Theoretical Computer Science 412 (2011) 1629–1644

Contents lists available at ScienceDirect

Theoretical Computer Science

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

Runtime analysis of the 1-ANT ant colony optimizer

Benjamin Doerr [a], Frank Neumann [b], Dirk Sudholt [c,*,1], Carsten Witt [d,1]

[a] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[b] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
[c] CERCIA, University of Birmingham, Birmingham, United Kingdom
[d] DTU Informatics, Technical University of Denmark, Kgs. Lyngby, Denmark

ARTICLE INFO

Article history:
Received 20 March 2009
Received in revised form 29 July 2010
Accepted 8 December 2010
Communicated by D. Corne

Keywords:
Ant colony optimization
Runtime analysis
Theory

ABSTRACT

The runtime analysis of randomized search heuristics is a growing field where, in the last two decades, many rigorous results have been obtained. First runtime analyses of ant colony optimization (ACO) have been conducted only recently. In these studies simple ACO algorithms such as the 1-ANT are investigated. The influence of the evaporation factor in the pheromone update mechanism and the robustness of this parameter w.r.t. the runtime behavior have been determined for the example function OneMax.

This work puts forward the rigorous runtime analysis of the 1-ANT on the example functions LeadingOnes and BinVal. With respect to Evolutionary Algorithms (EAs), such analyses were essential to develop methods for the analysis on more complicated problems. The proof techniques required for the 1-ANT, unfortunately, differ significantly from those for EAs, which means that a new reservoir of methods has to be built up. Again, the influence of the evaporation factor is analyzed rigorously, and it is proved that its choice has a crucial impact on the runtime. Moreover, the analyses provide insight into the working principles of ACO algorithms. Our theoretical results are accompanied by experimental results that give us a more detailed impression of the 1-ANT's performance. Furthermore, the experiments also deal with the question whether using many ant solutions in one iteration can decrease the total runtime.

1. Introduction

The analysis of randomized search heuristics is a growing research area where many results have been obtained in recent years. This class of heuristics contains not only well-known approaches such as Randomized Local Search, Simulated Annealing, and Evolutionary Algorithms (EAs) but also less-known and more modern instances such as Estimation-of-Distribution Algorithms, Particle Swarm Optimization, and ant colony optimization (ACO). Such heuristics are often applied to problems whose structure is not known or if there are not enough resources such as time, money, or knowledge to obtain good specific algorithms. It is widely acknowledged that a solid theoretical foundation for such heuristics is needed. An obvious and accepted theoretical approach stemming from theoretical computer science is to analyze the (expected) runtime of randomized search heuristics by adapting the probabilistic methods available for the analysis of randomized algorithms (e. g., [1]).

The first steps to a runtime analysis of randomized search heuristics were made for a very simple EA called $(1+1)$ EA. Initially, the $(1+1)$ EA was investigated for the optimization of example functions such as OneMax, LeadingOnes, BinVal,

* Corresponding author. Tel.: +44 121 414 3734.
E-mail address: d.sudholt@cs.bham.ac.uk (D. Sudholt).

[1] The work was done while the author was at the Technische Universität Dortmund.

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.030

trap functions etc. [2], all of which indubitably may be regarded as toy problems. Due to the simple structure of these problems, however, it was possible to develop methods for the analysis of EAs. This approach can be considered very successful since nowadays runtime analyses of EAs can be carried out w. r. t. well-known combinatorial optimization problems such as maximum matchings [3], minimum spanning trees [4], partition problems [5], Eulerian cycle problems [6], and graph coloring problems [7].

The theoretical runtime analysis for the modern and very popular randomized search heuristic ACO (e. g., [8]), however, lags far behind the results for the classical EAs. Until 2006, only convergence results, (e. g., [9]), and results on the dynamics of models of ACO (e. g., [10]) were known. In a survey on theoretical studies of ACO by Dorigo and Blum [11], researchers were encouraged to follow the approach taken for the analysis of EAs by starting a rigorous runtime analysis of simple ACO algorithms on OneMax. Soon after this appeal, first runtime analyses appeared in a work by Gutjahr [12], and, independently, in a work by Neumann and Witt [13]. These two papers study different types of ACO algorithms with possibly significantly different runtime behavior.

Neumann and Witt [13] define a simple ACO algorithm called 1-ANT based on the model of Gutjahr [9] and bound its runtime w. r. t. the fitness function OneMax from above and below. The investigations of 1-ANT with respect to its runtime behavior require deep insights into the pheromone update process used by an ACO algorithm. Therefore, not only the results but also the techniques developed by studying this algorithm are of great interest for understanding ACO algorithms from a theoretical point of view. Understanding gradual changes in the pheromone model of an ACO algorithm is perhaps the most fundamental task in the theory of ACO. It is also highly relevant today as recent theoretical studies of ACO in combinatorial optimization, such as minimum spanning trees [14] and the TSP [15], could only obtain results for a simplified pheromone model where only two values are attained. Increasing our knowledge on pheromone adaptations in pheromone models like the one investigated here may help to prove stronger results for ACO in combinatorial optimization.

In [13], it is shown that the so-called evaporation factor ρ, the important parameter that determines the update strength according to pheromone values in ACO algorithms, has a crucial impact on the runtime. More precisely, it is proved that there exists a threshold value for ρ below which no efficient optimization is possible. This threshold behavior shows that the 1-ANT is not robust w. r. t. the choice of ρ. Gutjahr [12] studies an ACO algorithm that is more directly inspired by the MAX–MIN ant system [16] and called MMAS in the following. Here the situation is different. Phase transitions do not occur and the expected runtime on simple functions (in particular OneMax and LeadingOnes, the latter being treated by Gutjahr and Sebastiani [17]) is polynomial if ρ is bounded by an inverse polynomial. From this perspective, MMAS is a far more efficient ACO algorithm than the 1-ANT. Still, the 1-ANT is an ACO variant of significant theoretical interest since it provides a detailed insight into the effect of pheromone updates in ACO and stimulated further studies (e. g., [18–20]).

The aim of this paper is to put forward the analysis of the 1-ANT on example problems in a similar fashion to Neumann and Witt [13]. As Gutjahr [21] has observed, such analyses are an important and emergent issue in the community of ACO. A closer look at [13] reveals that the mathematical methods employed for the analysis of the 1-ANT differ heavily from those for the analysis of EAs. Even more conspicuously, it seems that the mathematical tools are tailored for the symmetric function OneMax. It is by no means clear whether a comprehensive runtime analysis of the 1-ANT can be conducted on more complicated problems. A recent analysis of the 1-ANT on the combinatorial minimum spanning tree problem by the same authors [22] basically considers a special case of the 1-ANT with two pheromone values and fails to deliver statements on the choice of ρ. In this paper, we return to example problems by choosing the non-symmetric functions LeadingOnes and BinVal investigated by Droste et al. [2] and analyze the runtime of the 1-ANT on these functions w. r. t. n, the dimensionality of the search space, and the evaporation factor ρ.

The motivation for analyzing example problems is the same as for initial studies of evolutionary algorithms. Example functions are chosen that are simple enough to be attacked by rigorous arguments, have interesting properties, and yet reflect typical characteristics of practical problems. The simple function OneMax may be rediscovered in a practical problem when a specific target point has to be hit and the fitness function gives good hints towards this point. In the case of the functions LeadingOnes and BinVal, the bits have different priorities; some bits are more important than other bits. For BinVal this resembles a situation where the fitness gives good hints towards a target, but some local moves are rewarded more than others. The function LeadingOnes contains strong dependencies between specific bits, a characteristic often found in practical problems. Another interesting feature is that some bits are irrelevant to the fitness, until other bits have been set correctly. The investigations for LeadingOnes and BinVal in this paper put forward the understanding of ACO algorithms in such situations and lead to the development of new proof techniques.

Our upcoming analyses for these functions show that a similar phase transition behavior can be observed as proven by Neumann and Witt for OneMax [13]. Again, they give new insights into the impact of a pheromone update and its importance for the optimization process carried out by an ACO algorithm. We show that if ρ is asymptotically smaller than a threshold, no efficient optimization is possible; however, for values a little above the threshold, polynomial runtimes are very likely. Hence, our investigations again suggest that the 1-ANT is not robust w. r. t. the choice of ρ. The proofs contribute new methods for the runtime analysis of ACO algorithms and may serve as a basis for further theoretical studies. Still, recall that the 1-ANT is not the most efficient ACO algorithm for these problems. Different variants of the MAX–MIN ant system such as MMAS are not so sensitive to the choice of ρ, see [17,23].

