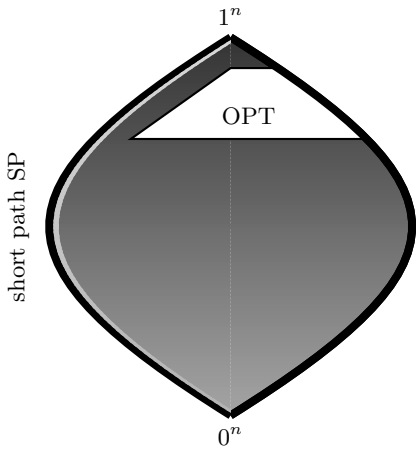

Fig. 4. Illustration of the Boolean hypercube and the function SP-Target. The vertical position of a search point is determined by the number of 1-bits. Its horizontal position is determined by the position of 1-bits in the bit string. The objective value is indicated by the brightness; dark areas indicate low values and light areas indicate high values.

Lemma 2. *If the best-so-far solution of MMAS* has never had more than* $2n/3$ 1*-bits, the probability of creating a solution with at least* $3n/4$ 1*-bits is* $2^{-\Omega(n)}$ *in each iteration.*

Proof. The proof is an application of Chernoff bounds with respect to the number of ones in the solutions created by MMAS*. Let the potential $P_t := p_1 + \cdots + p_n$ at time t denote the current sum of the probabilities of sampling ones over all bits, which, by definition of the construction procedure, equals the expected number of ones in the next constructed solution. Observe that $P_t \leq 2n/3$ implies by Chernoff bounds that the probability of creating a solution with at least $3n/4$ 1-bits is $2^{-\Omega(n)}$. We now show: if all best-so-far solutions up to time t have at most $2n/3$ ones, then $P_i \leq 2n/3$ for $0 \leq i \leq t$. This will prove the lemma.

For the last claim, we denote by k the number of ones in the best-so-far solution according to which pheromones are updated. Due to the pheromone update mechanism, the new potential P_{i+1} is obtained from P_i and k according to $P_{i+1} = (1-\rho)P_i + k\rho$. Hence, if $P_i \leq 2n/3$ and $k \leq 2n/3$ then also $P_{i+1} \leq 2n/3$. The claim follows by induction since $P_0 = n/2 \leq 2n/3$. □

Proof of Theorem 12. We distinguish two phases in the run according to the best-so-far solution x^*. Phase 1 holds as long as $x^* \notin \text{SP}$ and $|x^*|_1 \leq 2n/3$, and Phase 2 applies as long as $x^* \in \text{SP}$. Our aim is to show that a typical run passes through the two phases in their order with a failure probability of $2^{-\Omega(n^{2/9})}$. The probability of finishing the second phase will be bounded by $2^{-\Omega(n^{2/9})}$ for each step of the phase. This implies the theorem as, by the union bound, the total probability in $2^{cn^{2/9}}$ iterations, $c > 0$ a small constant, is still $2^{-\Omega(n^{2/9})}$.

Consider the first (and best-so-far) solution x^* created by MMAS*. By Chernoff bounds, $n/3 \leq |x^*|_1 \leq 2n/3$ with probability $1 - 2^{-\Omega(n)}$. There is only a single solution in SP for each value of $|x^*|_1$. By the symmetry of the construction procedure, we conclude $\text{Prob}(x^* \in \text{SP} \mid |x^*|_1 = k) = 1/\binom{n}{k}$. The last expression is $2^{-\Omega(n)}$ for $n/3 \leq k \leq 2n/3$. Hence, with probability $1-2^{-\Omega(n)}$, there is a non-empty Phase 1. By Lemma 2, the probability that a specific iteration in Phase 1 creates an optimum is $2^{-\Omega(n)}$. Otherwise, the behavior is as for MMAS* on the function $|x|_0$. Using $\rho = 1/\text{poly}(n)$ and the analyses for the symmetric function $\textsc{OneMax}(x) = |x|_1$ from Section 3.1, the expected time until the first phase is finished is polynomial. The total failure probability in Phase 1 is bounded by the product of its expected length and the failure probability in a single iteration. Therefore, the total failure probability for the first phase is still of order $2^{-\Omega(n)}$.

In Phase 2 we have $x^* \in \text{SP}$. The goal is now to show that a solution from SP with high probability can only be created if the sampling distribution is sufficiently concentrated around solutions in SP. This in turn makes creating solutions of high Hamming distance from SP, including OPT, very unlikely.

