Simple Max-Min Ant Systems and the Optimization of Linear Pseudo-Boolean Functions

Timo Kötzing
Max-Planck-Institut für
Informatik
66123 Saarbrücken, Germany

Frank Neumann
School of Computer Science
University of Adelaide
Adelaide, SA 5005, Australia

Dirk Sudholt
CERCIA, University of
Birmingham
Birmingham, B15 2TT, UK

Markus Wagner
School of Computer Science
University of Adelaide
Adelaide, SA 5005, Australia

ABSTRACT

With this paper, we contribute to the understanding of ant colony optimization (ACO) algorithms by formally analyzing their runtime behavior. We study simple MAX-MIN ant systems on the class of linear pseudo-Boolean functions defined on binary strings of length n. Our investigations point out how the progress according to function values is stored in the pheromones. We provide a general upper bound of $O((n^3 \log n)/\rho)$ on the running time for two ACO variants on all linear functions, where ρ determines the pheromone update strength. Furthermore, we show improved bounds for two well-known linear pseudo-Boolean functions called ONEMAX and BINVAL and give additional insights using an experimental study.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Performance, Theory

Keywords

Ant colony optimization, MMAS, runtime analysis, pseudo-Boolean optimization, theory

1. INTRODUCTION

Ant colony optimization (ACO) is an important class of stochastic search algorithms that has found many applications in combinatorial optimization as well as for stochastic and dynamic problems [5]. The basic idea behind ACO is

that ants construct new solutions for a given problem by carrying out random walks on a so-called construction graph. These random walks are influenced by the pheromone values that are stored along the edges of the graph. During the optimization process the pheromone values are updated according to good solutions found during the optimization which should lead to better solutions in further steps of the algorithm.

Building up a theoretical foundation of this kind of algorithms is a challenging task as these algorithms heavily rely on random decisions. The construction of new solutions depends on the current pheromone situation in the used system which varies highly during the optimization run. Capturing the theoretical properties of the pheromone constellation is a hard task but very important to gain new theoretical insights into the optimization process of ACO algorithms.

With this paper, we contribute to the theoretical understanding of ACO algorithms. Our goal is to gain new insights into the optimization process of these algorithms by studying them on the class of linear pseudo-Boolean functions. There are investigations of different depths on the behavior of simple evolutionary algorithms for this class of functions. The main result shows that each linear pseudo-Boolean function is optimized in expected time $O(n \log n)$ by the well known (1+1) EA [7, 10, 11]. Furthermore, estimation-of-distribution algorithms have been analyzed and compared to evolutionary algorithms [6, 2, 1].

With respect to ACO algorithms, initial results on simplified versions of the MAX-MIN ant system [18] have been obtained. These studies deal with specific pseudo-Boolean functions defined on binary strings of length n. Such studies are primarily focused on well-known linear example functions called ONEMAX and BINVAL or the function LEADINGONES [16, 9, 4, 14, 15]. Recently, some results on the running time of ACO algorithms on combinatorial optimization problems such as minimum spanning trees [17] or the traveling salesman [13] have been obtained. These analyses assume that the pheromone bounds are attained in each iteration of the algorithms. This is the case if a MAX-MIN ant system uses an aggressive pheromone update which forces the pheromone only to take on the maximum and minimum value. The analyses presented in [17] and [13] do not carry over to less aggressive pheromone updates. In particular,

there are no corresponding polynomial upper bounds if the number of different function values is exponential with respect to the given input size.

We provide new insights into the optimization of MAX-MIN ant systems for smaller pheromone updates on functions that may attain exponentially many functions values. Our study investigates simplified versions of the MAX-MIN ant system called MMAS* and MMAS [14] on linear pseudo-Boolean functions with non-zero weights. For these algorithms, general upper bounds of $O((n + (\log n)/\rho)D)$ and $O(((n^2 \log n)/\rho)D)$ respectively, have been provided for the running time on unimodal functions attaining D different function values [14]. As linear pseudo-Boolean function are unimodal, these bounds carry over to this class of functions. However, they only give weak bounds for linear pseudo-Boolean functions attaining many function values (e. g. for functions where the number of different function values is exponential in n).

We show an upper bound of $O((n^3 \log n)/\rho)$ for MMAS* and MMAS optimizing any linear pseudo-Boolean function. Furthermore, our studies show that the method of fitness-based partitions may also be used according to pheromone values as MAX-MIN ant systems quickly store knowledge about high-fitness solutions in the pheromone values. This is one of the key observations that we use for our more detailed analyses on ONEMAX and BINVAL in which we improve the results presented in [14].

To provide further insights that are not captured by our theoretical analyses, we carry out an experimental study. Our experimental investigations give comparisons to simple evolutionary algorithms, and consider the impact of the chosen weights of the linear functions and pheromone update strength with respect to the optimization time. One key observation of these studies is that ONEMAX is not the simplest linear function for the simple MAX-MIN ant systems under investigation. Additionally, the studies indicate that the runtime grows at most linearly with $1/\rho$ for a fixed value of n.

We proceed as follows. In Section 2, we introduce the simplified MAX-MIN ant systems that will be investigated in the following. In Section 3, we provide general runtime bounds for the class of linear pseudo-Boolean functions. We present specific results for ONEMAX and BINVAL in Section 4. Our experimental study which provides further insights is reported in Section 5. Finally, we discuss our results and finish with some concluding remarks.

2. SIMPLIFIED MAX-MIN ANT SYSTEMS

We first describe the simplified MAX-MIN ant systems that will be investigated in the sequel. The following construction graph is used to construct solutions for pseudo-Boolean optimization, i. e., bit strings of n bits. It is 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 then taken with probability $\tau(e_{i,1})/(\tau(e_{i,0}) + \tau(e_{i,1}))$. In the case of our construction graph, we identify the path taken by the ant with a corresponding binary solution x as described above and denote the path by $P(x)$.

All ACO algorithms considered here 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 traversal probabilities coincide.

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. To prevent pheromones from dropping to arbitrarily small values, we follow the MAX-MIN ant system by Stützle and Hoos [18] 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 flipping a bit and hence potentially reverting a "wrong" decision made for a bit.

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:

$$\tau'(e) = \min \left\{ (1 - \rho) \cdot \tau(e) + \rho, \ 1 - \frac{1}{n} \right\} \quad \text{if } e \in P(x) \text{ and}$$

$$\tau'(e) = \max \left\{ (1 - \rho) \cdot \tau(e), \ \frac{1}{n} \right\} \qquad \text{if } e \notin P(x).$$

The algorithm MMAS now works as follows. It records the best solution found so far, known as best-so-far solution. It repeatedly constructs a new solution. This solution is then compared against the current best-so-far and it replaces the previous best-so-far if the objective value of the new solution is not worse. Finally, the pheromones are updated with respect to the best-so-far solution. A formal description is given in Algorithm 1.