The outline of the paper is as follows. In Section 2, we provide the necessary definitions and recapitulate the previous results for OneMax. Section 3 proves general properties of the pheromone update mechanism that will be used in the following analyses. Sections 4 and 5 deal with the main results of the paper, namely lower and upper bounds on the runtime

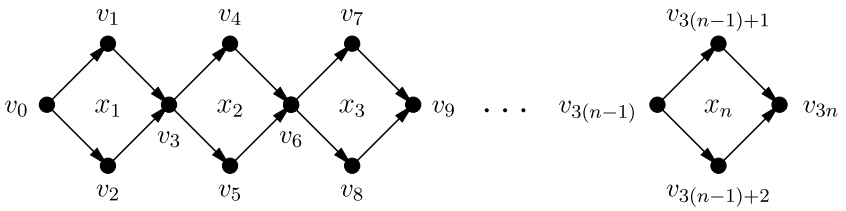

Fig. 1. Construction graph for pseudo-Boolean optimization.

of the 1-ANT on LeadingOnes, respectively. A generalization of the results to the function BinVal is discussed in Section 6. Our theoretical results are complemented by empirical results in Section 7 where we also consider an extension of the 1-ANT creating multiple ant solutions in every generation. We conclude in Section 8.

A preliminary version of the theoretical results without the experiments has appeared at GECCO 2007 [24]. In this article the proof techniques and the upper runtime bounds have been improved considerably. In particular, we are now able to present tight bounds for the phase transition on BinVal.

2. The algorithm

ACO algorithms construct solutions by random walks on so-called construction graphs. This random walk is influenced by values on the edges called pheromone values. In addition, the walk may be influenced by heuristic information about the problem. We use the setting of Neumann and Witt [13] where no heuristic information is used. As in [13], our main aim is to consider the effect of the pheromone update in a simple ACO algorithm called 1-ANT (see Algorithm 2) and to analyze its effect on the runtime for growing sizes of the optimization problem.

Let $C = (V, E)$ be the construction graph with a designated start vertex s and pheromone values τ on the edges. Starting at s, an ant traverses the construction graph depending on the pheromone value using Algorithm 1. Assuming that the ant is at vertex v, the ant moves to a successor w of v, where w is chosen with a probability that is proportional to the pheromone values of all non-visited successors of v. The process is iterated until all successors of the current vertex v have been visited.

Algorithm 1 (*Construct*(C, τ))**.**
(1) $v := s$, mark v as visited.
(2) While there is an unvisited successor of v in C:
  (a) Let N_v be the set of unvisited successors of v and $T := \sum_{(v,w)|w\in N_v} \tau_{(v,w)}$.
  (b) Choose one successor w of v where the probability of selecting $u \in N_v$ is $\tau_{(v,u)}/T$.
  (c) Mark w as visited, set $v := w$ and go to (2).
(3) Return the solution x and the path $P(x)$ constructed by this procedure.

In the initialization step of the 1-ANT, each edge gets a pheromone value of $1/|E|$ such that the pheromone values sum up to 1. After that, an initial solution x^* is produced by a random walk on the construction graph and the pheromone values are updated with respect to this walk. In each iteration, a new solution x is constructed and the pheromone values are updated if this solution is not inferior to the currently best solution x^*. We formulate our algorithm for maximization problems although it can be easily adapted to minimization.

Algorithm 2 (*1-ANT*)**.**
(1) Set $\tau_{(u,v)} = 1/|E|$ for all $(u, v) \in E$.
(2) Compute x (and $P(x)$) using Construct(C, τ).
(3) Update$(\tau, P(x))$ and set $x^* := x$.
(4) Compute x (and $P(x)$) using Construct(C, τ).
(5) If $f(x) \geq f(x^*)$, Update$(\tau, P(x))$ and set $x^* := x$.
(6) Go to (4).

For theoretical investigations, it is common to have no termination condition in such an algorithm. One is interested in the random optimization time which equals the number of constructed solutions until the algorithm has produced an optimal search point. Often one tries to bound the expected value of this time.

Considering the optimization for pseudo-Boolean fitness functions $f : \{0, 1\}^n \to \mathbb{R}$ with $n \geq 3$, we use the construction graph $C_{\text{bool}} = (V, E)$ (see Fig. 1) with $s = v_0$. In the literature, this graph is also known as *Chain* [12]. Optimizing bit strings of length n, the graph has $3n + 1$ vertices and $4n$ edges. The decision whether a bit x_i, $1 \leq i \leq n$, is set to 1 is made at node $v_{3(i-1)}$. In case that the edge $(v_{3(i-1)}, v_{3(i-1)+1})$ is chosen, x_i is set to 1 in the constructed solution. Otherwise $x_i = 0$ holds. After this decision has been made, there is only one single edge which can be traversed in the next step. In case that $(v_{3(i-1)}, v_{3(i-1)+1})$ has been chosen, the next edge is $(v_{3(i-1)+1}, v_{3i})$, and otherwise the edge $(v_{3(i-1)+2}, v_{3i})$ will be traversed. Hence, these edges have no influence on the constructed solution and we can assume

$$\tau_{(v_{3(i-1)},v_{3(i-1)+1})} = \tau_{(v_{3(i-1)+1},v_{3i})}$$

and

$$\tau_{(v_{3(i-1)},v_{3(i-1)+2})} = \tau_{(v_{3(i-1)+2},v_{3i})}$$

for $1 \le i \le n$. We call the edges $(v_{3(i-1)}, v_{3(i-1)+1})$ and $(v_{3(i-1)+1}, v_{3i})$ 1-*edges* and the other edges 0-*edges*.

The pheromone values are chosen such that at each time

$$\sum_{(u,v)\in E} \tau_{(u,v)} = 1$$

holds. In addition, it is useful to have bounds on the pheromone values (e. g., [16]) to ensure that each search point has a positive probability of being chosen in the next step. We restrict each $\tau_{(u,v)}$ to the interval $\left[\frac{1}{2n^2}, \frac{n-1}{2n^2}\right]$ and ensure $\sum_{(u,\cdot)\in E} \tau_{(u,\cdot)} = \frac{1}{2n}$ for $u = v_{3i}$, $0 \le i \le n-1$, and $\sum_{(\cdot,v)} \tau_{(\cdot,v)} = \frac{1}{2n}$ for $v = v_{3i}$, $1 \le i \le n$. This can be achieved by normalizing the pheromone values after an update and replacing the current value by $\frac{1}{2n^2}$ if $\tau_{(u,v)} < \frac{1}{2n^2}$ and by $\frac{n-1}{2n^2}$ if $\tau_{(u,v)} > \frac{n-1}{2n^2}$ holds. Depending on whether edge (u, v) is contained in the path $P(x)$ of the accepted solution x, the pheromone values are updated to τ' in the procedure Update$(\tau, P(x))$ as follows:

$$\tau'_{(u,v)} = \min\left\{\frac{(1-\rho)\cdot\tau_{(u,v)}+\rho}{1-\rho+2n\rho}, \frac{n-1}{2n^2}\right\} \quad \text{if } (u,v) \in P(x)$$

and

$$\tau'_{(u,v)} = \max\left\{\frac{(1-\rho)\cdot\tau_{(u,v)}}{1-\rho+2n\rho}, \frac{1}{2n^2}\right\} \quad \text{if } (u,v) \notin P(x),$$

where $\rho < 1$.

Let $p_i = \mathrm{P}(x_i = 1)$, $1 \le i \le n$, be the probability of setting the bit x_i to one in the next constructed solution. A consequence of the described setting is that $p_i \in [1/n, 1-1/n]$ due to the upper and lower bounds on the pheromone values. The choice of these values is inspired by standard mutation operators in evolutionary algorithms where there is always a probability of $1/n$ of exploring a different bit value.

Neumann and Witt [13] have shown a close connection between the 1-ANT and a simple evolutionary algorithm known as $(1+1)$ EA, which is defined as follows.

Algorithm 3 ($(1+1)$ *EA*)**.**
(1) Choose $x^* \in \{0,1\}^n$ uniformly at random.
(2) Create x by flipping every bit in x^* independently with probability $1/n$.
(3) If $f(x) \ge f(x^*)$ then $x^* := x$.
(4) Go to (2).

The $(1+1)$ EA sets a bit x_i in x to one with probability $1/n$ (if $x^*_i = 0$) or $1-1/n$ (if $x^*_i = 1$). Neumann and Witt have shown that the introduced 1-ANT behaves like the $(1+1)$ EA if $\rho \ge (n-2)/(3n-2)$. The reason is that then the 1-ANT's pheromone values always attain their upper or lower bounds after the first update has occurred. For the function OneMax with

$$\textsc{OneMax}(x) = \sum_{i=1}^{n} x_i$$

the influence of ρ has been analyzed in greater detail. Neumann and Witt have shown that there is a phase transition from exponential to polynomial runtime as ρ grows. In particular, they have given an exponential lower bound for the case $\rho = O(n^{1-\varepsilon})$ and a polynomial upper bound for $\rho = \Omega(n^{1+\varepsilon})$ where $\varepsilon > 0$ is in each case a positive constant. In addition, Doerr and Johannsen [19] have proven that no polynomial expected runtimes are possible for $\rho = o(1/(n \log n))$. The main argument for the lower bound is that the value of the currently best solution and the expected value of the one constructed in the next iteration may differ. This makes it difficult even to rediscover the best-so-far solution, which leads to an exponential optimization time. In contrast to this, the polynomial upper bound for large ρ relies on the observation that the function value of the last accepted solution determines the expected value of the next constructed solution almost exactly.

