# Chapter 8
# Ant Colony Optimization:
# Overview and Recent Advances

Marco Dorigo and Thomas Stützle

**Abstract** Ant Colony Optimization (ACO) is a metaheuristic that is inspired by the pheromone trail laying and following behavior of some ant species. Artificial ants in ACO are stochastic solution construction procedures that build candidate solutions for the problem instance under concern by exploiting (artificial) pheromone information that is adapted based on the ants' search experience and possibly available heuristic information. Since the proposal of the Ant System, the first ACO algorithm, many significant research results have been obtained. These contributions focused on the development of high-performing algorithmic variants, the development of a generic algorithmic framework for ACO algorithms, successful applications of ACO algorithms to a wide range of computationally hard problems, and the theoretical understanding of properties of ACO algorithms. This chapter reviews these developments and gives an overview of recent research trends in ACO.

## 8.1 Introduction

Ant Colony Optimization (ACO) [57, 59, 66] is a metaheuristic for solving hard combinatorial optimization problems. The inspiring source of ACO is the pheromone trail laying and following behavior of real ants, which use pheromones as a communication medium. In analogy to the biological example, ACO is based on indirect communication within a colony of simple agents, called (artificial) ants, mediated by (artificial) pheromone trails. The pheromone trails in ACO serve as a distributed, numerical information, which the ants use to probabilistically construct solutions to the problem being solved and which the ants adapt during the algorithm's execution to reflect their search experience.

Marco Dorigo and Thomas Stützle
IRIDIA, Université Libre de Bruxelles (ULB), Brussels, Belgium
e-mail: {mdorigo, stuetzle}@ulb.ac.be

The first example of such an algorithm is Ant System (AS) [55, 63–65], which was proposed using as example application the well-known traveling salesman problem (TSP) [6, 99, 128]. Despite encouraging initial results, AS could not compete with state-of-the-art algorithms for the TSP. Nevertheless, it had the important role of stimulating further research both on algorithmic variants, which obtain much better computational performance, and on applications to a large variety of different problems. In fact, there exist now a considerable number of applications of such algorithms where world class performance is obtained. Examples are applications of ACO algorithms to problems such as sequential ordering [76], scheduling [18], assembly line balancing [19], probabilistic TSP [7], 2D-HP protein folding [132], DNA sequencing [25], protein–ligand docking [98], and packet-switched routing in Internet-like networks [47]. The ACO metaheuristic provides a common framework for the existing applications and algorithmic variants [57, 59]. Algorithms which follow the ACO metaheuristic are called ACO algorithms.

The (artificial) ants in ACO implement a randomized construction heuristic which makes probabilistic decisions as a function of artificial pheromone trails and possibly available heuristic information based on the input data of the problem to be solved. As such, ACO can be interpreted as an extension of traditional construction heuristics, which are readily available for many combinatorial optimization problems. Yet, an important difference with construction heuristics is the adaptation of the pheromone trails during algorithm execution to take into account the cumulated search experience.

The rest of this chapter is organized as follows. In Section 8.2, we briefly overview construction heuristics and local search algorithms. In Section 8.3, we present a specific version of the ACO metaheuristic that focuses on applications to $\mathcal{NP}$-hard problems. Section 8.4 outlines the inspiring biological analogy and describes the historical developments leading to ACO. In Section 8.5, we illustrate how the ACO metaheuristic can be applied to different types of problems and we give an overview of its successful applications. Section 8.6 gives an overview of recent developments in ACO and Section 8.7 concludes the chapter.

## 8.2 Approximate Approaches

Many important combinatorial optimization problems are hard to solve. The notion of problem hardness is captured by the theory of computational complexity [79, 123] and for many important problems it is well known that they are $\mathcal{NP}$-hard, that is, the time needed to solve an instance in the worst case grows exponentially with instance size. Often, approximate algorithms are the only feasible way to obtain near-optimal solutions at relatively low computational cost.

Most approximate algorithms are either *construction* algorithms or *local search* algorithms.[1] These two types of methods are significantly different, because

---

[1] Other approximate methods are also conceivable. For example, when stopping exact methods, like Branch & Bound, before completion [10, 95] (for example, after some given time bound,

construction algorithms work on partial solutions trying to extend these in the best possible way to complete problem solutions, while local search methods move in the search space of complete solutions.

## 8.2.1 Construction Algorithms

Construction algorithms build solutions to a problem under consideration in an incremental way starting with an empty initial solution and iteratively adding appropriate solution components without backtracking until a complete solution is obtained. In the simplest case, solution components are added in random order. Often better results are obtained if a heuristic estimate of the myopic benefit of adding solution components is taken into account. *Greedy construction heuristics* add at each step a solution component that achieves the maximal myopic benefit as measured by some heuristic information. An algorithmic outline of a greedy construction heuristic is given in Figure 8.1. The function `GreedyComponent` returns the solution component $e$ with the best heuristic estimate as a function of the current partial solution $s_p$. Solutions returned by greedy algorithms are typically of (much) better quality than randomly generated solutions. Yet, a disadvantage of greedy construction heuristics is that they typically generate only a limited number of different solutions. Additionally, greedy decisions in early stages of the construction process constrain the available choices at later stages, often causing very poor moves in the final phases of the solution construction.

