Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 344 (2005) 243–278
Theoretical Computer Science
www.elsevier.com/locate/tcs

Ant colony optimization theory: A survey

Marco Dorigo[a,*,1], Christian Blum[b,2]

[a] *IRIDIA, Université Libre de Bruxelles, CP 194/6, Ave. F. Roosevelt 50, 1050 Brussels, Belgium*
[b] *ALBCOM, LSI, Universitat Politècnica de Catalunya, Jordi Girona 1-3, Campus Nord, 08034 Barcelona, Spain*

Received 21 March 2005; received in revised form 22 May 2005; accepted 30 May 2005

Communicated by T. Baeck

Abstract

Research on a new metaheuristic for optimization is often initially focused on proof-of-concept applications. It is only after experimental work has shown the practical interest of the method that researchers try to deepen their understanding of the method's functioning not only through more and more sophisticated experiments but also by means of an effort to build a theory. Tackling questions such as "how and why the method works'' is important, because finding an answer may help in improving its applicability. Ant colony optimization, which was introduced in the early 1990s as a novel technique for solving hard combinatorial optimization problems, finds itself currently at this point of its life cycle. With this article we provide a survey on theoretical results on ant colony optimization. First, we review some convergence results. Then we discuss relations between ant colony optimization algorithms and other approximate methods for optimization. Finally, we focus on some research efforts directed at gaining a deeper understanding of the behavior of ant colony optimization algorithms. Throughout the paper we identify some open questions with a certain interest of being solved in the near future.

Keywords: Ant colony optimization; Metaheuristics; Combinatorial optimization; Convergence; Stochastic gradient descent; Model-based search; Approximate algorithms

* Corresponding author. Tel.: +32 2 6503169; fax: +32 2 6502715.
E-mail addresses: mdorigo@ulb.ac.be (M. Dorigo), cblum@lsi.upc.edu (C. Blum).

[1] Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Research Director, and from the "ANTS'' project, an "Action de Recherche Concertée'' funded by the Scientific Research Directorate of the French Community of Belgium.

[2] Christian Blum acknowledges support from the "Juan de la Cierva'' program of the Spanish Ministry of Science and Technology of which he is a post-doctoral research fellow, and from the Spanish CICYT project TRACER (grant TIC-2002-04498-C05-03).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.05.020

1. Introduction

In the early 1990s, ant colony optimization (ACO) [20,22,23] was introduced by M. Dorigo and colleagues as a novel nature-inspired metaheuristic for the solution of hard combinatorial optimization (CO) problems. ACO belongs to the class of metaheuristics [8,32,40], which are approximate algorithms used to obtain good enough solutions to hard CO problems in a reasonable amount of computation time. Other examples of metaheuristics are tabu search [30,31,33], simulated annealing [44,13], and evolutionary computation [39,58,26]. The inspiring source of ACO is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. As it has been shown in [18], indirect communication between the ants via pheromone trails enables them to find shortest paths between their nest and food sources. This characteristic of real ant colonies is exploited in artificial ant colonies in order to solve CO problems.

According to Papadimitriou and Steiglitz [56], a CO problem $\mathcal{P} = (\mathcal{S}, f)$ is an optimization problem in which, given a finite set of solutions $\mathcal{S}$ (also called *search space*) and an objective function $f : \mathcal{S} \mapsto \mathbb{R}^+$ that assigns a positive cost value to each of the solutions, the goal is either to find a solution of minimum cost value,[3] or—as in the case of approximate solution techniques—a good enough solution in a reasonable amount of time. ACO algorithms belong to the class of metaheuristics and therefore follow the latter goal. The central component of an ACO algorithm is a parametrized probabilistic model, which is called the *pheromone model*. The pheromone model consists of a vector of model parameters $\mathcal{T}$ called *pheromone trail parameters*. The pheromone trail parameters $\mathcal{T}_i \in \mathcal{T}$, which are usually associated to components of solutions, have values τ_i, called *pheromone values*. The pheromone model is used to probabilistically generate solutions to the problem under consideration by assembling them from a finite set of solution components. At runtime, ACO algorithms update the pheromone values using previously generated solutions. The update aims to concentrate the search in regions of the search space containing high quality solutions. In particular, the reinforcement of solution components depending on the solution quality is an important ingredient of ACO algorithms. It implicitly assumes that good solutions consist of good solution components.[4] To learn which components contribute to good solutions can help assembling them into better solutions. In general, the ACO approach attempts to solve an optimization problem by repeating the following two steps:

- candidate solutions are constructed using a pheromone model, that is, a parametrized probability distribution over the solution space;
- the candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling toward high quality solutions.

[3] Note that minimizing over an objective function f is the same as maximizing over $-f$. Therefore, every CO problem can be described as a minimization problem.

[4] Note that this does not require the objective function to be (partially) separable. It only requires the existence of a fitness-distance correlation [41].

After the initial proof-of-concept application to the traveling salesman problem (TSP) [22,23], ACO was applied to many other CO problems. [5] Examples are the applications to assignment problems [14,47,46,63,66], scheduling problems [11,17,27,51,64], and vehicle routing problems [29,59]. Among other applications, ACO algorithms are currently state-of-the-art for solving the sequential ordering problem (SOP) [28], the resource constraint project scheduling (RCPS) problem [51], and the open shop scheduling (OSS) problem [4]. For an overview of applications of ACO we refer the interested reader to [24].

The first theoretical problem considered was the one concerning convergence. The question is: will a given ACO algorithm find an optimal solution when given enough resources? This is an interesting question, because ACO algorithms are stochastic search procedures in which the pheromone update could prevent them from ever reaching an optimum. When considering a stochastic optimization algorithm, there are at least two possible types of convergence that can be considered: *convergence in value* and *convergence in solution*. When studying convergence in value, we are interested in evaluating the probability that the algorithm will generate an optimal solution at least once. On the contrary, when studying convergence in solution we are interested in evaluating the probability that the algorithm reaches a state which keeps generating the same optimal solution. The first convergence proofs were presented by Gutjahr in [37,38]. He proved convergence with probability $1-\varepsilon$ to the optimal solution (in [37]), and more in general to any optimal solution (in [38]), of a particular ACO algorithm that he called graph-based ant system (GBAS). Notwithstanding its theoretical interest, the main limitation of this work was that GBAS is quite different from any implemented ACO algorithm and its empirical performance is unknown. A second strand of work on convergence focused therefore on a class of ACO algorithms that are among the best-performing in practice, namely, algorithms that apply a positive lower bound $\tau_{\min}$ to all pheromone values. The lower bound prevents the probability to generate any solution to become zero. This class of algorithms is denoted by $\text{ACO}_{\tau_{\min}}$. Dorigo and Stützle, first in [65] and later in [24], presented a proof for the convergence in value, as well as a proof for the convergence in solution, for algorithms from $\text{ACO}_{\tau_{\min}}$. With the convergence of ACO algorithms we deal in Section 3 of this paper.

Recently, researchers have been dealing with the relation of ACO algorithms to other methods for learning and optimization. One example is the work presented in [2] that relates ACO to the fields of optimal control and reinforcement learning. A more prominent example is the work that aimed at finding similarities between ACO algorithms and other probabilistic learning algorithms such as stochastic gradient ascent (SGA), and the cross-entropy (CE) method. Meuleau and Dorigo have shown in [52] that the pheromone update as outlined in the proof-of-concept application to the TSP [22,23] is very similar to a stochastic gradient ascent in the space of pheromone values. Based on this observation, the authors developed an SGA-based type of ACO algorithm whose pheromone update describes a stochastic gradient ascent. This algorithm can be shown to converge to a local optimum with probability 1. In practice, this SGA-based pheromone update has

[5] Note that the class of ACO algorithms also comprises methods for the application to problems arising in networks, such as routing and load balancing (see, for example, [19]), and for the application to continuous optimization problems (see, for example, [62]). However, in this review we exclusively focus on ACO for solving CO problems.

not been much studied so far. The first implementation of SGA-based ACO algorithms was proposed in [3] where it was shown that SGA-based pheromone updates avoid certain types of search bias. Zlochin et al. [67] have proposed a unifying framework for so-called *model-based search* (MBS) algorithms. An MBS algorithm is characterized by the use of a (parametrized) probabilistic model $M \in \mathcal{M}$ (where $\mathcal{M}$ is the set of all possible probabilistic models) that is used to generate solutions to the problem under consideration. The class of MBS algorithms can be divided into two subclasses with respect to the way the probabilistic model is used. The algorithms in the first subclass use a given probabilistic model without changing the model structure at run-time, whereas the algorithms of the second subclass use and change the probabilistic model in alternating phases. ACO algorithms are examples of algorithms from the first subclass. In this paper we deal with model-based search in Section 4.

While convergence proofs can provide insight into the working of an algorithm, they are usually not very useful to the practitioner that wants to implement efficient algorithms. This is because, generally, either infinite time or infinite space are required for a stochastic optimization algorithm to converge to an optimal solution (or to the optimal solution value). The existing convergence proofs for particular ACO algorithms are no exception. As more relevant for practical applications might be considered the research efforts that were aimed at a better understanding of the behavior of ACO algorithms. Blum [3] and Blum and Dorigo [5,7] made the attempt of capturing the behavior of ACO algorithms in a formal framework. This work is closely related to the notion of *deception* as used in the evolutionary computation field. The term deception was introduced by Goldberg in [34] with the aim of describing problems that are misleading for genetic algorithms (GAs). Well-known examples of GA-deceptive problems are n-bit trap functions [16]. These functions are characterized by (i) fix-points that correspond to sub-optimal solutions and that have large basins of attraction, and (ii) fix-points with relatively small basins of attraction that correspond to optimal solutions. Therefore, for these problems a GA will—in most cases—not find an optimal solution. In [3,5,7], Blum and Dorigo adopted the term deception for the field of ant colony optimization, similarly to what had previously been done in evolutionary computation. It was shown that ant colony optimization algorithms in general suffer from *first order deception* in the same way as GAs suffer from deception. Blum and Dorigo further introduced the concept of *second order deception*, which is caused by a bias that leads to decreasing algorithm performance over time. Among the principal causes for this search bias were identified situations in which some solution components on average receive update from more solutions than others they compete with. This was shown for scheduling problems in [9,10], and for the k-cardinality tree problem in [12]. Recently, Montgomery et al. [54] made an attempt to extend the work by Blum and Sampels [9,10] to assignment problems, and to attribute search bias to different algorithmic components. Merkle and Middendorf [48,49] were the first to study the behavior of a simple ACO algorithm by analyzing the dynamics of its *model*, which is obtained by applying the expected pheromone update. Their work deals with the application of ACO to idealized permutation problems. When applied to constrained problems such as permutation problems, the solution construction process of ACO algorithms consists of a sequence of random decisions in which later decisions depend on earlier ones. Therefore, the later decisions of the construction process are inherently biased by the earlier ones. The work of Merkle and Middendorf

shows that this leads to a bias which they call *selection bias*. Furthermore, the competition between the ants was identified as the main driving force of the algorithm. Some of the principal aspects of the above mentioned works are discussed in Section 5.

Outline. In Section 2 we introduce ACO in a way that suits its theoretical study. For this purpose we take inspiration from the way of describing ACO as done in [6]. We further outline successful ACO variants and introduce the concept of models of ACO algorithms, taking inspiration from [49]. In Section 3 we deal with existing convergence proofs for ACO algorithms, while in Section 4 we present the work on establishing the relation between ACO and other techniques for optimization. In Section 5 we deal with some important aspects of the works on search bias in ACO algorithms. Finally, in Section 6 we draw conclusions and propose an outlook to the future.

2. Ant colony optimization

ACO algorithms are stochastic search procedures. Their central component is the pheromone model, which is used to probabilistically sample the search space. The pheromone model can be derived from a *model* of the tackled CO problem, defined as follows:

Definition 1. A *model* $\mathcal{P} = (\mathcal{S}, \Omega, f)$ of a *CO* problem consists of:
- a *search* (*or solution*) *space* $\mathcal{S}$ defined over a finite set of discrete decision variables and a set Ω of *constraints* among the variables;
- an *objective function* $f : \mathcal{S} \to \mathbb{R}^+$ to be minimized.

The search space $\mathcal{S}$ is defined as follows: Given is a set of *n discrete variables* X_i with values $v_i^j \in D_i = \{v_i^1, \ldots, v_i^{|D_i|}\}$, $i = 1, \ldots, n$. A variable instantiation, that is, the assignment of a value v_i^j to a variable X_i, is denoted by $X_i = v_i^j$. A feasible solution $s \in \mathcal{S}$ is a complete assignment (i.e., an assignment in which each decision variable has a domain value assigned) that satisfies the constraints. If the set of constraints Ω is empty, then each decision variable can take any value from its domain independently of the values of the other decision variables. In this case we call $\mathcal{P}$ an *unconstrained* problem model, otherwise a *constrained* problem model. A feasible solution $s^* \in \mathcal{S}$ is called a *globally optimal solution* (or global optimum), if $f(s^*) \leqslant f(s)\ \forall s \in \mathcal{S}$. The set of globally optimal solutions is denoted by $\mathcal{S}^* \subseteq \mathcal{S}$. To solve a CO problem one has to find a solution $s^* \in \mathcal{S}^*$.

A model of the CO problem under consideration implies a finite set of solution components and a pheromone model as follows. First, we call the combination of a decision variable X_i and one of its domain values v_i^j a *solution component* denoted by c_i^j. Then, the pheromone model consists of a *pheromone trail parameter* $\mathcal{T}_i^j$ for each solution component c_i^j. The set of all solution components is denoted by $\mathfrak{C}$. The value of a pheromone trail parameter $\mathcal{T}_i^j$—called *pheromone value*—is denoted by τ_i^j.[6] The vector of all pheromone

[6] Note that pheromone values are in general a function of the algorithm's iteration t: $\tau_i^j = \tau_i^j(t)$. This dependence on the iteration will however be made explicit only when necessary.

trail parameters is denoted by $\mathcal{T}$. As a CO problem can be modeled in different ways, different models of a CO problem can be used to define different pheromone models.

As an example, we consider the asymmetric traveling salesman problem (ATSP): a completely connected, directed graph $G(V, A)$ with a positive weight d_{ij} associated to each arc $a_{ij} \in A$ is given. The nodes of the graph represent cities and the arc weights represent distances between the cities. The goal consists in finding among all (directed) Hamiltonian cycles in G one for which the sum of the weights of its arcs is minimal, that is, a shortest Hamiltonian cycle. This NP-hard CO problem can be modeled as follows: we model each city $i \in V$ by a decision variable X_i whose domain consists of a domain value v_i^j for each outgoing arc a_{ij}. A variable instantiation $X_i = v_i^j$ means that arc a_{ij} is part of the corresponding solution. The set of constraints must be defined so that only candidate solutions that correspond to Hamiltonian cycles in G are valid solutions. The set of solution components $\mathfrak{C}$ consists of a solution component $\mathfrak{c}_i^j$ for each combination of variable X_i and domain value v_i^j, and the pheromone model $\mathcal{T}$ consists of a pheromone trail parameter $\mathcal{T}_i^j$, with value τ_i^j, associated to each solution component $\mathfrak{c}_i^j$.

2.1. The framework of a basic ACO algorithm

When trying to prove theoretical properties for the ACO metaheuristic, the researcher faces a first major problem: ACO's very general definition. Although generality is a desirable property, it makes theoretical analysis much more complicated, if possible at all. It is for this reason that we introduce ACO in a form that covers all the algorithms that were theoretically studied, but that is not as general as the definition of the ACO metaheuristic as given, for example, in Chapter 2 of [24] (see also footnote 5).

