Electronic Colloquium on Computational Complexity, Report No. 143 (2006)

Ant Colony Optimization and the Minimum Spanning Tree Problem

Frank Neumann
Algorithms and Complexity,
Max-Planck-Institut für Informatik,
66123 Saarbrücken, Germany
fne@mpi-inf.mpg.de

Carsten Witt[1]
FB Informatik, LS 2,
Universität Dortmund,
44221 Dortmund, Germany
carsten.witt@cs.uni-dortmund.de

November 15, 2006

Abstract

Ant Colony Optimization (ACO) is a kind of randomized search heuristic that has become very popular for solving problems from combinatorial optimization. Solutions for a given problem are constructed by a random walk on a so-called construction graph. This random walk can be influenced by heuristic information about the problem. In contrast to many successful applications, the theoretical foundation of this kind of randomized search heuristic is rather weak. Theoretical investigations with respect to the runtime behavior of ACO algorithms have been started only recently for the optimization of pseudo-boolean functions.

We present the first comprehensive rigorous analysis of a simple ACO algorithms for a combinatorial optimization problem. In our investigations we consider the minimum spanning tree problem and examine the effect of two construction graphs with respect to the runtime behavior. The choice of the construction graph in an ACO algorithm seems to be crucial for the success of such an algorithm. First, we take the input graph itself as the construction graph and analyze the use of a construction procedure that is similar to Broder's algorithm [1] for choosing a spanning tree uniformly at random. After that, a more incremental construction procedure is analyzed. It turns out that this procedure is superior to the Broder-based algorithm and produces additionally in a constant number of iterations a minimum spanning tree if the influence of the heuristic information is large enough.

[1]Financial support by the Deutsche Forschungsgemeinschaft (SFB) in terms of the Collaborative Research Center "Computational Intelligence" (SFB 531) is gratefully acknowledged.

ISSN 1433-8092

1 Introduction

Using Ant Colony Optimization (ACO) algorithms to obtain good solutions for combinatorial optimization problems has become very popular in recent years. In contrast to other kinds of randomized search heuristis such as Simulated Annealing or evolutionary algorithms, ACO algorithms have the ability to integrate knowledge about the problem into the construction of a new solution. In the case of a new combinatorial optimization problem, there is often some knowledge about the problem which can be incorporated into this kind of randomized search heuristic. Therefore, the main application of ACO algorithms lies in the field of combinatorial optimization and the first problem to which this kind of heuristic has been applied was the traveling salesperson problem [4]. ACO is inspired by a colony of ants that search for a common source of food. It has been observed that ants are able to find a shortest path to such a source under certain circumstances by indirect communication. This communication is done by so-called pheromone values. The behavior of ants is put into an algorithmic framework to obtain solutions for a given problem. Solutions are constructed by random walks of artificial ants on a so-called construction graph, which has weights – the pheromone values – on the edges. Larger pheromone values lead to higher probability of being traversed in the next walk. In addition, the random walk is usually influenced by heuristic information about the problem.

In contrast to successful applications, the theoretical foundation of the mentioned search heuristics is still in its infancy. A lot of applications show their practical evidence, but for a long time they were not analyzed with respect to their runtime or approximation qualities. We concentrate on the analysis of such heuristics with respect to their runtime behavior in a similar fashion what is usually done for randomized algorithms. In this case, either the expected optimization time, which equals the number of constructed solutions until an optimal one has been obtained, or the success probability after a certain number of steps is analyzed.

The first result with respect to the runtime of a simple ACO algorithm called 1-ANT has been obtained by Neumann and Witt [10]. They have shown that the 1-ANT for the optimization of pseudo-Boolean functions behaves as a well-known evolutionary algorithm called (1+1) EA. Many combinatorial optimization problems can be considered as the optimization of a specific pseudo-boolean function. Especially in the case of polynomially solvable problems, we can not hope that more or less general search heuristics outperform the best-known algorithms for a specific problem. Nevertheless, it is interesting to analyze them on such problems as this shows how the heuristics work and therefore improve the understanding of these, in practice successful, algorithms. The (1+1) EA has been considered for a wide class of combinatorial optimization problems in the context of optimizing a pseudo-Boolean function. All results with respect to the (1+1) EA transfer to the 1-ANT in this context. This includes runtime bounds on some of the best-known polynomially solvable combinatorial optimization problems such as maximum matching, and the minimum spanning tree problem. In the case of NP-hard problems, the result of Witt [13] on the partition problem transfers to the 1-ANT.