**procedure** *Greedy Construction Heuristic*
    $s_p = empty\ solution$
    **while** $s_p$ not_a_complete_solution **do**
       $e = $ `GreedyComponent`$(s_p)$
       $s_p = s_p \otimes e$
    **end**
    **return** $s_p$
**end** *Greedy Construction Heuristic*

**Fig. 8.1** Algorithmic skeleton of a greedy construction heuristic. The addition of component $e$ to a partial solution $s_p$ is denoted by the operator $\otimes$.

As an example, consider a greedy construction heuristic for the TSP. In the TSP we are given a complete weighted graph $G = (V,E)$ with $V$ being the set of vertices, representing the cities, and $E$ the set of edges fully connecting the vertices. Each edge is assigned a value $d_{ij}$, which is the length of edge $(i,j) \in E$. The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where an Hamiltonian circuit is a closed tour visiting exactly once each of the $n = |V|$ vertices of $G$. For symmetric TSPs, the distances between the cities are independent of the

---

or when some guarantee on the solution quality is obtained through the use of lower and upper bounds), we can convert exact algorithms into approximate ones.

direction of traversing the edges, that is, $d_{ij} = d_{ji}$ for every pair of vertices. In the more general asymmetric TSP (ATSP) at least for one pair of vertices $i, j$ we have $d_{ij} \neq d_{ji}$.

A simple rule of thumb to build a tour is to start from some initial city and to always choose to go to the closest still unvisited city before returning to the start city. This algorithm is known as the *nearest neighbor* tour construction heuristic.

Construction algorithms are typically the fastest approximate methods, but the solutions they generate often are not of a very high quality and they are not guaranteed to be optimal with respect to small changes; the results produced by constructive heuristics can therefore often be improved by local search algorithms.

### 8.2.2 Local Search Algorithms

Local search algorithms start from a complete initial solution and try to find a better solution in an appropriately defined *neighborhood* of the current solution. In its most basic version, known as *iterative improvement*, the algorithm searches the neighborhood for an improving solution. If such a solution is found, it replaces the current solution and the local search continues. These steps are repeated until no improving neighbor solution can be found and the algorithm ends in a *local optimum*. An outline of an iterative improvement algorithm is given in Figure 8.2. The procedure `Improve` returns a better neighbor solution if one exists, otherwise it returns the current solution, in which case the algorithm stops.

> **procedure** *IterativeImprovement (s ∈ S)*
>     $s' =$ `Improve(`$s$`)`
>     **while** $s' \neq s$ **do**
>         $s = s'$
>         $s' =$ `Improve(`$s$`)`
>     **end**
>     **return** $s$
> **end** *IterativeImprovement*

**Fig. 8.2** Algorithmic skeleton of iterative improvement. As input is given a complete solution $s$ in search space $S$.

The choice of an appropriate neighborhood structure is crucial for the performance of local search algorithms and has to be done in a problem-specific way. The neighborhood structure defines the set of solutions that can be reached from $s$ in one single step of the algorithm. An example neighborhood for the TSP is the *k-exchange* neighborhood in which neighbor solutions differ by at most $k$ edges. Figure 8.3 shows an example of a 2-exchange neighborhood. The 2-exchange algorithm systematically tests whether the current tour can be improved by replacing two edges. To fully specify a local search algorithm, it is necessary to designate
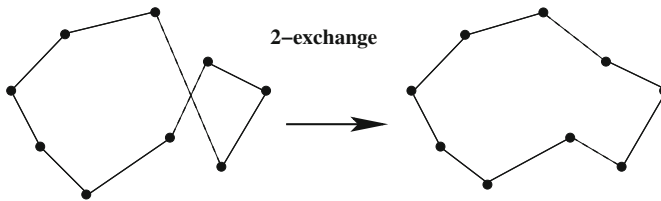
**Fig. 8.3** Schematic illustration of a 2-exchange move. The proposed move reduces the total tour length if we consider the Euclidean distance between the points.

a particular neighborhood examination scheme that defines how the neighborhood is searched and which neighbor solution replaces the current one. In the case of iterative improvement algorithms, this rule is called the *pivoting rule* [157] and examples are the *best-improvement* rule, which chooses the neighbor solution giving the largest improvement of the objective function, and the *first-improvement* rule, which uses the first improved solution found when scanning the neighborhood to replace the current one. A common problem with local search algorithms is that they easily get trapped in local minima and that the result strongly depends on the initial solution.

## 8.3 The ACO Metaheuristic

Artificial ants used in ACO are stochastic solution construction procedures that probabilistically build a solution by iteratively adding solution components to partial solutions by taking into account (i) heuristic information about the problem instance being solved, if available, and (ii) (artificial) pheromone trails which change dynamically at runtime to reflect the agents' acquired search experience.

A stochastic component in ACO allows the ants to build a wide variety of different solutions and hence to explore a much larger number of solutions than greedy heuristics. At the same time, the use of heuristic information, which is readily available for many problems, can guide the ants toward the most promising solutions. More important, the ants' search experience can be used to influence, in a way reminiscent of reinforcement learning [149], the solution construction in future iterations of the algorithm. Additionally, the use of a colony of ants can give the algorithm increased robustness and in many ACO applications the collective interaction of a population of agents is needed to efficiently solve a problem.