Algorithm 1 captures the framework of a basic ACO algorithm. It works as follows. At each iteration, n_a ants probabilistically construct solutions to the combinatorial optimization problem under consideration, exploiting a given pheromone model. Then, optionally, a local search procedure is applied to the constructed solutions. Finally, before the next iteration starts, some of the solutions are used for performing a pheromone update. This framework is explained with more details in the following.

InitializePheromoneValues($\mathcal{T}$). At the start of the algorithm the pheromone values are all initialized to a constant value $c > 0$.

ConstructSolution($\mathcal{T}$). The basic ingredient of any ACO algorithm is a constructive heuristic for probabilistically constructing solutions. A constructive heuristic assembles solutions as sequences of elements from the finite set of solution components $\mathfrak{C}$. A solution construction starts with an empty partial solution $\mathfrak{s}^p = \langle\rangle$. Then, at each construction step the current partial solution $\mathfrak{s}^p$ is extended by adding a feasible solution component from the set $\mathfrak{N}(\mathfrak{s}^p) \subseteq \mathfrak{C} \setminus \{\mathfrak{s}^p\}$. This set is determined at each construction step by the solution construction mechanism in such a way that the problem constraints are met. The process of constructing solutions can be regarded as a walk (or a path) on the so-called *construction graph* $\mathcal{G}_C = (\mathfrak{C}, \mathfrak{L})$, which is a fully connected graph whose vertices are the solution components in $\mathfrak{C}$ and whose edges are the elements of $\mathfrak{L}$. The allowed walks on $\mathcal{G}_C$ are implicitly

Algorithm 1 The framework of a basic ACO algorithm

input: An instance P of a CO problem model $\mathcal{P} = (\mathcal{S}, f, \Omega)$.
InitializePheromoneValues($\mathcal{T}$)
$\mathfrak{s}_{\text{bs}} \leftarrow$ NULL
while termination conditions not met **do**
  $\mathfrak{S}_{\text{iter}} \leftarrow \emptyset$
  for $j = 1, \ldots, n_a$ **do**
    $\mathfrak{s} \leftarrow$ ConstructSolution($\mathcal{T}$)
    if $\mathfrak{s}$ is a valid solution **then**
      $\mathfrak{s} \leftarrow$ LocalSearch($\mathfrak{s}$) {optional}
      if ($f(\mathfrak{s}) < f(\mathfrak{s}_{\text{bs}})$) or ($\mathfrak{s}_{\text{bs}} =$ NULL) **then** $\mathfrak{s}_{\text{bs}} \leftarrow \mathfrak{s}$
      $\mathfrak{S}_{\text{iter}} \leftarrow \mathfrak{S}_{\text{iter}} \cup \{\mathfrak{s}\}$
    end if
  end for
  ApplyPheromoneUpdate($\mathcal{T}$,$\mathfrak{S}_{\text{iter}}$,$\mathfrak{s}_{\text{bs}}$)
end while
output: The best-so-far solution $\mathfrak{s}_{\text{bs}}$

defined by the solution construction mechanism that defines the set $\mathfrak{N}(\mathfrak{s}^p)$ with respect to a partial solution $\mathfrak{s}^p$. The choice of a solution component $\mathfrak{c}_i^j \in \mathfrak{N}(\mathfrak{s}^p)$ is, at each construction step, done probabilistically with respect to the pheromone model. The probability for the choice of $\mathfrak{c}_i^j$ is proportional to $[\tau_i^j]^\alpha \cdot [\eta(\mathfrak{c}_i^j)]^\beta$, where η is a function that assigns to each valid solution component—possibly depending on the current construction step—a heuristic value which is also called the *heuristic information*. The value of parameters α and β, $\alpha > 0$ and $\beta > 0$, determines the relative importance of pheromone value and heuristic information. The heuristic information is optional, but often needed for achieving a high algorithm performance. In most ACO algorithms the probabilities for choosing the next solution component—also called the *transition probabilities*—are defined as follows:

$$\mathbf{p}(\mathfrak{c}_i^j \mid \mathfrak{s}^p) = \frac{[\tau_i^j]^\alpha \cdot [\eta(\mathfrak{c}_i^j)]^\beta}{\sum_{\mathfrak{c}_k^l \in \mathfrak{N}(\mathfrak{s}^p)} [\tau_k^l]^\alpha \cdot [\eta(\mathfrak{c}_k^l)]^\beta}, \quad \forall\, \mathfrak{c}_i^j \in \mathfrak{N}(\mathfrak{s}^p). \tag{1}$$

Note that potentially there are many different ways of choosing the transition probabilities. The above form has mainly historical reasons, because it was used in the first ACO algorithms [22,23] to be proposed in the literature. In the rest of the paper we assume that the construction of a solution is aborted if $\mathfrak{N}(\mathfrak{s}^p) = \emptyset$ and $\mathfrak{s}$ is not a valid solution.[7]

As an example of this construction mechanism let us consider again the ATSP (see Section 2). Let $\mathcal{I}$ denote the set of indices of the current decision variable and of the decision variables that have already a value assigned. Let i_c denote the index of the current decision variable (i.e., the decision variable that has to be assigned a value in the current construction step). The solution construction starts with an empty partial solution

[7] Alternatively, non-valid solutions might be punished by giving them an objective function value that is higher than the value of any feasible solution.

$\mathfrak{s}^p = \langle\rangle$, with $i_c \in \{1, \ldots, |V|\}$ randomly chosen, and with $\mathcal{I} = \{i_c\}$. Also, the index of the first decision variable is stored in variable i_f (i.e., $i_f \leftarrow i_c$). Then, at each of the $|V| - 1$ construction steps a solution component $\mathfrak{c}_{i_c}^j \in \mathfrak{N}(\mathfrak{s}^p)$ is added to the current partial solution, where $\mathfrak{N}(\mathfrak{s}^p) = \{\mathfrak{c}_{i_c}^k \mid k \in \{1, \ldots, |V|\} \setminus \mathcal{I}\}$. This means that at each construction step a domain value is chosen for the decision variable with index i_c. Once the solution component $\mathfrak{c}_{i_c}^j$ is added to $\mathfrak{s}^p$, i_c is set to j. The transition probabilities used in each of the first $|V| - 1$ construction steps are those of Equation 1, where the heuristic information can, in the case of the ATSP, be defined as $\eta(\mathfrak{c}_i^j) = 1/d_{ij}$ (this choice introduces a bias towards short arcs). The last construction step consists of adding solution component $\mathfrak{c}_{i_c}^{i_f}$ to the partial solution $\mathfrak{s}^p$, which corresponds to closing the Hamiltonian cycle. [8]

LocalSearch($\mathfrak{s}$). A local search procedure may be applied for improving the solutions constructed by the ants. The use of such a procedure is optional, though experimentally it has been observed that, if available, its use improves the algorithm's overall performance.

ApplyPheromoneUpdate($\mathcal{T}$,$\mathfrak{S}_{\text{iter}}$,$\mathfrak{s}_{\text{bs}}$). The aim of the pheromone value update rule is to increase the pheromone values on solution components that have been found in high quality solutions. Most ACO algorithms use a variation of the following update rule:

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \frac{\rho}{\mathfrak{S}_{\text{upd}}}\cdot\sum_{\{\mathfrak{s}\in\mathfrak{S}_{\text{upd}}\mid\mathfrak{c}_i^j\in\mathfrak{s}\}} F(\mathfrak{s}), \tag{2}$$

for $i = 1, \ldots, n$, and $j = 1, \ldots, |D_i|$. Instantiations of this update rule are obtained by different specifications of $\mathfrak{S}_{\text{upd}}$, which—in all the cases that we consider in this paper—is a subset of $\mathfrak{S}_{\text{iter}} \cup \{\mathfrak{s}_{\text{bs}}\}$, where $\mathfrak{S}_{\text{iter}}$ is the set of solutions that were constructed in the current iteration, and $\mathfrak{s}_{\text{bs}}$ is the best-so-far solution. The parameter $\rho \in (0, 1]$ is called *evaporation rate*. It has the function of uniformly decreasing all the pheromone values. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a sub-optimal region. It implements a useful form of *forgetting*, favoring the exploration of new areas in the search space. $F : \mathfrak{S} \mapsto \mathbb{R}^+$ is a function such that $f(\mathfrak{s}) < f(\mathfrak{s}') \Rightarrow +\infty > F(\mathfrak{s}) \geqslant F(\mathfrak{s}')$, $\forall \mathfrak{s} \neq \mathfrak{s}' \in \mathfrak{S}$, where $\mathfrak{S}$ is the set of all the sequences of solution components that may be constructed by the ACO algorithm and that correspond to feasible solutions. $F(\cdot)$ is commonly called the *quality function*. Note that the factor $1/\mathfrak{S}_{\text{upd}}$ is usually not used. We introduce it for the mathematical purpose of studying the expected update of the pheromone values. In the cases that we study in this paper the factor is constant. Hence it does not change the algorithms' qualitative behaviour.

2.2. ACO variants

Variants of the ACO algorithm generally differ from each other in the pheromone update rule that is applied. A well-known example of an instantiation of update rule (2) is the

[8] Note that this description of the ACO solution construction mechanism for the ATSP is equivalent to the original description as given in [23].

AS-update rule, that is, the update rule of Ant System (AS) [23]. The AS-update rule is obtained from update rule 2 by setting

$$\mathfrak{S}_{\text{upd}} \leftarrow \mathfrak{S}_{\text{iter}}. \tag{3}$$

This update rule is well-known due to the fact that AS was the first ACO algorithm to be proposed in the literature. An example of a pheromone update rule that is more used in practice is the *IB-update* rule (where IB stands for *iteration-best*). The IB-update rule is given by

$$\mathfrak{S}_{\text{upd}} \leftarrow \text{argmax}\{F(\mathfrak{s}) \mid \mathfrak{s} \in \mathfrak{S}_{\text{iter}}\}. \tag{4}$$

The IB-update rule introduces a much stronger bias towards the good solutions found than the AS-update rule. However, this increases the danger of premature convergence. An even stronger bias is introduced by the *BS-update* rule, where BS refers to the use of the *best-so-far* solution $\mathfrak{s}_{\text{bs}}$, that is, the best solution found since the first algorithm iteration. In this case, $\mathfrak{S}_{\text{upd}}$ is set to $\{\mathfrak{s}_{\text{bs}}\}$.

In practice, ACO algorithms that use variations of the IB-update or the BS-update rule and that additionally include mechanisms to avoid premature convergence achieve better results than algorithms that use the AS-update rule. Examples are ant colony system (ACS) [21] and $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [66], which are among the most successful ACO variants in practice.

ACS works as follows. First, instead of choosing at each step during a solution construction the next solution component according to Eq. (1), an ant chooses, with probability q_0, the solution component that maximizes $[\tau_i^j]^{\alpha} \cdot [\eta(c_i^j)]^{\beta}$, or it performs, with probability $1 - q_0$, a probabilistic construction step according to Eq. (1). This type of solution construction is called *pseudo-random proportional*. Second, ACS uses the BS-update rule with the additional particularity that the pheromone evaporation is only applied to values of pheromone trail parameters that belong to solution components that are in $\mathfrak{s}_{\text{bs}}$. Third, after each solution construction step, the following additional pheromone update is applied to pheromone values τ_i^j whose corresponding solution components c_i^j have been added to the solutions under construction:

$$\tau_i^j \leftarrow (1 - \xi) \cdot \tau_i^j + \xi \cdot \tau_0, \tag{5}$$

where τ_0 is a small positive constant such that $F_{\min} \geqslant \tau_0 \geqslant c$, $F_{\min} = \min\{F(\mathfrak{s}) \mid \mathfrak{s} \in \mathfrak{S}\}$, and c is the initial value of the pheromones. In practice, the effect of this local pheromone update is to decrease the pheromone values on the visited solution components, making in this way these components less desirable for the following ants. We want to remark already at this point that ACS belongs to the class $\text{ACO}_{\tau_{\min}}$ of algorithms, that is, the class of ACO algorithms that apply a lower bound $\tau_{\min} > 0$ to all the pheromone values. In the case of ACS, this lower bound is given by τ_0. This follows from the fact that (i) $\tau_i^j \geqslant \tau_0, \forall\, \mathcal{T}_i^j \in \mathcal{T}$, and (ii) $F(\mathfrak{s}_{\text{bs}}) \geqslant \tau_0$.

$\mathcal{MM}$AS algorithms are characterized as follows. Depending on some convergence measure, at each iteration either the IB-update or the BS-update rule (both as explained above) are used for updating the pheromone values. At the start of the algorithm the IB-update rule is used more often, while during the run of the algorithm the frequency with which the

BS-update rule is used increases. Instead of using an implicit lower bound in the pheromone values like ACS, $\mathcal{MM}$AS algorithms use an explicit lower bound $\tau_{\min} > 0$. Therefore, also $\mathcal{MM}$AS belongs to the class $\text{ACO}_{\tau_{\min}}$ of ACO algorithms. In addition to the lower bound, $\mathcal{MM}$AS algorithms use $F(\mathfrak{s}_{\text{bs}})/\rho$ as an upper bound to the pheromone values. The value of this bound is updated each time a new improved solution is found by the algorithm.

It is interesting to note that $F(\mathfrak{s}_{\text{bs}})/\rho$ is an approximation of the real upper bound $\tau_{\max}$ to the value of the pheromones, given below.

Proposition 1. *Given Algorithm* 1 *that is using the pheromone update rule from Eq.* (2)*, for any pheromone value* τ_i^j*, the following holds*:

$$\lim_{t\to\infty} \tau_i^j(t) \leqslant \frac{F(\mathfrak{s}^*)\cdot|\{\mathfrak{S}_{\text{upd}}\}|}{\rho}, \tag{6}$$

where $\mathfrak{s}^*$ *is an optimal solution, and* $\tau_i^j(t)$ *denotes the pheromone value* τ_i^j *at iteration* t*.*

Proof. The maximum possible increase of a pheromone value τ_i^j is—at any iteration—$F(\mathfrak{s}^*)\cdot|\{\mathfrak{S}_{\text{upd}}\}|$ if all the solutions in $\mathfrak{S}_{\text{upd}}$ are equal to the optimal solution $\mathfrak{s}^*$ with $\mathfrak{c}_i^j \in \mathfrak{s}^*$. Therefore, due to evaporation, the pheromone value τ_i^j at iteration t is bounded by

$$\tau_i^{j\,\max}(t) = (1-\rho)^t \cdot c + \sum_{k=1}^{t} (1-\rho)^{t-k} \cdot F(\mathfrak{s}^*)\cdot|\{\mathfrak{S}_{\text{upd}}\}|, \tag{7}$$

where c is the initial value for all the pheromone trail parameters. Asymptotically, because $0 < \rho \leqslant 1$, this sum converges to $F(\mathfrak{s}^*)\cdot|\{\mathfrak{S}_{\text{upd}}\}|/\rho$. □

From this proposition it is clear that the pheromone value upper bound in the case of the IB- or the BS-update rule is $F(\mathfrak{s}^*)/\rho$.