In this paper, we conduct a first comprehensive runtime analysis of ACO algorithms on a combinatorial optimization problem. We have chosen the well-known minimum spanning tree (MST) problem as a promising starting point since different randomized search heuristics, in particular the (1+1) EA, have been studied w. r. t. this problem before, e. g., by Neumann and Wegener [8,9] and Wegener [11]. Due to [10] and the result on the (1+1) EA in [8], the expected optimization time of the 1-ANT for the MST problem is $O(m^2(\log n + \log w_{\max}))$, where $w_{\max}$ is the largest weight of the input. In addition, a class of instances with polynomial weights has been presented in [8] where the expected time to obtain on optimal solution is $\Theta(n^4 \log n)$.

It is widely assumed and observed in experiments that the choice of the construction graph has a great effect on the runtime behavior of an ACO algorithm. The construction graph used by Neumann and Witt [10] is a general one for the optimization of pseudo-Boolean functions, which does not take knowledge about the given graph into account. ACO algorithms have the advantage that more knowledge about the structure of a given problem can be incorporated into the construction of solutions. This is done by choosing an appropriate construction graph together with a procedure which allows to obtain feasible solutions. The choice of such a construction graph together with its procedure has been observed experimentally as a crucial point for the success of such an algorithm.

We examine ACO algorithms that work on a construction graphs which seem to be more suitable for the considered problem. First, we consider a random walk on the input graph to construct solutions for the MST problem. It is well known how to choose a spanning tree of a given graph uniformly at random using random walk algorithms (see e. g. [1] [12]). Our construction procedure produces solutions by a variant of Broder's algorithm [1]. We show a polynomial, but relatively large, upper bound for obtaining a minimum spanning tree by this procedure if no heuristic information influences the random walk. Using only heuristic information for constructing solutions, we show that the 1-ANT together with the Broder-based construction procedure with high probability does not find a minimum spanning tree or even does not present a feasible solution in polynomial time.

After that, we consider a more incremental construction procedure that follows a general approach proposed by Dorigo and Stützle [5] to obtain an ACO construction graph. We call this the Kruskal-based construction procedure as in each step an edge that does not create a cycle is chosen to be included into the solution. It turns out that the expected optimization time of the 1-ANT using the Kruskal-based construction procedure is $O(mn(\log n + \log w_{\max}))$. This beats the 1-ANT in the case that the minimum spanning tree problem is more generally modeled as an optimization problem of a special pseudo-boolean function since the lower bound for this approach is $\Omega(n^4 \log n)$ for a special graph with polynomial weights. Using the 1-ANT together with the Kruskal-based construction procedure and a large influence of the heuristic information, the algorithm has even a constant expected optimization time. All our analyses show that and how ACO algorithms for combinatorial optimization can be analyzed rigorously using the toolbox from the analyses of randomized algorithms. In particular, we provide insight into the working principles of ACO algorithms by studying the effect of the (guided) random walks that these algorithms perform.

After having motivated our work, we introduce the model of the minimum spanning tree problem and the 1-ANT in Section 2. In Section 3, we consider a construction procedure which is influenced by Broder's algorithm and consider its effect with respect to the runtime behavior. Section 4 deals with the analysis of the 1-ANT using the Kruskal-based construction graph. We finish with conclusions and the discussion of some open problems.

2 Minimum spanning trees and the 1-ANT

Throughout the paper, we consider the well-known MST problem. Given an undirected graph $G = (V, E)$ with edge costs (weights) $w\colon E \to \mathbb{N}_{\geq 1}$, the goal is to find a spanning tree $E^* \subseteq E$ such that the total cost $\sum_{e \in E^*} w(e)$ becomes minimal. Denote $n := |V|$ and $m := |E|$ and assume w. l. o. g. that $E := \{1, \ldots, m\}$. Moreover, let $m \geq n$ since an existing spanning tree is unique if $m = n - 1$. The MST problem can be solved in time $O(m \log n)$ or $O(n^2)$ using the Greedy algorithms by Kruskal respectively Prim, see, e. g., [2].

We study the simple ACO algorithm called 1-ANT (see Algorithm 1), already analyzed in [10] for the optimization of pseudo-boolean functions. In the 1-ANT, solutions are constructed iteratively by different construction procedures on a given directed construction graph $C = (X, A)$. In the initialization step, each edge $(u, v) \in A$ gets a pheromone value $\tau_{(u,v)} = 1/|A|$ such that the pheromone values sum up to 1. Afterwards, an initial solution x^* is produced by a random walk of an imaginary ant on the construction graph and the pheromone values are updated with respected to this walk. In each iteration, a new solution is constructed and the pheromone values are updated if this solution is not inferior (w. r. t. a fitness function f) to the best solution obtained so far.

Algorithm 1 (1-ANT)

1.) Set $\tau_{(u,v)} = 1/|A|$ for all $(u, v) \in A$.