The domain of application of ACO algorithms is vast. In principle, ACO can be applied to any discrete optimization problem for which some solution construction mechanism can be conceived. In Section 8.3.1, we first define a generic problem representation that the ants in ACO may exploit to construct solutions, and then we define the ACO metaheuristic.

### *8.3.1 Problem representation*

Let us consider minimization problems[2] and define a general model of a combina-
torial optimization problem.

**Definition 8.1** *A model $P = (S, \Omega, f)$ of a combinatorial optimization problem con-
sists of*

- *a search space $S$ that is defined by a finite set of decision variables, each with a
  finite domain, and a set $\Omega$ of constraints among the variables;*
- *an objective function $f : S \mapsto \mathbf{R}_0^+$ to be minimized.*

The search space is defined by a finite set of variables $X_i, i = 1, \ldots, n$, each having
an associated domain $D_i$ of values that can be assigned to it. An instantiation of a
variable consists of an assignment of a value $v_i^j \in D_i$ to variable $X_i$ and it is denoted
by $X_i = v_i^j$. A feasible solution $s \in S$ is an assignment to each variable of a value in its
domain such that all the problem constraints in $\Omega$ are satisfied. If $\Omega$ is empty, then
the problem is unconstrained and each decision variable can take any value from
its domain, independent of the other variables. In this case, $P$ is an *unconstrained*
problem model; otherwise it is called *constrained*. A feasible solution $s^* \in S$ is called
a global minimum of $P$ if and only if we have that $f(s^*) \leq f(s) \; \forall s \in S$. We denote
by $S^* \subseteq S$ the set of all global minima.                                          □

   This model of a combinatorial optimization problem can be directly used to de-
rive a generic pheromone model that is exploited by ACO algorithms. To see how, let
us call the instantiation of a variable $X_i$ with a particular value $v_i^j$ of its domain a so-
lution component, which is denoted by $c_i^j$. Ants then need to appropriately combine
solution components to form high-quality, feasible solutions. To do so, each solution
component $c_i^j$ will have an associated pheromone variable $\mathrm{T}_{ij}$. We denote the set of
all solution components by $C$ and the set of all pheromone variables by $\mathrm{T}$. Each
pheromone variable $\mathrm{T}_{ij}$ has a pheromone value $\tau_{ij}$; this value indicates the desir-
ability of choosing solution component $c_i^j$. Note that, as said before, the pheromone
values are time varying and therefore they are a function of the algorithm iteration
$t$. In what follows we will, however, omit the reference to the iteration counter and
write simply $\tau_{ij}$ instead of $\tau_{ij}(t)$.

   As an example of this formalization, consider the TSP. In this case, the solution
components are the moves from one city to another one. This can be formalized
by associating one variable to each city. Each variable $X_i$ has then associated $n-1$
values, $j = 1, \ldots, n, \; j \neq i$. As a result, a pheromone value $\tau_{ij}$ is associated to each
edge connecting two cities. An instantiation of the decision variables corresponds
to a feasible solution, if and only if the set of edges corresponding to the values
of the decision variables forms a Hamiltonian cycle. (Note that for the TSP it is
easy to guarantee that ants generate feasible solutions.) The objective function $f(\cdot)$
computes for each feasible solution the sum of the edge lengths, that is, the length
of the Hamiltonian cycle.

---

[2] The adaptation to maximization problems is straightforward.

## 8.3.2 The Metaheuristic

A general outline of the ACO metaheuristic for applications to static combinatorial optimization problems[3] is given in Figure 8.4. After initializing parameters and pheromone trails, the main loop consists of three main steps. First, $m$ ants construct solutions to the problem instance under consideration, biased by the pheromone information and possibly by the available heuristic information. Once the ants have completed their solutions, these may be improved in an optional local search phase. Finally, before the start of the next iteration, the pheromone trails are adapted to reflect the search experience of the ants. The main steps of the ACO metaheuristic are explained in more detail in the following.

**procedure** *ACO algorithm for combinatorial optimization problems*
    `Initialization`
    **while** (termination condition not met) **do**
        ConstructAntSolutions
        `ApplyLocalSearch`     % optional
        UpdatePheromones
    **end**
**end** *ACO algorithm for combinatorial optimization problems*

**Fig. 8.4** Algorithmic skeleton for ACO algorithms applied to combinatorial optimization problems. The application of a local search algorithm is a typical example of a possible daemon action in ACO algorithms.

`Initialization`. At the start of the algorithm, parameters are set and all pheromone variables are initialized to a value $\tau_0$, which is a parameter of the algorithm.

`ConstructAntSolutions`. A set of $m$ ants constructs solutions to the problem instance being tackled. To do so, each ant starts with an initially empty solution $s_p = \emptyset$. At each construction step, an ant extends its current partial solution $s_p$ by choosing one feasible solution component $c_i^j \in N(s_p) \subseteq C$ and adding it to its current partial solution. $N(s_p)$ is the set of solution components that may be added while maintaining feasibility and it is defined implicitly by a solution construction process that the ants implement. If a partial solution cannot be extended maintaining feasibility, it depends on the particular construction mechanism whether the solution construction is abandoned or an infeasible, complete solution is constructed. In the latter case, infeasible solutions may be penalized in dependence of the degree to which they violate problem constraints.