We consider the function LeadingOnes (proposed by Rudolph [25]) with

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

whose function value equals the number of leading ones in the considered bit string x. A non-optimal solution may always be improved by appending a single one to the leading ones. LeadingOnes differs from OneMax in the essential way that the assignment of the bits after the leading ones do not contribute to the function value. This implies that bits at the beginning of the bit string have a stronger influence on the function value than bits at the end. Because of this, the methods developed by Neumann and Witt [13] cannot be used for analyzing the 1-ANT on LeadingOnes as these methods make particular use

of the fact that all bits contribute equally to the function value. We will develop new methods to deal with the circumstance that different bits may have different priorities for the optimization process. A well-known linear function that relies on the different priorities is BinVal (introduced by Droste et al. [2]) defined as

$$\textsc{BinVal}(x) = \sum_{i=1}^{n} 2^{n-i} x_i,$$

which interprets a bit string as the binary representation of an integer. After having analyzed the 1-ANT on LeadingOnes, we will show how to adapt the developed methods for analyzing the 1-ANT on BinVal.

3. On the pheromone update mechanism

To analyze the 1-ANT for pseudo-Boolean optimization, it is necessary to understand the development of pheromone values for single bits. As defined in Section 2, for each bit there is a pair of 1-edges and a pair of 0-edges. The pheromone values on the edges in a pair are always equal. Therefore, we speak of only a single 1-edge and a single 0-edge for each bit when considering the pheromone values for a bit. The probability of setting a bit to 1 is called *success probability*. We already know that the success probability is proportional to the pheromone value on the corresponding 1-edge by a factor of $2n$. An analogous statement holds for the probability of setting the bit to 0 and the 0-edge. Finally, throughout the paper it is crucial to note that the bits are processed independently by the 1-ANT.

Consider an arbitrary but fixed bit x_i. If this bit is set to 1 in the next constructed solution, we speak of a success and a failure otherwise. Obviously, if a success occurs, the success probability in the next step is increased, unless the pheromone bounds have been hit in the previous step. The amount of increase depends on the previous pheromone value on the 1-edge (or, equivalently, the previous success probability). We introduce a notation for a success probability that has been increased t times.

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 bounds,*

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

where $\beta = (1-\rho)/(1+2n\rho-\rho)$.

Proof. Let $\tau = p/(2n)$ and $\tau^{(t)} = p^{(t)}/(2n)$ be the pheromone values on the corresponding 1-edge. We prove the claim by induction on t. The case $t = 0$ is obvious. For $t \geq 1$, we obtain

$$p^{(t)} = 2n \cdot \frac{(1-\rho)\tau^{(t-1)} + \rho}{1+2n\rho-\rho} = \frac{(1-\rho)p^{(t-1)} + 2n\rho}{1+2n\rho-\rho}$$

according to the pheromone update formula and the mapping from pheromones to probabilities. Using the induction hypothesis,

$$\begin{aligned} p^{(t)} &= \frac{(1-\rho)(1-(1-p)\cdot\beta^{t-1}) + 2n\rho}{1+2n\rho-\rho} \\ &= 1 - \frac{(1-\rho)(1-p)\beta^{t-1}}{1+2n\rho-\rho} = 1 - (1-p)\cdot\frac{(1-\rho)\beta^{t-1}}{1+2n\rho-\rho} \\ &= 1-(1-p)\cdot\beta^t \end{aligned}$$

according to the definition of β. □

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 bound 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.

We will also use the following lemma that bounds the success probability of a certain bit after t successes. It is an immediate consequence of Lemma 1.

Lemma 2.

$$p^{(t)} \leq p + t \cdot 2n\rho.$$

Proof. Let $\beta = (1-\rho)/(1+2n\rho-\rho)$ and observe $\beta^t = (1+(\beta-1))^t \geq 1 - t(1-\beta)$ by Bernoulli's inequality. Then

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

and the claim follows since $1-\beta = 2n\rho/(1+2n\rho-\rho) \leq 2n\rho$. □

4. A lower bound for LeadingOnes

In this section, we show that the 1-ANT is very inefficient on LeadingOnes if $\rho = o(1/(n \log n))$, i. e., if ρ is asymptotically smaller than $1/(n \log n)$. The following theorem shows that then even polynomially many multistarts fail within polynomial time with overwhelming probability.

One of the main reasons for the failure is that with the small evaporation factor, the success probabilities at single bits can reach large enough values only slowly. In consequence, the 1-ANT is faster in finding good solutions than in storing this knowledge in the pheromone values. Using Lemma 2, the following lower bound on the runtime of the 1-ANT can be shown.

Theorem 1. *With probability* $1-2^{-\Omega(\min\{1/(n\rho),\, n\})}$*, the runtime of the 1-ANT on* LeadingOnes *is at least* $2^{c\cdot\min\{1/(n\rho),\, n\}}$ *for some constant* $c > 0$.

Note that the statement of the theorem is only meaningful for $\rho = O(1/(n \log n))$. However, if we choose, e. g., $\rho = 1/(n \log^2 n)$, the bound is already superpolynomially large.

Proof. Let $k = 1/(8n\rho)$. Consider the state of the 1-ANT at the earliest time when one of the following two conditions is fulfilled.

(i) The fitness f_c of the current solution is at least $n/2$.
(ii) 1-ANT has performed k accepted steps.

We first convince ourselves that in this situation, all success probabilities never left the interval $[1/4, 3/4]$, then, that with high probability we are not done yet, and finally, that the next accepted step takes the time claimed in the theorem.

Consider the success probability of a certain bit. From Lemma 2, we know that at most k accepted steps, independent of the particular accepted solutions, can increase the success probability by at most $k \cdot 2n\rho$. Hence the success probability p after k such steps is bounded by $p \leq 1/2 + 2nk\rho \leq 3/4$. By symmetry, the same holds for the failure probability $1 - p$. Hence $p \in [1/4, 3/4]$.

Now let us regard the last (accepted) step before the system reached the state fixed above. At the start of this step, we have all success probabilities in $[1/4, 3/4]$ and our current solution has fitness $f_0 < n/2$. Hence the probability that the 1-ANT finds the optimal solution in this single step is bounded by $(3/4)^{(n/2)}$—recall that we know already that this step will be accepted, hence the first f_0 bits of this solution are one with probability one. Nevertheless, we see that with probability $1 - 2^{-\Omega(n)}$, we have not found the optimum yet.

Finally, let us estimate the time to obtain an accepted step from the state fixed above. We first estimate the current fitness f_c. If we did not perform k accepted steps then clearly $f_c \geq n/2$. Hence let us assume that we actually did perform k accepted steps. Conditional on the fact that a step was accepted, the probability that this leads to a fitness increase is at least $1/4$ since the probability of a success at the leftmost zero-bit is at least $1/4$. Hence $E(f_c) \geq k/4$, and the usual Chernoff bounds imply $\mathrm{P}(f_c \geq k/8) = 1 - 2^{-\Omega(k)} = 1 - 2^{-\Omega(1/(n\rho))}$. Combining the two cases, we have

$$\mathrm{P}\big(f_c \geq \min\{n/2, k/8\}\big) = 1 - 2^{-\Omega(1/(n\rho))}.$$

If $f_c \geq \min\{n/2, k/8\}$ then the probability that the next step is accepted is at most $(3/4)^{\min\{n/2,\, \lfloor k/8 \rfloor\}}$, i. e., at most $2^{-\Omega(\min\{1/(n\rho),\, n\})}$. Consequently, there is some small constant $c > 0$ such that a phase consisting of $2^{c\cdot\min\{1/(n\rho),\, n\}}$ steps does not produce another accepted step with probability $1 - 2^{-\Omega(\min\{1/(n\rho),\, n\})}$. □

5. An upper bound for LeadingOnes

In contrast to the situation from the last section, large values of ρ allow the 1-ANT to rediscover the leadings ones of previous solutions efficiently. In order to prove an upper bound for the 1-ANT, we make use of the observations from Section 3 and the following characteristics of LeadingOnes.

If k is the current best LeadingOnes-value, the success probabilities for the first k bits are increased in every pheromone update as an update is only performed when the leading ones are rediscovered. We say that these bits are in *increasing state*. The bit at position $k+1$ is called an *essential bit* since it is essential for the fitness evaluation. All bits following the first 0-bit have been irrelevant to the fitness evaluation up to now. Since the pheromone update process is symmetric for 0-edges and 1-edges, the probability that an ant creates a 1 on such a bit equals the probability that the ant creates a 0. These bits are therefore said to be in *random state*.

Due to this randomness, an improvement may increase the current best LeadingOnes-value from k to a value larger than $k + 1$ if the 1-ANT happens to create further bits with value 1. If the best LeadingOnes-value increases to $k + \ell$, the bits at positions $k + 2, \ldots, k + \ell$ are called *free riders* (see [2]) if $\ell \geq 2$. In other words, free riders are 1-bits that we get "for free" in addition to the essential 1-bit at position $k + 1$.

Free riders may help to increase the fitness, but on the other hand they may also prove as a burden for the 1-ANT. If an improvement creates many free riders, the 1-ANT is forced to rediscover all these free riders in the following steps in order to have another update. If ρ is small, this effect slows down the optimization. This is taken into account in the proof of the following upper bound.