We make this idea precise and consider a point $1^i0^{n-i} \in \text{SP}$. This search point consists of a prefix of i ones and a suffix of $n-i$ zeros. For a newly constructed solution x we define $P(i) := p_1 + \cdots + p_i$ as the expected number of ones in the prefix and $S(i) := (1-p_{i+1}) + \cdots + (1-p_n)$ as the expected number of zeros in the suffix. The number of ones in the prefix plus the number of zeros in the suffix yields the number of bits equaling in 1^i0^{n-i} and x, i. e., $n - H\big(1^i0^{n-i}, x\big)$. We call $P(i)$ ($S(i)$) *insufficient* if and only if $P(i) \leq i - i^{2/3}$ ($S(i) \leq (n-i) - (n-i)^{2/3}$) holds. We now show that with insufficiencies it is very unlikely to create 1^i0^{n-i}. As this holds for all i, we conclude that if SP is reached after a certain number of iterations, the pheromones do not have insufficiencies, with high probability.

Let $s(i)$ denote the probability of constructing the solution 1^i0^{n-i}. We distinguish three cases and apply Chernoff bounds to prove the following implications:

Case 1: $i < n^{2/3}$. Then insufficient $S(i)$ implies $s(i) = 2^{-\Omega(n^{1/3})}$.

Case 2: $i > n - n^{2/3}$. Then insufficient $P(i)$ implies $s(i) = 2^{-\Omega(n^{1/3})}$.

Case 3: $n^{2/3} \leq i \leq n - n^{2/3}$. Then insufficient $P(i)$ and insufficient $S(i)$ each imply $s(i) = 2^{-\Omega(n^{2/9})}$.

We assume that the described insufficiencies do not occur whenever a best-so-far solution $x^* = 1^i0^{n-i}$ in Phase 2 is accepted. The failure probability is $2^{-\Omega(n^{2/9})}$ for each new best-so-far solution x^*. Iterations in between two exchanges of x^* cannot create insufficiencies as $P(i)$ and $S(i)$ can only increase as long as x^* is maintained. Hence, we do not have insufficiencies in Phase 2 for at least $2^{\Omega(n^{2/9})}$ iterations with probability at least $1 - 2^{-\Omega(n^{2/9})}$.

Being in Phase 2 without insufficiencies, we show depending on the three cases for the current $x^* = 1^i0^{n-i}$ that creating an optimal solution has probability $2^{-\Omega(n^{2/9})}$. In the first case, the expected number of zeros in the suffix of x is at least $(n-i) - (n-i)^{2/3}$. By Chernoff bounds, the random number of zeros is at least $(n-i) - 2(n-i)^{2/3}$ with probability at least $1 - 2^{-\Omega(n^{1/3})}$. Along with $i < n^{2/3}$, it follows that then the solution has Hamming distance at

most $3n^{2/3}$ from SP. By the definition of SP-Target, this is not enough to reach OPT. The second case is treated analogously. In the third case, the probability of obtaining less than $i-2i^{2/3}$ ones in the prefix or less than $(n-i)-2(n-i)^{2/3}$ zeros in the suffix is altogether bounded by $2^{-\Omega(n^{2/9})}$. Then the solution has Hamming distance at most $4n^{2/3}$ from SP, which is also not enough to reach the optimum. This finishes the analysis of the second phase, and, therefore, proves the theorem. □

The following theorem proves the benefits of local search. Recall that the number of function evaluations is at most by a factor of $O(n^2)$ larger than the stated optimization time.

Theorem 13. *Choosing* $1/\mathrm{poly}(n) \le \rho \le 1/16$*, the optimization time of MMAS-LS* on* SP-Target *is* $O(1/\rho)$ *with probability* $1-2^{-\Omega(n)}$*. If* $\gamma \ge 1$ *is chosen large enough but constant, the expected optimization time is also* $O(1/\rho)$*.*

Proof. Note that every call of local search ends either with 1^n or with a global optimum. If the initial local search creates $x^* = 1^n$, all pheromone values increase simultaneously and uniformly from their initial value 1/2 towards their upper border $1-1/n$. We divide a run into two phases. The first phase ends when either all pheromones become larger than 27/32 or when a global optimum has been found. The second phase ends when a global optimum has been found, hence it is empty if the first phase ended with an optimum.