---

[3] Static problems are those whose topology and costs do not change while they are being solved. This is the case, for example, for the classic TSP, in which city locations and intercity distances do not change during the algorithm's runtime. In contrast, in dynamic problems the topology and costs can change while solutions are built. An example of such a problem is routing in telecommunications networks [47], in which traffic patterns change all the time.

The choice of the solution component to add is done probabilistically at each construction step. Various ways for defining the probability distributions have been considered. The most widely used rule is that of Ant System (AS) [65]:

$$p\left(c_i^j|s_p\right) = \frac{\tau_{ij}^{\alpha} \cdot \left[\eta\left(c_i^j\right)\right]^{\beta}}{\sum_{c_i^l \in N(s_p)} \tau_{il}^{\alpha} \cdot \left[\eta\left(c_i^l\right)\right]^{\beta}}, \quad \forall c_i^j \in N(s_p), \tag{8.1}$$

where $\eta(\cdot)$ is a function that assigns to each feasible solution component $c_i^j \in N(s_p)$ a heuristic value, which is usually called the heuristic information. Parameters $\alpha$ and $\beta$ determine the relative influence of the pheromone trails and the heuristic information and have the following influence on the algorithm behavior. If $\alpha = 0$, the selection probabilities are proportional to $[\eta_{ij}]^{\beta}$ and a solution component with a high heuristic value will more likely be selected: this case corresponds to a stochastic greedy algorithm. If $\beta = 0$, only pheromone amplification is at work.

`ApplyLocalSearch`. Once complete candidate solutions are obtained, these may further be improved by applying local search algorithms. In fact, for a wide range of combinatorial optimization problems, ACO algorithms reach best performance when coupled with local search algorithms [66]. More generally, local search is one example of what have been called *daemon actions* [57, 59]. These are used to implement problem specific or centralized actions that cannot be performed by individual ants.

`UpdatePheromones`. The pheromone update is intended to make solution components belonging to good solutions more desirable for ants operating in the following iterations. There are essentially two mechanisms that are used to achieve this goal. The first is pheromone deposit, which increases the level of the pheromone of solution components that are associated with a chosen set $S_{\text{upd}}$ of good solutions. The second is pheromone trail evaporation, which is the mechanism that decreases over time the pheromone deposited by previous ants. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a suboptimal region. It implements a useful form of *forgetting*, favoring the exploration of new areas of the search space. The pheromone update is commonly implemented as:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{\text{upd}}|c_i^j \in s} g(s), \tag{8.2}$$

where $S_{\text{upd}}$ is the set of solutions that are used to deposit pheromone, $\rho \in (0,1]$ is a parameter called evaporation rate, $g(\cdot) : S \mapsto \mathbb{R}^+$ is a function such that $f(s) < f(s') \Rightarrow g(s) \geq g(s')$. It determines the quality of a solution and is commonly called *evaluation function*.

ACO algorithms typically differ in the way pheromone update is implemented: different specifications of how to determine $S_{\text{upd}}$ result in different instantiations of update rule 8.2. Typically, $S_{\text{upd}}$ is a subset of $S_{\text{iter}} \cup \{s_{\text{gb}}\}$, where $S_{\text{iter}}$ is the set of all solutions constructed in the current iteration of the main loop and $s_{\text{gb}}$ is the best solution found since the start of the algorithm (gb stands for global-best).

## 8.4 History

The first ACO algorithm to be proposed was Ant System (AS). AS was applied to some rather small TSP instances with up to 75 cities. It was able to reach the performance of other general-purpose heuristics like evolutionary computation [55, 65]. Despite these initial encouraging results, AS did not prove to be competitive with state-of-the-art algorithms specifically designed for the TSP. Therefore, a substantial amount of research in ACO has focused on extending/modifying AS to improve its performance. In Section 8.4.1 we first briefly introduce the biological metaphor by which AS and ACO are inspired, and then we present a brief history of the early developments that have led from the original AS to more performing ACO algorithms.

### 8.4.1 Biological Analogy

In many ant species, individual ants may deposit a pheromone (a chemical that ants can smell) on the ground while walking [43, 80]. By depositing pheromone, ants create a trail that is used, for example, to mark the path from the nest to food sources and back. Foragers can sense the pheromone trails and follow the path to food discovered by other ants. Several ant species are capable of exploiting pheromone trails to determine the shortest among the available paths leading to the food.

Deneubourg and colleagues [43, 80] used a double bridge connecting a nest of ants and a food source to study pheromone trail laying and following behavior in controlled experimental conditions.[4] They ran a number of experiments in which they varied the ratio between the length of the two branches of the bridge. The most interesting, for our purposes, of these experiments is the one in which one branch was longer than the other. In this experiment, at the start the ants were left free to move between the nest and the food source and the percentage of ants that chose one or the other of the two branches was observed over time. The outcome was that, although in the initial phase random oscillations could occur, in most experiments all the ants ended up using the shorter branch.

This result can be explained as follows. When a trial starts there is no pheromone on the two branches. Hence, the ants do not have a preference and they select with the same probability either of the two branches. It can be expected that, on average, half of the ants choose the short branch and the other half the long branch, although stochastic oscillations may occasionally favor one branch over the other. However, because one branch is shorter than the other, the ants choosing the short branch are the first to reach the food and to start their travel back to the nest.[5] But then, when they must make a decision between the short and the long branch, the higher

---

[4] The experiment described was originally executed using a laboratory colony of Argentine ants (*Iridomyrmex humilis*). It is known that these ants deposit pheromone both when leaving and when returning to the nest [80].