Theorem 2. *The expected runtime of the 1-ANT on* LeadingOnes *is bounded from above by* $O(n^2 \cdot (6e)^{1/(n\rho)})$.

Note that the bound is polynomial for $\rho = \Omega(1/(n \log n))$ and only $O(n^2)$ for $\rho = \Omega(1/n)$. For $\rho = o(1/(n \log n))$, it is superpolynomially large.

Proof. We show that the expected time until increasing the so far maximum LeadingOnes-value is always bounded by $O(n \cdot (6e)^{1/(n\rho)})$ provided the optimum has not been reached. Multiplying the expected time for an improvement by the maximum number of improvements, n, will yield the theorem.

Suppose the currently best LeadingOnes-value equals $k < n$. For an improvement, it is necessary and sufficient to set the first $k+1$ bits in a newly constructed solution to 1. Since the essential bit at position $k+1$ was set to 0 in the last accepted solution, its success probability was decreased in the last pheromone update. Therefore, we estimate the success probability from below by $1/n$ for this bit. If we can prove that the first k bits are all set to 1 after an expected number of $O((6e)^{1/(n\rho)})$ steps, we obtain the theorem as the 1-ANT processes all bits independently.

The first k bits are all in increasing state, but their success probabilities may differ significantly. To simplify the considerations, we remove the pheromone bounds for a bit as soon as it enters the increasing state. This implies that the success probabilities of these bits are allowed to exceed $1 - 1/n$. We argue that we then underestimate the expected optimization time by only a constant factor. The success probabilities with and without pheromone bounds differ by at most $1/n$. Hence, with at most $n-1$ bits in increasing state, the probability that solution constructions with and without pheromone bounds differ in at least one of these bits is at most $1 - (1-1/n)^{n-1} \leq 1 - 1/e$. If a solution construction with pheromone bounds creates value 0 for a bit in increasing state, this solution is rejected by the 1-ANT as otherwise the bit would not be in increasing state. Moreover, in this case the current best-so-far solution and all pheromone values remain untouched. Hence, compared to the setting without pheromone bounds for bits in increasing state, we have "idle" steps for the 1-ANT with pheromone bounds. Our modification disregards these idle steps, therefore we underestimate the expected optimization time by a factor of at most $1/(1-1/e)$.

As a result of our modification, the success probability of the i-th bit, $1 \leq i \leq k-1$, has been increased at least as often as the one of the $(i+1)$-st bit. However, the leading k bits we want to rediscover are still of two different types. Such a bit has either been essential in the past or it has been added to the leading bits as a free rider. This distinction is crucial since the premises for the unrewarded success probabilities are different:

1. The success probability of an essential bit x_i is decreased every time a best-so-far solution with $i-1$ leading ones is created. At the time x_i enters increasing state, we therefore expect its unrewarded success probability to be biased towards the lower bound $1/n$.
2. A free rider x_i enters increasing state directly from random state, hence the unrewarded success probability p_i is random. More precisely, due to the 1-ANT's symmetric treatment of bits in random state, the expected unrewarded success probability equals $1/2$. The distribution of p_i is symmetric around the expectation, i. e., $\mathrm{P}(p_i = x) = \mathrm{P}(p_i = 1-x)$ for $0 \leq x \leq 1$. We are dealing with a discrete distribution since the underlying state space is countably infinite.

In the following, when speaking of "rediscovering an essential bit (a free rider)", this means a success for a bit that has been an essential bit (a free rider) in the past, but now has reached the increasing state. We first concentrate on rediscovering all essential bits in increasing state. The unrewarded success probability of one such bit is bounded below by the general lower bound for success probabilities, $w^{(0)} := 1/n$. Fortunately the rewarded success probabilities can be much larger. Temporarily scanning the leading ones from right to left, the t-th essential bit in increasing state has been rewarded at least t times. Hence, its success probability is bounded below by the worst-case lower bound $w^{(t)}$, recalling the notation for rewarded success probabilities from Definition 1. As the 1-ANT sets bits to 1 independently, the probability for successes on all these bits is bounded by the product over $w^{(t)}$. The expected number of steps until this happens is bounded by the reciprocal since the considered random variable is geometrically distributed. We use the following estimation, the proof of which is deferred to Lemma 4 in the Appendix.

$$\prod_{t=1}^{\infty} \frac{1}{w^{(t)}} \leq \prod_{t=1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) = O\big((2e)^{1/(n\rho)}\big).$$

We are left with the task of rediscovering all free riders. We claim that this expected time is bounded by $O(3^{1/(n\rho)})$, implying a total bound of $O((6e)^{1/(n\rho)})$ for rediscovering all leading ones.

The consideration of free riders is more difficult than the arguments for essential bits. This is due to the fact that in an improvement the number of gained free riders is random. This motivates us to divide the k leading ones into blocks of random length. Denote by b the number of improving steps so far. All free riders gained during the i-th improvement, $1 \leq i \leq b$, form a block of free riders. Let us pessimistically assume that an unbounded number of free riders is possible, i. e., we prolong the bit string if necessary until the first 0-bit is created. We first estimate the success probability of a block, i. e., the probability of setting all bits of the block to 1, in isolation and argue later how to extend this analysis to multiple blocks. In the following, we will derive bounds on the expected time to rediscover a single block after the block has received t updates.

Consider the bit string of all m bits in random state. Let p_i be the success probability of the i-th bit in random state at the time an improvement takes place. Consider the new block of free riders created during this improvement and let $T^{(t)}$

denote the random time until this new block of free riders is rediscovered (or the global optimum is found), provided it has been rewarded t times. Our aim is to bound $E\big(T^{(t)}\big)$. Therefore, we abbreviate $\mathbf{p} = (p_1, \ldots, p_m)$ and apply the law of total expectation

$$E\big(T^{(t)}\big) = E\big(E\big(T^{(t)} \mid \mathbf{p}\big)\big)$$

as it is easier to estimate the inner, conditional expectation. Let L denote the random length of the new block of free riders. By the law of total probability

$$E\big(T^{(t)} \mid \mathbf{p}\big) = \sum_{\ell=0}^{m} E\big(T^{(t)} \mid L = \ell \wedge \mathbf{p}\big) \cdot \mathrm{P}(L = \ell \mid \mathbf{p}).$$

In order to have block length ℓ, the 1-ANT has to be successful on the first ℓ bits and to fail on the $(\ell+1)$-st bit. Therefore $\mathrm{P}(L = \ell \mid \mathbf{p}) = \prod_{i=1}^{\ell} p_i \cdot (1-p_{\ell+1})$. The probability of rediscovering the first ℓ bits equals the product of the rewarded success probabilities $p_i^{(t)}$ since the 1-ANT processes bits independently. Since LeadingOnes is considered, no pheromone update takes place unless these bits are rediscovered. Hence, the waiting time is geometrically distributed with the mentioned product as parameter. This yields

$$\begin{aligned} E\big(T^{(t)} \mid \mathbf{p}\big) &= \sum_{\ell=0}^{m-1} \prod_{i=1}^{\ell} p_i \cdot (1-p_{\ell+1}) \cdot \prod_{i=1}^{\ell} \frac{1}{p_i^{(t)}} \\ &= \sum_{\ell=0}^{m-1} (1-p_{\ell+1}) \prod_{i=1}^{\ell} \frac{p_i}{p_i^{(t)}} \\ &= \sum_{\ell=0}^{m-1} (1-p_{\ell+1}) \prod_{i=1}^{\ell} \frac{p_i}{1-(1-p_i)\cdot\beta^t}, \end{aligned} \tag{1}$$

where the last equation follows from Lemma 1. Note that in case $L = m$ the global optimum has been found, hence we only deal with the case $L \le m-1$.

To estimate the unconditional expectation $E\big(T^{(t)}\big)$, we repeat that the p_i are independent random variables and their distributions satisfy the symmetry $\mathrm{P}(p_i = x) = \mathrm{P}(p_i = 1-x)$ for $0 \le x \le 1$, implying $E(p_i = 1/2)$. Note, however, that the distributions are time-dependent. In the initial step, $\mathrm{P}(p_i = 1/2) = 1$, which need not be the case in later steps. We may ignore this time dependence by stating a bound which holds for all distributions of the above kind.

$$\begin{aligned} E\big(T^{(t)}\big) &= E\big(E\big(T^{(t)} \mid \mathbf{p}\big)\big) \\ &= E\left(\sum_{\ell=0}^{\infty} (1-p_{\ell+1}) \prod_{i=1}^{\ell} \frac{p_i}{1-(1-p_i)\cdot\beta^t}\right) \\ &= \sum_{\ell=0}^{\infty} E(1-p_{\ell+1}) \prod_{i=1}^{\ell} E\left(\frac{p_i}{1-(1-p_i)\cdot\beta^t}\right) \\ &= \frac{1}{2} \sum_{\ell=0}^{\infty} \prod_{i=1}^{\ell} E\left(\frac{p_i}{1-(1-p_i)\cdot\beta^t}\right). \end{aligned}$$