2.3. The hyper-cube framework

Rather than being an ACO variant, the hyper-cube framework (HCF) for ACO (proposed in [6]) is a framework for implementing ACO algorithms that comes with several benefits. In ACO algorithms, the vector of pheromone values can be regarded as a $|\mathfrak{C}|$-dimensional vector[9] $\vec{\tau}$. The application of a pheromone value update rule changes this vector. It moves in a $|\mathfrak{C}|$-dimensional hyper-space defined by the lower and upper limits of the range of values that the pheromone trail parameters can assume. We will denote this hyper-space in the following by $\mathcal{H}_{\mathcal{T}}$. Proposition 1 shows that the upper limit for the pheromone values depends on the quality function $F(\cdot)$, which implies that the limits of $\mathcal{H}_{\mathcal{T}}$ can be very different depending on the quality function and therefore depending on the problem instance tackled. In contrast, the pheromone update rule of the HCF as described in the following implicitly defines the hyper-space $\mathcal{H}_{\mathcal{T}}$ independently of the quality function $F(\cdot)$ and of

[9] Remember that we denote by $\mathfrak{C}$ the set of all solution components.

the problem instance tackled. For example, the pheromone update rule from Eq. (2), once written in HCF-form, becomes

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \rho\cdot \sum_{\{\mathfrak{s}\in\mathfrak{S}_{\text{upd}}\,|\,\mathfrak{c}_i^j\in\mathfrak{s}\}} \frac{F(\mathfrak{s})}{\sum_{\{\mathfrak{s}'\in\mathfrak{S}_{\text{upd}}\}} F(\mathfrak{s}')}, \tag{8}$$

for $i = 1,\ldots,n$, $j = 1,\ldots,|D_i|$. The difference between this pheromone update rule and the one that is used in standard ACO algorithms consists in the normalization of the added amount of pheromone.

In order to give a graphical interpretation of the pheromone update in the HCF, we consider a solution $\mathfrak{s}$ from a different point of view. With respect to a solution $\mathfrak{s}\in\mathfrak{S}$, we partition the set of solution components $\mathfrak{C}$ into two subsets, the set $\mathfrak{C}_{\text{in}}$ that contains all solution components $\mathfrak{c}_i^j\in\mathfrak{s}$, and $\mathfrak{C}_{\text{out}} = \mathfrak{C}\setminus\mathfrak{C}_{\text{in}}$. In this way, we can associate to a solution $\mathfrak{s}$ a binary vector $\vec{\mathfrak{s}}$ of dimension $|\mathfrak{C}|$ in which the position corresponding to solution component $\mathfrak{c}_i^j$ is set to 1 if $\mathfrak{c}_i^j\in\mathfrak{C}_{\text{in}}$, to 0 otherwise. This means that we can regard a solution $\mathfrak{s}$ as a corner of the $|\mathfrak{C}|$-dimensional unit hyper-cube, and that the set of feasible solutions $\mathfrak{S}$ can be regarded as a (sub)set of the corners of this same hypercube. In the following, we denote the convex hull of $\mathfrak{S}$ by $\tilde{\mathfrak{S}}$. It holds that

$$\vec{\tau}\in\tilde{\mathfrak{S}} \Leftrightarrow \vec{\tau} = \sum_{\mathfrak{s}\in\mathfrak{S}} \alpha_{\mathfrak{s}}\vec{\mathfrak{s}}, \quad \alpha_{\mathfrak{s}}\in[0,1], \quad \sum_{\mathfrak{s}\in\mathfrak{S}}\alpha_{\mathfrak{s}} = 1. \tag{9}$$

As an example see Fig. 1(a). In the following, we give a graphical interpretation of the pheromone update rule in the HCF. When written in vector form, Eq. (8) can be expressed as

$$\vec{\tau} \leftarrow (1-\rho)\cdot\vec{\tau} + \rho\cdot\vec{m}, \tag{10}$$

where $\vec{m}$ is a $|\mathfrak{C}|$-dimensional vector with

$$\vec{m} = \sum_{\mathfrak{s}\in\mathfrak{S}_{\text{upd}}} \gamma_{\mathfrak{s}}\cdot\vec{\mathfrak{s}} \quad \text{where } \gamma_{\mathfrak{s}} = \frac{F(\mathfrak{s})}{\sum_{\mathfrak{s}'\in\mathfrak{S}_{\text{upd}}} F(\mathfrak{s}')}. \tag{11}$$

Vector $\vec{m}$ is a vector in $\tilde{\mathfrak{S}}$, the convex hull of $\mathfrak{S}$, as $\sum_{\mathfrak{s}\in\mathfrak{S}_{\text{iter}}}\gamma_{\mathfrak{s}} = 1$ and $0\leqslant\gamma_{\mathfrak{s}}\leqslant 1\,\forall\,\mathfrak{s}\in\mathfrak{S}_{\text{iter}}$. It also holds that vector $\vec{m}$ is the weighted average of binary solution vectors. The higher the quality $F(\mathfrak{s})$ of a solution $\mathfrak{s}$, the higher its influence on vector $\vec{m}$. Simple algebra allows us to express Eq. (10) as

$$\vec{\tau} \leftarrow \vec{\tau} + \rho\cdot(\vec{m} - \vec{\tau}). \tag{12}$$

This shows that the application of the pheromone update rule in the HCF shifts the current pheromone value vector $\vec{\tau}$ toward $\vec{m}$ (see Fig. 1(b)). The size of this shift is determined by the value of parameter ρ. In the extreme cases there is either very little update (when ρ is very close to zero), or the current pheromone value vector $\vec{\tau}$ is replaced by $\vec{m}$ (when $\rho = 1$). Furthermore, if the initial pheromone value vector $\vec{\tau}$ is in $\tilde{\mathfrak{S}}$, it remains in $\tilde{\mathfrak{S}}$, and the pheromone values are bounded to the interval $[0,1]$. This means that the HCF,

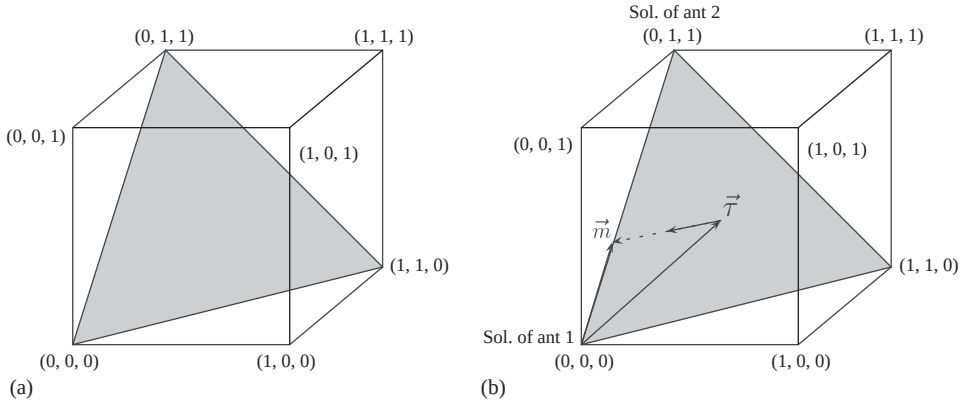

Fig. 1. (a) Example of the convex hull of binary solution vectors; (b) example of the pheromone update in the HCF. In this example, the set $\mathfrak{S}$ of feasible solutions in binary vector form consists of the three vectors $(0, 0, 0)$, $(1, 1, 0)$ and $(0, 1, 1)$. The gray shaded area depicts the set $\widetilde{\mathfrak{S}}$. In (b), two solutions have been created by two ants. The vector $\vec{m}$ is the weighted average of these two solutions (where we assume that $(0, 0, 0)$ is of higher quality), and $\vec{\tau}$ will be shifted toward $\vec{m}$ as a result of the pheromone value update rule (Eq. (8)). Figure from [6]. © IEEE Press.

independently of the problem instance tackled, defines the hyper-space for the pheromone values to be the $|\mathfrak{C}|$-dimensional unit hypercube. [10]

It is interesting to note that in the case of the IB- and BS-update rules (in which only one solution $\mathfrak{s}_{\text{upd}} \in \{\mathfrak{s}_{\text{ib}}, \mathfrak{s}_{\text{bs}}\}$ is used for updating) the old pheromone vector $\vec{\tau}$ is shifted toward the updating solution $\mathfrak{s}_{\text{upd}}$ in binary vector form:

$$\vec{\tau} \leftarrow \vec{\tau} + \rho \cdot (\vec{\mathfrak{s}}_{\text{upd}} - \vec{\tau}). \tag{13}$$

As a notational convention we use HCF-AS-update, HCF-IB-update, and HCF-BS-update, if the corresponding update rules are considered for an ACO algorithm that is implemented in the HCF.

2.4. Models of ACO algorithms

Merkle and Middendorf introduced the use of models of ACO algorithms in [49] for the study of the dynamics of the ACO algorithm search process. A model of an ACO algorithm is a deterministic dynamical system obtained by applying the expected pheromone update instead of the real pheromone update. The advantage of studying an ACO algorithm model is that it—being deterministic—behaves always in the same way, in contrast to the behavior of the ACO algorithm itself which in each run slightly differs due to the stochasticity. There are several ways of studying an ACO model. For example, one might study the evolution of the pheromone values over time, or one might study the evolution of the expected quality of

[10] Note that earlier attempts to normalize pheromone values exist in the literature (see, for example, [35]). However, existing approaches do not provide a framework for doing it automatically.

the solutions that are generated per iteration. This expected iteration quality is henceforth denoted by $W_F(\mathcal{T})$, or by $W_F(\mathcal{T} \mid t)$, where $t > 0$ is the iteration counter.

We use the following notation for defining ACO models. The template for this notation is

$$M(< \text{problem} >, < \text{update\_rule} >, < \text{nr\_of\_ants} >), \tag{14}$$

where $<$ problem $>$ is the considered problem (or problem type, such as, for example, unconstrained problems), $<$ update_rule $>$ is the pheromone update rule that is considered, and $<$ nr_of_ants $>$ is the number of ants that build solutions at each iteration. The number $<$ nr_of_ants $>$ can be a specific integer $n_a \geqslant 1$, any finite integer (denoted by $n_a < \infty$), or $n_a = \infty$. In all cases, the character $*$ denotes any possible entry. As an example, consider the model $M(*, \text{AS}, n_a < \infty)$: this is the model of an ACO algorithm that can be applied to any problem, and that uses the AS-update rule and a finite number of ants at each iteration. The expected iteration quality of model $M(*, \text{AS}, n_a < \infty)$ is

$$W_F(\mathcal{T}) = \sum_{S^{n_a} \in \mathfrak{S}^{n_a}} \left(\mathbf{p}(S^{n_a} \mid \mathcal{T}) \cdot \frac{1}{n_a} \cdot \sum_{\mathfrak{s} \in S^{n_a}} F(\mathfrak{s}) \right), \tag{15}$$

where $\mathfrak{S}^{n_a}$ is the set of all multi-sets of cardinality n_a consisting of elements from $\mathfrak{S}$, and $\mathbf{p}(S^{n_a} \mid \mathcal{T})$ is the probability that the n_a ants produce the multi-set $S^{n_a} \in \mathfrak{S}^{n_a}$, given the current pheromone values. The expected pheromone update of model $M(*, \text{AS}, n_a < \infty)$ is

$$\tau_i^j \leftarrow (1-\rho) \cdot \tau_i^j + \frac{\rho}{n_a} \cdot \sum_{S^{n_a} \in \mathfrak{S}^{n_a}} \left(\mathbf{p}(S^{n_a} \mid \mathcal{T}) \sum_{\mathfrak{s} \in S^{n_a} \mid c_i^j \in \mathfrak{s}} F(\mathfrak{s}) \right), \tag{16}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, |D_i|$.

In order to reduce the computational complexity we may consider model $M(*, \text{AS}, n_a = \infty)$, which assumes an infinite number of ants per iteration.[11] In this case, the expected iteration quality is given by

$$W_F(\mathcal{T}) = \sum_{\mathfrak{s} \in \mathfrak{S}} F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T}), \tag{17}$$

where $\mathbf{p}(\mathfrak{s} \mid \mathcal{T})$ is the probability to produce solution $\mathfrak{s}$ given the current pheromone values. The expected pheromone update of model $M(*, \text{AS}, n_a = \infty)$ is given by

$$\tau_i^j \leftarrow (1-\rho) \cdot \tau_i^j + \rho \cdot \sum_{\{\mathfrak{s} \in \mathfrak{S} \mid c_i^j \in \mathfrak{s}\}} F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T}). \tag{18}$$

If, instead, we consider model $M(*, \text{HCF-AS}, n_a = \infty)$, that is, the AS algorithm implemented in the HCF using an infinite number of ants, the expected iteration quality is the same as in model $M(*, \text{AS}, n_a = \infty)$ (see Eq. (17)), but the expected pheromone

[11] The way of examining the expected behavior of an algorithm by assuming an infinite number of solutions per iteration has already been used in the field of evolutionary computation (see for example the *zeroth order model* proposed in [57]).

update becomes

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \rho\cdot \sum_{\{\mathfrak{s}\in\mathfrak{S}\mid \mathfrak{c}_i^j\in\mathfrak{s}\}} \frac{F(\mathfrak{s})\cdot \mathbf{p}(\mathfrak{s}|\mathcal{T})}{W_F(\mathcal{T})}, \tag{19}$$

for $i = 1,\ldots,n$, $j = 1,\ldots,|D_i|$. The study of models of ACO algorithms will play an important role in Section 5 of this paper.

3. Convergence of ACO algorithms

In this section, we discuss convergence of two classes of ACO algorithms: $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ and $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$. These two classes are defined as follows. First, in order to ease the derivations, both $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ and $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$ use simplified transition probabilities that do not consider heuristic information: Eq. (1) (see Section 2.1), becomes

$$\mathbf{p}(\mathfrak{c}_i^j \mid \mathfrak{s}^p) = \frac{[\tau_i^j]^\alpha}{\sum_{\mathfrak{c}_k^l\in\mathfrak{N}(\mathfrak{s}^p)}[\tau_k^l]^\alpha}, \quad \forall\ \mathfrak{c}_i^j \in \mathfrak{N}(\mathfrak{s}^p). \tag{20}$$

Second, both algorithm classes use the BS-update rule (see Section 2.2). Third, both $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ and $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$ use a lower limit $\tau_{\min} > 0$ for the value of pheromone trails, chosen so that $\tau_{\min} < F(\mathfrak{s}^*)$, where $\mathfrak{s}^*$ is an optimal solution. $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$ differs from $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ because it allows the change of the value of $\tau_{\min}$ at run-time.

For $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ convergence in value is proven via Theorem 1 which essentially says that, because of the use of a fixed positive lower bound on the pheromone values, $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ is guaranteed to find an optimal solution if given enough time.

For $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$, first convergence in value is proven via Theorem 2, under the condition that the bound $\tau_{\min}$ decreases to zero slowly enough.[12] Then, convergence in solution is proven via Theorem 3, which shows that a sufficiently slow decrement of the lower pheromone trail limits leads to the effect that the algorithm converges to a state in which all the ants construct the optimal solution over and over again (made possible by the fact that the pheromone trails go to zero).

3.1. Convergence in value

In this subsection, we state that $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ is guaranteed to find an optimal solution with a probability that can be made arbitrarily close to 1 if given enough time (convergence in value). However, as we will indicate in Section 3.2, the convergence in solution for $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ cannot be proved.

In Proposition 1 (see Section 2.2) it was proved that, due to pheromone evaporation, the pheromone values are asymptotically bounded from above with $\tau_{\max}$ as the limit. The following proposition follows directly from Proposition 1.

[12] Unfortunately, Theorem 2 cannot be proven for the exponentially fast decrement of the pheromone trails obtained by a constant pheromone evaporation rate, which most ACO algorithms use.