Note that when a worse solution is constructed then the old best-so-far solution is reinforced again. In case no improvement is found for some time, this means that the same solution x^* is reinforced over and over again and the pheromones move towards the respective borders in x^*. Previous studies [9] have shown that after $(\ln n)/\rho$ iterations of reinforcing the same solution all pheromones have reached their respective borders; this time is often called "freezing time" [14] (see also Lemma 1).

Algorithm 1 MMAS

1: Set $\tau_{(u,v)} = 1/2$ for all $(u, v) \in E$.
2: Construct a solution x^*.
3: Update pheromones w. r. t. x^*.
4: **repeat forever**
5:      Construct a solution x.
6:      **if** $f(x) \geq f(x^*)$ **then** $x^* := x$.
7:      Update pheromones w. r. t. x^*.

We also consider a variant of MMAS known as MMAS* [14] (see Algorithm 2). The only difference is that
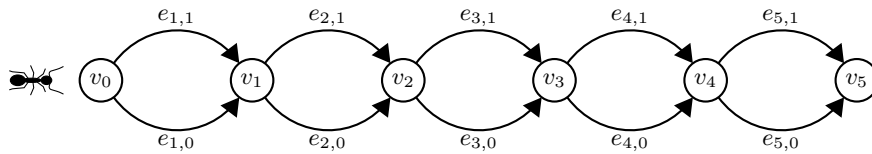
Figure 1: Construction graph for pseudo-Boolean optimization with $n = 5$ bits.

the best-so-far solution is only changed in case the new solution is strictly better. This kind of strategy is often used in applications of ACO. However, in [14] it was argued that MMAS works better on functions with plateaus as MMAS is able to perform a random walk on equally fit solutions.

Algorithm 2 MMAS*

1: Set $\tau_{(u,v)} = 1/2$ for all $(u, v) \in E$.
2: Construct a solution x^*.
3: Update pheromones w.r.t. x^*.
4: **repeat forever**
5:     Construct a solution x.
6:     **if** $f(x) > f(x^*)$ **then** $x^* := x$.
7:     Update pheromones w.r.t. x^*.

In the following we analyze the performance of MMAS and MMAS* on linear functions. We are interested in the number of iterations of the main loop of the algorithms until the first global optimum is found. This time is commonly called the optimization time. In our setting this time equals the number of function evaluations.

Note that the pheromones on the 1-edges $e_{i,1}$ suffice to describe all pheromones as $\tau(e_{i,0}) + \tau(e_{i,1}) = 1$. When speaking of pheromones, we therefore often focus on pheromones on 1-edges.

3. GENERAL RESULTS

We first derive general upper bounds on the expected optimization time of MMAS and MMAS* on linear pseudo-Boolean functions. A linear pseudo-Boolean function for an input vector $x = (x_1, \ldots, x_n)$ is a function $f : \{0,1\}^n \mapsto \mathbb{R}$, with $f(x) = \sum_{i=1}^n w_i x_i$ and *weights* $w_i \in \mathbb{R}$. We only consider positive weights since a function with a negative weight w_i may be transformed into a function with a positive weight $w_i' = -w_i$ by exchanging the meaning of bit values 0 and 1 for bit i. This results in a function whose value is by an additive term of w_i' larger. This and exchanging bit values does not impact the behavior of our algorithms. We also exclude weights of 0 as these bits do not contribute to the fitness. Finally, in this section we assume without loss of generality that the weights are ordered according to their values: $w_1 \geq w_2 \geq \ldots \geq w_n$.

Two well-known linear functions are the function ONE-MAX where $w_1 = w_2 = \cdots = w_n = 1$ and the function BINVAL where $w_i = 2^{n-i}$. These functions represent two extremes: for ONEMAX all bits are of equal importance, while in BINVAL a bit at position i can dominate all bits at positions $i + 1, \ldots, n$.

3.1 Analysis Using Fitness-based Partitions

We exploit a similarity between MMAS, MMAS* and evolutionary algorithms to obtain a first upper bound. We use the method of fitness-based partitions, also called fitness-level method, to estimate the expected optimization time. This method has originally been introduced for the analysis of elitist evolutionary algorithms (see, e.g., Wegener [20]) where the fitness of the current search point can never decrease. The idea is to partition the search space into sets $A_1, \ldots, A_m$ that are ordered with respect to fitness. Formally, we require that for all $1 \leq i \leq m - 1$ all search points in A_i have a strictly lower fitness than all search points in A_{i+1}. In addition, A_m must contain all global optima.

Now, if s_i is (a lower bound on) the probability of discovering a new search point in $A_{i+1} \cup \cdots \cup A_m$, given that the current best solution is in A_i, the expected optimization time is bounded by $\sum_{i=1}^{m-1} 1/s_i$ as $1/s_i$ is (an upper bound on) the expected time until fitness level i is left and each fitness level has to be left at most once.

Gutjahr and Sebastiani [9] as well as Neumann, Sudholt, and Witt [14] have adapted this method for MMAS*. If the algorithm does not find a better search point for some time, the same solution x^* is reinforced over and over again, until eventually all pheromones attain their borders corresponding to the bit values in x^*. This happens after at most $(\ln n)/\rho$ iterations (see [9, 14] or Lemma 1). We say that then all pheromones are saturated. In this setting the solution creation process of MMAS* equals a standard bit mutation of x^* in an evolutionary algorithm. If s_i is (a lower bound on) the probability that the a mutation of x^* creates a search point in $A_{i+1} \cup \cdots \cup A_m$, then the expected time until MMAS* leaves fitness level A_i is bounded by $(\ln n)/\rho + 1/s_i$ as either the algorithm manages to find an improvement before the pheromones saturate or the pheromones saturate and the probability of finding an improvement is at least s_i. This results in an upper bound of $m \cdot (\ln n)/\rho + \sum_{i=1}^{m-1} 1/s_i$ for MMAS*.

One restriction of this method is, however, that fitness levels are only allowed to contain a single fitness value; in the above bound m must equal the number of different fitness values. Without this condition—when a fitness level contains multiple fitness values—MMAS* may repeatedly exchange the current best-so-far solution within a fitness level. This can prevent the pheromones from saturating, so that the above argument breaks down. For this reason, all upper bounds in [14] grow at least linearly in the number of function values.

The following lemma gives an explanation for the time bound $(\ln n)/\rho$ for saturating pheromones (recall that this time is also called *freezing time*). We present a formulation that holds for arbitrary sets of bits. Though we do not make use of the larger generality, this lemma may be of independent interest.