2.) Compute a solution x using a construction procedure.

3.) Update the pheromone values and set $x^* := x$.

4.) Compute x using a construction procedure.

5.) If $f(x) \leq f(x^*)$, update the pheromone values and set $x^* := x$.

6.) Go to 4.).

We analyze the influence of different construction procedures on the runtime behavior of the 1-ANT algorithm. This is done by considering the expected number of solutions that are constructed by the algorithm until a minimum spanning tree has been obtained for the first time. We call this the *expected optimization time* of the 1-ANT.

3 Broder-based construction graph

Since the MST problem is a graph problem, the first idea is to use the input graph G to the MST problem itself as the construction graph C of the 1-ANT. (Note that each undirected edge $\{u, v\}$ can be considered as two directed edges (u, v) and (v, u).) However, it is not obvious how a random walk of an ant on G is translated into a spanning tree. Interestingly, the famous algorithm of Broder [1], which chooses uniformly at random from all spanning trees of G, is a random walk algorithm. We will use an ACO variant as given in Algorithm 2 of Broder's algorithm. As usual in ACO algorithms, the construction procedure maintains pheromone values τ and heuristic information η for all edges of the construction graph G. Considering the MST problem, we assume that the heuristic information $\eta_{\{u,v\}}$ of an edge $\{u, v\}$ is the inverse of the weight of the edge $\{u, v\}$ in G. α and β are parameters that control the extent to which pheromone values respectively heuristic information is used.

Algorithm 2 (BroderConstruct(G, τ, η))

1.) Choose an arbitrary node $s \in V$.

2.) $u := s$, $T = \emptyset$

3.) Let $R := \sum_{\{u,v\} \in E} [\tau_{\{u,v\}}]^\alpha \cdot [\eta_{\{u,v\}}]^\beta$.

4.) Choose one neighbor v of u where the probability of selection of any fixed v is $\frac{[\tau_{\{u,v\}}]^\alpha \cdot [\eta_{\{u,v\}}]^\beta}{R}$.

5.) If v has not been visited before, set $T := T \cup \{u, v\}$.

6.) Set $u := v$.

7.) If each node of G has been visited return T, otherwise go to 3.)

Obviously, Algorithm 2 outputs a spanning tree T whose cost $f(T)$ is measured by the sum of the w-values of its edges. After a new solution has been accepted, the pheromone values τ are updated w. r. t. the constructed spanning tree T. We maintain upper and lower bounds on these values, which is a common measure to ensure convergence [3] and was also proposed in the previous runtime analysis of the 1-ANT [10]. We assume that after each update, the τ-value of each edge in the construction graph attains either the upper bound h or lower bound ℓ. Hence, for the new pheromone values τ' after an update, it holds that

$$\tau'_{\{u,v\}} = h \quad \text{if} \quad \{u, v\} \in T$$

and

$$\tau'_{\{u,v\}} = \ell \quad \text{if} \quad \{u, v\} \notin T.$$

So the last constructed solution is indirectly saved by the $n-1$ undirected edges that obtain the high pheromone value h. The ratio of the parameters ℓ and h is crucial since too large values of ℓ will lead to too large changes of the tree in subsequent steps whereas too large values of h will make changes of the tree too unlikely. We choose h and l such that $h = n^3\ell$ holds and will argue later on the optimality of this choice.

Note that choosing $\beta = 0$ or $\alpha = 0$ in Algorithm 2, only the pheromone value respectively the heuristic information influence the random walk. We examine the cases where one of these values is 0 to study the effect of the pheromone values respectively the heuristic information separately. First, we consider the case $\alpha = 1$ and $\beta = 0$ for the Broder-based construction graph. This has the following consequences. Let u be the current node of the random walk and denote by $R := \sum_{\{u,\cdot\}} \tau_{\{u,\cdot\}}$ the sum over the pheromone values of all edges that are incident on u. Then the next node is chosen proportionally to the pheromone values on the corresponding edges, which means that a neighbor v of u is chosen with probability $\tau_{\{u,v\}}/R$.

For simplicity, we call the described setting of α, β, h and ℓ the *cubic update scheme*. To become acquainted therewith, we derive the following simple estimations on the probabilities of traversing edges depending on the pheromone values. Assume that a node v has k adjacent edges with value h and i adjacent edges with value ℓ. Note that $k + i \leq n - 1$ and $h = n^3\ell$. Then the probability of choosing an edge with value h is

$$\frac{kh}{kh + i\ell} \;=\; 1 - \frac{i}{kn^3 + i} \;\geq\; 1 - \frac{1}{n^2},$$

where among the edges with values h one edge is chosen uniformly at random. The probability of choosing a specific edge with value ℓ is at least

$$\frac{\ell}{\ell + (n-2)h} \;\geq\; \frac{\ell}{nh} \;\geq\; \frac{1}{n^4}.$$

This leads us to the following theorem, which shows that the 1-ANT in the described setting is able to construct MSTs in expected polynomial time.