Proposition 2. *Once an optimal solution $\mathfrak{s}^*$ has been found by algorithm $ACO_{\mathrm{bs},\tau_{\min}}$, it holds that*

$$\forall c_i^j \in \mathfrak{s}^* : \lim_{t\to\infty} \tau_i^j(t) = \tau_{\max} = \frac{F(\mathfrak{s}^*)}{\rho}. \tag{21}$$

The proof of this proposition is basically a repetition of the proof of Proposition 1, restricted to the solution components of the optimal solution $\mathfrak{s}^*$. Additionally, τ_0 has—for each $c_i^j \in \mathfrak{s}^*$—to be replaced by $\tau_i^j(t^*)$ (where t^* is the iteration in which $\mathfrak{s}^*$ was found).

Proposition 1 implies that, for the proof of Theorem 1 (see below), the only essential point is that $\tau_{\min} > 0$, because from above the pheromone values will anyway be bounded by $\tau_{\max}$. Proposition 2 additionally states that, once an optimal solution $\mathfrak{s}^*$ has been found, the pheromone values on all solution components of $\mathfrak{s}^*$ converge to $\tau_{\max} = F(\mathfrak{s}^*)/\rho$.

Theorem 1. *Let $\mathbf{p}^*(t)$ be the probability that $ACO_{\mathrm{bs},\tau_{\min}}$ finds an optimal solution at least once within the first t iterations. Then, for an arbitrarily small $\varepsilon > 0$ and for a sufficiently large t it holds that*

$$\mathbf{p}^*(t) \geqslant 1 - \varepsilon, \tag{22}$$

and asymptotically $\lim_{t\to\infty} \mathbf{p}^*(t) = 1$.

Proof. The proof of this theorem consists in showing that, because of $\tau_{\min} > 0$, at each algorithm iteration any generic solution, including any optimal solution, can be generated with a probability greater than zero. Therefore, by choosing a sufficiently large number of iterations, the probability of generating any solution, and in particular an optimal one, can be made arbitrarily close to 1. For a detailed proof see [65] or [24]. □

3.2. Convergence in solution

In this subsection we deal with the convergence in solution of algorithm $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$. For proving this property, it has to be shown that, in the limit, any arbitrary ant of the colony will construct the optimal solution with probability one. This cannot be proven if, as done in $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$, a small, positive lower bound is imposed on the lower pheromone value limits because in this case at any iteration t each ant can construct any solution with a non-zero probability. The key of the proof is therefore to allow the lower pheromone trail limits to decrease over time toward zero, but making this decrement slow enough to guarantee that the optimal solution is eventually found.

The proof of convergence in solution as presented in [24] was inspired by an earlier work of Gutjahr [38]. It is organized in two theorems. First, Theorem 2 proves convergence in value of $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$ when its lower pheromone trail limits decrease toward zero at not more than logarithmic speed. Next, Theorem 3 states, under the same conditions, convergence in solution of $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$.

Theorem 2. *Let the lower pheromone trail limits in $ACO_{\mathrm{bs},\tau_{\min}(t)}$ be*

$$\forall t \geqslant 1, \quad \tau_{\min}(t) = \frac{d}{\ln(t+1)}, \tag{23}$$

with d being a constant, and let $\mathbf{p}^*(t)$ be the probability that $ACO_{\mathrm{bs},\tau_{\min}(t)}$ finds an optimal solution at least once within the first t iterations. Then it holds that

$$\lim_{t\to\infty} \mathbf{p}^*(t) = 1. \tag{24}$$

Proof. The proof consists in showing that there is an upper bound to the probability of not constructing an optimal solution whose value goes to zero in the limit. A detailed proof can be found in [24]. □

It remains to be proved that any ant will in the limit construct the optimal solution with probability 1 (i.e., convergence in solution). This result is stated in Theorem 3.

Theorem 3. *Let t^* be the iteration in which the first optimal solution $\mathfrak{s}^*$ has been found and $\mathbf{p}(\mathfrak{s}^*, t, k)$ be the probability that an arbitrary ant k constructs $\mathfrak{s}^*$ in the t-th iteration, with $t > t^*$. Then it holds that $\lim_{t\to\infty} \mathbf{p}(\mathfrak{s}^*, t, k) = 1$.*

Proof. The proof of this theorem consists in showing that the pheromone values of solution components that do not belong to the optimal solution asymptotically converge to 0. For details see [24]. □

3.3. Extension to include additional features of ACO algorithms

Most, if not all, ACO algorithms in practice include some features that are present neither in $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ nor in $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$. Of particular interest is how the use of local search to improve the constructed solutions and the use of heuristic information affect the convergence proof for $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$.[13] Concerning the use of local search, it is rather easy to see that it neither affects the convergence properties of $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$, nor those of $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$. This is because the validity of both convergence proofs (as presented in [24]) depends only on the way solutions are constructed and not on the fact that the solutions are taken or not to their local optima by a local search routine.

The second question concerns the consequences of the use of heuristic information, that is, when considering Eq. (1) instead of Eq. (20) for computing the transition probabilities during solution construction. In fact, neither Theorem 1 nor Theorems 2 and 3 are affected by the heuristic information, if we have $0 < \eta(\mathfrak{c}_i^j) < +\infty$ for each $\mathfrak{c}_i^j \in \mathfrak{C}$ and $\beta < \infty$. In fact, with these assumptions $\eta(\cdot)$ is limited to some (instance specific) interval $[\eta_{\min}, \eta_{\max}]$, with $\eta_{\min} > 0$ and $\eta_{\max} < +\infty$. Then, the heuristic information has the only effect to change the lower bounds on the probability of making a specific decision (which is an important component of the proofs of Theorems 2 and 3).

[13] Note that, here and in the following, although the remarks made on $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$ in general also apply to $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}(t)}$, for simplicity we often refer only to $\mathrm{ACO}_{\mathrm{bs},\tau_{\min}}$.

3.4. Convergence proofs for other types of ACO algorithms

A pioneering study from which much of the inspiration for later works was taken is that of Gutjahr [36–38]. In [37] he presented the first piece of research on the convergence properties of ACO algorithms, which deals with the convergence in solution for the so-called graph-based ant system (GBAS). GBAS is very similar to $\text{ACO}_{\text{bs},\tau_{\min}(t)}$ except that $\tau_{\min} = 0$ and the pheromone update rule changes the pheromones only when, in the current iteration, a solution at least as good as the best one found so far is generated. The following theorems were proved for GBAS:

Theorem 4. *For each $\varepsilon > 0$, for a fixed evaporation rate ρ, and for a sufficiently large number of ants, the probability* $\mathbf{p}$ *that a fixed ant constructs the optimal solution at iteration t is $\mathbf{p} \geqslant 1 - \varepsilon$ for all $t \geqslant t_0$, with $t_0 = t_0(\varepsilon)$.*

Theorem 5. *For each $\varepsilon > 0$, for a fixed number of ants, and for an evaporation rate ρ sufficiently close to zero, the probability* $\mathbf{p}$ *that a fixed ant constructs the optimal solution at iteration t is $\mathbf{p} \geqslant 1 - \varepsilon$ for all $t \geqslant t_0$, with $t_0 = t_0(\varepsilon)$.*

One of the limitations of these proofs is that they require the problem to have a single optimal solution. This limitation has been removed in an extension of the above two results in [36]. Another limitation is the way of updating the pheromone values. While the convergence results presented in previous sections hold independently of the way the pheromone values are updated, the theorems for GBAS hold only for its particular pheromone update rule. In [36] this limitation was weakened by only requiring the GBAS update rule in the final phases of the algorithm.

Finally, Gutjahr [38] provided a proof of convergence in solution for two variants of GBAS that gave the inspiration for the proof of Theorem 2. The first variant was called GBAS/tdlb (for time-dependent lower pheromone bound), and the second one GBAS/tdev (for time-dependent evaporation rate). GBAS/tdlb uses a lower bound on the pheromone values very similar to the one that is used in Theorem 2. Differently, in GBAS/tdev it is the pheromone evaporation rate that is varied during the run of the algorithm: for proving that GBAS/tdev converges in solution, pheromone evaporation is decreased slowly, and in the limit it tends to zero.

3.5. Final remarks on convergence proofs

From the point of view of the researcher interested in practical applications of the algorithms, the interesting part of the discussed convergence proofs is Theorem 1, which guarantees that $\text{ACO}_{\text{bs},\tau_{\min}}$ will find an optimal solution if it runs long enough. It is therefore interesting that this theorem also applies to ACO algorithms that differ from $\text{ACO}_{\text{bs},\tau_{\min}}$ in the way the pheromone update procedure is implemented. In general, Theorem 1 applies to any ACO algorithm for which the probability $\mathbf{p}(\mathfrak{s})$ of constructing a solution $\mathfrak{s} \in \mathfrak{S}$ always remains greater than a small constant $\varepsilon > 0$. In $\text{ACO}_{\text{bs},\tau_{\min}}$ this is a direct consequence of the fact that $0 < \tau_{\min} < \tau_{\max} < +\infty$, which was obtained by (i) explicitly setting a minimum value $\tau_{\min}$ for pheromone trails, (ii) limiting the amount of pheromone that the

ants may deposit after each iteration to finite values, (iii) letting pheromone evaporate over time, that is, by setting $\rho > 0$, and by (iv) the particular form of choosing the transition probabilities. As mentioned in Section 2.2, we call the class of ACO algorithms that impose a lower bound (and, implicitly, an upper bound) to the pheromone values $\text{ACO}_{\tau_{\min}}$. By definition, Theorem 1 holds therefore for any algorithm in $\text{ACO}_{\tau_{\min}}$, which contains practically relevant algorithms such as ACS and $\mathcal{MM}$AS.

Open problem 1. *The proofs that were presented in this section do not say anything about the time required to find an optimal solution, which can be astronomically large. It would be interesting to obtain results on convergence speed for ACO algorithms, in spirit similar to what has been done in evolutionary computation for relatively simple problems such as, for example, ONE-MAX* [43].

4. Model-based search

Up to now we have regarded ACO algorithms as a class of stochastic search procedures working in the space of the solutions of a combinatorial optimization problem. Under this interpretation, artificial ants are stochastic constructive heuristics that build better and better solutions to a combinatorial optimization problem by using and updating pheromone trails. In other words, our attention has been directed to the stochastic constructive procedure used by the ants and to how the ants use the solutions they build to bias the search of future ants by changing pheromone values. In the following, we show that by changing the point of view, we can clarify the intrinsic relation of ACO algorithms to algorithms such as stochastic gradient ascent (SGA) [53,60] and the cross-entropy (CE) method [15,61]. This is done by studying these algorithms under a common algorithmic framework called model-based search (MBS) [67]. The results presented in this section were obtained in [25,52,67].

An MBS algorithm is characterized by the use of a (parametrized) probabilistic model $M \in \mathcal{M}$ (where $\mathcal{M}$ is the set of all possible probabilistic models) that is used to generate solutions to the problem under consideration. At a very general level, a model-based search algorithm attempts to solve an optimization problem by repeating the following two steps:

- Candidate solutions are constructed using some parametrized probabilistic model, that is, a parametrized probability distribution over the solution space.
- Candidate solutions are evaluated and then used to modify the probabilistic model in a way that is deemed to bias future sampling toward low cost solutions. Note that the model's structure may be fixed in advance, with solely the model's parameter values being updated, or alternatively, the structure of the model may be allowed to change as well.

In the following, we focus on the use of fixed model structures based on a vector of model parameters $\mathcal{T}$, and identify a model M with its vector of parameters $\mathcal{T}$. The way of sampling solutions (i.e., the way of constructing solutions) induces a probability function $\mathbf{p}(\cdot \mid \mathcal{T})$ on the search space of the tackled optimization problem. Given this probability function and a certain setting τ of the parameter values, the probability of a solution $\mathfrak{s} \in \mathfrak{S}$ to be sampled is denoted by $\mathbf{p}(\mathfrak{s} \mid \tau)$. We assume that

- $\forall\, \mathfrak{s} \in \mathfrak{S}$ the model parameters can assume values $\tau_{\mathfrak{s}}$ such that the distribution $\mathbf{p}(\cdot \mid \tau_{\mathfrak{s}})$ defined by $\mathbf{p}(\mathfrak{s} \mid \tau_{\mathfrak{s}}) = 1$ and $\mathbf{p}(\mathfrak{s}' \mid \tau_{\mathfrak{s}}) = 0\ \forall\, \mathfrak{s}' \neq \mathfrak{s}$ is obtained. This "expressiveness"

assumption is needed in order to guarantee that the sampling can concentrate in the proximity of any solution, an optimal solution in particular; [14]
- and that the probability function $\mathbf{p}(\cdot \mid \mathcal{T})$ is continuously differentiable with respect to $\mathcal{T}$.

In MBS algorithms, the view on an algorithm is dominated by its probabilistic model. Therefore, the tackled optimization problem is replaced by the following continuous maximization problem:

$$\tau^* \leftarrow \operatorname*{argmax}_{\tau} W_F(\mathcal{T}), \tag{25}$$

where $W_F(\mathcal{T})$ (as introduced in Section 2.4) denotes the expected quality of a generated solution depending on the values of the parameters $\mathcal{T}$. It may be easily verified that, under the "expressiveness'' assumption we made about the space of possible probability distributions, the support of $\mathbf{p}(\cdot \mid \tau^*)$ (i.e., the set $\{\mathfrak{s} \mid \mathbf{p}(\mathfrak{s} \mid \tau^*) > 0\}$) is necessarily contained in $\mathfrak{S}^*$. This implies that solving the problem given by Eq. (25) is equivalent to solving the original combinatorial optimization problem.

In the following we first outline the SGA and the CE methods in the MBS framework, before we show the relation of the two methods to ACO algorithms. In particular, we will see that the pheromone update rules as proposed in the ACO literature have a theoretical justification.

4.1. Stochastic gradient ascent

A possible way of searching for a (possibly local) optimum of the problem given by Eq. (25) is to use the gradient ascent method. In other words, gradient ascent may be used as a heuristic to change τ with the goal of solving Eq. (25). The gradient ascent procedure starts from some initial model parameter value setting τ (possibly randomly generated). Then, at each iteration it calculates the gradient $\nabla W_F(\mathcal{T})$ and updates τ to become $\tau + \alpha \nabla\, W_F(\mathcal{T})|_\tau$, [15] where α is a step-size parameter.

The gradient can be calculated (bearing in mind that $\nabla \ln f = \nabla f / f$) as follows:

$$\begin{aligned}
\nabla W_F(\mathcal{T}) &= \nabla \sum_{\mathfrak{s} \in \mathfrak{S}} F(\mathfrak{s}) \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) = \sum_{\mathfrak{s} \in \mathfrak{S}} F(\mathfrak{s}) \nabla \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) \\
&= \sum_{\mathfrak{s} \in \mathfrak{S}} \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) F(\mathfrak{s}) \frac{\nabla \mathbf{p}(\mathfrak{s} \mid \mathcal{T})}{\mathbf{p}(\mathfrak{s} \mid \mathcal{T})} \\
&= \sum_{\mathfrak{s} \in \mathfrak{S}} \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) F(\mathfrak{s}) \nabla \ln \mathbf{p}(\mathfrak{s} \mid \mathcal{T}).
\end{aligned} \tag{26}$$

However, the gradient ascent algorithm cannot be implemented in practice, as for its evaluation a summation over the whole search space is needed. A more practical alternative is the use of *stochastic gradient ascent*, which replaces—for a given parameter setting τ—the expectation in Eq. (26) by an empirical mean of a sample generated from $\mathbf{p}(\cdot \mid \tau)$.

[14] Note that this condition may be relaxed by assuming that the probability distribution induced by a parameter value setting τ is in the closure of all inducible probability distributions.