LEMMA 1. *Given an index set $I \subseteq \{1, \ldots, n\}$ we say that a bit is in I if its index is in I. Let x^* be the current best-*

so-far solution of MMAS or MMAS* optimizing an arbitrary function. After $(\ln n)/\rho$ further iterations either all pheromones corresponding to bits in I have reached their respective bounds in $\{1/n, 1-1/n\}$ or x^* has been replaced by some search point x^{**} with $f(x^{**}) \geq f(x^*)$ for MMAS and $f(x^{**}) > f(x^*)$ for MMAS* such that x^{**} differs from x^* in at least one bit in I.

PROOF. Assume the bit values of the bits in I remain fixed in the current best-so-far solution for $(\ln n)/\rho$ iterations as otherwise there is nothing to prove. In this case for every bit x_i^* with $i \in I$ the same bit value x_i^* has been reinforced for $(\ln n)/\rho$ iterations. This implies that for the edge in the construction graph representing the opposite bit value the lower pheromone border $1/n$ has been reached, as for any initial pheromone $0 \leq \tau_i \leq 1$ on this edge we have $(1-\rho)^{(\ln n)/\rho}\tau_i \leq e^{-\ln n}\tau_i \leq 1/n$. $\square$

So far, the best known general upper bounds for MMAS and MMAS* that apply to every linear function are $O((n^2 \log n)/\rho \cdot 2^n)$ and $O((\log n)/\rho \cdot 2^n)$, respectively, following from upper bounds for unimodal functions [14]. The term 2^n results from the fact that in the worst case a linear function has 2^n different function values. This is the case, for instance, for the function BINVAL. An exponential upper bound for linear functions is, of course, unsatisfactory. The following theorem establishes a polynomial upper bound (with respect to n and $1/\rho$) for both algorithms.

THEOREM 2. *The expected optimization time of MMAS and MMAS* on every linear function is in $O((n^3 \log n)/\rho)$.*

PROOF. The proof is an application of the above-described fitness-based partitions method. In the first step, we consider the time needed to sample a solution which is at least on the next higher fitness level. We analyze the two situations when the pheromones are either saturated or not. Our upper bound is the result of the repetition of such advancing steps between fitness levels.

Following Wegener [19], we define the fitness levels $A_0, \ldots, A_n$ with

$$A_i = \left\{ x \in \{0,1\}^n \,\middle|\, \sum_{j=1}^{i} w_j \leq f(x) < \sum_{j=1}^{i+1} w_j \right\}$$

for $0 \leq i \leq n-1$ and $A_n := \{1^n\}$. Recall that $\sum_{j=1}^{i} w_j$ is the sum of the i largest weights. Since the weights are non-decreasing, a solution x is at least on fitness level A_i if the leftmost i bits have value 1.

The expected time spent sampling solutions on fitness level A_i (i.e., without sampling a solution of a higher fitness level) is the sum of the time spent in A_i with saturated pheromone values and the time spent in A_i with unsaturated pheromone values. In the following, we analyze for both situations the probabilities to sample a solution of a higher fitness level. In the end, as MMAS and MMAS* might not remain in one of both situations exclusively, but alternates between situations of saturated and unsaturated pheromone values, we take the sum of both run times as an upper bound.

First, when the pheromone values are saturated, the probability of setting the leftmost zero and keeping all other bits as in the best-so-far solution is $1/n \cdot (1-1/n)^{n-1} \geq 1/n \cdot 1/e$, as $(1 - 1/n)^{n-1} \geq 1/e \geq (1 - 1/n)^n$ holds for all $n \in \mathbb{N}$. This results in a probability of $\Omega(1/n)$ of advancing in such

a situation. Thus, after an expected number of $O(n)$ steps with saturated pheromone values, the algorithm will sample a solution on a higher fitness level.

For the second case, when the pheromone values are not saturated, let $i < n$, and suppose $x^* \in A_i$ is the current best solution. Our argument is intuitively as follows. If we don't find solutions with better fitness while unsaturated, then we will be saturated after at most $(\ln n)/\rho$ steps. Otherwise, we either make a *good* or a *bad* improvement: a good improvement gets the EA to a higher fitness level; a bad improvement is an improvement which does not go to a higher fitness level. This latter case is bad, since it prevents the pheromones from getting saturated, and we cannot quantify progress in this case. We will show that good improvements are at least $1/n^2$ times as likely as bad ones, which gives a limit on the number of bad improvements one can expect.

Then, let us denote by $G = \bigcup_{j>i} A_j$ all *good* solutions that are at least on the next higher fitness level, and by $B = \{x \in \{0,1\}^n \,|\, f(x) \geq f(x^*), x \notin G\}$ all *bad* solutions that are in A_i with an equal or a higher function value than x^*. Thus, every improving sampled solution belongs to $G \cup B$.

Let $h : \{0,1\}^n \mapsto \{0,1\}^n$ be the function that returns for a given solution $x \in G \cup B$ a solution $x' \in G$, where the leftmost 0 (if any) was flipped to 1.

Let $P(x)$ be the probability of sampling a new solution x. Then the probability $q(x)$ of sampling any of x and $h(x)$ is greater than or equal to $P(x)$. The probability of sampling $h(x)$ is the probability of sampling any of x and $h(x)$, times the probability that the leftmost zero of x was sampled as a one. Thus, for all x, $P(h(x)) = q(x) \cdot 1/n \geq P(x)/n$, as the pheromone values are at least $1/n$.

Furthermore, each solution $h(x)$ has at most n preimages with respect to h. Note that, for all $x \in B$, $h(x) \in G$.

Thus, the probability of sampling the next solution $x \in G$ is

$$P(x \in G) = \sum_{x \in G} P(x) \geq \frac{\sum_{x \in B} P(h(x))}{n}$$
$$\geq \frac{\sum_{x \in B} P(x)}{n^2} = \frac{P(x \in B)}{n^2}.$$

So, sampling a good solution is at least $1/n^2$ times as likely as sampling a bad one.

Furthermore, while no bad solutions are sampled, at most $(\ln n)/\rho$ steps are spent before the pheromone values are saturated (based on Lemma 1).

Thus, up to $(\ln n)/\rho$ steps are spent with unsaturated pheromone values before sampling a new solution, and $O((n^2 \log n)/\rho)$ steps are spent in total sampling solutions in B before sampling a solution in G.

Consequently, the time spent on one fitness level is the sum of the times spent in either situation of the pheromone values, that is, $O(n) + O((n^2 \log n)/\rho) = O((n^2 \log n)/\rho)$.

Finally, as there are n fitness levels, the afore-described steps have to be performed at most n-times, which yields a total runtime of $O((n^3 \log n)/\rho)$. $\square$

3.2 Fitness-based Partitions for Pheromones

We describe an approach for extending the argument on f-based partitions to pheromones instead of the best-so-far solution. This alternate approach is based on *weighted pheromone sums*. Given a vector of pheromones τ and a linear function f, the *weighted pheromone sum (wps)* is $f(\tau)$.