We bound the length of the first phase by the first point of time t^* where all pheromone values exceed 27/32. By Lemma 1 after t steps the pheromone values are at least $\min\{1-1/n,\ 1-(1/2)(1-\rho)^t\}$. Solving the equation

$$1-\left(\frac{1}{2}\right)(1-\rho)^t \;=\; 27/32 \iff (1-\rho)^t \;=\; 5/16$$

yields the upper bound

$$t^* \;\le\; \left\lceil \frac{\ln(5/16)}{\ln(1-\rho)} \right\rceil \;\le\; \frac{\ln(16/5)}{\rho} + 1 \;=\; O(1/\rho).$$

The assumption $\rho \le 1/16$ implies that at the last step in the first phase the pheromone value at every bit is within the interval $[25/32, 27/32]$, pessimistically assuming that a global optimum has not been found before (neither by a constructed ant solution, nor by local search). The next constructed search point x then fulfills the following two properties with probability $1-O(2^{-n/2400})$:

1. $\frac{3n}{4} \le |x|_1 \le \frac{7n}{8}$,
2. $H(x, \mathrm{SP}) \ge n/(\gamma \log n)$.

Using Chernoff bounds with $\delta := 1/25$, the failure probability for the first event is at most $2e^{-(25n/32)(\delta^2/3)} = 2e^{-n/2400}$. To bound the failure probability of the second event, given the first event, we exploit that all pheromone values are equal. Therefore, if we know that $|x|_1 = k$ then x is uniform over all search

points with k ones. Since the number of search points with k ones is monotone decreasing for $3n/4 \leq k \leq 7n/8$, we only consider search points with $k = 7n/8$ ones as a worst case. The number of such search points is $\binom{n}{n/8}$, and the number of search points of Hamming distance at most $m := n/(\gamma \log n)$ from SP is at most $m \cdot \binom{n}{m}$. Altogether, the probability of $H(x, \mathrm{SP}) \leq m$, given that $3n/4 \leq |x|_1 \leq 7n/8$, is bounded from above by

$$\frac{m\binom{n}{m}}{\binom{n}{n/8}} \;\leq\; \frac{m\left(\frac{en}{m}\right)^m}{\left(\frac{n}{n/8}\right)^{n/8}} \;\leq\; m \cdot 2^{o(n)} \cdot 8^{-n/8}.$$

The last expression is even $O(2^{-n/8})$. Altogether, the sum of the failure probabilities is $O(2^{-n/2400})$ as suggested, and the first statement follows.

For the second statement we estimate the time in the second phase, provided that the first phase has been unsuccessful. Using the bound $(\ln n)/\rho$ on the expected freezing time from Section 3.1 and $\rho = 1/\mathrm{poly}(n)$, the time to reach the pheromone border is $O((\log n)/\rho) = \mathrm{poly}(n)$, or an optimum is created anyway. With all pheromones at the upper border, the solution construction process equals a standard mutation of 1^n, i. e., flipping each bit in 1^n independently with probability $1/n$. Flipping the first m bits results in a global optimum as 0^m1^{n-m} has Hamming distance at least m to 1^i0^{n-i} for every i. The probability of creating 0^m1^{n-m} in a standard mutation is at least

$$\left(\frac{1}{n}\right)^{n/(\gamma \log n)} \left(1 - \frac{1}{n}\right)^{n - n/(\gamma \log n)} \geq e^{-1} \cdot 2^{-n/\gamma}.$$

This means that the expected time in the second phase is $O(\mathrm{poly}(n)2^{n/\gamma})$. Using that the first phase is unsuccessful only with probability $O(2^{-n/2400})$ and applying the law of total probability, the expected optimization time altogether is $O(1/\rho) + O(2^{-n/2400}) \cdot O(\mathrm{poly}(n)2^{n/\gamma})$. The latter is $O(1)$ for $\gamma > 2400$, which proves the second statement. □

5.2 A Function where Local Search is Detrimental

Similarly to the function SP-Target, we design another function SP-Trap (short path with trap) where local search is detrimental, using ideas from Section 5. We take over the path with increasing function value, $\mathrm{SP} = \{1^i0^{n-i} \mid 0 \leq i \leq n\}$, but in contrast to SP-Target, the former region of global optima now becomes a trap, $\mathrm{TRAP} = \{x \mid |x|_1 \geq (3/4) \cdot n \wedge H(x, \mathrm{SP}) \geq n/\log n\}$. The unique global optimum is placed within distance 2 from the local optimum: $\mathrm{OPT} = \{0^21^{n-2}\}$. This ensures that local search climbing the path SP cannot reach the global optimum. All remaining search points give hints to reach the start of the path.

$$\text{SP-Trap}(x) := \begin{cases} |x|_0 & \text{if } x \notin (\mathrm{SP} \cup \mathrm{TRAP} \cup \mathrm{OPT}), \\ n + i & \text{if } x = 1^i0^{n-i} \in \mathrm{SP}, \\ 3n & \text{if } x \in \mathrm{TRAP}, \\ 4n & \text{if } x \in \mathrm{OPT}. \end{cases}$$

The function SP-Trap is sketched in Figure 5.