[5] In the ACO literature, this is often called *differential path length* effect.

level of pheromone on the short branch biases their decision in its favor.[6] Therefore, pheromone starts to accumulate faster on the short branch, which will eventually be used by the great majority of the ants.

It should be clear by now how real ants have inspired AS and later algorithms: the double bridge was substituted by a graph, and pheromone trails by artificial pheromone trails. Also, because we wanted artificial ants to solve problems more complicated than those solved by real ants, we gave artificial ants some extra capacities, like a memory (used to implement constraints and to allow the ants to retrace their solutions without errors) and the capacity for depositing a quantity of pheromone proportional to the quality of the solution produced (a similar behavior is observed also in some real ant species in which the quantity of pheromone deposited while returning to the nest from a food source is proportional to the quality of the food source [9]).

In Section 8.4.2 we will see how, starting from AS, new algorithms have been proposed that, although retaining some of the original biological inspiration, are less and less biologically inspired and more and more motivated by the need of making ACO algorithms competitive with other state-of-the-art algorithms. Nevertheless, many aspects of the original Ant System remain: the need for a colony, the role of autocatalysis, the cooperative behavior mediated by artificial pheromone trails, the probabilistic construction of solutions biased by artificial pheromone trails and local heuristic information, the pheromone updating guided by solution quality, and the evaporation of pheromone trails are present in all ACO algorithms.

## 8.4.2 Historical Development

As said, AS was the first ACO algorithm to be proposed in the literature. In fact, AS was originally a set of three algorithms called *ant-cycle*, *ant-density*, and *ant-quantity*. These three algorithms were proposed in Dorigo's doctoral dissertation [55] and first appeared in a technical report [63, 64] that was published a few years later in the IEEE Transactions on Systems, Man, and Cybernetics [65]. Other early publications are [34, 35].

While in *ant-density* and *ant-quantity* the ants updated the pheromone directly after a move from a city to an adjacent one, in *ant-cycle* the pheromone update was only done after all the ants had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality. Because *ant-cycle* performed better than the other two variants, it was later called simply Ant System (and in fact, it is the algorithm that we will present in Section 8.4.2.1), while the other two algorithms were no longer studied.

The major merit of AS, whose computational results were promising but not competitive with other more established approaches, was to stimulate a number of

---

[6] A process like this, in which a decision taken at time $t$ increases the probability of making the same decision at time $T > t$ is said to be an *autocatalytic* process. Autocatalytic processes exploit *positive feedback*.

researchers to develop extensions and improvements of its basic ideas so as to produce better performing, and often state-of-the-art, algorithms.

### 8.4.2.1  The First ACO Algorithm: Ant System and the TSP

The TSP is a paradigmatic $\mathcal{NP}$-hard combinatorial optimization problem that has attracted an enormous amount of research effort [6, 94, 99]. The TSP is a very important problem also in the context of Ant Colony Optimization because it is the problem to which the original AS was first applied [55, 63–65], and it has later often been used as a benchmark to test new ideas and algorithmic variants.

In AS each ant is initially put on a randomly chosen city and has a memory, in which the partial solution it has constructed so far is stored (initially the memory contains only the start city). Starting from its start city, an ant iteratively moves from city to city, which corresponds to adding iteratively solution components as explained in Section 8.3.2. When being at a city $i$, an ant $k$ chooses to go to an as yet unvisited city $j$ with a probability given by Equation (8.1). The heuristic information is given by $\eta_{ij} = 1/d_{ij}$ and $N(s_p)$ is the set of cities that ant $k$ has not yet visited.

The solution construction ends after each ant has completed a tour, that is, after each ant has constructed a sequence of length $n$, corresponding to a permutation of the city indices. Next, the pheromone trails are updated. In AS this is done by using Equation (8.2), where we have

$$S_{\mathrm{upd}} = S_{\mathrm{iter}} \tag{8.3}$$

and

$$g(s) = 1/f(s), \tag{8.4}$$

where $f(s)$ is the length of the tour $s$. Hence, the shorter the ant's tour is, the more pheromone is received by edges (solution components) belonging to the tour.[7] In general, edges which are used by many ants and which are contained in shorter tours will receive more pheromone and therefore are also more likely to be chosen in future iterations of the algorithm.

### 8.4.2.2  Ant System and Its Extensions

As previously stated, AS was not competitive with state-of-the-art algorithms for the TSP. Researchers then started to extend it to try to improve its performance.

A first improvement, called the *elitist strategy*, was introduced in [55, 65]. It consists in giving the best tour since the start of the algorithm (called $s_{\mathrm{gb}}$) a strong additional weight. In practice, each time the pheromone trails are updated

---

[7] Note that when applied to symmetric TSPs the edges are considered to be bidirectional and edges $(i, j)$ and $(j, i)$ are both updated. This is different for the ATSP, where edges are directed; in this case an ant crossing edge $(i, j)$ will update only this edge and not edge $(j, i)$.

by Equation (8.2), we have that $S_{\text{upd}} = S_{\text{iter}} \cup \{s_{\text{gb}}\}$ and that $g(s), s \neq s_{\text{gb}}$, is given by Equation (8.4). For $s_{\text{gb}}$ we have that $g(s_{\text{gb}}) = e/f(s_{\text{gb}})$, where $e$ is a positive integer. Note that this type of pheromone update is a first example of daemon action as described in Section 8.3.2.