The idea is that, during a run of the algorithm, the wps should rise until *value-saturated* with respect to the current best search point, and then a significant improvement should have a decent probability.

Define a function v on bit strings as follows.

$$v(x) = \sum_{i=1}^{n} \begin{cases} (1 - \frac{1}{n})w_i, & \text{if } x_i = 1; \\ \frac{1}{n}w_i, & \text{otherwise.} \end{cases}$$

A pheromone vector τ is called *value-saturated* with respect to a search point x^* iff

$$f(\tau) \geq v(x^*).$$

Note that this definition of value-saturation is very much different from previous notions of saturation.

Let, for all i, a_i be the bit string starting with i ones and then having only zeros.

We let

$$A_i = \{x \mid f(a_i) \leq f(x) < f(a_{i+1})\}$$

and

$$B_i = \{\tau \mid v(a_i) \leq f(\tau) < v(a_{i+1})\}.$$

While $(A_i)_i$ captures the progress of the search points towards the optimum, $(B_i)_i$ captures the progress of the pheromones.

LEMMA 3. *For all i, if the best-so-far solution was in $\bigcup_{j \geq i} A_j$ for at least $(\ln n)/\rho$ iterations, then $\tau \in \bigcup_{j \geq i} B_j$.*

PROOF. Let h be such that $\forall s : h(s) = \min(s(1 - \rho) + \rho, 1 - 1/n)$. Let τ^0 be the vector of pheromones when the algorithm samples a solution in $\bigcup_{j \geq i} A_j$ for the first time, and let $(\tau^t)_t$ be the vectors of pheromones in the successive rounds. For all t we let h^t be the t-times composition of h with itself. Further, we define the sequence of *capped-out pheromones* $(\tau^{\text{cap},t})_t$ such that, for all t, $\tau_j^{\text{cap},t} = \min(h^t(1/n), \tau_j^t)$. For this capped-out version of pheromones we have, for all t,

$$f(\tau^{\text{cap},t+1}) \geq \min((1-\rho)f(\tau^{\text{cap},t}) + \rho f(a_i), (1 - 1/n)f(a_i)),$$

as pheromones will evaporate and at least an $f(a_i)$ weighted part of them will receive new pheromone ρ (note that the old capped-out pheromone raised by h cannot exceed the new cap). Thus, we get inductively

$$\forall t : f(\tau^{\text{cap},t}) \geq h^t(1/n)f(a_i).$$

As we know from Lemma 1, for $t \geq (\ln n)/\rho$ we have $h^t(1/n) \geq 1 - 1/n$, and, thus, $\tau^t \in \bigcup_{j \geq i} B_j$. $\square$

This argument opens up new possibilities for analyses and we believe it to be of independent interest.

If it is possible to show, for all i, if $\tau \in B_i$, then the probability of sampling a new solution in $\bigcup_{j > i} A_j$ is $\Omega(1/n)$, then Lemma 3 would immediately improve the bound in Theorem 2 to $O(n^2 + (n \log n)/\rho)$. However, this claim is not easy to show.

It is possible to prove this for the special case of ONE-MAX using the following theorem by Gleser [8]. This theorem gives a very nice handle on estimating probabilities for sampling above-average solutions for ONEMAX.

THEOREM 4 (GLESER [8]). *Let τ, τ' be two pheromone vectors such that, for all $j \leq n$, the sum of the j least values*

of τ is at least the sum of the j least values of τ'. Let λ be the sum of the elements of τ'. Then it is at least as likely to sample $\lfloor \lambda + 1 \rfloor$ ones with τ as it is with τ'.

We can use this theorem to get a good bound for ONE-MAX: the worst case for the probability of an improvement is attained when all but at most one pheromones are at their respective borders. In this situation, when there are still i ones missing, the probability of an improvement is at least

$$i \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{i}{en}.$$

Combining this with Lemma 3, we get the following result.

COROLLARY 5. *The expected optimization time of MMAS and MMAS* on ONEMAX is bounded by*

$$\sum_{i=1}^{n} \frac{en}{i} + n \cdot (\ln n)/\rho = O((n \log n)/\rho).$$

This re-proves the $O((n \log n)/\rho)$-bound for MMAS* in [14] and it improves the current best known bound for MMAS on ONEMAX by a factor of n^2. We will present an even improved bound for ONEMAX in Section 4.1.

However, with regard to general linear functions it is not clear how this argument can be generalized to arbitrary weights. In the following section we therefore turn to the investigation of concrete linear functions.

4. IMPROVED BOUNDS FOR SELECTED LINEAR FUNCTIONS

Using insights from Section 3.2 we next present improved upper bounds for the function ONEMAX (Section 4.1). Afterwards, we focus on the special function BINVAL (Section 4.2). These resulting bounds are much stronger than the general ones given in Section 3 above.

4.1 OneMax

Recall the bound $O((n \log n)/\rho)$ for MMAS and MMAS* from Corollary 5. In the following we prove a bound of $O(n \log n + n/\rho)$ for both MMAS and MMAS* by more detailed investigations on the pheromones and their dynamic growth over time. This shows in particular that the term $1/\rho$ has at most a linear impact on the total expected optimization time.

Let v and a_i be as in Section 3.2. For all $i \leq n$, we let $\alpha_i = i(1 - 1/n)$. Observe that $v(a_i) = \alpha_i + (n - i)/n$ and, in particular, $\alpha_i \leq v(a_i) \leq \alpha_i + 1$.

The following lemma gives a lower bound on the pheromone $f(\tau^+)$ after on iteration. Note that for ONE-MAX $f(\tau^+)$ corresponds to the sum of pheromones.

LEMMA 6. *Let $i < j$ and let τ be the current pheromones with $v(a_i) \leq f(\tau) < v(a_{i+1})$ and suppose that the best-so-far solution has at least j ones. We denote by τ^+ the pheromones after one iteration of MMAS or MMAS*. Then we have $f(\tau^+) \geq v(a_{i+1})$ or*

$$f(\tau^+) - \alpha_i \geq (f(\tau) - \alpha_i)(1 - \rho) + (j - i)\rho \geq v(a_i). \quad (1)$$