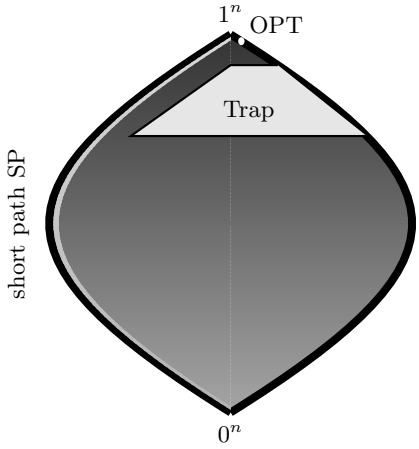

Fig. 5. Illustration of the Boolean hypercube and the function SP-Trap. The vertical position of a search point is determined by the number of 1-bits. Its horizontal position is determined by the position of 1-bits in the bit string. The objective value is indicated by the brightness; dark areas indicate low values and light areas indicate high values.

In the remainder of this section, we prove that MMAS* is efficient on SP-Trap while MMAS-LS* fails dramatically. Tuning the definition of SP-Trap, we could also extend the following theorem by a polynomial bound on the expected optimization time. We refrain from such modifications to illustrate the main effects.

Theorem 14. *Choosing $\rho = 1/poly(n)$, the optimization time of MMAS* on* SP-Trap *is $O((n \log n)/\rho + n^3)$ with probability $1 - 2^{-\Omega(n^{2/9})}$.*

Proof. By the argumentation from the proof of Theorem 12, the probability that a solution in TRAP is produced within $O(n^3)$ iterations is at most $2^{-\Omega(n^{2/9})}$.

Under the assumption that TRAP is never reached until the global optimum is found, MMAS* behaves equally on SP-Trap and a modified function where $x \in$ TRAP receives value $|x|_0$. We apply the fitness-level method described in Section 3.1 to estimate the expected optimization time on the latter, easier function. The number of fitness levels is $O(n)$. On every fitness level, the number of iterations until either all pheromones are frozen or the current best-so-far solution has improved is bounded by $(\ln n)/\rho$ with probability 1. We pessimistically assume that an improvement can only happen once all pheromones have been frozen. Then the optimization time is bounded by $O((n \log n)/\rho)$ plus the sum of waiting times for improvements on all fitness levels. Showing that the latter quantity is bounded by $O(n^3)$ with probability $1 - 2^{-\Omega(n)}$ completes the proof.

After freezing, the probability for an improvement from $x^* = 1^n$ equals $(1/n)^2 \cdot (1 - 1/n)^{n-2} \geq 1/(en^2)$. For all other $x^* \notin$ TRAP, there is always a

better Hamming neighbor, hence the probability for an improvement is at least $1/(en)$. Together, the expected waiting times for improvements on all fitness levels sum up to $en^2 + O(n) \cdot en = O(n^2)$. By Markov's inequality the probability of waiting more than cn^2 steps is at most $1/2$ for a suitable constant $c > 0$. Hence, the probability that more than n independent phases of length cn^2 are needed is bounded by $2^{-\Omega(n)}$. Therefore, the bound $O(n^3)$ holds with probability $1 - 2^{-\Omega(n)}$. □

Theorem 15. *Choosing* $1/\mathrm{poly}(n) \leq \rho \leq 1/16$*, the optimization time of MMAS-LS* on* SP-Trap *is at least* 2^{cn} *with probability* $1-2^{-\Omega(n)}$ *for some constant* $c > 0$*.*

Proof. We follow the lines of the proof of Theorem 13. As long as OPT $= 0^2 1^{n-2}$ is not created, the behavior of MMAS-LS* on SP-Trap and SP-Target is identical. Reconsider the first phase described in the proof of Theorem 13 (with the former OPT replaced by TRAP) and denote by $P := p_1 + \cdots + p_n$ the sum of probabilities of sampling ones over all bits. Throughout the phase, $P \leq 27n/32$, hence the probability of sampling at least $n-2$ ones, which is necessary to reach OPT, is $2^{-\Omega(n)}$ according to Chernoff bounds.