Theorem 1 *The expected optimization time of the 1-ANT using the procedure* BroderConstruct *with cubic update scheme is $O(n^6(\log n + \log w_{\max}))$. The expected number of traversed edges in a run of* BroderConstruct *is bounded above by $O(n^2)$ except for the initial run, where it is $O(n^3)$.*

Proof We use the following idea for Theorem 2 in [8]. Suppose the spanning tree T^* was constructed in the last accepted solution. Let $T = T^* \setminus \{e\} \cup \{e'\}$ be any spanning tree that is obtained from T^* by including one edge e' and removing another edge e, and let $s(m,n)$ be a lower bound on the probability of producing T from T^* in the next step. Then the expected number of steps until a minimum spanning tree has been obtained is $O(s(m,n)^{-1}(\log n + \log w_{max}))$. To prove the theorem, it therefore suffices to show that the probability of the 1-ANT producing T by the next constructed solution is $\Omega(1/n^6)$.

To simplify our argumentation, we first concentrate on the probability of rediscovering T^* in the next constructed solution. This happens if the ant traverses all edges of T^* in some arbitrary order and no other edges in between, which might require that an edge has to be taken more than once. (This is a pessimistic assumption since newly traversed edges are not necessarily included in the solution.) Hence, we are confronted with the cover time for the tree T^*. The cover time for trees on n nodes in general is bounded above by $2n^2$ [7], i. e., by Markov's inquality, it is at most $4n^2$ with probability at least $1/2$. We can apply this result if no so-called error occurs that an edge with pheromone value ℓ is taken. According to the above calculations, the probabilty of an error is bounded above by $1/n^2$ in a single step of the ant. Hence, there is no error in $O(n^2)$ steps with probability $\Omega(1)$. Therefore, the probability of rediscovering T^* in the next solution (using $O(n^2)$ steps of *BroderConstruct*) is at least $\Omega(1)$. Additionally taking into account the number of steps $O(n^3)$ for the initial solution [1], we have already bounded the expected number of traversed edges in a run of *BroderConstruct.*

To construct T instead of T^*, exactly one error is desired, namely e' has to be traversed instead of e. Consider the ant when it is for the first time on a node on which e' is incident. By the calculations above, the probability of including e' is $\Omega(1/n^4)$. Note that inserting e' into T^* closes a cycle c. Hence, when e' has been included, there may be at most $n-2$ edges of $\tilde{T} := T^* \setminus \{e\}$ left to traverse. We partition the edges of the forest $\tilde{T}$ into two subsets: The edges that belong to the cycle c are called critical and the remaining ones are called uncritical. The order of inclusion for the uncritical edges is irrelevant. However, all critical edges have to be included before the ant traverses e.

We are faced with the following problem: Let $v_1, \ldots, v_k, v_1$ describe the cycle c and suppose w. l. o. g. that $e' = \{v_1, v_k\}$. It holds that $e = \{v_i, v_{i+1}\}$ for some $1 \leq i \leq k-1$. Moreover, let v_s be the node of c that is visited first by the ant. W. l. o. g., $1 \leq s \leq i$. With probability $\Omega(1/n^4)$, the edge e' is traversed exactly once until a new solution has been constructed. Hence, after e' has been taken, the ant must visit the nodes $v_k, v_{k-1}, \ldots, v_{i+1}$ in the described order (unless an error other than including e' occurs), possibly traversing uncritical edges in between. To ensure that e is traversed before, we would like the ant to visit all the nodes in $\{v_2, \ldots, v_i\}$, without visiting nodes in $\{v_{i+1}, \ldots, v_k\}$, before visiting v_1 and subsequently traversing e'. We apply results on the Gambler's Ruin Problem [6]. The probability of going from v_s to v_i before visiting v_1 is at least $\Omega(1/n)$. The same lower bound holds on the probability of going from v_i to v_1 before visiting v_{i+1}. These random walks are still completed in expected time $O(n^2)$. Hence, in total, the probability of constructing T is $\Omega((1/n^4) \cdot (1/n) \cdot (1/n)) = \Omega(1/n^6)$ as suggested. □

We see that the ratio $h/\ell = n^3$ leads to relatively high exponents in the expected optimization time. However, this ratio seems to be necessary for our argumentation. Consider the complete graph on n nodes where the spanning tree T^* equals a path of length $n-1$. The cover time for this special tree T^* is bounded below by $\Omega(n^2)$. To each node of the path, at most 2 edges with value h and at least $n-3$ edges with value ℓ are incident. Hence, the ratio is required to obtain an error probability of $O(1/n^2)$. It is much more difficult to improve the upper bound of Theorem 1 or to come up with a matching lower bound. The reasons are twofold. First, we cannot

control the effects of steps where the ant traverses edges to nodes that have been visited before in the construction step. These steps might reduce the time until certain edges of T^* are reached. Second, our argumentation concerning the cycle $v_1, \ldots, v_k, v_1$ makes a worst-case assumption on the starting node v_s. It seems more likely that v_s is uniform over the path, which could improve the upper bound of the theorem by a factor $\Omega(n)$. However, a formal proof of this is open.