PROOF. Suppose that in rewarding bit positions from τ to get τ^+, exactly k positions cap out at the upper pheromone border $1 - 1/n$. From $f(\tau^+) < v(a_{i+1})$ (otherwise there is nothing left to show), we have $k \leq i$. We decompose $f(\tau^+)$

into the contribution of the capped-out bits (which is α_k, being k of the j rewarded positions) and the rest. Then we have

$$f(\tau^+) \geq \alpha_k + (f(\tau) - \alpha_k)(1 - \rho) + (j - k)\rho.$$

We now get

$$\alpha_i + (f(\tau) - \alpha_i)(1 - \rho) + (j - i)\rho$$
$$= \alpha_k + \alpha_{i-k} + (f(\tau) - \alpha_k - \alpha_{i-k})(1 - \rho) + (j - k)\rho + (k - i)\rho$$
$$\leq f(\tau^+) + \alpha_{i-k} - \alpha_{i-k}(1 - \rho) + (k - i)\rho$$
$$= f(\tau^+) + \rho(\alpha_{i-k} - (i - k))$$
$$\leq f(\tau^+).$$

From $v(a_i) \leq f(\tau)$ we get

$$f(\tau^+) \geq \alpha_i + (f(\tau) - \alpha_i)(1 - \rho) + (j - i)\rho$$
$$\geq \alpha_i + (v(a_i) - \alpha_i)(1 - \rho) + (j - i)\rho$$
$$= v(a_i) + \rho(j - i - v(a_i) + \alpha_i).$$

From $v(a_i) - \alpha_i < 1$ and $j > i$ we get the desired conclusion. $\square$

One important conclusion from Lemma 6 is that once the sum of pheromones is above some value $v(a_i)$, it can never decrease below this term.

Now we extend Lemma 6 towards multiple iterations. The following lemma shows that, unless a value of $v(a_{i+1})$ is reached, the sum of pheromones quickly converges to α_j when j is the number of ones in the best-so-far solution.

LEMMA 7. *Let $i < j$ and let τ be the current pheromones with $v(a_i) \leq f(\tau) < v(a_{i+1})$ and suppose that the best-so-far solution has at least j ones. For all t, we denote by τ^t the pheromones after t iterations of MMAS or MMAS*. Then we have for all t $f(\tau^t) \geq v(a_{i+1})$ or*

$$f(\tau^t) - \alpha_i \geq (j - i)(1 - (1 - \rho)^t).$$

PROOF. Inductively for all t, we get from Lemma 6 $f(\tau^t) \geq v(a_{i+1})$ or

$$f(\tau^t) - \alpha_i \geq (f(\tau^0) - \alpha_i)(1 - \rho)^t + (j - i)\rho \sum_{i=0}^{t-1}(1 - \rho)^i$$
$$= (f(\tau^0) - \alpha_i)(1 - \rho)^t + (j - i)\rho\frac{1 - (1 - \rho)^t}{1 - (1 - \rho)}$$
$$= (f(\tau^0) - \alpha_i)(1 - \rho)^t + (j - i)(1 - (1 - \rho)^t)$$
$$\geq (j - i)(1 - (1 - \rho)^t).$$

$\square$

THEOREM 8. *The expected optimization time of MMAS and MMAS* on ONEMAX is $O(n \log n + n/\rho)$.*

PROOF. Define $v(a_{-1}) = 0$. Let τ be the current pheromones and τ^t be the pheromones after t iterations of MMAS or MMAS*. We divide a run of MMAS or MMAS* into phases: the algorithm is in Phase j if $f(\tau) \geq f(a_{j-1})$, the current best-so-far solution contains at least j ones, and the conditions for Phase $j + 1$ are not yet fulfilled. We estimate the expected time until each phase is completed, resulting in an upper bound on the expected optimization time.

We first deal with the last $n/2$ phases and consider some Phase j with $j \geq n/2$. By Lemma 7 after t iterations we

either have $f(\tau^t) \geq v(a_{j+1})$ or $f(\tau^t) \geq \alpha_{j-1} + 1 - (1 - \rho)^t$. Setting $t := \lceil 1/\rho \rceil$, this implies $f(\tau^t) \geq \alpha_{j-1} + 1 - e^{-\rho t} \geq \alpha_{j-1} + 1 - 1/e$. We claim that then the probability of creating $j + 1$ ones is $\Omega((n - j)/n)$.

Using $j \geq n/2$, the total pheromone $\alpha_{j-1} + 1 - 1/e$ can be distributed on an artificially constructed pheromone vector τ' as follows. We assign value $1 - 1/n$ to $j - 1$ entries and value $1/n$ to $n - j$ entries. As $(n-j)/n \leq 1/2$, we have used pheromone of $\alpha_{j-1} + 1/2$ and so pheromone $1 - 1/2 - 1/e = \Omega(1)$ remains for the last entry. We now use Theorem 4 to see that it as least as likely to sample a solution with $j + 1$ ones with the real pheromone vector τ^t as it is with τ'. By construction of τ' this probability is at least $(1 - 1/n)^{j-1} \cdot (1/2 - 1/e) \cdot (n-j)/n \cdot (1 - 1/n)^{n-j-1} \geq (n-j)(1/2-1/e)/(en)$ as a sufficient condition is setting all bits with pheromone larger than $1/n$ in τ' to 1 and adding exactly one 1-bit out of the remaining $n - j$ bits.

Invoking Lemma 7 again for at least $j + 1$ ones in the best-so-far solution, we get $f(\tau^{t+t'}) \geq \alpha_{j-1} + 2(1 - (1 - \rho)^{t'})$, which for $t' := \lceil 2/\rho \rceil$ yields $f(\tau^{t+t'}) \geq \alpha_{j-1} + 3/2 \geq \alpha_j + 1/2 \geq v(a_j)$ as $j \geq n/2$.

For the phases with index $j < n/2$ we construct a pessimistic pheromone vector τ' in a similar fashion. We assign value $1 - 1/n$ to $j - 2$ entries, value $1/n$ to $n - j$ entries, and put the remaining pheromone on the two last bits such that either only one bit receives pheromone above $1/n$ or one bit receives pheromone $1 - 1/n$ and the other bit gets the rest. To show that the pheromones raise appropriately, we aim at a larger gain in the best number of ones. The probability of constructing at least $j + 2$ ones with any of the above-described vectors is at least $(1 - 1/n)^{j-2} \cdot \binom{n-j+2}{3} \cdot 1/n^3 \cdot (1 - 1/n)^{n-j} \geq 1/(48e) = \Omega((n - j)/n)$.

Using the same choice $t' := \lceil 2/\rho \rceil$ as above, Lemma 7 yields $f(\tau^{t+t'}) \geq \alpha_{j-1} + 3(1 - (1 - \rho)^{t'}) \geq \alpha_{j-1} + 2 \geq v(a_j)$.

Summing up the expected times for all phases yields a bound of

$$O\left(\sum_{i=0}^{n-1}\frac{n}{n - i} + n(t + t')\right) = O(n \log n + n/\rho).$$

$\square$

4.2 BinVal

The function BINVAL has similar properties as the well-known function LEADINGONES$(x) := \sum_{i=1}^{n}\prod_{j=1}^{i}x_j$ that counts the number of leading ones. For both MMAS and MMAS* the leading ones in x^* can never be lost as setting one of these bits to 0 will definitely result in a worse solution. This implies for both algorithms that the pheromones on the first LEADINGONES(x^*) bits will strictly increase over time, until the upper pheromone border is reached.