Other improvements were rank-based Ant System ($\text{AS}_{\text{rank}}$), *MAX–MIN* Ant System (*MM*AS), and Ant Colony System (ACS). $\text{AS}_{\text{rank}}$ [30] is in a sense an extension of the elitist strategy: it sorts the ants according to the lengths of the tours they generated and, after each tour construction phase, only the $(w - 1)$ best ants and the global-best ant are allowed to deposit pheromone. The $r$th best ant of the colony contributes to the pheromone update with a weight given by $\max\{0, w - r\}$ while the global-best tour reinforces the pheromone trails with weight $w$. This can easily be implemented by an appropriate choice of $S_{\text{upd}}$ and $g(s)$ in Equation (8.2).

*MM*AS [144, 147, 148] introduces upper and lower bounds to the values of the pheromone trails, as well as a different initialization of their values. In practice, in *MM*AS the allowed range of the pheromone trail strength is limited to the interval $[\tau_{\min}, \tau_{\max}]$, that is, $\tau_{\min} \leq \tau_{ij} \leq \tau_{\max} \; \forall \tau_{ij}$, and the pheromone trails are initialized to the upper trail limit, which causes a higher exploration at the start of the algorithm. In [144, 148] it is discussed how to set the upper and lower pheromone trail limits in a principled way. Pheromone updates are performed using a strong elitist strategy: only the best solution generated is allowed to update pheromone trails. This can be the *iteration-best* solution, that is, the best in the current iteration, or the *global-best* solution. The amount of pheromone deposited is then given by $g(s_b) = 1/f(s_b)$, where $s_b$ is either $s_{\text{ib}}$, the iteration-best solution, or $s_{\text{gb}}$. In fact, in *MM*AS the iteration-best ant and the global-best ant can be used alternately in the pheromone update. Computational results have shown that best results are obtained when pheromone updates are performed using the global-best solution with increasing frequency during the algorithm execution [144, 148]. As an additional means for increasing the explorative behavior of *MM*AS, occasional pheromone trail reinitialization is used. *MM*AS has been improved also by the addition of local search routines that take the solution generated by ants to their local optimum just before the pheromone update.

ACS [60, 61, 75] improves over AS by increasing the importance of exploitation of information collected by previous ants with respect to exploration of the search space.[8] This is achieved via two mechanisms. First, a strong elitist strategy is used to update pheromone trails. Second, ants choose a solution component (that is, the next city in the TSP case) using the so-called pseudo-random proportional rule [61]: with probability $q_0$, $0 \leq q_0 < 1$, they move to the city $j$ for which the product between pheromone trail and heuristic information is maximum, that is, $j = \arg\max_{c_i^j \in N(s_p)} \left\{ \tau_{ij} \cdot \eta_{ij}^{\beta} \right\}$, while with probability $1 - q_0$ they operate a biased exploration in which the probability $p_{ij}(t)$ is the same as in AS (see Equation (8.1)).

---

[8] ACS was an offspring of Ant-Q [74], an algorithm intended to create a link between reinforcement learning [149] and Ant Colony Optimization. Computational experiments have shown that some aspects of Ant-Q, in particular the pheromone update rule, could be strongly simplified without affecting performance. It is for this reason that Ant-Q was abandoned in favor of the simpler and equally good ACS.

The value $q_0$ is a parameter: when it is set to a value close to 1, as it is the case in most ACS applications, exploitation is favored over exploration. Obviously, when $q_0 = 0$ the probabilistic decision rule becomes the same as in AS.

Also, as in *MM*AS, in ACS only the best ant (the global-best or the iteration-best ant) is allowed to add pheromone after each algorithm iteration; the former is the most common choice in applications of ACS. The amount of pheromone deposited is then given by $g(s_b) = \rho/f(s_{gb})$, where $\rho$ is the pheromone evaporation.

Finally, ACS also differs from most ACO algorithms because ants update the pheromone trails while building solutions (as it was the case in *ant-quantity* and in *ant-density*, see [55, 63, 64]). In practice, ACS ants "eat" some of the pheromone trail on the edges they visit. This has the effect of decreasing the probability that the same path is used by all ants (that is, it favors exploration, counterbalancing this way the other two above-mentioned modifications that strongly favor exploitation of the collected knowledge about the problem). Similar to *MM*AS, ACS also usually exploits local search to improve its performance.

One could continue by enumerating the modifications that have been proposed in various other ACO algorithms that have been reported in the literature. Instead, we refer the interested reader to [66] and here we just give a brief overview of the various developments on ACO algorithms for $\mathcal{NP}$-hard problems. In Table 8.1 we give for each of the main ACO variants that have been proposed, the main references to these algorithms, the year in which they have been proposed, and whether they have been tested on the TSP. In fact (published) tests of most ACO variants have been done on the TSP, which again confirms the central role of this problem in ACO research.