For the third equation we have used the independence of the random variables and for the fourth equation we have used $E(1-p_i) = 1/2$. The remaining expectation can be bounded by the following inequality, which is proven by Lemma 5 in the Appendix. The proof only relies on the symmetry of p_i and hence it holds regardless of the time dependence.

$$E\left(\frac{p_i}{1-(1-p_i)\cdot\beta^t}\right) \le \frac{1}{2-\beta^t}.$$

Together, we bound $E\big(T^{(t)}\big)$ by

$$\frac{1}{2} \sum_{\ell=0}^{\infty} \prod_{i=1}^{\ell} \frac{1}{2-\beta^t} \le \frac{1}{2}\left(1 + \frac{1}{1-\beta^t}\right) = 1 + \frac{1}{2}\frac{1}{\beta^{-t}-1},$$

where we have estimated the sum by an infinite series. Plugging in $\beta \le 1/(1+2n\rho)$ yields

$$E\big(T^{(t)}\big) \le 1 + \frac{1}{2}\frac{1}{(1+2n\rho)^t - 1} := \alpha(t).$$

We have successfully bounded the expected time to rediscover a single block of free riders. Although the distributions of success probabilities in later blocks differ from those in earlier blocks as is the case with the actual block lengths, our upper

bound $\alpha(t)$ on $E\left(T^{(t)}\right)$ holds regardless on the configuration of earlier blocks. This intuitively explains that the product over the corresponding $\alpha(t)$ for all blocks represents a bound for the expected time to rediscover all blocks of free riders in increasing state. A formal proof of this is given by Lemma 6 in the Appendix. Recall that the t-th newest block has been rewarded at least t times. Finally, the product of the $\alpha(t)$ can be bounded similarly to the product of inverse worst-case lower bounds from Lemma 4, it is proven by Lemma 7 in the Appendix. We obtain

$$\prod_{t=1}^{\infty}\alpha(t) \le \prod_{t=1}^{\infty}\left(1+\frac{1}{2}\frac{1}{(1+2n\rho)^t-1}\right) = O\left(3^{1/(n\rho)}\right)$$

which completes the proof. □

Theorems 1 and 2 reveal a phase transition in the behavior of the 1-ANT on LeadingOnes for $\rho \sim 1/(n\log n)$. Below this threshold, no efficient optimization is possible since the effect of pheromone updates is more or less irrelevant. This is similar to the behavior observed on OneMax, where the threshold value has been confined to a small region around $1/n$, more precisely between $\Omega(1/(n\log n))$ [19] and $O(n^{\varepsilon}/n)$ [13]. This shows that the 1-ANT is not robust w. r. t. the choice of the parameter ρ on two well-known and simple example functions.

6. Generalization to BinVal

The example function BinVal (see Section 2) is a linear function, although, in some respect an extreme example. The coefficient 2^{n-i} of the i-th bit outweighs the sum of all smaller coefficients. This leads to the following relation to LeadingOnes: If LeadingOnes$(x') >$ LeadingOnes(x) for $x, x' \in \{0,1\}^n$ then also BinVal$(x') >$ BinVal(x). This allows us to treat the LeadingOnes-value of the current solution as a potential function while BinVal is optimized. It is sufficient to increase the potential at most n times to reach the optimal solution (albeit the number of different BinVal-values is 2^n). Similarly, with probability $1-2^{-\Omega(n)}$, it is necessary to increase the potential altogether by at least $n/2$ to reach the optimum since the initial LeadingOnes-value does not exceed $n/2$ with this probability.

When the $(1+1)$ EA optimizes BinVal, the described approach allows us to immediately take over the upper bound $O(n^2)$ for the expected optimization time on LeadingOnes. This is not the best bound possible since $O(n\log n)$ can be shown by a direct approach. Such a direct approach seems difficult for the 1-ANT on BinVal. Therefore, we rather try to transfer our results from LeadingOnes to BinVal using the potential function.

With respect to the lower bound, we inspect the proof of Theorem 1. Instead of considering the real BinVal, we take the LeadingOnes-value of a bit string as a pseudo-fitness. The arguments on the pheromone values are still valid, and moreover, it is still necessary to create a solution with pseudo-fitness at least n to optimize BinVal, implying that the optimum is not found with probability $1-2^{-\Omega(n)}$ before the crucial point of time in the proof is reached. The Chernoff-bound-type arguments on the (pseudo-)fitness carry over, too. Moreover, for an accepted step, it is afterwards still necessary to create a search point with pseudo-fitness at least f_c. We have shown that $\rho = o(1/(n\log n))$ leads to superpolynomial runtimes also on BinVal.

Theorem 3. *With probability* $1-2^{-\Omega(\min\{1/(n\rho),\, n\})}$*, the runtime of the 1-ANT on* BinVal *is at least* $2^{c\cdot(\min\{1/(n\rho),\, n\})}$ *for some constant* $c > 0$.

We can also adapt the proof of the upper bound on LeadingOnes for BinVal. As for LeadingOnes, the first leading ones are in increasing state since a failure on a leading one is not accepted by the 1-ANT. In addition, a sufficient condition for an improvement, provided we have k leading ones, is to have successes for the first $k+1$ bits. We can also speak of essential bits and free riders in the context of BinVal, where the distinction is again made whenever a bit enters increasing state. Using Lemma 4, the expected time to rediscover all essential bits is bounded by $O((2e)^{1/(n\rho)})$.

The only but essential difference between LeadingOnes and BinVal is that BinVal is influenced by the configuration of the bits following the leftmost zero. Hence, the former "random state" is now biased towards high success probabilities and the probability of a bit being a "free rider" is likely to exceed $1/2$. Recall $E\left(T^{(t)}\right) = E\left(E\left(T^{(t)} \mid p_1,\ldots,p_m\right)\right)$. While the estimation of the inner conditional expectation from Eq. (1) still holds true for BinVal, we cannot rely on the symmetry of the p_i-variables to estimate the outer expectation. However, we can make an estimation that does not rely on specific properties of the success probabilities $p_1,\ldots,p_m$. Instead, we show a bound that holds for all success probabilities.

Lemma 3. *For all* $m, t \ge 1$ *and all* $p_1,\ldots,p_m \in [0,1]$

$$\sum_{\ell=0}^{m-1}(1-p_{\ell+1})\prod_{i=1}^{\ell}\frac{p_i}{1-(1-p_i)\beta^t} \le \frac{1}{1-\beta^t}.$$

Proof. For $0 \le j \le m-1$ define

$$S_j := \sum_{\ell=j}^{m-1}(1-p_{\ell+1})\prod_{i=j+1}^{\ell}\frac{p_i}{1-(1-p_i)\beta^t}.$$

Observe that for $1 \le j \le m-1$ the S_j are due to the following recursive formulation:

$$S_{j-1} = (1-p_j) + \frac{p_j}{1-(1-p_j)\beta^t} \cdot S_j.$$

We prove $S_j \le 1/(1-\beta^t)$ by induction on j. The case $j=0$ proves the lemma. We start with $j = m-1$:

$$S_{m-1} = (1-p_m) \le 1 \le \frac{1}{1-\beta^t}.$$

Assume $S_j \le 1/(1-\beta^t)$ and note that S_j does only depend on $p_{j+1}, \ldots, p_m$. This yields

$$\begin{aligned} S_{j-1} &= 1 - p_j + \frac{p_j}{1-(1-p_j)\beta^t} \cdot S_j \\ &\le \max_{p_j} \left\{ 1 - p_j + \frac{p_j}{1-(1-p_j)\beta^t} \cdot \frac{1}{1-\beta^t} \right\} \end{aligned}$$

since S_{j-1} is monotone increasing with S_j. Call the term in brackets $r(p_j)$. We now argue that $r(p_j)$ is monotone increasing with p_j. Hence, the maximum is attained for $p_j = 1$ and it follows

$$S_{j-1} \le 1 - 1 + \frac{1}{1-(1-1)\beta^t} \cdot \frac{1}{1-\beta^t} = \frac{1}{1-\beta^t}.$$

The derivative of $r(p_j)$ is

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}p_j} r(p_j) &= -1 + \frac{1-(1-p_j)\beta^t - p_j\beta^t}{(1-(1-p_j)\beta^t)^2} \cdot \frac{1}{1-\beta^t} \\ &= -1 + \frac{1-\beta^t}{(1-(1-p_j)\beta^t)^2} \cdot \frac{1}{1-\beta^t} \\ &= -1 + \frac{1}{(1-(1-p_j)\beta^t)^2} \end{aligned}$$

and this term is positive as $(1-(1-p_j)\beta^t)^2 \le 1$. □

Lemma 3 bounds the expected time to rediscover a single block of "free rider" bits after t updates by $\alpha(t) := 1/(1-\beta^t)$. Lemma 6 proves that the product over the $\alpha(t)$ is an upper bound on the expected time to rediscover all "free riders". We estimate

$$\prod_{t=1}^{\infty} \frac{1}{1-\beta^t} \le \prod_{t=1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) = O\big((2e)^{1/(n\rho)}\big)$$

where the first inequality follows from $\beta \le 1/(1+2n\rho)$ and the second estimate follows from Lemma 4. This holds independently from the success probability of a bit at the time when it first becomes a "free rider". Together with the bound $O((2e)^{1/(n\rho)})$ to rediscover the essential bits in increasing state, we have shown the following theorem.