In [14] the following upper bound for LEADINGONES was shown.

THEOREM 9 ([14]). *The expected optimization time of MMAS and MMAS* on LEADINGONES is bounded by $O(n^2 + n/\rho)$ and $O\left(n^2 \cdot (1/\rho)^{\varepsilon} + \frac{n/\rho}{\log(1/\rho)}\right)$ for every constant $\varepsilon > 0$.*

The basic proof idea is that after an average waiting time of ℓ iterations the probability of rediscovering the leading ones in x^* is at least $\Omega(e^{-5/(\ell\rho)})$. Plugging in appropriate values for ℓ then gives the claimed bounds.

DEFINITION 10. *For $\ell \in \mathbb{N}$ and a sequence of bits $x_1, \ldots, x_i$ ordered with respect to increasing pheromones we say that these bits form an (i, ℓ)-layer if for all $1 \leq j \leq i$*

$$\tau_j \geq \min(1 - 1/n, 1 - (1 - \rho)^{j\ell}).$$

With such a layering the considered bits can be rediscovered easily, depending on the value of ℓ. The following lemma was implicitly shown in [14, proof of Theorem 6].

LEMMA 11. *The probability that in an (i, ℓ)-layer all i bits defining the layer are set to 1 in an ant solution is $\Omega(e^{-5/(\ell\rho)})$.*

Assume we have $k = \text{LEADINGONES}(x^*)$ and the pheromones form a (k, ℓ)-layer. Using Lemma 11 and the fact that a new leading one is added with probability at least $1/n$, the expected waiting time until we have $\text{LEADINGONES}(x^*) \geq k + 1$ and a $(k + 1, \ell)$-layer of pheromones is at most $O(n \cdot e^{5/(\ell\rho)} + \ell)$. As this is necessary at most n times, this gives us an upper bound on the expected optimization time. Plugging in $\ell = \lceil 5/\rho \rceil$ and $\ell = \lceil 5/(\varepsilon\rho \ln(1/\rho)) \rceil$ yields the same upper bounds for BIN-VAL as we had in Theorem 9 for LEADINGONES.

THEOREM 12. *The expected optimization time of MMAS and MMAS* on BINVAL is bounded by $O(n^2 + n/\rho)$ and $O\left(n^2 \cdot (1/\rho)^{\varepsilon} + \frac{n/\rho}{\log(1/\rho)}\right)$ for every constant $\varepsilon > 0$.*

These two bounds show that the second term "$+n/\rho$" in the first bound—that also appeared in the upper bound for ONEMAX—can be lowered, at the expense of an increase in the first term. It is an interesting open question whether for all linear functions when ρ is very small the runtime is $o(n/\rho)$, i.e., sublinear in $1/\rho$ for fixed n. The relation between the runtime and ρ is further discussed from an experimental perspective in the next section.

5. EXPERIMENTS

In this section, we investigate the behavior of our algorithms using experimental studies. Our goal is to examine the effect of the pheromone update strength as well as the impact of the weights of the linear function that should be optimized.

Our experiments are related to those in [3, 14]. The authors of the first article concentrate their analyses of 1-ANT and MMAS on ONEMAX, LEADINGONES, and random linear functions. A single problem size n for each function is used, and a number of theory-guided indicators monitors the algorithms' progress at every time step, in order to measure the algorithms' progress within individual runs. In the second article, the runtime of MMAS and MMAS* is investigated on ONEMAX and other functions, for two values of n and a wide range of values of ρ.

First, we investigate the runtime behavior of our algorithms for different settings of ρ (see Figure 2). We investigate ONEMAX, BINVAL, and random functions where the weights are chosen uniformly at random from the interval $]0, 1]$. For our investigations, we consider problems of size $n = 100, 150, \ldots, 1000$. Each fixed value of n is run for $\rho \in \{1.0, 0.5, 0.1, 0.05\}$ and the results are averaged over 1000 runs. Remember that, for $\rho = 1.0$, MMAS* is equivalent to the $(1+1)$ EA*, and MMAS is equivalent to the $(1+1)$ EA (see [12] for the definition of the evolutionary algorithms). As a result of the 1000 repetitions, the curves in

the plots of Figure 2 are fairly smooth, as the standard error of the mean of the average of 1000 identically distributed random variables is only $1/\sqrt{1000} \approx 3.2\%$ of the standard error of the mean of any single such variable.

One general observation is that the performance of MMAS* on random linear functions and BINVAL is practically identical to that of MMAS. This was expected, as several bit positions have to have the same associated weights in order for MMAS to benefit from its weakened acceptance condition. Furthermore, we notice that ONEMAX is not the simplest linear function for MMAS* to optimize. In fact, for certain values of ρ, ONEMAX is as difficult to optimize as BINVAL.

For all experiments, the performance of MMAS* with $\rho = 1.0$ is very close to that of MMAS with $\rho = 1.0$. However, with different values of ρ, several performance differences are observed. For example, MMAS* with $\rho = 0.5$ and $\rho = 0.1$ optimizes random linear functions faster than MMAS* with $\rho = 1.0$, which is on the other hand the fastest setup for ONEMAX. Furthermore, the performance of MMAS increased significantly with values of $\rho < 1.0$, e.g., MMAS with $\rho = 0.1$ is 30% faster than MMAS with $\rho = 1.0$. Another general observation is that MMAS* performs better on random linear functions, than on ONEMAX, e.g. for $n = 1000$ and $\rho = 0.1$ the runtime decreases by roughly 10%.

In the following, we give an explanation for this behavior. During the optimization process, it is possible to replace a lightweight 1 at bit i (i.e., a bit with a relatively small associated weight w_i) with a heavyweight 0 at bit j (i.e., a bit with a relatively large associated weight w_j). Afterwards, during the freezing process, the probability for sampling again the lightweight 1 at bit i (whose associated τ_i is in the process of being reduced to $\tau_{\min}$) is relatively high. Unlike in the case of ONEMAX, it is indeed possible for MMAS* to collect heavyweight 1-bits in between, and the "knowledge" of the lightweight 1-bits is available for a certain time, stored as a linear combination of the pheromones' values. This effect occurs in the phase of adjusting the pheromones, not when the pheromone values are saturated. Otherwise, the effect could be observed for the $(1+1)$ EA and the $(1+1)$ EA* as well.