[15] Note that $\nabla W_F(\mathcal{T})|_\tau$ denotes the gradient of $W_F(\mathcal{T})$ evaluated in τ.

The update rule for the stochastic gradient then becomes

$$\tau(t+1) = \tau(t) + \alpha \sum_{\mathfrak{s} \in \mathfrak{S}_{\text{upd}}} F(\mathfrak{s}) \nabla \ln \mathbf{p}(\mathfrak{s} \mid \tau(t)), \tag{27}$$

where $\mathfrak{S}_{\text{upd}}$ is the sample at iteration t. In order to derive a practical algorithm from the SGA approach, we need a model for which the derivatives of $\ln \mathbf{p}(\cdot \mid \mathcal{T})$ can be calculated efficiently. In Section 4.3 we will show how this can be done within the context of the ACO metaheuristic.

4.2. The cross-entropy method

Starting from some initial distribution that is given by the probability function $\mathbf{p}(\cdot \mid \tau(0))$ (denoted in the following by $\mathbf{p}_0$), the CE method inductively builds a series of distributions $\mathbf{p}_t = \mathbf{p}(\cdot \mid \tau(t))$ in an attempt to increase the probability of generating high quality solutions after each iteration. A tentative way to achieve this goal is to set $\mathbf{p}_{t+1}$ equal to $\hat{\mathbf{p}}$, where $\hat{\mathbf{p}}$ is proportional to $\mathbf{p}_t$ as follows:

$$\hat{\mathbf{p}} \propto \mathbf{p}_t F(\cdot), \tag{28}$$

where $F(\cdot)$ is, again, some quality function, depending on the objective function.

If this were possible, then, for time independent quality functions, after t iterations we would obtain $\mathbf{p}_t \propto \mathbf{p}_0 F(\cdot)^t$. Consequently, as $t \to \infty$, $\mathbf{p}_t$ would converge to a probability distribution restricted to $\mathfrak{S}^*$. Unfortunately, even if the distribution $\mathbf{p}_t$ is such that it can be induced by some setting τ of the parameter values, for the distribution $\hat{\mathbf{p}}$ as defined by Eq. (28) this does not necessarily hold, hence some sort of projection is needed. A natural candidate for the projection $\mathbf{p}_{t+1}$ is the distribution $\mathbf{p}$ that minimizes the *Kullback–Leibler divergence* [45], which is a commonly used measure of the difference between two distributions:

$$D(\hat{\mathbf{p}} \| \mathbf{p}) = \sum_{\mathfrak{s} \in \mathfrak{S}} \hat{\mathbf{p}}(\mathfrak{s} \mid \mathcal{T}) \ln \frac{\hat{\mathbf{p}}(\mathfrak{s} \mid \mathcal{T})}{\mathbf{p}(\mathfrak{s} \mid \mathcal{T})} \tag{29}$$

or equivalently the *cross-entropy*:

$$- \sum_{\mathfrak{s} \in \mathfrak{S}} \hat{\mathbf{p}}(\mathfrak{s} \mid \mathcal{T}) \ln \mathbf{p}(\mathfrak{s} \mid \mathcal{T}). \tag{30}$$

Since $\hat{\mathbf{p}} \propto \mathbf{p}_t F(\cdot)$, the cross-entropy minimization is equivalent to the following maximization problem:

$$\mathbf{p}_{t+1} = \operatorname*{argmax}_{\mathbf{p}(\cdot \mid \tau)} \sum_{\mathfrak{s} \in \mathfrak{S}} \mathbf{p}(\mathfrak{s} \mid \tau(t)) F(\mathfrak{s}) \ln \mathbf{p}(\mathfrak{s} \mid \tau). \tag{31}$$

In a way similar to what happened with the gradient of Eq. (26), the maximization problem given by Eq. (31) cannot be solved in practice, because the evaluation of the function on the right-hand side requires summation over the whole search space. As before, however, a finite sample approximation can be used

$$\mathbf{p}_{t+1} = \operatorname*{argmax}_{\mathbf{p}(\cdot \mid \tau)} \sum_{\mathfrak{s} \in \mathfrak{S}_{\text{upd}}} F(\mathfrak{s}) \ln \mathbf{p}(\mathfrak{s} \mid \tau), \tag{32}$$

where $\mathfrak{S}_{\text{upd}}$ is the sample at iteration t. In some relatively simple cases this problem can be solved exactly. In general, however, the analytical solution is unavailable. Still, even if the exact solution is not known, some iterative methods such as SGA for solving this optimization problem may be used. It should be noted that, since the new vector of pheromone values $\tau(t+1)$ is a random variable, depending on a sample, there is no use in running the SGA process till full convergence. Instead, in order to obtain some robustness against sampling noise, a fixed number of SGA updates may be used. One particular choice, which is of special interest, is the use of a single gradient ascent update, leading to an update rule which is identical to the SGA update shown in Eq. (27). However, in general the CE method imposes less restrictions on the quality function (e.g., allowing it to change over time), hence the resulting algorithm may be seen as a generalization of SGA.

4.3. Relation of ACO with SGA and the CE method

The discussion of SGA and of the CE method in the previous two sections was focused on the update of the model parameter values. However, this is only one of the components needed in any model-based search algorithm. In the following we focus on the probability function $\mathbf{p}(\cdot \mid \mathcal{T})$ that is implicitly given by the solution construction process of ACO algorithms. We show that the calculation of the derivatives of this probability function can be carried out in a reasonable time, and we outline the existing work on deriving updates of the parameter values that describe a SGA, respectively a CE method, in the space of the parameter values.

The SGA update in ACO. The SGA parameter value update that we describe in the following is a generalization of the one that was presented in [67] (which was itself a generalization of the one that was given in [52]).

As described in Section 2.1, in ACO algorithms a solution $\mathfrak{s}$ is constructed as a finite-length sequence $\langle \mathfrak{c}_i^j, \ldots, \mathfrak{c}_k^l, \ldots, \mathfrak{c}_r^s \rangle$ of solution components $\mathfrak{c}$ from the set $\mathfrak{C}$ of solution components. For the sake of simplicity, we rename the components of the sequence so to obtain $\langle \mathfrak{c}_1, \mathfrak{c}_2, \ldots, \mathfrak{c}_{|\mathfrak{s}|} \rangle$. By defining the transition probabilities as done in Eq. (1) (see p. 8), the probability function in ACO algorithms can be written as

$$\mathbf{p}(\mathfrak{s} \mid \mathcal{T}) = \prod_{h=1}^{|\mathfrak{s}|-1} \mathbf{p}(\mathfrak{c}_{h+1} \mid \mathfrak{s}_h^p), \tag{33}$$

where $\mathfrak{s}_h^p$ is the partial sequence $\langle \mathfrak{c}_1, \ldots, \mathfrak{c}_h \rangle$, and consequently

$$\nabla \ln \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) = \sum_{h=1}^{|\mathfrak{s}|-1} \nabla \ln \mathbf{p}(\mathfrak{c}_{h+1} \mid \mathfrak{s}_h^p). \tag{34}$$

Let us now consider an arbitrary solution construction step $h \in \{1, \ldots, |\mathfrak{s}|\}$ with $\mathfrak{N}(\mathfrak{s}_h^p)$ being the set of solution components that can be added to the current partial sequence $\mathfrak{s}_h^p$. For the sake of readability let us also denote the "desirability'' $[\tau_i^j]^\alpha \cdot [\eta(\mathfrak{c}_i^j)]^\beta$ of a solution component $\mathfrak{c}_i^j$ (as used for determining the transition probabilities in Eq. (1)) by $\mathrm{d}(\mathfrak{c}_i^j)$.

If $c_i^j \in \mathfrak{N}(s_h^p)$ and $c_i^j = c_{h+1}$ it holds that

$$\begin{aligned}
\frac{\partial}{\partial \mathcal{T}_i^j}\left\{\ln \mathbf{p}(c_i^j \mid s_h^p)\right\} &= \frac{\partial}{\partial \mathcal{T}_i^j}\left\{\ln\left(d(c_i^j)\Big/\sum_{c_k^l \in \mathfrak{N}(s_h^p)} d(c_k^l)\right)\right\} \\
&= \frac{\partial}{\partial \mathcal{T}_i^j}\left\{\ln d(c_i^j) - \ln \sum_{c_k^l \in \mathfrak{N}(s_h^p)} d(c_k^l)\right\} \\
&= d'(c_i^j)\Big/ d(c_i^j) - d'(c_i^j)\Big/ \sum_{c_k^l \in \mathfrak{N}(s_h^p)} d(c_k^l) \\
&= \left\{1 - d(c_i^j)\Big/ \sum_{c_k^l \in \mathfrak{N}(s_h^p)} d(c_k^l)\right\} \frac{d'(c_i^j)}{d(c_i^j)} \\
&= \left\{1 - \mathbf{p}(c_i^j \mid s_h^p)\right\} \frac{d'(c_i^j)}{d(c_i^j)}. \qquad (35)
\end{aligned}$$

Otherwise, if $c_i^j \in \mathfrak{N}(s_h^p)$ but $c_i^j \neq c_{h+1}$ it holds (by a similar argument) that

$$\frac{\partial}{\partial \mathcal{T}_i^j}\left\{\ln \mathbf{p}(c_i^j \mid s_h^p)\right\} = -\mathbf{p}(c_i^j \mid s_h^p) \frac{d'(c_i^j)}{d(c_i^j)}. \qquad (36)$$

Finally, if $c_i^j \notin \mathfrak{N}(s_h^p)$ then $\mathbf{p}(c_i^j \mid s_h^p)$ is independent of $\mathcal{T}_i^j$ and therefore we have that

$$\frac{\partial}{\partial \mathcal{T}_i^j}\left\{\ln \mathbf{p}(c_i^j \mid s_h^p)\right\} = 0. \qquad (37)$$

By combining these results, the SGA pheromone update procedure is derived as follows. Let s be the solution for which pheromone updates have to be performed. First, because of Eqs. (27) and (35), pheromones associated to solution components $c_i^j \in s$ are reinforced with the amount $\alpha F(s) \cdot d'(c_i^j)/d(c_i^j)$. Then, because of Eqs. (27), (35) and (36), pheromones that are associated to all the solution components that were considered [16] during the construction of s are decreased by an amount given by $\alpha F(s) \cdot \mathbf{p}(c_i^j \mid s_h^p) \cdot d'(c_i^j)/d(c_i^j)$. Last, because of Eq. (37), all the remaining pheromones are not updated.

In order to guarantee stability of the resulting algorithm, it is desirable to have a bounded gradient $\nabla \ln \mathbf{p}(s \mid \mathcal{T})$. This means that a function $d(\cdot)$, for which $d'(\cdot)/d(\cdot)$ is bounded, should be used. In [52] the authors suggest using $d(\cdot) = \exp(\cdot)$, which leads to $d'(\cdot)/d(\cdot) \equiv 1$. It should be further noted that if, in addition, $F(\cdot) = 1/f(\cdot)$ and $\alpha = 1$, the reinforcement part becomes $1/f(\cdot)$ as in the original Ant System algorithm (see Section 2.2).

The cross-entropy update in ACO. As we have shown in Section 4.2, the CE method requires at each step the solution of the problem stated in Eq. (32). Since, in general, the exact solution is not available, an iterative scheme such as gradient ascent could be employed. As we have shown in the previous section, the gradient of the log-probability for ACO-type probabilistic functions can be efficiently calculated. The obtained values may be plugged into any general

[16] We say that a solution component c_i^j was "considered'' at construction step h, if $c_i^j \in \mathfrak{N}(s_h^p)$.

iterative solution scheme of the cross-entropy minimization problem, for example, the one described by Eq. (27). If, for example, we use a single-step gradient ascent for solving Eq. (32), we obtain a generalization of the SGA pheromone update, in which the quality function is permitted to change over time.

In some special cases, such as for example unconstrained problems, it can be shown (see [24]) that the parameter update of the CE method is exactly the same as the update of Ant System implemented in the HCF (see Section 2.3) with a setting of $\rho = 1$ (remember that ρ is the evaporation rate in standard ACO update rules).

Open problem 2. *The relation of ACO algorithms to other probabilistic learning algorithms such as estimation of distribution algorithms* [55], *or graphical models and Bayesian networks* [42], *is relatively unexplored. More work on these subjects could be of interest.*

5. Search bias in ACO algorithms

In ACO algorithms we find different forms of search bias. A first type of desirable bias, whose goal is to direct the search towards good zones of the search space, is given by the pheromone update rule. A less desirable form of bias, however, can be caused by algorithm features such as the pheromone model and the solution construction process. In fact, sometimes this additional bias is harmful and results in a decrease of algorithm performance over time. There are basically two different strands of work on this type of potentially harmful bias in ACO algorithms. In this section we first review results obtained by Blum et al. in [12,7], and then we summarize those obtained by Merkle and Middendorf in [49].

5.1. Negative search bias caused by an unfair competition

The fact that the average quality of the generated solutions improves over time is, in general, considered to be a desirable characteristic for a metaheuristic. This is because the generation of better average quality solutions during the algorithms' execution is often positively correlated with the probability to generate improved best solutions. Therefore, situations in which this is not the case might be labeled *negative search bias*, as it was done by Blum and Dorigo in [7]. For detecting this type of search bias, they studied the evolution of the expected iteration quality $W_F(\mathcal{T} \mid t)$ [17] of the solutions that are generated by ACO algorithm models.

First, the application of ACO algorithms to unconstrained CO problems was considered, that is, CO problems in which the set Ω of constraints is empty (see Definition 1 at p. 6). By relating the expected pheromone update to a type of function called *growth transformation* [1], the following result was proved in [6]:

Theorem 6. *The expected iteration quality $W_F(\mathcal{T})$ of $M(U, HCF\text{-}AS, n_a = \infty)$, where U stands for the application to unconstrained problems, is continuously non-decreasing.*

[17] $W_F(\mathcal{T} \mid t)$ is the value of $W_F(\mathcal{T})$ at iteration t. For a definition of $W_F(\mathcal{T})$ see Section 2.4.

More formally, it holds that

$$W_F(\mathcal{T} \mid t+1) > W_F(\mathcal{T} \mid t), \tag{38}$$

as long as at least one pheromone value changes from iteration t to iteration $t+1$.

An extension of this result to the model $M(U, AS, n_a = \infty)$ was later presented in [5]. These results indicate that the AS algorithm shows a desired behavior when applied to unconstrained problems. However, this result is not exceedingly useful, because most of the relevant optimization problems tackled with ACO algorithms are constrained. Therefore, the focus of research shifted to constrained problems. An example is the case study concerning the NP-hard *k*-cardinality tree (KCT) problem [12,3], a generalization of the well-known minimum spanning tree (MST) problem. It is defined as follows: Given is an undirected graph $G = (V, E)$ (where $|V| = n$ and $|E| = m$) with edge-weights $w(e) \in \mathbb{N}^+, \forall\, e \in E$. The set of all trees in G with exactly k edges is henceforth denoted by $\mathfrak{T}_k$. The goal is to find a tree $T_k \in \mathfrak{T}_k$ that minimizes

$$f(T_k) = \sum_{e \in E(T_k)} w(e). \tag{39}$$

This means that the objective function value of a *k*-cardinality tree is given by the sum of the weights of all its edges. We consider the following CO problem model of the KCT problem: We assign a binary decision variable X_e to each edge $e \in E$. If $X_e = 1$, then e is part of the *k*-cardinality tree that is built. The pheromone model is derived as follows. We introduce for each of the binary decision variables X_e two solution components: c_e^0, which corresponds to $X_e = 0$, and c_e^1 corresponding to $X_e = 1$. The pheromone model consists of a pheromone trail parameter $\mathcal{T}_e^j$ for each solution component $\mathfrak{c}_e^j$, with $j \in \{0, 1\}$.