Theorem 4. *The expected runtime of the 1-ANT on* BinVal *is bounded from above by* $O(n^2 \cdot (4e^2)^{1/(n\rho)})$.

As for LeadingOnes, Theorems 3 and 4 show a phase transition behavior for $\rho \sim 1/(n \log n)$. This strengthens the impression that the efficiency of the 1-ANT on pseudo-Boolean optimization problems is not robust w. r. t. the pheromone update mechanism and the choice of ρ.

7. Experiments

This section complements our theoretical investigations with experimental studies. The theoretical results are asymptotic ones. The aim of our experimental investigations is to show that the properties proven in the previous sections can also be observed for realistic input sizes. Later on, we also examine what happens if more than one ant is used in each iteration.

As our asymptotic results focus on n, we first consider the growth of the average runtime with n, for various values of ρ. We choose $n \in \{8, 16, 24, \ldots, 64\}$ and $\rho \in \{0.01, 0.002, 0.001, 0.0005\}$. The average runtime in 100 runs was recorded; the result is shown in Fig. 2. Note that the runtime is displayed on a logarithmic scale. For small n most ρ-values were too small to allow for an efficient behavior and in these cases the average runtime increases exponentially with n. However, as n grows further, larger values of ρ gradually make the 1-ANT more efficient as we move past the predicted phase transition with respect to n and ρ.

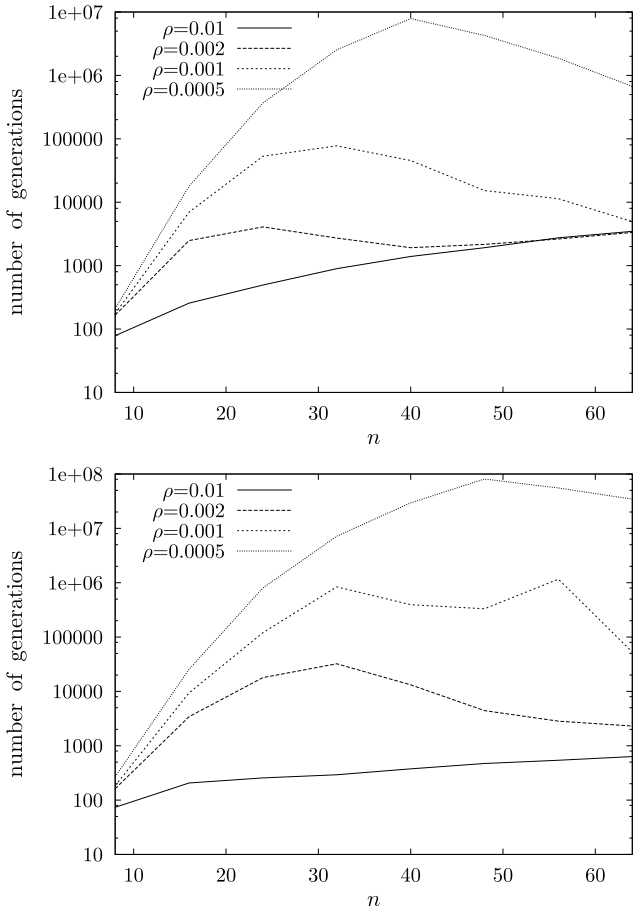

Fig. 2. Number of generations for the 1-ANT for finding the global optimum on LeadingOnes (top) and BinVal (bottom) for $n \in \{8, 16, 24, \ldots, 64\}$ and $\rho \in \{0.01, 0.002, 0.001, 0.0005\}$, averaged over 100 runs in each setting.

The variance was very high, in particular for small values of ρ. While for $\rho = 0.01$ the standard deviation was typically somewhere between a quarter and half the mean, it increased to values between the mean and twice the mean for $\rho = 0.0005$.

Our theoretical results only show a phase transition behavior for $\rho \sim 1/(n \log n)$, without making a statement on the precise constant. In order to get more insights on the exact location of the phase transition, we fix $n = 64$ and investigate values of ρ proportional to $1/(n \log n)$. More precisely, we choose $\rho = c/(n \log n)$ for $c \in \{0.15, 0.2, \ldots, 1.0\}$.

For the upper bound on LeadingOnes we argued that free riders slow down the optimization if ρ is small. For BinVal, we expect to have more free riders and we were only able to prove a worse upper bound than for LeadingOnes. It is therefore interesting to ask whether this intuition about free riders is true, i. e., whether the 1-ANT really performs worse on BinVal than on LeadingOnes.

The average number of generations to obtain an optimal solution is shown in Fig. 3 in dependence of the evaporation factor ρ. The results are again averaged over 100 runs in each setting and a logarithmic scale is used. We see that with decreasing values of ρ the runtime increases drastically for both functions and that the 1-ANT becomes inefficient for $\rho < 0.3/(n \log n)$. For $n = 64$ this corresponds to a ρ-value of 0.00059 and it matches our observations from Fig. 2. It can also be observed that the runtime of the 1-ANT on BinVal is larger than for LeadingOnes if ρ is small. This supports our argumentation that free riders can be a burden for the 1-ANT. On the other hand, for larger values of ρ the 1-ANT is more efficient on BinVal than on LeadingOnes. This effect can also be explained by theoretical arguments. With growing ρ the 1-ANT behaves more and more like the $(1 + 1)$ EA and the $(1 + 1)$ EA is more efficient on BinVal than on LeadingOnes [2]. This difference is not reflected in our upper bound for BinVal as we focus on the runtime behavior for small ρ.

Another goal of our experimental studies is to examine situations related to our theoretical model where our rigorous analyses do not give a picture of the runtime behavior of ACO algorithms. A possible weakness of the 1-ANT is that it produces in each iteration just one single solution. Usually, many ants are used in one iteration for updating the pheromone values. Therefore, we also study a more realistic ACO algorithm that is closely related to the algorithm analyzed in the previous sections. Our algorithm called λ-ANT uses λ ants instead of a single one. The best solution found among the λ solutions is

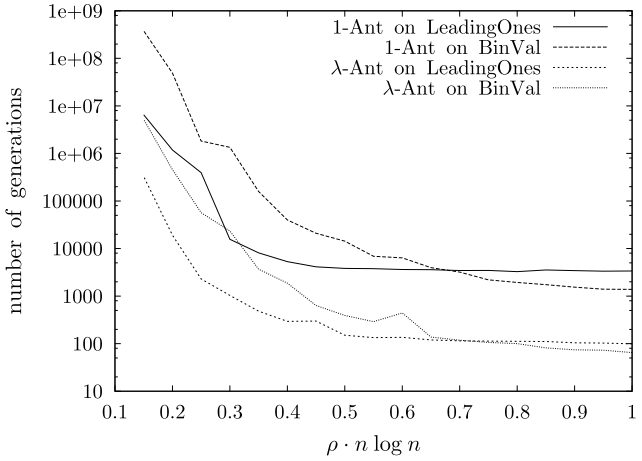

Fig. 3. Number of generations for the 1-ANT and the λ-ANT to find the global optimum on LeadingOnes and BinVal, resp., for $\lambda = n = 64$, averaged over 100 runs for each value of ρ.

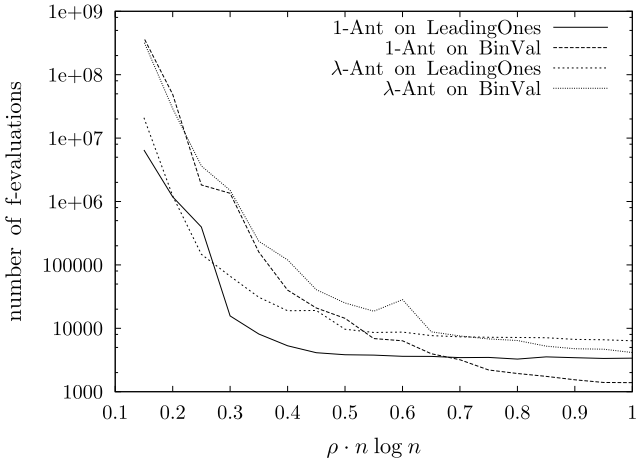

Fig. 4. Number of fitness evaluations for the 1-ANT and the λ-ANT to find the global optimum on LeadingOnes and BinVal, resp., for $\lambda = n = 64$, averaged over 100 runs for each value of ρ.

used for updating the pheromone values if this solution is at least as good as the best solution found so far during the run of the algorithm.

For our experimental studies, we choose $\lambda = n = 64$ such that the number of ants used in one iteration is sufficiently large to examine the effect of using more than one ant. As ACO algorithms can easily be parallelized, we report the number of generations (see Fig. 3) as well as the number of fitness evaluations (see Fig. 4) the λ-ANT needs to reach the optimal solution for LeadingOnes and BinVal. These results show that the number of generations can be reduced by using more than one ant in each iteration. This has already been observed in the context of evolutionary algorithms where the number of iterations can be reduced by producing λ offspring instead of a single one (see e. g. [26]).