ACO algorithms often use heuristic information to direct the search process. In the following, we set $\alpha = 0$ and examine the effect of heuristic information for the MST problem. Recall that the heuristic information for an edge e is given by $\eta(e) = 1/w(e)$. Interestingly, for the obvious Broder-based graph, heuristic information alone does not help to find MSTs in reasonable time regardless of β. On the following example graph G^*, either the runtime of *BroderConstruct* explodes or MSTs are found only with exponentially small probability. W. l. o. g., $n = 4k + 1$. Then G^*, a connected graph on the nodes $\{1, \ldots, n\}$, consists of k triangles with weights $(1, 1, 2)$ and two paths of length k with exponentially increasing weights along the path. More precisely, let

$$T^* := \bigcup_{i=1}^{k} \big\{\{1, 2i\}, \{1, 2i + 1\}, \{2i, 2i + 1\}\big\},$$

where $w(\{1, 2i\}) = w(\{2i, 2i + 1\}) := 1$ and $w(\{1, 2i + 1\}) := 2$. Moreover, denote

$$P_1^* := \{1, 2k + 2\} \cup \bigcup_{i=2}^{k} \big\{2k + i, 2k + i + 1\big\},$$

where $w(\{1, 2k + 2\}) := 2$ and $w(\{2k + i, 2k + i + 1\}) := 2^i$, and, similarly,

$$P_2^* := \{1, 3k + 2\} \cup \bigcup_{i=2}^{k} \big\{3k + i, 3k + i + 1\big\},$$

where $w(\{1, 3k+2\}) := 2$ and $w(\{3k+i, 3k+i+1\}) := 2^i$. Finally, the edge set of G^* is $T^* \cup P_1^* \cup P_2^*$. Hence, all triangles and one end of each path are glued by node 1.

Theorem 2 *Choosing $\alpha = 0$ and β arbitrarily, the probability that the 1-ANT using* BroderConstruct *finds an MST for G^*, or the probability of termination within polynomial time is $2^{-\Omega(n)}$.*

Proof Regardless of the ant's starting point, at least one path, w. l. o. g. P_1^*, must be traversed from 1 to its other end, and for least $k - 1$ triangles, both nodes $2i$ and $2i + 1$ must be visited through node 1. For each of these initially undiscovered triangles, the first move into the triangle must go from 1 to $2i$, otherwise the resulting tree will not be minimal. If the triangle is entered at node $2i$, we consider it a success, otherwise (entrance at $2i + 1$) an error. The proof idea is to show that for too small β, i. e., when the influence of heuristic information is low, with overwhelming probability at least one triangle contains an error. If, on the other hand, β is too large, the ant with overwhelming probability will not be able to traverse P_1^* in polynomial time due to its exponentially increasing edge weights.

We study the success probabilities for the triangles and the path P_1. Given that the ant moves from 1 to either $2i$ or $2i + 1$, the probability of going to $2i$ equals

$$\frac{(\eta(\{1, 2i\}))^\beta}{(\eta(\{1, 2i\}))^\beta + (\eta(\{1, 2i + 1\}))^\beta} = \frac{1}{1 + 2^{-\beta}}$$

since $\eta(e) = 1/w(e)$. Therefore, the probability of $k-1$ successes equals, due to independence, $(1+2^{-\beta})^{-k+1}$. This probability increases with β. However, for $\beta \leq 1$, it is still bounded above by $(2/3)^{k-1} = 2^{-\Omega(n)}$.