**Table 8.1** Overview of the main ACO algorithms for $\mathcal{NP}$-hard problems that have been proposed in the literature. Given are the ACO algorithm name, the main references where these algorithms are described, the year they first have been published, and whether the corresponding algorithms have been tested on the TSP.

| ACO algorithm | Main references | Year | TSP |
|---|---|---|---|
| Ant System | [55, 63, 65] | 1991 | Yes |
| Elitist AS | [55, 63, 65] | 1992 | Yes |
| Ant-Q | [74] | 1995 | Yes |
| Ant Colony System | [60, 61, 75] | 1996 | Yes |
| *MM*AS | [146–148] | 1996 | Yes |
| Rank-based AS | [29, 30] | 1997 | Yes |
| ANTS | [104, 105] | 1998 | No |
| Best-worst AS | [36, 37] | 2000 | Yes |
| Population-based ACO | [82] | 2002 | Yes |
| Beam-ACO | [17, 18] | 2004 | No |

### 8.4.2.3 Applications to Dynamic Network Routing Problems

The application of ACO algorithms to dynamic problems, that is, problems whose characteristics change while being solved, is among the main success stories in ACO. The first such application [131] concerned routing in circuit-switched

networks (e.g., classical telephone networks). The proposed algorithm, called ABC, was demonstrated on a simulated version of the British Telecom network. The main merit of ABC was to stimulate the interest of ACO researchers in dynamic problems. In fact, only rather limited comparisons were made between ABC and state-of-the-art algorithms, so that it is not possible to judge on the quality of the results obtained.

A very successful application of ACO to dynamic problems is the AntNet algorithm, proposed by Di Caro and Dorigo [45–48] and discussed in Section 8.5.3. AntNet was applied to routing in packet-switched networks. It contains a number of innovations with respect to AS and it has been shown experimentally to outperform a whole set of state-of-the-art algorithms on numerous benchmark problems. Later, AntNet has also been extended to routing problems in mobile ad hoc networks, obtaining again excellent performance [68].

#### 8.4.2.4 Toward the ACO Metaheuristic

Given the initial success of ACO algorithms in the applications to $\mathcal{NP}$-hard problems as well as to dynamic routing problems in networks, Dorigo and Di Caro [57] made the synthesis effort that led to the definition of a first version of the ACO metaheuristic (see also [57, 59, 66]). In other words, the ACO metaheuristic was defined a posteriori with the goal of providing a common characterization of a new class of algorithms and a reference framework for the design of new instances of ACO algorithms.

The first version of the ACO metaheuristic was aimed at giving a comprehensive framework for ACO algorithm to "classical" $\mathcal{NP}$-hard problems *and* for applications to highly dynamic problems in network routing applications. As such, this early version of the ACO metaheuristic left a lot of freedom to the algorithm designer in the definition of solution components, of the construction mechanism, of the pheromone update, and of the ants' behavior. This more comprehensive variant of the ACO metaheuristic is presented in many publications on this topic [57, 59, 66]. Differently, the version of the ACO metaheuristic described in Section 8.3 is targeted toward the application of ACO algorithms to $\mathcal{NP}$-hard problems and therefore it is also more precise with respect to the definition of solution components and of the solution construction procedure. It follows mainly the versions presented in chapter 3 of [66] or in [21, 22].

## 8.5 Applications

The versatility and the practical use of the ACO metaheuristic for the solution of combinatorial optimization problems is best illustrated via example applications to a number of different problems.

The ACO application to the TSP has already been illustrated in Section 8.4.2.4. Here, we additionally discuss applications to two $\mathcal{NP}$-hard optimization problems:

the single machine total weighted tardiness problem (SMTWTP) and the set covering problem (SCP). We have chosen these problems since they are in several aspects different from the TSP. Although the SMTWTP is also a permutation problem, it differs from the TSP in the interpretation of the permutations. In the SCP a solution is represented as a subset of the available solution entities.

Applications of ACO to dynamic problems focus mainly on routing in data networks. To give a flavor of these applications, as a third example, we present the AntNet algorithm [47].

### 8.5.1 Example 1: The Single Machine Total Weighted Tardiness Scheduling Problem (SMTWTP)

In the SMTWTP $n$ jobs have to be processed sequentially without interruption on a single machine. Each job has an associated processing time $p_j$, a weight $w_j$, and a due date $d_j$ and all jobs are available for processing at time 0. The tardiness of job $j$ is defined as $T_j = \max\{0, C_j - d_j\}$, where $C_j$ is its completion time in the current job sequence. The goal in the SMTWTP is to find a job sequence that minimizes the sum of the weighted tardiness given by $\sum_{i=1}^{n} w_i \cdot T_i$.

For the ACO application to the SMTWTP, we can have for each position $i$ in the sequence one variable $X_i$ and each variable has $n$ associated values $j = 1, \ldots, n$. The solution components model the assignment of a job $j$ to position $i$ in the sequence.