As in the case of the investigations for evolutionary algorithms it is hard to reduce the number of fitness evaluations. Fig. 4 shows larger average runtimes for the λ-ANT compared to the 1-ANT if ρ is large. For smaller ρ the curves partly overlap. Therefore, we use statistical tests for the hypothesis that the 1-ANT outperforms the λ-ANT. A non-parametric Mann–Whitney test shows a significant advantage for the 1-ANT for all observed values of ρ. The significance levels are $p < 0.05$ for LeadingOnes and $\rho = 0.15/(n \log n)$, $p < 0.01$ for LeadingOnes and $\rho = 0.20/(n \log n)$, and $p < 0.0001$ for all other ρ-values and all results on BinVal. This in particular shows that the averages are very sensitive to outliers, i. e., runs that took a very long time to finish. Even though for $\rho = 0.25/(n \log n)$ the 1-ANT has a larger average runtime than the λ-ANT on LeadingOnes, our statistical test clearly state that the 1-ANT performs better than the λ-ANT in this setting.

Due to these tests and insights from our theoretical investigations for the 1-ANT, we conjecture that using just one ant is the optimal choice for our model when considering the functions LeadingOnes and BinVal. However, a rigorous proof for this conjecture remains open.

8. Conclusions

We have investigated the runtime behavior of a simple ACO algorithm called 1-ANT. Our investigations show some general properties for the update scheme used in the 1-ANT. Based on these investigations, we have shown that there is a phase transition for $\rho \sim 1/(n \log n)$ from exponential to polynomial runtimes for the functions LeadingOnes and BinVal. Our experimental investigations point out that this can even be observed for instances of moderate size as well as for the λ-ANT using a linear number of ants in each iteration. Moreover, the experiments indicate that using many ants cannot decrease the number of function evaluations on the considered functions.

We want to state some open problems regarding ACO algorithms. First, it would be desirable to examine standard ACO algorithms, e. g., MMAS, using more than one ant by rigorous analyses. Here, it would be especially interesting in which situations the use of more ants is provably beneficial. Another challenging problem is to analyze the choice of the evaporation factor in ACO algorithms for classical combinatorial optimization problems.

Acknowledgements

The third and fourth authors were partly supported by the German Science Foundation (DFG) as a part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

Appendix

Lemma 4. *Let $w^{(0)} = 1/n$, then*

$$\prod_{t=1}^{\infty} \frac{1}{w^{(t)}} \leq \prod_{t=1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) = O\big((2e)^{1/(n\rho)}\big).$$

Proof. By Lemma 1 and $\beta \leq 1/(1+2n\rho)$,

$$\begin{aligned}\prod_{t=1}^{\infty} \frac{1}{w^{(t)}} &= \prod_{t=1}^{\infty} \frac{1}{1-(1-1/n)\beta^t} \\ &\leq \prod_{t=1}^{\infty} \frac{1}{1-(1+2n\rho)^{-t}} \\ &= \prod_{t=1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right).\end{aligned}$$

For the second statement, we bound the first $\alpha = \lceil 1/(2n\rho) \rceil$ factors of the above product separately.

$$\begin{aligned}\prod_{t=1}^{\alpha} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) &\leq \prod_{t=1}^{\alpha} \left(1 + \frac{1}{2n\rho t}\right) \\ &\leq \prod_{t=1}^{\alpha} \left(1 + \frac{\alpha}{t}\right) = \frac{(2\alpha)!}{(\alpha!)^2} \\ &= \frac{\Theta(\sqrt{\alpha})(2\alpha)^{2\alpha}}{\Theta(\alpha)\alpha^{2\alpha}} = O(4^{\alpha})\end{aligned}$$

where the last estimations follow from Stirling's approximation and $\alpha \geq 1$. The remaining terms are bounded as follows, using $(1 + 1/\alpha)^{\alpha} \geq 2$.

$$\begin{aligned}\prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) &\leq \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{(1+1/\alpha)^t - 1}\right) \\ &\leq \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{2^{t/\alpha} - 1}\right) \\ &\leq \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{2}{2^{t/\alpha}}\right) \\ &\leq \prod_{t=\alpha+1}^{\infty} \exp\left(\frac{2}{2^{t/\alpha}}\right) = \exp\left(\sum_{t=\alpha+1}^{\infty} \frac{2}{2^{t/\alpha}}\right). \qquad (2)\end{aligned}$$

Now,

$$\sum_{t=\alpha+1}^{\infty} \frac{2}{2^{t/\alpha}} \leq \sum_{t=\alpha}^{\infty} \frac{2}{2^{\lfloor t/\alpha \rfloor}}$$

and grouping α summands and performing an index transformation with $j = t\alpha$, we arrive at

$$\sum_{t=\alpha+1}^{\infty} \frac{2}{2^{t/\alpha}} \leq \sum_{j=1}^{\infty} \frac{2\alpha}{2^j} = 2\alpha.$$

Plugging this into (2) yields

$$\prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) \leq e^{2\alpha}$$

and for the whole product the bound $O\left((4e^2)^\alpha\right) = O((2e)^{1/(n\rho)})$. □

Lemma 5. *Consider a random variable $X \in [0, 1]$ with finite support where $\mathrm{P}(X = x) = \mathrm{P}(X = 1 - x)$. If $0 \leq \beta^t \leq 1$,*

$$E\left(\frac{X}{1-(1-X)\beta^t}\right) \leq \frac{1}{2-\beta^t}.$$

Proof. We show that the expectancy is maximized if $\mathrm{P}(X = 1/2) = 1$. Let $\Omega(X)$ denote the (finite) support of X. By the symmetry of the distribution,

$$\begin{aligned} E\left(\frac{X}{1-(1-X)\beta^t}\right) &= \sum_{x\in\Omega(X)} \mathrm{P}(X = x)\frac{x}{1-(1-x)\beta^t} \\ &= \sum_{x\in\Omega(X)\cap[1/n,1/2)} \mathrm{P}(X = x)\left(\frac{x}{1-(1-x)\beta^t} + \frac{1-x}{1-x\beta^t}\right) + \mathrm{P}(x = 1/2)\cdot\frac{1/2}{1-\beta^t/2}. \end{aligned}$$

Since a finite set is considered, the term in parentheses attains a maximum at some point x^*. The expectation is maximized if the distribution gives maximal mass to x^* and $1 - x^*$, which means mass $1/2$ each if $x^* < 1/2$ and mass 1 otherwise. It is easy to see that $\frac{x}{1-(1-x)\beta^t}$ is concave for positive x and $0 \leq \beta \leq 1$, hence the term in parentheses is non-decreasing, and we must choose $x^* = 1/2$ to maximize the expectation. This implies

$$E\left(\frac{X}{1-(1-X)\beta^t}\right) \leq \frac{1/2}{1-(1-1/2)\beta^t} = \frac{1}{2-\beta^t}. \quad \square$$

We justify why blocks of free riders can be treated in isolation in the proofs of the Theorems 2 and 4. The product of bounds on waiting times for single blocks yields an upper bound on the waiting times for all blocks.

Lemma 6. *Consider the 1-ANT on either* LeadingOnes *or* BinVal. *Let $L_k \in \mathbb{N}_0$ be the (random) block length of the k-th block of "free rider" bits gained during the k-th increase in the number of leading ones. Abbreviate $\ell[1, i] := \{L_1 = \ell_1 \wedge \cdots \wedge L_i = \ell_i\}$ for a given history of lengths for the first i blocks. Given $\ell[1, i]$, let $p_{i,j}^{\ell[1,i]}$ denote the concrete success probability of the j-th bit in the i-th block at a current point of time.*

Let $T_{\mathrm{all}}(k)$ be the unconditional random number of further steps until the 1-ANT for the first time rediscovers the first k blocks of free rider bits. If there are upper bounds $\alpha_1, \ldots, \alpha_k$ such that

$$\sum_{\ell_i=0}^{n} \mathrm{P}(L_i = \ell_i \mid \ell[1, i-1]) \cdot \prod_{j=1}^{\ell_i} \frac{1}{p_{i,j}^{\ell[1,i]}} \leq \alpha_i$$

for all $1 \leq i \leq k$, then

$$E(T_{\mathrm{all}}(k)) \leq \prod_{i=1}^{k} \alpha_i.$$

Proof. Note that

$$\mathrm{P}(\ell[1, k]) = \prod_{i=1}^{k} \mathrm{P}(L_i = \ell_i \mid \ell[1, i-1]). \tag{3}$$

Applying the law of total probability with respect to the k block lengths, we have

$$E(T_{\mathrm{all}}(k)) = \sum_{\ell[1,k]} \mathrm{P}(\ell[1, k]) \cdot E(T_{\mathrm{all}}(k) \mid \ell[1, k]). \tag{4}$$

In the conditional expectation on the right-hand side, the lengths of all considered blocks are known and the concrete success probabilities are given as well. As all blocks cover disjoint portions of the bit string and the 1-ANT processes all bits independently, the probability of rediscovering all these bits is just the product of all success probabilities involved. Since LeadingOnes or BinVal is considered, no pheromone update takes place unless this event happens. Hence, the waiting time is geometrically distributed with the product of success probabilities as parameter, i. e.,

$$E(T_{\text{all}}(k) \mid \ell[1,k]) = \prod_{i=1}^{k} \prod_{j=1}^{\ell_i} \frac{1}{p_{i,j}^{\ell[1,i]}}.$$