Considering the path P_1^*, we are faced with the Gambler's Ruin Problem. At each of the nodes $2k+i$, $2 \leq i \leq k-1$, the probability of going to a lower-numbered node and the probability of going to a higher-numbered have the same ratio of $r := (2^{-i+1})^\beta/(2^i)^\beta = 2^\beta$. Hence, starting in $2k+2$, the probability of reaching $3k+1$ before returning to 1 equals (see [6])

$$\frac{r}{r^k - 1} = \frac{2^\beta}{2^{k\beta} - 1}.$$

This probability decreases with β. However, for $\beta \geq 1$, it is still bounded above by $2/(2^k - 1) = 2^{-\Omega(n)}$. Then the probability of reaching the end in a polynomial number of trials is also $2^{-\Omega(n)}$. □

4 A Kruskal-based construction procedure

Dorigo and Stützle [5] state a general approach how to obtain an ACO construction graph from any combinatorial optimization algorithm. The idea is to identify the so-called components of the problem, which may be objects, binary variables etc., with nodes of the construction graph and to allow the ant to choose from these components by moving to the corresponding nodes. In our setting, the components to choose from are the edges from the edge set $\{1, \ldots, m\}$ of the input graph G. Hence, the canonical construction graph $C(G)$ for the MST problem is a directed graph on the $m+1$ nodes $\{0, 1, \ldots, m\}$ with the designated start node $s := 0$. Its edge set A of cardinality m^2 is given by

$$A := \big\{(i,j) \mid 0 \leq i \leq m,\ 1 \leq j \leq m,\ i \neq j\big\},$$

i. e., $C(G)$ is obtained from the complete directed graph by removing all self-loops and the edges pointing to s. When the 1-ANT visits node e in the construction graph $C(G)$, this corresponds to choosing the edge e for a spanning tree. To ensure that a walk of the 1-ANT actually constructs a tree, we define the feasible neigborhood $N(v_k)$ of node v_k depending on the nodes $v_1, \ldots, v_k$ visited so far:

$$\begin{aligned} N(v_k) := \big(&E \setminus \{v_1, \ldots, v_k\}\big) \\ &\setminus \big\{e \in E \;\big|\; \big(V, \{v_1, \ldots, v_k, e\}\big) \text{ contains a cycle}\big\}. \end{aligned}$$

Note that the feasible neighborhood depends on the memory of the ant about the path followed so far, which is very common in ACO algorithms, see, e. g., [5].

A new solution is constructed using Algorithm 3. Again, the random walk of an ant is controlled by the pheromone values τ and the heuristic information η on the edges. Similarly to the Broder-based construction graph, we assume that the $\eta_{(u,v)}$-value of an edge (u,v) is the inverse of the weight of the edge of G corresponding to the node v in $C(G)$.

Algorithm 3 (Construct($C(G), \tau, \eta$))

1.) $v_0 := s$; $k := 0$.

2.) While $N(v_k)$ *is nonempty:*

a.) Let $R := \sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha \cdot [\eta_{(v_k,y)}]^\beta$.

b.) Choose one neighbor v_{k+1} of v_k where the probability of selection of any fixed $y \in N(v_k)$ is $\frac{[\tau_{(v_k,y)}]^\alpha \cdot [\eta_{(v_k,y)}]^\beta}{R}$.

c.) Set $k := k+1$ and go to 2.).

3.) Return the path $p = (v_0, \ldots, v_k)$ constructed by this procedure.

A run of Algorithm 3 returns a sequence of $k+1$ nodes of $C(G)$. It is easy to see that $k := n-1$ after the run, hence the number of steps is bounded above by n, and that $v_1, \ldots, v_{n-1}$ is a sequence of edges that form a spanning tree for G. Accordingly, we measure the fitness $f(p)$ of a path $p = (v_0, \ldots, v_{n-1})$ simply by $w(v_1) + \cdots + w(v_{n-1})$, i. e., the cost of the corresponding spanning tree. It remains to specify the update scheme for the pheromone values. As in the case of the Broder-based construction procedure, we only consider two different values h and ℓ. To allow the ant to rediscover the edges of the previous spanning tree equiprobably in each order, we reward all edges pointing to nodes from p except s, i. e., we reward $(m+1)(n-1)$ edges. Hence, the τ'-values are

$$\tau'_{(u,v)} = h \qquad \text{if } v \in p \text{ and } v \neq s$$

and

$$\tau'_{(u,v)} = \ell \qquad \text{otherwise.}$$

We choose h and ℓ such that $h = (m-n+1)(\log n)\ell$ holds. In this case, the probability of taking a rewarded edge (if applicable) is always at least $1 - 1/\log n$.

We first consider the case where the random walk to construct solutions is only influenced by the pheromone values on the edges of $C(G)$.

Theorem 3 *Choosing $\alpha = 1$ and $\beta = 0$, the expected optimization time of the 1-ANT with construction graph $C(G)$ is bounded by $O(mn(\log n + \log w_{\max}))$.*

Proof Again we use the proof idea for Theorem 2 in [8]. It suffices to show the following claim. Suppose the 1-ANT has constructed the spanning tree T^* in the last accepted solution. Let $T = T^* \setminus \{e\} \cup \{e'\}$ be any spanning tree that is obtained from T^* by including one edge e' and removing another edge e. Then the probability of producing T by the next constructed solution is $\Omega(1/(nm))$.