The considered solution construction mechanism works as follows. The algorithm starts with an empty partial solution $\mathfrak{s}_0^p = \langle\rangle$, and with empty sets E_T (for collecting the added edges) and V_T (for collecting the implicitly added vertices). Then, at construction step h, $0 < h < k$, a solution component $\mathfrak{c}_e^1 \in \mathfrak{N}(\mathfrak{s}_h^p)$ is added to the current partial solution $\mathfrak{s}_h^p$. When adding the solution component $\mathfrak{c}_e^1$ to $\mathfrak{s}_h^p$ we also add $e = \{v, v'\}$ to E_T and v and v' to V_T. For the first construction step, set $\mathfrak{N}(\mathfrak{s}_0^p)$ is defined as $\mathfrak{N}(\mathfrak{s}_0^p) = \{\mathfrak{c}_e^1 \mid e = \{v, v'\} \in E\}$ and for each subsequent construction step h as

$$\mathfrak{N}(\mathfrak{s}_h^p) = \{\mathfrak{c}_e^1 \mid (e = \{v, v'\} \in E \setminus E_T) \wedge ((v \in V_T) \vee (v' \in V_T))\}. \tag{40}$$

The definition of $\mathfrak{N}(\mathfrak{s}_h^p)$ is such that only feasible *k*-cardinality trees can be generated. After k construction steps are performed we add, for all $e \in E$ with $\mathfrak{c}_e^1 \notin \mathfrak{s}_k^p$, the solution component $\mathfrak{c}_e^0$ to $\mathfrak{s}_k^p$. By this last step a sequence $\mathfrak{s}$ is completed that corresponds to a feasible solution. The transition probabilities are at each construction step defined by Eq. (1), with $\alpha = 1$ and $\beta = 0$. The setting of $\beta = 0$ means that no heuristic information for biasing the transition probabilities was considered in order to study the pure behavior of the algorithm.

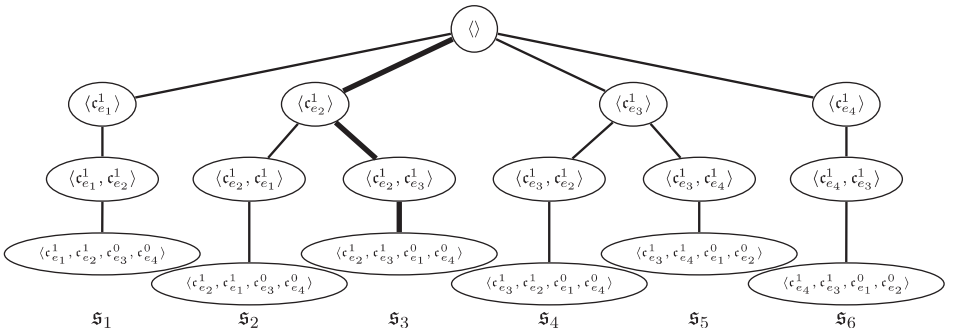

Fig. 2. The complete search tree defined by the solution construction mechanism of the ACO algorithm and the problem instance kct_simple_inst. The bold path in the search tree shows the steps of constructing solution $\mathfrak{s}_3 = \langle \mathfrak{c}^1_{e_2}, \mathfrak{c}^1_{e_3}, \mathfrak{c}^0_{e_1}, \mathfrak{c}^0_{e_4} \rangle$. Figure from [3]. © Akademische Verlagsgesellschaft Aka GmbH.

The evolution of the model M(KCT, AS, $n_a = \infty$) and the behavior of the AS algorithm was studied when applied to the following small problem instance:

The weight settings for this instance are $w(e_1) = w(e_4) = 1$ and $w(e_2) = w(e_3) = 2$. Let us denote this problem instance by kct_simple_inst, and let us consider the problem of solving the 2-cardinality tree problem in kct_simple_inst. An ACO algorithm using the above described solution construction mechanism can produce six different sequences of solution components that map to valid solutions. All six possible solution constructions are shown in form of a search tree in Fig. 2, where sequences $\mathfrak{s}_1$ and $\mathfrak{s}_2$ correspond to the solution (1, 1, 0, 0), that is, ($X_1 = 1$, $X_2 = 1$, $X_3 = 0$, $X_4 = 0$), $\mathfrak{s}_3$ and $\mathfrak{s}_4$ correspond to the solution (0, 1, 1, 0), and $\mathfrak{s}_5$ and $\mathfrak{s}_6$ map to the solution (0, 0, 1, 1). The objective function values are $f(\mathfrak{s}_1) = f(\mathfrak{s}_2) = f(\mathfrak{s}_5) = f(\mathfrak{s}_6) = 3$ and $f(\mathfrak{s}_3) = f(\mathfrak{s}_4) = 4$. This means that $\mathfrak{s}_1$, $\mathfrak{s}_2$, $\mathfrak{s}_5$, and $\mathfrak{s}_6$ are optimal solutions to this problem instance, whereas $\mathfrak{s}_3$ and $\mathfrak{s}_4$ are sub-optimal solutions.

The results of applying the model M(KCT, AS, $n_a = \infty$) to kct_simple_inst are graphically shown in Fig. 3. The expected iteration quality W_F continuously decreases over time. At first sight, this is a surprising result, as we would expect the exact opposite from an ACO algorithm. However, this behavior can be easily explained by taking a closer look at the search tree that is shown in Fig. 2. In the first construction step, there are four different possibilities to extend the empty partial solution. The four solution components that can be added are $\mathfrak{c}^1_{e_1}$, $\mathfrak{c}^1_{e_2}$, $\mathfrak{c}^1_{e_3}$, and $\mathfrak{c}^1_{e_4}$. However, solution components $\mathfrak{c}^1_{e_1}$ and $\mathfrak{c}^1_{e_4}$ in expectation only receive update from two solutions (i.e., sequences $\mathfrak{s}_1$ and $\mathfrak{s}_2$ in case of $\mathfrak{c}^1_{e_1}$, respectively sequences $\mathfrak{s}_5$ and $\mathfrak{s}_6$ in case of $\mathfrak{c}^1_{e_4}$), whereas solution components $\mathfrak{c}^1_{e_2}$ and $\mathfrak{c}^1_{e_3}$ in expectation receive update from four solutions (i.e., sequences $\mathfrak{s}_i$, where $i \in \{1, \ldots, 4\}$, in case of $\mathfrak{c}^1_{e_2}$, respectively sequences $\mathfrak{s}_i$, where $i \in \{3, \ldots, 6\}$, in case of $\mathfrak{c}^1_{e_3}$). This means that for many initial settings of the pheromone values (e.g., when the initial pheromone values are set to the same positive constant $c > 0$) $\mathcal{T}^1_{e_2}$ and $\mathcal{T}^1_{e_3}$ receive in expectation

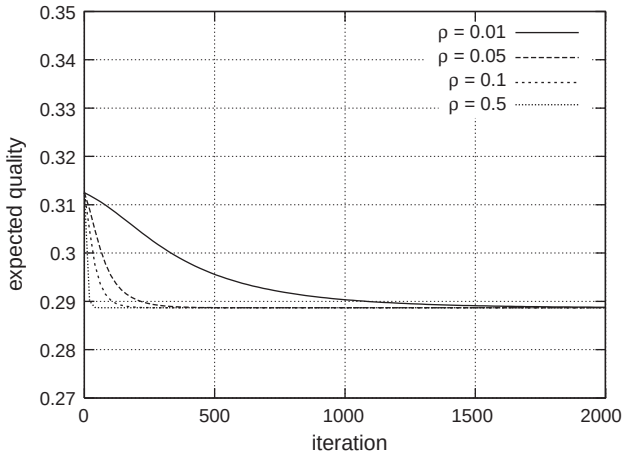

Fig. 3. The evolution of the expected iteration quality W_F of the model $M(\mathrm{KCT}, \mathrm{AS}, n_a = \infty)$ applied to problem instance kct_simple_inst for different settings of the evaporation parameter ρ. All the pheromone values were initialized to 0.5. The plots show that the expected iteration quality continuously decreases. Moreover, for increasing ρ the impact of the pheromone value update increases and the expected iteration quality decreases faster. Figure from [3]. © Akademische Verlagsgesellschaft Aka GmbH.

more updates than $\mathcal{T}^1_{e_1}$ and $\mathcal{T}^1_{e_4}$ just because the number of solutions that contribute to their updates is higher. Therefore, over time the probability of constructing the sub-optimal solutions $\mathfrak{s}_3$ and $\mathfrak{s}_4$ increases, whereas the probability of constructing the optimal solutions $\mathfrak{s}_i$, where $i \in \{1, 2, 5, 6\}$, decreases. This means that the expected iteration quality decreases over time.

The behavior of the real AS algorithm was compared to the behavior of its model by applying AS to the same problem instance kct_simple_inst. Fig. 4 shows the evolution of the empirically obtained average quality of the solutions per iteration for two different evaporation rate values. The results show that when ρ is small the empirical behavior of the algorithm approximates quite well the behavior of its model. However, the stochastic error makes the real AS algorithm, after approximately 1000 iterations, decide for one of the two optimal solutions, which results in a turn from decreasing average iteration quality to increasing average iteration quality.

The bias that is introduced by the fact that some solution components receive in expectation updates from more solutions than others results from an unfair competition between the solution components. In contrast, a fair competition can be defined as follows (see also [7]):

Definition 2. Given a model $\mathcal{P}$ of a CO problem, we call the combination of an ACO algorithm and a problem instance P of $\mathcal{P}$ a *competition-balanced system* (*CBS*), if the following holds: given a feasible partial solution $\mathfrak{s}^p$ and the set of solution components $\mathfrak{N}(\mathfrak{s}^p)$ that can be added to extend $\mathfrak{s}^p$, each solution component $\mathfrak{c} \in \mathfrak{N}(\mathfrak{s}^p)$ is a component of the same number of feasible solutions (in terms of sequences built by the algorithm) as

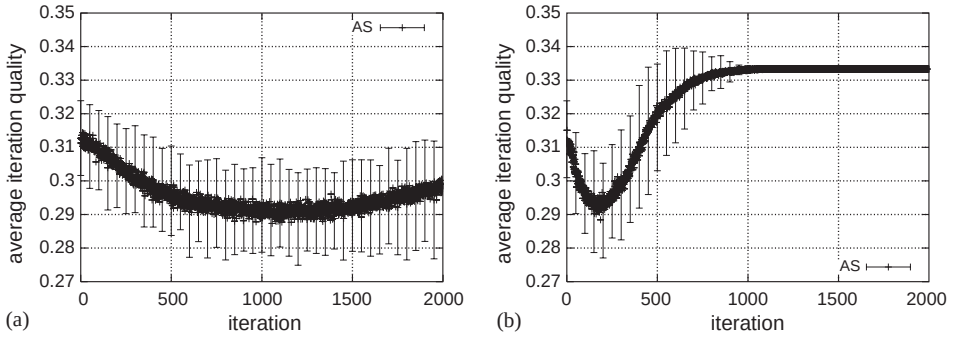

Fig. 4. (a) Instance kct_simple_inst, $n_a = 10$, $\rho = 0.01$; (b) instance kct_simple_inst, $n_a = 10$, $\rho = 0.05$. The plots show the evolution of the average iteration quality obtained by the AS algorithm applied to problem instance kct_simple_inst for $n_a = 10$ (i.e., 10 ants per iteration) and two different settings of the evaporation parameter ρ ($\rho \in \{0.01, 0.05\}$). All the pheromone values were initialized to 0.5. The results are averaged over 100 runs (error bars show the standard deviation and are plotted every 50-th iteration). Figure from [3]. © Akademische Verlagsgesellschaft Aka GmbH.

any other solution component $\mathfrak{c}' \in \mathfrak{N}(\mathfrak{s}^p)$, $\mathfrak{c} \neq \mathfrak{c}'$.[18] In this context, we call the competition between the solution components a *fair competition* if the combination of an ACO algorithm and a problem instance is a CBS.

The application to the small KCT example instance has shown that ACO algorithms applied to the KCT problem when modeled as shown above are—in general—not CBSs. The question is now if we can expect an algorithm **not** to suffer from a negative search bias in case an algorithm/problem instance combination is a CBS. In [7], the authors started to investigate this question by studying the application of ACO algorithms to the asymmetric traveling salesman problem (ATSP). The results of applying the model $M(\text{ATSP}, \text{AS}, n_a = \infty)$ to randomly generated ATSP instances suggest that the expected iteration quality is continuously non-decreasing. However, in general this question is still open.

Open problem 3. *Is the property of being a competition-balanced system sufficient to ensure the absence of any negative search bias*? *In this context, it would be interesting to see if the result that is stated in Theorem* 6 *can be extended to the model* $M(*, AS, n_a = \infty)$ *applied to problems for which the combination of AS and the problem is a competition-balanced system*; *as, for example, AS applied to the ATSP*.

Finally, we note that a (temporary) decrease in expected iteration quality is not necessarily an indicator for the existence of a negative search bias. In other words, the evolution of the expected iteration quality is not a reliable indicator for negative search bias. As an example, let us consider the application of the model $M(\text{U}, \text{IB}, n_a = \infty)$ to a very simple unconstrained maximization problem consisting of two binary variables X_1 and X_2. This

[18] Note that there exist ACO algorithms in which partial solutions are extended by "groups'' of solution components. In these cases the definition of a CBS has to be adapted accordingly.

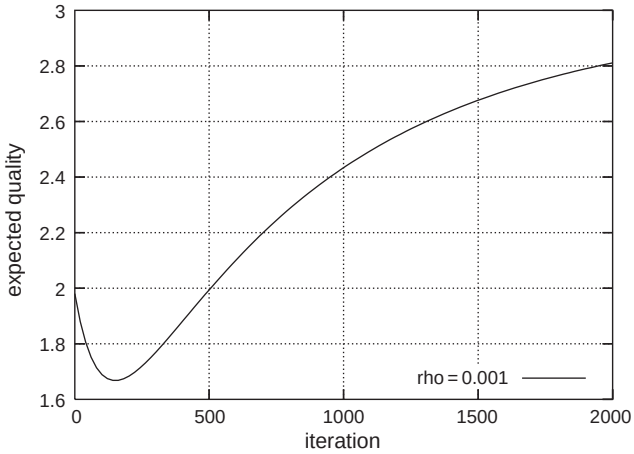

Fig. 5. The evolution of the expected iteration quality W_F of the model $M(\mathrm{U}, \mathrm{IB}, n_a = \infty)$ applied an unconstrained problem instance with two variables (see text for details).

problem has 4 solutions: $\mathfrak{s}_1 = \langle \mathfrak{c}_1^0, \mathfrak{c}_2^0 \rangle$, $\mathfrak{s}_2 = \langle \mathfrak{c}_1^1, \mathfrak{c}_2^0 \rangle$, $\mathfrak{s}_3 = \langle \mathfrak{c}_1^0, \mathfrak{c}_2^1 \rangle$, and $\mathfrak{s}_4 = \langle \mathfrak{c}_1^1, \mathfrak{c}_2^1 \rangle$. Let us assign the following objective function values: $f(\mathfrak{s}_1) = 2$, $f(\mathfrak{s}_2) = f(\mathfrak{s}_3) = 1$, and $f(\mathfrak{s}_4) = 3$. Note that, as we are maximizing, we choose $F(\cdot) = f(\cdot)$. Any ACO algorithm using the solution construction process for unconstrained problems as outlined at the beginning of this section is a CBS. Let us assume that at the start of the algorithm the pheromone values are initialized so that $\mathbf{p}(X_1 = 1) = \mathbf{p}(X_2 = 1) = \varepsilon$, with ε a positive number close to zero. With this setting, the probability to construct any of the four solutions is greater than zero. Therefore, as model $M(\mathrm{U}, \mathrm{IB}, n_a = \infty)$ considers an infinite number of ants per iteration, for sure at each iteration solution $\mathfrak{s}_4$ will be the iteration-best solution, and will therefore be used for updating the pheromone values. The graphic in Fig. 5 shows the evolution of the expected quality of the solutions generated by $M(\mathrm{U}, \mathrm{IB}, n_a = \infty)$ starting from a setting such that $\varepsilon = 0.01$. During the first approximately 200 iterations, the expected quality decreases. For a sufficiently large population size, this result will be approximated by the IB algorithm. Clearly, in this case, the decrease of expected iteration quality is not due to a negative bias. This leads to another open question:

Open problem 4. *Can it be shown that the models $M(U, IB, n_a = \infty)$ and/or $M(U, BS, n_a = \infty)$ have stable attractors that correspond to solutions to the problems*?