We have already seen that the choice of ρ may have a high impact on the optimization time. The runtime bounds given in this paper increase with decreasing ρ. In the following, we want to investigate the impact of ρ closer by conducting experimental studies. We study the effect of ρ by fixing $n = 100$ and varying $\rho = 1/x$ with $x = 1, 11, 21, \ldots, 1001$. Note that these ρ-values are much smaller than the ones shown in Figure 2. The results are shown in Figure 3 and are averaged over 10.000 runs. As a result of the large number of repetitions, the standard error of the mean of the average is only $1/\sqrt{10000} = 1\%$ of the standard error of the mean of any run. The effect that small values of ρ can improve the performance on ONEMAX is observable again. The fitted linear regression lines, which are based on the mean iterations for $1/\rho \in (500, 1000]$, support our claim that the runtime grows at most linear with $1/\rho$ for a fixed value of n. In fact, the fitted lines indicate that the growth of the average runtime is very close to a linear function in $1/\rho$. The real curves appear to be slightly concave, which corresponds to a sublinear growth. However, the observable effects are too small to allow for general conclusions.
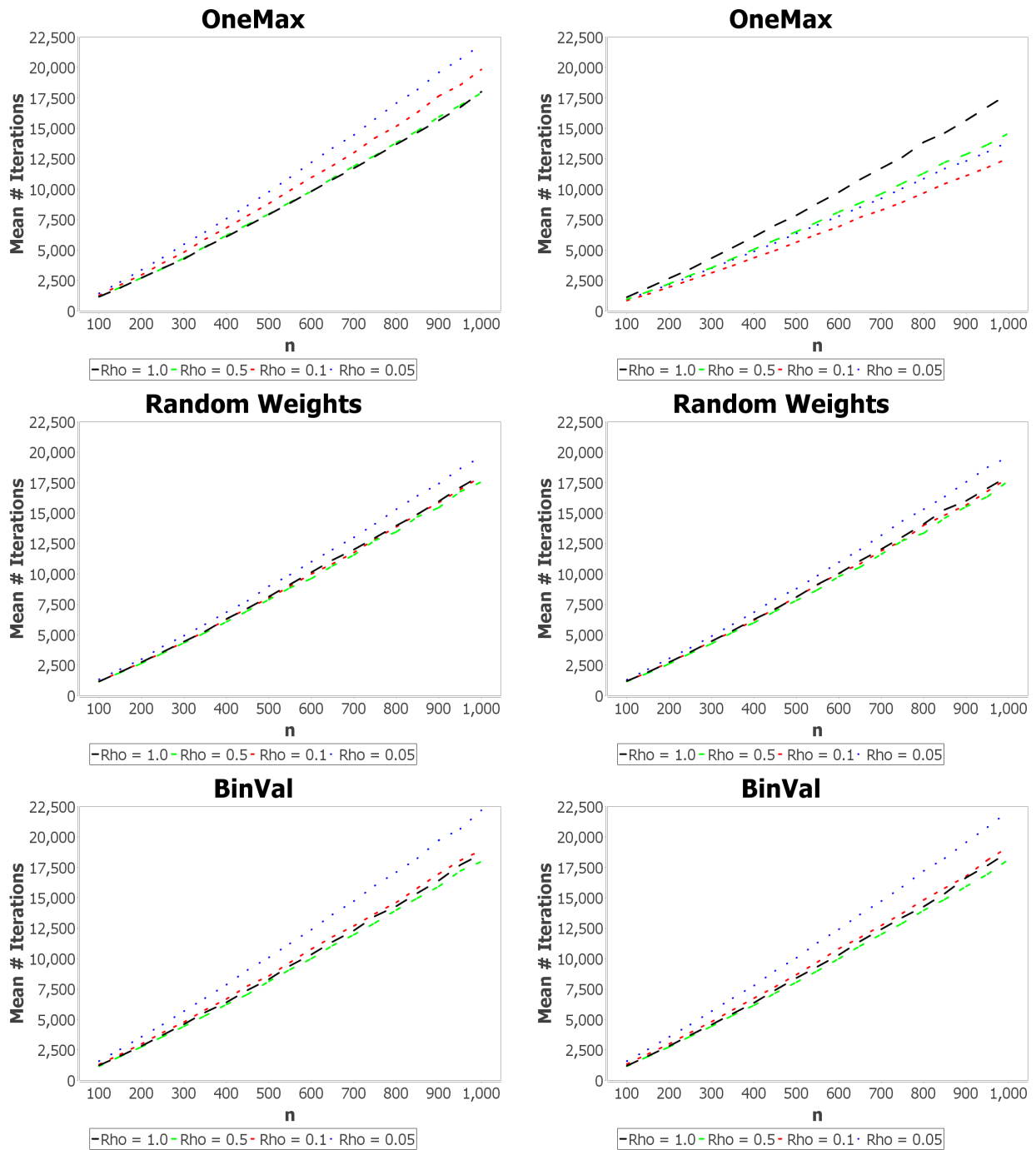
Figure 2: Runtime of MMAS* (left column) and MMAS (right column).

6. CONCLUSIONS AND FUTURE WORK

The rigorous analysis of ACO algorithms is a challenging task as these algorithms are of a high stochastic nature. Understanding the pheromone update process and the information that is stored in pheromone during the optimization run plays a key role in strengthening their theoretical foundations.

We have presented improved upper bounds for the performance of ACO on the class of linear pseudo-Boolean func-

tions. The general upper bound of $O((n^3 \log n)/\rho)$ from Theorem 2 applies to all linear functions, but in the light of the smaller upper bounds for ONEMAX and BINVAL we believe that this bound is still far from optimal. Stronger arguments are needed in order to arrive at a stronger result.