Let $e_1, \ldots, e_{n-1}$ be the edges of T^* and suppose w. l. o. g. that the edges of T are $e_1, \ldots, e_{n-2}, e'$ where $e' \neq e_i$ for $1 \leq i \leq n-1$. We show that with probability $\Omega(1)$, exactly $n-2$ (but not $n-1$) out of the $n-1$ nodes visited by the 1-ANT in $C(G)$ form a uniformly random subset of $\{e_1, \ldots, e_{n-1}\}$. Hence, e_{n-1} is missing with probability $1/(n-1)$. Furthermore, we will show that the probability of visiting e' rather than e_{n-1} as the missing node has probability at least $\Omega(1/m)$. Hence, in total, T is constructed with probability $\Omega(1/(nm))$.

We still have to prove the statements on the probabilities in detail. We study the events E_i, $1 \leq i \leq n-1$, defined as follows. E_i occurs iff the first $i-1$ and the last $n-i-1$ nodes visited by the 1-ANT (excluding s) correspond to edges of T^* whereas the i-th one does not. Edges in $C(G)$ pointing to nodes of T^* have pheromone value h and all remaining edges have value ℓ. Hence, if $j-1$ edges of T^* have been found, the probability of not choosing another edge of T^* by the next node visited in $C(G)$ is at most

$$\frac{(m-(n-1))\ell}{((n-1)-(j-1))h} = \frac{1}{(n-j)\log n}.$$

Therefore, the first $i-1$ and last $n-i-1$ nodes (excluding s) visited correspond to edges of T^* with probability at least

$$1 - \sum_{\substack{j=1 \\ j \neq i}}^{n-1} \frac{1}{(n-j)\log n} \;\geq\; 1 - \frac{(\ln(n-1)+1)}{\log n} + \frac{1}{\log n}$$

$$\geq\; 1 - \frac{\ln n}{\log n} \;=\; \Omega(1)$$

(estimating the $(n-1)$-th Harmonic number by $\ln(n-1)+1$) and, due to the symmetry of the update scheme, each subset of T^* of size $n-2$ is equally likely, i. e., has probability $\Omega(1/n)$. Additionally, the probability of choosing by the i-th visited node an edge e' not contained in T^* equals

$$\frac{\ell}{(n-i)h + k\ell} \;\geq\; \frac{1}{(n-i+1)(m-n+1)\log n},$$

where k is the number of edges outside T^* that can still be chosen; note that $k\ell \leq h$. Hence, with probability at least $c/((n-i+1)mn\log n)$ for some small enough constant c (and large enough n), E_i occurs and the tree T is constructed. Since the E_i are mutually disjoint events, T is constructed instead of T^* with probability at least

$$\sum_{i=1}^{n-1} \frac{c}{(n-i+1)mn\log n} \;=\; \Omega(1/(mn))$$

as suggested. □

In the following, we examine the use of heuristic information for the Kruskal-based construction graph. Here it can be proven that strong heuristic information helps the 1-ANT mimicking the greedy algorithm by Kruskal.

Theorem 4 *Choosing $\alpha = 0$ and $\beta \geq 6w_{\max}\log n$, the expected optimization time of the 1-ANT using the construction graph $C(G)$ is constant.*

Proof We show that the next solution which the 1-ANT constructs is with probability at least $1/e$ a minimum spanning tree, where e is Euler's number. This implies that the expected number of solutions that have to be constructed until a minimum spanning tree has been computed is bounded above by e.

Let $(w_1, w_2, \ldots, w_{n-1})$ the weights of edges of a minimum spanning tree. Let $w_i \leq w_{i+1}$, $1 \leq i \leq n-2$ and asumme that the ant has already included $i-1$ edges that have weights $w_1, \ldots, w_{i-1}$ and consider the probability of choosing an edge of weight w_i in the next step. Let $M = \{e_1, \ldots, e_r\}$ be the set of edges that can be included without creating a cycle and denote by $M_i = \{e_1, \ldots, e_s\}$ the subset of M that includes all edges of weight w_i. W. l. o. g. we assume $w(e_i) \leq w(e_{i+1})$, $1 \leq i \leq r-1$.