5.2. Search bias caused by selection fix-points

Merkle and Middendorf [48–50] studied the dynamics of models of ACO algorithms. One of the questions they tackled concerns the source of the driving force of the algorithm. Often ACO practitioners ask themselves: Why to use more than one ant per iteration? Wouldn't the algorithm work with only one ant? The work presented in [49] gives a theory-based answer to this question. Let us in the following consider the AS algorithm implemented in the HCF for the application to unconstrained problems; and let us consider the use of one

ant per iteration. The expected pheromone update of the algorithms' model, that is, model $M(\mathrm{U}, \text{AS-HCF}, n_a = 1)$,[19] is given by

$$\tau_i^j(t+1) = (1-\rho)\cdot\tau_i^j(t) + \rho\cdot\sum_{\{\mathfrak{s}\in\mathfrak{S}\mid \mathfrak{c}_i^j\in\mathfrak{s}\}}\mathbf{p}(\mathfrak{s}\mid\mathcal{T}). \tag{41}$$

However, it holds that $\sum_{\{\mathfrak{s}\in\mathfrak{S}\mid \mathfrak{c}_i^j\in\mathfrak{s}\}}\mathbf{p}(\mathfrak{s}\mid\mathcal{T}) = \mathbf{p}(\mathfrak{c}_i^j\mid\mathcal{T})$ (because the tackled problem is unconstrained). Moreover, it holds that $\mathbf{p}(\mathfrak{c}_i^j\mid\mathcal{T}) = \tau_i^j(t)$, because the algorithm is implemented in the HCF. Therefore, Eq. (41) can be rewritten as

$$\tau_i^j(t+1) = (1-\rho)\cdot\tau_i^j(t) + \rho\cdot\tau_i^j(t), \tag{42}$$

and therefore it holds that $\tau_i^j(t+1) = \tau_i^j(t)$. This means that the expected pheromone update does not change the pheromone values, which shows that—without the competition among the ants—in ACO algorithm models applied to unconstrained problems, when considering either the AS or the IB updates in the HCF, there is no driving force of the algorithm. This suggests that (i) the competition between the ants is a driving force of ACO algorithms, and that (ii) for the above mentioned type of ACO algorithm models, no negative search bias exists. In the following, we show how this result relates to ACO algorithms that are not implemented in the HCF.

Let us indicate by Δ_i^j the amount of update that a pheromone value τ_i^j receives. Then, we can express a general pheromone update rule by

$$\tau_i^j(t+1) = (1-\rho)\cdot\tau_i^j(t) + \rho\cdot\Delta_i^j. \tag{43}$$

In case of model $M(\mathrm{U}, \mathrm{AS}, n_a = 1)$, it holds that

$$\Delta_i^j = \mathbf{p}(\mathfrak{c}_i^j\mid\mathcal{T}) = \tau_i^j(t)\Big/\sum_{k=1}^{|D_i|}\tau_i^k(t)\ , \tag{44}$$

which in general is not equal to $\tau_i^j(t)$. Therefore, in case the algorithm is not implemented in the HCF, Eq. (43) makes the pheromone values move towards a situation in which $\tau_i^j(t) = \Delta_i^j$. Such situations were labeled by Merkle and Middendorf *selection fix-points* in [49], where they showed the relevance of the *selection fix-point bias* caused by the selection fix-points of ACO algorithms applied to constrained problems.

In particular, the above mentioned work focused on the behavioral study of models $M(\mathrm{PP}, \text{IB-HCF}, n_a < \infty)$ when applied to the following type of permutation problems (PP): A problem instance P consists of a set of n items $I = \{1,\ldots,n\}$ and an $n\times n$ cost matrix $C = [c_{i,j}]_{i,j=1,\ldots,n}$ with integer entries (costs) $c_{i,j}\geqslant 0$. The set of solutions $\mathcal{S}$ to the problem consists of all possible permutations of the n items. Given such a permutation π, its objective function value $f(\pi)$ is given by $\sum_{i=1}^n c_{i,\pi(i)}$. The problem consists in finding a permutation $\pi\in\mathcal{S}$ with minimal objective function value. Given a problem instance P,

[19] Note that with $n_a = 1$ models $M(\mathrm{U}, \text{AS-HCF}, n_a = 1)$ and $M(\mathrm{U}, \text{IB-HCF}, n_a = 1)$ are equivalent.

one can produce "restricted'' permutation problem instances of different sizes. Given, for example, a problem instance P with cost matrix

$$C = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}, \tag{45}$$

a problem instance P^2 with the following cost matrix can be produced:

$$C^2 = \begin{pmatrix} 0 & 1 & 2 & \infty & \infty & \infty \\ 1 & 0 & 1 & \infty & \infty & \infty \\ 2 & 1 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 1 & 2 \\ \infty & \infty & \infty & 1 & 0 & 1 \\ \infty & \infty & \infty & 2 & 1 & 0 \end{pmatrix}. \tag{46}$$

In a similar way, bigger problem instances (i.e., P^k, $k > 2$) can be produced from the elementary subproblem P. The objective function value for a solution π for a restricted problem instance P^k becomes $\sum_{j=0}^{k-1} \sum_{i=1}^{n} c_{jk+i,\pi(jk+i)}$ (where the c-entries are from the matrix C^k). These restricted permutation problems simulate real world problems that consist of subproblems more or less independent of each other. The CO problem model that was used to tackle these problems with an ACO algorithm is the following: Given a restricted permutation problem instance P^k, to each position $r = 1, \ldots, kn$ of a permutation to be built is assigned a decision variable X_r. The domain D_r for a variable X_r contains all elements $l \in I^k = \{1, \ldots, kn\}$ with $c_{r,l} \neq \infty$ (which means that we do not allow position/item combinations with infinite cost). Again, we introduce for each variable/value combination $(X_r, l \in D_r)$ a solution component $\mathfrak{c}_r^l$, which has associated a pheromone trail parameter $\mathcal{T}_r^l$. All the pheromone values are initially set to $1/n$. The solution construction works as follows: We start from an empty partial solution $\mathfrak{s}_0^p = \langle\rangle$. Set I_p, which is the set of already placed items, is set to the empty set at the start of solution construction. Then, at each step h we assign to each decision variable (in the order $r = 1, \ldots, kn$) a value by selecting one of the solution components $\mathfrak{c}_r^l$ from $\mathfrak{N}(\mathfrak{s}_h^p) = \{\mathfrak{c}_r^q \mid q \in D_r \setminus I_p\}$. The transition probabilities are at each construction step defined by Eq. (1) (see Section 2.1), with $\alpha = 1$ and $\beta = 0$.

We have seen above that, when ACO algorithms that are implemented in the HCF are applied to unconstrained problems, every setting of the pheromone values (given that they are probabilities) is a selection fix-point. However, what are the selection fix-points of, for example, a constrained problem such as the elementary subproblem of size $n = 3$ with the cost matrix shown in Eq. (45)? If we depict the vector of pheromone values in matrix form (i.e., in the same form as cost matrix C) we get

$$\begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 = 1 - \tau_1^1 - \tau_1^2 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 = 1 - \tau_2^1 - \tau_2^2 \\ \tau_3^1 = 1 - \tau_1^1 - \tau_2^1 & \tau_3^2 = 1 - \tau_1^2 - \tau_2^2 & \tau_3^3 = \tau_1^1 + \tau_1^2 + \tau_2^1 + \tau_2^2 - 1 \end{pmatrix}. \tag{47}$$

It is easy to verify that with the initial setting of the pheromone values to $1/n$, the equations in this matrix hold. Furthermore, the pheromone update in the HCF preserves this property. Therefore, we only have to care about the four pheromone values τ_1^1, τ_1^2, τ_2^1, and τ_2^2. Assuming one ant per iteration, it is clear that for the pheromone values τ_1^1 and τ_1^2 it holds that $\Delta_1^1 = \tau_1^1$ and $\Delta_1^2 = \tau_1^2$, respectively. This is because the choice of an item for the first position of the permutation to be built does not depend on any other decision. This shows that the crucial pheromone values are τ_2^1 and τ_2^2. The Δ-values for these pheromone values are:

$$\Delta_2^1 = \frac{\tau_1^2\tau_2^1}{1-\tau_2^2} + \frac{(1-\tau_1^1-\tau_1^2)\tau_2^1}{\tau_2^1+\tau_2^2}, \tag{48}$$

$$\Delta_2^2 = \frac{\tau_1^1\tau_2^2}{1-\tau_2^1} + \frac{(1-\tau_1^1-\tau_1^2)\tau_2^2}{\tau_2^1+\tau_2^2}. \tag{49}$$

As shown in [49], there are four solutions to the equations $\Delta_2^1 - \tau_2^1 = 0$ and $\Delta_2^2 - \tau_2^2 = 0$. Depending on the exact setting of the pheromone values τ_1^1, τ_1^2, and τ_1^3, exactly one of these solutions is a stable selection fix-point.[20] Fig. 6 shows examples of selection fix-points of different pheromone value settings.

The interesting question is, if and how these selection fix-points will influence the search process when more than one ant per iteration is used, that is, under competition conditions. Merkle and Middendorf applied the model M(PP, HCF-IB, $n_a = 2$), for example, to the restricted permutation problem P^{64}, where P is the elementary subproblem discussed above. Observing the evolution of $(\tau_2^1, \tau_2^2, \tau_2^3)$ and $(\tau_3^1, \tau_3^2, \tau_3^3)$ one can notice a clear bias introduced by the selection fix-points (which are changing, with changing pheromone values). This is shown in Fig. 7. Merkle and Middendorf observed that the influence of the selection fix-points—when the model is still far from convergence—increases with increasing problem size. This is due to the fact that with increasing problem size the influence of the competition between the ants on one elementary subproblem decreases, and the model approaches the behavior of the model which only uses one ant. Summarizing, we can conclude that—even if an algorithm/instance combination is a CBS (see Definition 2)—when applied to constrained problems the search process is influenced by a bias towards the selection fix-points.

Recently, Merkle and Middendorf [50] extended their work by introducing a pheromone update which they call *the competition controlled pheromone update.* This update is based on the observation that the decisions of an ant (during solution construction) do not all have the same importance. They introduced a measure, based on the Kullback–Leibler divergence [45], in order to determine this importance. Based on this measure, solution components that were chosen in decisions with higher importance receive proportionally more update than other solution components. The usefulness of this update, that was shown for an ACO model, has still to be tested in an implemented algorithm.

[20] Note that the stability of a selection fix-point can be determined by analyzing the eigenvalues of the Jacobian matrix of the vector function $[f_1, f_2] = [\Delta_2^1 - \tau_2^1, \Delta_2^2 - \tau_2^2]$.

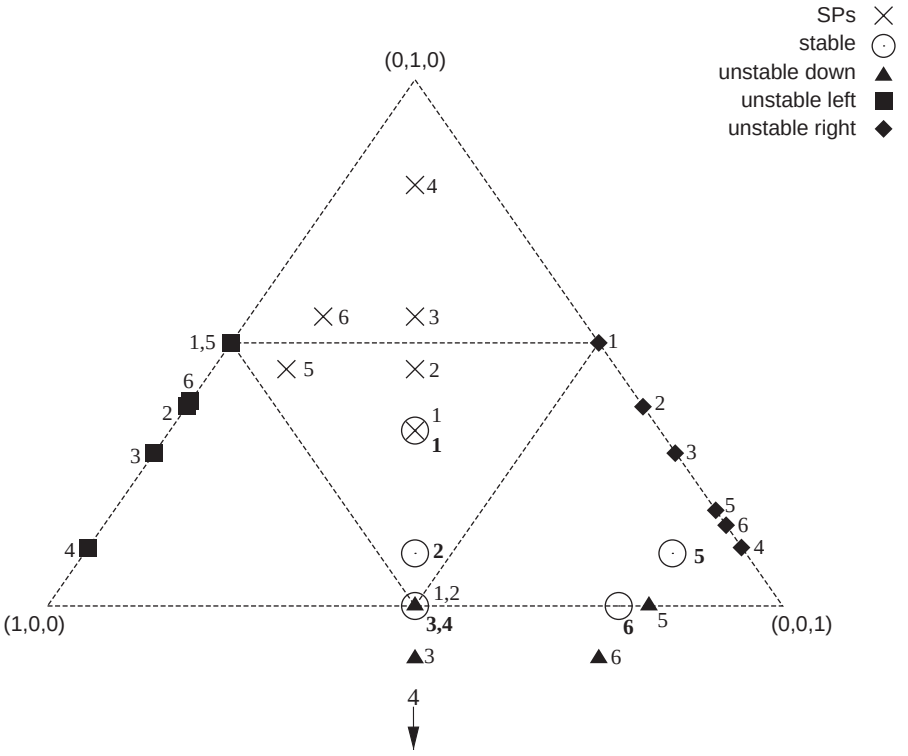

Fig. 6. The graphic shows the stable as well as the three unstable selection fix-points (SPs) (given by $(\tau_2^1, \tau_2^2, \tau_2^3)$) for six different settings of the first row $(\tau_1^1, \tau_1^2, \tau_1^3)$. All SPs and pheromone settings (which are triples of points) are shown in the following form: The first coordinate is the distance from the line between $(0, 1, 0)$ and $(0, 0, 1)$. The second coordinate is the distance from the line between $(1, 0, 0)$ and $(0, 0, 1)$, and the third coordinate is the distance from the line between $(0, 1, 0)$ and $(1, 0, 0)$. The inner triangle contains the triples with all coordinates $\leqslant \frac{1}{2}$. When all the coordinates are $\geqslant 0$, the corresponding point appears inside the big triangle. If not, the point is placed outside, as it happens for some of the unstable fix-points. Numbers denote the corresponding SPs and pheromone settings. The authors would like to express their thanks to Daniel Merkle and Martin Middendorf for providing this graphic which appeared in [49]. © MIT Press.

Open problem 5. *How does selection fix-point bias relate to the bias introduced by the fact that an algorithm/instance combination is not a competition-balanced system*? *Is, in such a case, also the selection bias a negative force*? *Can something be said about the nature of the selection fix-points for certain types of optimization problems*?

Open problem 6. *The development of new algorithmic components for ACO based on theoretical foundation* (*in the same spirit as the competition controlled pheromone update introduced in* [50]) *is an interesting research direction. The extraction of guidelines concerning the choice of ACO algorithmic components as a function of the characteristics of the considered CO problem could improve the applicability of ACO algorithms in practice*.