With probability $1 - 2^{-\Omega(n)}$, the first best-so-far solution 1^n is replaced by some $x^{**} \in$ TRAP where $|x^{**}|_1 \leq 7n/8$ when the first phase is ended. Due to strict selection, x^{**} then can only be replaced if OPT is created. The latter has probability $2^{-\Omega(n)}$ for the following reasons: the P-value is at most $27n/32 \leq 7n/8$ when x^{**} is accepted. Hence, following the argumentation from the proof of Lemma 2, the P-value will not exceed $7n/8$ unless x^{**} is replaced. With a P-value of at most $7n/8$, creating OPT has probability $2^{-\Omega(n)}$ and the claim follows for a suitable constant $c > 0$. □

6 Conclusions

Ant colony optimization is a powerful metaheuristic that has found many applications for combinatorial and adaptive problems. In contrast to the rich number of successful applications, the theoretical understanding lags far behind practical success. A solid theoretical foundation is necessary to get a rigorous understanding of how ACO algorithms work. We have shown how simple ACO algorithms can be analyzed with respect to their computational complexity on example functions with different properties. This enables practitioners to gain new insights into their dynamic behavior and to clarify design issues, so that better algorithms can be developed. In particular, we have addressed the question whether the best-so-far solution should be replaced by new solutions of the same quality. As it is common practice to hybridize ACO with local search, we have discussed possible effects of introducing local search from a theoretical perspective and pointed out situations where the use of local search is either provably beneficial or provably disastrous.

Bibliography

Aickelin U, Burke EK, Li J (2007) An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. Journal of the Operational Research Society 58:1574–1585

Attiratanasunthron N, Fakcharoenphol J (2008) A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs. Information Processing Letters 105(3):88–92

Balaprakash P, Birattari M, Stützle T, Dorigo M (2006) Incremental local search in ant colony optimization: Why it fails for the quadratic assignment problem. In: Proceedings of ANTS Workshop (ANTS '06), pp 156–166

Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to Algorithms, 2nd edn. The MIT Press

Doerr B, Johannsen D (2007) Refined runtime analysis of a basic ant colony optimization algorithm. In: Proceedings of the Congress of Evolutionary Computation (CEC '07), IEEE Press, pp 501–507

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), ACM Press, pp 33–40

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. MIT Press

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

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: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03), Springer, pp 415–426

Gutjahr WJ (2000) A graph-based ant system and its convergence. Future Generation Computer Systems 16:873–888

Gutjahr WJ (2003) A generalized convergence result for the graph-based ant system metaheuristic. Probability in the Engineering and Informational Sciences 17:545–569

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

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

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

Hart WE, Krasnogor N, Smith JE (eds) (2004) Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, vol 166. Springer
Hoos HH, Stützle T (2004) Stochastic Local Search: Foundations & Applications. Elsevier/Morgan Kaufmann
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 5:589–599
Kennedy J, Eberhart RC, Shi Y (2001) Swarm Intelligence. Morgan Kaufmann
Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9(5):474–488
Levine J, Ducatelle F (2004) Ant colony optimisation and local search for bin packing and cutting stock problems. Journal of the Operational Research Society 55(7):705–716
Lourenço HR, Martin O, Stützle T (2002) Iterated local search. In: Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol 57, Kluwer Academic Publishers, Norwell, MA, pp 321–353
Merkle D, Middendorf M (2002) Modelling the dynamics of Ant Colony Optimization algorithms. Evolutionary Computation 10(3):235–262
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: Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC '06), Springer, LNCS, vol 4288, pp 618–627
Neumann F, Witt C (2008) Ant Colony Optimization and the minimum spanning tree problem. In: Proceedings of Learning and Intelligent Optimization (LION '07), Springer, LNCS, vol 5313, pp 153–166
Neumann F, Sudholt D, Witt C (2008) Rigorous analyses for the combination of ant colony optimization and local search. In: Proceedings of the Sixth International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS '08), Springer, LNCS, vol 5217, pp 132–143
Neumann F, Sudholt D, Witt C (2009) Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intelligence 3:35–68
Stützle T, Hoos HH (2000) MAX-MIN ant system. Journal of Future Generation Computer Systems 16:889–914
Sudholt D (2008) Memetic algorithms with variable-depth search to overcome local optima. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '08), ACM Press, pp 787–794
Sudholt D (2009) The impact of parametrization in memetic evolutionary algorithms. Theoretical Computer Science (to appear)
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: Proceedings of the 22nd Symposium on Theoretical

Aspects of Computer Science (STACS '05), Springer, LNCS, vol 3404, pp 44–56