The probability of choosing an edge of M_i in the next step is given by

$$\frac{\sum_{k=1}^{s}(\eta(e_k))^\beta}{\sum_{l=1}^{r}(\eta(e_l))^\beta} = \frac{\sum_{k=1}^{s}(\eta(e_k))^\beta}{\sum_{l=1}^{s}(\eta(e_l))^\beta + \sum_{l=s+1}^{r}(\eta(e_l))^\beta},$$

where $\eta(e_j) = 1/w(e_j)$ holds. Let $a = \sum_{k=1}^{s}(\eta(e_i))^\beta = \sum_{k=1}^{s}(1/w_i)^\beta$ and $b = \sum_{l=s+1}^{r}(\eta(e_l))^\beta$. The probability of choosing an edge of weight w_i is $a/(a+b)$, which is at least $1-1/n$ if $b \leq a/n$. The

number of edges in $M \setminus M_i$ is bounded above by m, and the weight of such an edge is at least $w_i + 1$. Hence, $b \leq m \cdot (1/(w_i + 1))^\beta$.

We would like $m \cdot (1/(w_i + 1))^\beta \leq s \cdot (1/w_i)^\beta/n$ to hold. This can be achieved by choosing

$$\beta \geq \frac{\log(mn/s)}{\log((w_i + 1)/w_i)} = \frac{\log(mn/s)}{\log(1 + 1/w_i)},$$

which is at most

$$\frac{\log(mn/s)}{w_i/2} \leq 6w_{\max} \log n$$

since $mn \leq n^3$ and $e^x \leq 1 + 2x$ for $0 \leq x \leq 1$. Due to our choices, the ant traverses the edge with weight w_i with probability at least $1 - 1/n$. Therefore, the probability that in every step i such an edge is taken is at least $(1 - 1/n)^{n-1} \geq 1/e$ as suggested. □

Of course, the result of Theorem 4 does not necessarily improve upon Kruskal's algorithm since the computational efforts in a run of the construction algorithm and for initializing suitable random number generators (both of which are assumed constant in our cost measure for the optimization time) must not be neglected. With a careful implementation of the 1-ANT, however, the expected computational effort w. r. t. the well-known uniform cost measure could be at least bounded above by the runtime $O(m \log m)$ of Kruskal's algorithm.

5 Conclusions

ACO algorithms have in particular shown to be successful in solving problems from combinatorial optimization. In contrast to many applications, first theoretical estimations of the runtime of such algorithms for the optimization of pseudo-boolean functions have been obtained only recently. In the case of combinatorial optimization problems, the construction graphs used are more related to the problem at hand. For the first time, the effect of such graphs have been investigated by rigorous runtime analyses. We have considered a simple ACO algorithm 1-ANT for the well-known minimum spanning tree problem. In the case of the Broder-based construction procedure a polynomial, but relatively large, upper bound has been proven. In addition, it has been shown that heuristic information can mislead the algorithm such that an optimal solution is not found within a polynomial number of steps with high probability. In the case of the Kruskal-based construction procedure, the upper bound obtained shows that this construction graph leads to a better optimization process than the 1-ANT and simple evolutionary algorithms in the context of the optimization of pseudo-boolean functions. In addition, a large influence of heuristic information makes the algorithm mimic Kruskal's algorithm for the minimum spanning tree problem. All analyses provide insight into the guided random walks that the 1-ANT performs in order to create solutions of our problem.

There are several interesting open questions concerning ACO algorithms. First, it would be desirable to obtain the expected optimization time for the considered algorithms asymptotically exactly. For the Broder-based construction graph, we have argued why we expect relatively large lower bounds. Nevertheless, a formal proof for that is open. On the other hand, the influence of the pheromone values and the heuristic information has been analyzed only separately. The same bounds should also hold if the effect of one of these parameters is low compared with the other one. But it would be interesting to consider cases where both have a large influence and to obtain bounds in these cases.

References

[1] A. Broder. Generating random spanning trees. In *Proc. of FOCS '89*, pages 442–447, 1989.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2. edition, 2001.

[3] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theor. Comput. Sci.*, 344:243–278, 2005.

[4] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: An autocatalytic optimizing process. Technical Report 91-016 Revised, Politecnico di Milano, 1991.

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

[6] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 3rd edition, 1968.

[7] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[8] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In *Proc. of GECCO '04*, volume 3102 of *LNCS*, pages 713–724, 2004.

[9] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. In *Proc. of GECCO '05*, pages 763–770. ACM Press, 2005.

[10] F. Neumann and C. Witt. Runtime analysis of a simple ant colony optimization algorithm. In *Proc. of ISAAC '06*, volume 4288 of *LNCS*, pages 618–627. Springer, 2006.

[11] I. Wegener. Simulated annealing beats Metropolis in combinatorial optimization. In *Proc. of ICALP '05*, volume 3580 of *LNCS*, pages 589–601, 2005.

[12] D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proc. of STOC '96*, pages 296–303, 1996.

[13] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS '05*, volume 3404 of *LNCS*, pages 44–56, 2005.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/