Along with Eqs. (3) and (4),

$$E(T_{\text{all}}(k)) = \sum_{\ell[1,k]} \prod_{i=1}^{k} \mathrm{P}(L_i = \ell_i \mid \ell[1,i-1]) \cdot \prod_{j=1}^{\ell_i} \frac{1}{p_{i,j}^{\ell[1,i]}}.$$

Abbreviate

$$F_i := \mathrm{P}(L_i = \ell_i \mid \ell[1,i-1]) \cdot \prod_{j=1}^{\ell_i} \frac{1}{p_{i,j}^{\ell[1,i]}}.$$

Pessimistically assuming that all blocks can have lengths between 0 and n, we have

$$\begin{aligned} E(T_{\text{all}}(k)) &= \sum_{\ell[1,k]} \prod_{i=1}^{k} F_i \le \sum_{\ell_1=0}^{n} \sum_{\ell_2=0}^{n} \cdots \sum_{\ell_k=0}^{n} \prod_{i=0}^{k-1} F_i \\ &= \sum_{\ell_1=0}^{n} F_1 \cdot \sum_{\ell_2=0}^{n} F_2 \cdots \sum_{\ell_k=0}^{n} F_k \\ &\le \prod_{i=1}^{k} \alpha_i \end{aligned}$$

according to the given bounds. □

Lemma 7.

$$\prod_{t=1}^{\infty} \left(1 + \frac{1}{2} \frac{1}{(1+2n\rho)^t - 1}\right) = O\left(3^{1/(n\rho)}\right).$$

Proof. The proof is very similar to the proof of Lemma 4 as the calculations only differ in some constants. We now bound the first $\alpha = \lceil 1/(4n\rho) \rceil$ factors separately.

$$\begin{aligned} \prod_{t=1}^{\alpha} \left(1 + \frac{1}{2} \frac{1}{(1+2n\rho)^t - 1}\right) &\le \prod_{t=1}^{\alpha} \left(1 + \frac{1}{4n\rho t}\right) \\ &\le \prod_{t=1}^{\alpha} \left(1 + \frac{\alpha}{t}\right) = \frac{(2\alpha)!}{(\alpha!)^2} \\ &= \frac{\Theta(\sqrt{\alpha})(2\alpha)^{2\alpha}}{\Theta(\alpha)\alpha^{2\alpha}} = O(4^{\alpha}) \end{aligned}$$

where the last estimations follow from Stirling's approximation and $\alpha \ge 1$. The remaining terms are bounded as follows, using $(1 + 1/(2\alpha))^{\alpha} \ge 3/2$.

$$\begin{aligned} \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{2} \frac{1}{(1+2n\rho)^t - 1}\right) &\le \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{2} \frac{1}{(1+1/(2\alpha))^t - 1}\right) \\ &\le \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{2} \frac{1}{(3/2)^{t/\alpha} - 1}\right) \\ &\le \prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{2} \frac{3}{(3/2)^{t/\alpha}}\right) \\ &\le \prod_{t=\alpha+1}^{\infty} \exp\left(\frac{3}{2} \frac{1}{(3/2)^{t/\alpha}}\right) = \exp\left(\frac{3}{2} \sum_{t=\alpha+1}^{\infty} \frac{1}{(3/2)^{t/\alpha}}\right). \end{aligned} \tag{5}$$

Now,

$$\sum_{t=\alpha+1}^{\infty} \frac{1}{(3/2)^{t/\alpha}} \leq \sum_{t=\alpha}^{\infty} \frac{1}{(3/2)^{\lfloor t/\alpha \rfloor}}$$

and grouping α summands and performing an index transformation with $j = t\alpha$, we arrive at

$$\frac{3}{2} \sum_{t=\alpha+1}^{\infty} \frac{1}{(3/2)^{t/\alpha}} \leq \frac{3}{2} \cdot \alpha \sum_{j=1}^{\infty} (2/3)^j = 3\alpha.$$

Plugging this into (5) yields

$$\prod_{t=\alpha+1}^{\infty} \left(1 + \frac{1}{(1+2n\rho)^t - 1}\right) \leq e^{3\alpha}$$

and for the whole product the bound $O\big((4e^3)^\alpha\big) = O\big(3^{1/(n\rho)}\big)$ as $(4e^3)^{1/4} < 3$. □

References

[1] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
[2] S. Droste, T. Jansen, I. Wegener, On the analysis of the $(1 + 1)$ evolutionary algorithm, Theoretical Computer Science 276 (2002) 51–81.
[3] O. Giel, I. Wegener, Evolutionary algorithms and the maximum matching problem, in: H. Alt, M. Habib (Eds.), Annual Symposium on Theoretical Aspects of Computer Science, STACS, in: Lecture Notes in Computer Science, vol. 2607, Springer, 2003, pp. 415–426.
[4] F. Neumann, I. Wegener, Randomized local search, evolutionary algorithms, and the minimum spanning tree problem, Theoretical Computer Science 378 (2007) 32–40.
[5] C. Witt, Worst-case and average-case approximations by simple randomized search heuristics, in: V. Diekert, B. Durand (Eds.), Annual Symposium on Theoretical Aspects of Computer Science, STACS, in: Lecture Notes in Computer Science, vol. 3404, Springer, 2005, pp. 44–56.
[6] B. Doerr, N. Hebbinghaus, F. Neumann, Speeding up evolutionary algorithms through asymmetric mutation operators, Evolutionary Computation 15 (2007) 401–410.
[7] D. Sudholt, Crossover is provably essential for the Ising model on trees, in: Genetic and Evolutionary Computation Conference, GECCO, ACM Press, 2005, pp. 1161–1167.
[8] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004.
[9] W.J. Gutjahr, A generalized convergence result for the graph-based ant system metaheuristic, Probability in the Engineering and Informational Sciences 17 (2003) 545–569.
[10] D. Merkle, M. Middendorf, Modelling the dynamics of Ant Colony Optimization algorithms, Evolutionary Computation 10 (2002) 235–262.
[11] M. Dorigo, C. Blum, Ant colony optimization theory: a survey, Theoretical Computer Science 344 (2005) 243–278.
[12] W.J. Gutjahr, First steps to the runtime complexity analysis of ant colony optimization, Computers and Operations Research 35 (2008) 2711–2727.
[13] F. Neumann, C. Witt, Runtime analysis of a simple ant colony optimization algorithm, Algorithmica 54 (2009) 243–255.
[14] F. Neumann, C. Witt, Ant colony optimization and the minimum spanning tree problem, Theoretical Computer Science 411 (2010) 2406–2413.
[15] T. Kötzing, F. Neumann, H. Röglin, C. Witt, Theoretical properties of two ACO approaches for the traveling salesman problem, in: Proceedings of the Seventh International Conference on Swarm Intelligence, ANTS 2010, 2010.
[16] T. Stützle, H.H. Hoos, MAX–MIN ant system, Journal of Future Generation Computer Systems 16 (2000) 889–914.
[17] W.J. Gutjahr, G. Sebastiani, Runtime analysis of ant colony optimization with best-so-far reinforcement, Methodology and Computing in Applied Probability 10 (2008) 409–433.
[18] N. Attiratanasunthron, J. Fakcharoenphol, A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs, Information Processing Letters 105 (2008) 88–92.
[19] B. Doerr, D. Johannsen, Refined runtime analysis of a basic ant colony optimization algorithm, in: Congress on Evolutionary Computation, CEC, IEEE Press, 2007, pp. 501–507.
[20] B. Doerr, D. Johannsen, C.H. Tang, How single ant ACO systems optimize pseudo-Boolean functions, in: G. Rudolph, T. Jansen, S.M. Lucas, C. Poloni, N. Beume (Eds.), Parallel Problem Solving from Nature—PPSN X, in: Lecture Notes in Computer Science, vol. 5199, Springer, 2008, pp. 378–388.
[21] W.J. Gutjahr, Mathematical runtime analysis of ACO algorithms: survey on an emerging issue, Swarm Intelligence 1 (2007) 59–79.
[22] F. Neumann, C. Witt, Ant colony optimization and the minimum spanning tree problem, Theoretical Computer Science 411 (2010) 2406–2413.
[23] F. Neumann, D. Sudholt, C. Witt, Analysis of different MMAS ACO algorithms on unimodal functions and plateaus, Swarm Intelligence 3 (2009) 35–68.
[24] B. Doerr, F. Neumann, D. Sudholt, C. Witt, On the runtime analysis of the 1-ANT ACO algorithm, in: Genetic and Evolutionary Computation Conference, GECCO, ACM, 2007, pp. 33–40.
[25] G. Rudolph, Convergence Properties of Evolutionary Algorithms, Verlag Dr. Kovač, 1997.
[26] T. Jansen, K.A. De Jong, I. Wegener, On the choice of the offspring population size in evolutionary algorithms, Evolutionary Computation 13 (2005) 413–440.