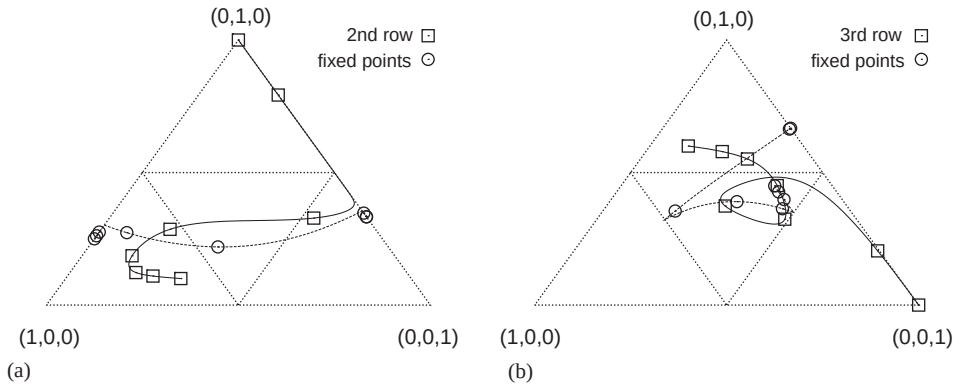

Fig. 7. (a) Evolution of $(\tau_2^1, \tau_2^2, \tau_2^3)$ (i.e., the 2nd row); (b) evolution of $(\tau_3^1, \tau_3^2, \tau_3^3)$ (i.e., the 3rd row). The graphics show the evolution of the pheromone values $(\tau_2^1, \tau_2^2, \tau_2^3)$ starting from (0.6, 0.1, 0.3) (in (a)), respectively $(\tau_3^1, \tau_3^2, \tau_3^3)$ starting from (0.3, 0.6, 0.1) (in (b)), of the model M(PP, HCF-IB, $n_a = 2$) applied to problem instance P^{64}. The pheromone values as well as the corresponding fix-points are shown at iterations 0, 10, 20, 50, 100, 200, 500, and 1000. The authors would like to express their thanks to Daniel Merkle and Martin Middendorf for providing these two graphics which appeared in [49]. © MIT Press.

6. Conclusions

In this paper we have over-viewed some recent efforts to develop a theory of ant colony optimization. After giving a brief introduction to the algorithms and problems considered in the overview, we have discussed convergence, presented connections between ACO algorithms and the stochastic gradient ascent and cross-entropy methods within the framework of model-based search, and finally discussed the influence of search bias on the working of ACO algorithms. For each of these different research directions we explicitly listed those that are, in our opinion, some of the most interesting open problem. As the ACO research field is currently flourishing, we expect to see many of these problems solved in the near future.

As a final comment, we note that ACO research is not only about theory. On the contrary, most of the field is concerned with experimental work. To the reader that, after learning the theoretical underpinnings of ACO as presented in this paper, becomes interested in the more practical aspects of the development of ACO algorithms, we suggest the recent book by Dorigo and Stützle [24]. This book describes in detail all the different types of ACO algorithms proposed in the literature, suggests how to apply them to different classes of combinatorial optimization problems and provides hints on how to efficiently implement them. Source code for ACO algorithms treated in the book is available for download in the software section of the ACO web-page (`http://www.aco-metaheuristic.org`).

Acknowledgements

We wish to thank Mauro Birattari, Nicolas Meuleau, Michael Sampels, Thomas Stützle, and Mark Zlochin for the time they spent with us discussing subjects strictly related to this

paper (and for writing with us many of the cited papers). Without them this paper would probably not exist. Additionally, we wish to thank Prasanna Balaprakash, Daniel Merkle and Paola Pellegrini, as well as the TCS editor Thomas Bäck, for the many useful comments and the careful proof-reading of a draft version of this paper.

References

[1] L.E. Baum, G.R. Sell, Growth transformations for functions on manifolds, Pacific J. Math. 27 (2) (1968) 211–227.
[2] M. Birattari, G. Di Caro, M. Dorigo, Toward the formal foundation of ant programming, in: M. Dorigo, G. Di Caro, M. Sampels (Eds.), Ant Algorithms, Proc. ANTS 2002, Third Internat. Workshop, Lecture Notes in Computer Science, Vol. 2463, Springer, Berlin, Germany, 2002, pp. 188–201.
[3] C. Blum, Theoretical and Practical Aspects of Ant Colony Optimization, Dissertations in Artificial Intelligence, Vol. 282, Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany, 2004.
[4] C. Blum, Beam-ACO—Hybridizing ant colony optimization with beam search: an application to open shop scheduling, Comput. Oper. Res. 32 (6) (2005) 1565–1591.
[5] C. Blum, M. Dorigo, Deception in ant colony optimization, in: M. Dorigo, M. Birattari, C. Blum, L.M. Gambardella, F. Mondada, T. Stützle (Eds.), Proc. ANTS 2004, Fourth Internat. Workshop on Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, Vol. 3172, Springer, Berlin, Germany, 2004, pp. 119–130.
[6] C. Blum, M. Dorigo, The hyper-cube framework for ant colony optimization, IEEE Trans. Systems, Man, Cybernet.-Part B 34 (2) (2004) 1161–1172.
[7] C. Blum, M. Dorigo, Search bias in ant colony optimization: on the role of competition-balanced systems, IEEE Trans. Evol. Comput. 9 (2) (2005) 159–174.
[8] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surveys 35 (3) (2003) 268–308.
[9] C. Blum, M. Sampels, Ant Colony Optimization for FOP shop scheduling: a case study on different pheromone representations, Proc. 2002 Congr. on Evolutionary Computation (CEC'02), Vol. 2, IEEE Computer Society Press, Los Alamitos, CA, 2002, pp. 1558–1563.
[10] C. Blum, M. Sampels, When model bias is stronger than selection pressure, in: J.J. Merelo Guervós et al. (Eds.), Proc. PPSN-VII, Seventh Internat. Conf. on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 2439, Springer, Berlin, Germany, 2002, pp. 893–902.
[11] C. Blum, M. Sampels, An ant colony optimization algorithm for shop scheduling problems, J. Math. Model. Algorithms 3 (3) (2004) 285–308.
[12] C. Blum, M. Sampels, M. Zlochin, On a particularity in model-based search, in: W.B. Langdon et al. (Eds.), Proc. Genetic and Evolutionary Computation Conf. (GECCO-2002), Morgan Kaufmann Publishers, San Francisco, CA, 2002, pp. 35–42.
[13] V. Černý, A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm, J. Optim. Theory Appl. 45 (1985) 41–51.
[14] D. Costa, A. Hertz, Ants can color graphs, J. Oper. Res. Soc. 48 (1997) 295–305.
[15] J.S. de Bonet, C.L. Isbell Jr., P. Viola, MIMIC: finding optima by estimating probability densities, in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), Adv. Neural Inform. Process. Systems, Vol. 7 (NIPS7), MIT Press, Cambridge, MA, 1997, pp. 424–431.
[16] K. Deb, D.E. Goldberg, Analyzing deception in trap functions, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms, Vol. 2, Morgan Kaufmann, San Mateo, CA, 1993, pp. 93–108.
[17] M.L. den Besten, T. Stützle, M. Dorigo, Ant colony optimization for the total weighted tardiness problem, in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.-P. Schwefel (Eds.), Proc. PPSN-VI, Sixth Internat. Conf. on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 1917, Springer, Berlin, Germany, 2000, pp. 611–620.
[18] J.-L. Deneubourg, S. Aron, S. Goss, J.-M. Pasteels, The self-organizing exploratory pattern of the argentine ant, J. Insect Behav. 3 (1990) 159–168.

[19] G. Di Caro, M. Dorigo, AntNet: distributed stigmergetic control for communications networks, J. Artif. Intel. Res. 9 (1998) 317–365.
[20] M. Dorigo, Optimization, learning and natural algorithms (in Italian), Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
[21] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[22] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Tech. Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.
[23] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Systems, Man, Cybernet.-Part B 26 (1) (1996) 29–41.
[24] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[25] M. Dorigo, M. Zlochin, N. Meuleau, M. Birattari, Updating ACO pheromones using stochastic gradient ascent and cross-entropy methods, in: S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, G.R. Raidl (Eds.), Applications of Evolutionary Computing, Proc. EvoWorkshops 2002, Lecture Notes in Computer Science, Vol. 2279, Springer, Berlin, Germany, 2002, pp. 21–30.
[26] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence Through Simulated Evolution, Wiley, New York, 1966.
[27] C. Gagné, W.L. Price, M. Gravel, Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times, J. Oper. Res. Soc. 53 (2002) 895–906.
[28] L.M. Gambardella, M. Dorigo, Ant colony system hybridized with a new local search for the sequential ordering problem, INFORMS J. Comput. 12 (3) (2000) 237–255.
[29] L.M. Gambardella, É.D. Taillard, G. Agazzi, MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 63–76.
[30] F. Glover, Tabu search—Part I, ORSA J. Comput. 1 (3) (1989) 190–206.
[31] F. Glover, Tabu search—Part II, ORSA J. Comput. 2 (1) (1990) 4–32.
[32] F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Norwell, MA, 2002.
[33] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Dordrecht, 1997.
[34] D.E. Goldberg, Simple genetic algorithms and the minimal deceptive problem, in: L. Davis (Ed.), Genetic Algorithms and Simulated Annealing, Pitman, London, UK, 1987, pp. 74–88.
[35] M. Guntsch, M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic TSP, in: E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, H. Tijink (Eds.), Applications of Evolutionary Computing: Proc. EvoWorkshops 2001, Lecture Notes in Computer Science, Vol. 2037, Springer, Berlin, Germany, 2001, pp. 213–222.
[36] W.J. Gutjahr, A generalized convergence result for the graph-based ant system metaheuristic, Tech. Report 99-09, Department of Statistics and Decision Support Systems, University of Vienna, Austria, 1999.
[37] W.J. Gutjahr, A graph-based ant system and its convergence, Future Gen. Comput. Systems 16 (9) (2000) 873–888.
[38] W.J. Gutjahr, ACO algorithms with guaranteed convergence to the optimal solution, Inform. Process. Lett. 82 (3) (2002) 145–153.
[39] J.H. Holland, Adaption in Natural and Artificial Systems, The University of Michigan Press, Ann Harbor, MI, 1975.
[40] H.H. Hoos, T. Stützle, Stochastic Local Search: Foundations and Applications, Elsevier, Amsterdam, The Netherlands, 2004.
[41] T. Jones, S. Forrest, Fitness distance correlation as a measure of problem difficulty for genetic algorithms, in: L.J. Eshelman (Ed.), Proc. 6th Internat. Conf. on Genetic Algorithms, Kaufman, LosAltos, CA, 1995, pp. 184–192.
[42] M.I. Jordan (Ed.), Learning in Graphical Models, MIT Press, Cambridge, MA, 1998.
[43] L. Kallel, B. Naudts, A. Rogers (Eds.), Theoretical Aspects of Evolutionary Computing, Natural Computing Series, Springer, Berlin, Germany, 2001.
[44] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[45] S. Kullback, Information Theory and Statistics, Wiley, New York, 1959.

[46] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, INFORMS J. Comput. 11 (4) (1999) 358–369.
[47] V. Maniezzo, A. Colorni, The ant system applied to the quadratic assignment problem, IEEE Trans. Data Knowledge Eng. 11 (5) (1999) 769–778.
[48] D. Merkle, M. Middendorf, Modelling ACO: composed permutation problems, in: M. Dorigo, G. Di Caro, M. Sampels (Eds.), Ant Algorithms, Proc. ANTS 2002, Third Internat. Workshop, Lecture Notes in Computer Science, Vol. 2463, Springer, Berlin, Germany, 2002, pp. 149–162.
[49] D. Merkle, M. Middendorf, Modelling the dynamics of ant colony optimization algorithms, Evol. Comput. 10 (3) (2002) 235–262.
[50] D. Merkle, M. Middendorf, Competition controlled pheromone update for ant colony optimization, in: M. Dorigo, M. Birattari, C. Blum, L.M. Gambardella, F. Mondada, T. Stützle (Eds.), Proc. ANTS 2004, Fourth Internat. Workshop on Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, Vol. 3172, Springer, Berlin, Germany, 2004, pp. 95–105.
[51] D. Merkle, M. Middendorf, H. Schmeck, Ant colony optimization for resource-constrained project scheduling, IEEE Trans. Evol. Comput. 6 (4) (2002) 333–346.
[52] N. Meuleau, M. Dorigo, Ant colony optimization and stochastic gradient descent, Artif. Life 8 (2) (2002) 103–121.
[53] T. Mitchell, Machine Learning, McGraw-Hill, Boston, 1997.
[54] J. Montgomery, M. Randall, T. Hendtlass, Search bias in constructive metaheuristics and implications for ant colony optimization, in: M. Dorigo, M. Birattari, C. Blum, L.M. Gambardella, F. Mondada, T. Stützle (Eds.), Proc. ANTS 2004, Fourth Internat. Workshop on Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, Vol. 3172, Springer, Berlin, Germany, 2004, pp. 391–398.
[55] H. Mühlenbein, G. Paaß. From recombination of genes to the estimation of distributions, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Proc. 4th Conf. on Parallel Problem Solving from Nature, PPSN IV, Lecture Notes in Computer Science, Vol. 1411, Springer, Berlin, 1996, pp. 178–187.
[56] C.H. Papadimitriou, K. Steiglitz, Combinatorial Optimization—Algorithms and Complexity, Dover Publications Inc., New York, 1982.
[57] A. Prügel-Bennett, A. Rogers, Modelling genetic algorithm dynamics, in: L. Kallel et al. (Eds.), Theoretical Aspects of Evolutionary Computation, Natural Computing Series, Springer, Berlin, Germany, 2001, pp. 59–85.
[58] I. Rechenberg, Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, Frommann-Holzboog, 1973.
[59] M. Reimann, K. Doerner, R.F. Hartl, D-ants: savings based ants divide and conquer the vehicle routing problems, Comput. Oper. Res. 31 (4) (2004) 563–591.
[60] G.E. Robbins, H. Monroe, A stochastic approximation method, Ann. Math. Statist. 22 (1951) 400–407.
[61] R.Y. Rubinstein, Combinatorial optimization, cross-entropy, ants and rare events, in: Stochastic Optimization: Algorithms and Applications, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 303–364.
[62] K. Socha, ACO for continuous and mixed-variable optimization, in: M. Dorigo, M. Birattari, C. Blum, L.M. Gambardella, F. Mondada, T. Stützle (Eds.), Proc. ANTS 2004, Fourth Internat. Workshop on Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, Vol. 3172, Springer, Berlin, Germany, 2004, pp. 25–36.
[63] K. Socha, M. Sampels, M. Manfrin, Ant algorithms for the university course timetabling problem with regard to the state-of-the-art, in: G. Raidl et al. (Ed.), Applications of Evolutionary Computing, Proc. EvoWorkshops 2003, Vol. 2611, 2003, pp. 334–345.
[64] T. Stützle, An ant approach to the flow shop problem, in: Fifth European Congr. on Intelligent Techniques and Soft Computing, EUFIT'98, 1998, pp. 1560–1564.
[65] T. Stützle, M. Dorigo, A short convergence proof for a class of ACO algorithms, IEEE Trans. Evol. Comput. 6 (4) (2002) 358–365.
[66] T. Stützle, H.H. Hoos, $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, Future Gen. Comput. Systems 16 (8) (2000) 889–914.
[67] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo, Model-based search for combinatorial optimization: a critical survey, Ann. Oper. Res. 131 (1–4) (2004) 373–395.