We also have developed novel methods for analyzing ACO algorithms without relying on pheromones freezing at pheromone borders. Fitness-level arguments on a pheromone level have revealed one possible way of reason-
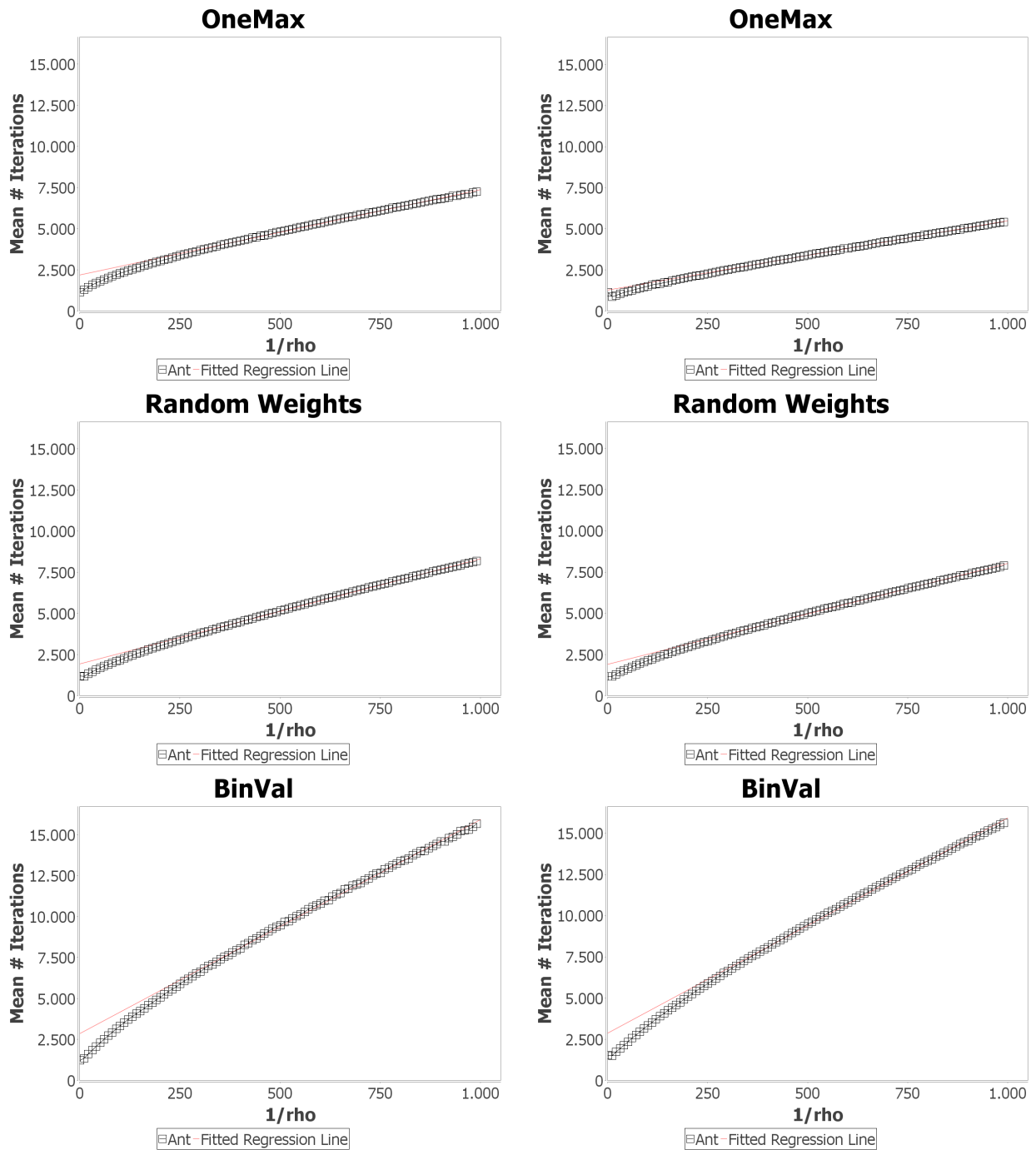
Figure 3: Impact of pheromone evaporation factor in MMAS* (left column) and MMAS (right column).

ing. For ONEMAX this approach, in combination with results from [8], has led to a bound of $O(n \log n + n/\rho)$, both for MMAS and MMAS*. This is a major improvement to the previous best known bounds $O((n^3 \log n)/\rho)$ for MMAS and $O((n \log n)/\rho)$ for MMAS* and it finally closes the gap of size n^2 between the upper bounds for MMAS and MMAS*. We conjecture that our improved bound holds for all linear functions, but this is still a challenging open problem.

Furthermore, our proof arguments have given more detailed insight into the precise dynamic growth of pheromones. For a fixed sum of pheromones, we have learned which distributions of pheromones among n bits are best and worst possible for increasing the best-so-far ONE-MAX value. We believe that this will prove useful for further theoretical studies of ACO.

The experimental results have revealed that a slow adaption of pheromone is beneficial for MMAS on ONEMAX as MMAS was faster than the (1+1) EA for all investigated ρ-

values not larger than 0.5. We also argued why MMAS* is faster on random-weight linear functions than on ONEMAX. The experiments also gave more detailed insights into the impact of the evaporation factor on the average runtime.

We conclude with the following two open questions and tasks for future work:

1. Do MMAS and MMAS* optimize all linear pseudo-Boolean functions in expected time $O(n \log n + n/\rho)$?

2. Analyze ACO for combinatorial problems like minimum spanning trees and the TSP in settings with slow pheromone adaptation.

Acknowledgments

7. REFERENCES

[1] T. Chen, P. K. Lehre, K. Tang, and X. Yao. When is an estimation of distribution algorithm better than an evolutionary algorithm? In *Proc. of CEC 2009*, pages 1470–1477. IEEE Press, 2009.

[2] T. Chen, K. Tang, G. Chen, and X. Yao. Analysis of computational time of simple estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):1 –22, 2010.

[3] B. Doerr, D. Johannsen, and C. H. Tang. How single ant ACO systems optimize pseudo-Boolean functions. In *Proc. of PPSN 2008*, pages 378–388. Springer, 2008.

[4] B. Doerr, F. Neumann, D. Sudholt, and C. Witt. On the runtime analysis of the 1-ANT ACO algorithm. In *Proc. of GECCO 2007*, pages 33–40. ACM, 2007.

[5] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[6] S. Droste. A rigorous analysis of the compact genetic algorithm for linear functions. *Natural Computing*, 5(3):257–283, 2006.

[7] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[8] L. Gleser. On the distribution of the number of successes in independent trials. *Annals of Probability*, 3(1):182–188, 1975.

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

[10] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

[11] J. Jägersküpper. A blend of Markov-chain and drift analysis. In *Proc. of PPSN 2008*, pages 41–51. Springer, 2008.

[12] T. Jansen and I. Wegener. 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, 2001.

[13] T. Kötzing, F. Neumann, H. Röglin, and C. Witt. Theoretical properties of two ACO approaches for the traveling salesman problem. In *Proc. of ANTS 2010*, pages 324–335. Springer, 2010.

[14] F. Neumann, D. Sudholt, and C. Witt. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence*, 3(1):35–68, 2009.

[15] F. Neumann, D. Sudholt, and C. Witt. A few ants are enough: ACO with iteration-best update. In *Proc. of GECCO 2010*, pages 63–70. ACM, 2010.

[16] F. Neumann and C. Witt. Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica*, 54(2):243–255, 2009.

[17] F. Neumann and C. Witt. Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science*, 411(25):2406–2413, 2010.

[18] T. Stützle and H. H. Hoos. MAX-MIN ant system. *Journal of Future Generation Computer Systems*, 16:889–914, 2000.

[19] I. Wegener. Theoretical aspects of evolutionary algorithms. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP 2001)*, pages 64–78. Springer, 2001.

[20] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In R. Sarker, X. Yao, and M. Mohammadian, editors, *Evolutionary Optimization*, pages 349–369. Kluwer, 2002.