The SMTWTP was tackled in [42] using ACS (ACS-SMTWTP). In ACS-SMTWTP, the positions of the sequence are filled in their canonical order, that is, first position one, next position two, and so on, until position $n$. At each construction step, an ant assigns a job to the current position using the *pseudo-random-proportional* action choice rule, where the feasible neighborhood of an ant is the list of yet unscheduled jobs. Pheromone trails are therefore defined as follows: $\tau_{ij}$ refers to the desirability of scheduling job $j$ at position $i$. This definition of the pheromone trails is, in fact, used in many ACO applications to scheduling problems [8, 42, 110, 142]. Concerning the heuristic information, the use of three priority rules allowed to define three different types of heuristic information for the SMTWTP [42]. The investigated priority rules were (i) the earliest due date rule, that puts the jobs in non-decreasing order of the due dates $d_j$; (ii) the modified due date rule, that puts the jobs in non-decreasing order of the modified due dates given by $mdd_j = \max\{C + p_j, d_j\}$ [8], where $C$ is the sum of the processing times of the already sequenced jobs; and (iii) the apparent urgency rule, that puts the jobs in non-decreasing order of the apparent urgency [118], given by $au_j = (w_j/p_j) \cdot \exp(-(\max\{d_j - C_j, 0\})/k\overline{p})$, where $k$ is a parameter of the priority rule. In each case, the heuristic information was defined as $\eta_{ij} = 1/h_j$, where $h_j$ is either $d_j$, $mdd_j$, or $au_j$, depending on the priority rule used.

The global and the local pheromone updates are carried out as in the standard ACS described in Section 8.4.2, where in the global pheromone update $g(s_{gb})$ is the total weighted tardiness of the global-best solution.

In [42], ACS-SMTWTP was combined with a powerful local search algorithm. The final ACS algorithm was tested on a benchmark set available from ORLIB at http://people.brunel.ac.uk/~mastjjb/jeb/info.html. Within the computation time limits given,[9] ACS reached a very good performance and could find in each single run the optimal or best-known solutions on all instances of the benchmark set [42].

### 8.5.2 Example 2: The Set Covering Problem (SCP)

In the set covering problem (SCP) we are given a finite set $A = \{a_1, \ldots, a_n\}$ of elements and a set $B = \{B_1, \ldots, B_l\}$ of subsets, $B_i \subseteq A$, that covers $A$, that is, we have $\bigcup_{i=1}^{l} B_i = A$. We say that a set $B_i$ covers an element $a_j$, if $a_j \in B_i$. Each set $B_i$ has an associated cost $c_i$. The goal in the SCP is to choose a subset $C$ of the sets in $B$ such that (i) every element of $A$ is covered and that (ii) $C$ has minimum total cost, that is, the sum of the costs of the subsets in $C$ is minimal.

ACO can be applied in a very straightforward way to the SCP. A binary variable $X_i$ is associated with every set $B_i$ and a solution component $c_i^1$ indicates that $B_i$ is selected for set $C$ (that is, $X_i = 1$), while a solution component $c_i^0$ indicates it is not selected (that is, $X_i = 0$). Each solution component $c_i^1$ is associated with a pheromone trail $\tau_i$ and a heuristic information $\eta_i$ that indicates the learned and the heuristic desirability of choosing subset $B_i$. (Note that no pheromone trails are associated with solution components $c_i^0$.) Solutions can be constructed as follows. Each ant starts with an empty solution and then adds at each step one subset until a cover is completed. A solution component $c_i^1$ is chosen with probability

$$p_i(s_p) = \frac{\tau_i^\alpha \cdot [\eta_i(s_p)]^\beta}{\sum_{l \in N(s_p)} \tau_l^\alpha \cdot [\eta_l(s_p)]^\beta}, \quad \forall c_i^1 \in N(s_p), \tag{8.5}$$

where $N(s_p)$ consists of all subsets that cover at least one still uncovered element of $A$. The heuristic information $\eta_i(s_p)$ can be chosen in several different ways. For example, a simple static information could be used, taking into account only the subset cost: $\eta_i = 1/c_i$. A more sophisticated approach would be to consider the total number of elements $d_i$ covered by a set $B_i$ and to set $\eta_i = d_i/c_i$. These two ways of defining the heuristic information do not depend on the partial solution. Typically, more accurate heuristics can be developed taking into account the partial solution of an ant. In this case, it can be defined as $\eta_i(s_p) = e_i(s_p)/c_i$, where $e_i(s_p)$ is the so-called cover value, that is, the number of additional elements covered when adding subset $B_i$ to the current partial solution $s_p$. In other words, the heuristic information measures the unit cost of covering one additional element.

An ant ends the solution construction when all the elements of $A$ are covered. In a post-optimization step, an ant can remove redundant subsets—subsets that only cover elements that are already covered by other subsets in the final solution—or

---

[9] The maximum time for the largest instances was 20 min on a 450 MHz Pentium III PC with 256 MB RAM. Programs were written in C++ and the PC was run under Red Hat Linux 6.1.

apply some additional local search to improve solutions. The pheromone update can be carried out in a standard way as described in earlier sections.

When applying ACO to the SCP, one difference with the previously presented applications is that the number of solution components in the ant's solutions may differ among the ants and, hence, the solution construction only ends when all the ants have terminated their corresponding walks.

There have been a few applications of ACO algorithms to the SCP [4, 90, 101]. The best results of these ACO algorithms are obtained by the variants that are tested by Lessing et al. [101]. In their article, they compared the performance of a number of ACO algorithms with and without the usage of a local search algorithm based on 3-flip neighborhoods [156]. The best performance results were obtained, as expected, when including local search and for a large number of instances the computational results were competitive with state-of-the-art algorithms for the SCP.

### 8.5.3 Example 3: AntNet for Network Routing Applications

Given a graph representing a telecommunications network, the problem solved by AntNet is to find the minimum cost path between each pair of vertices of the graph. It is important to note that, although finding a minimum cost path on a graph is an easy problem (it can be efficiently solved by algorithms having polynomial worst case complexity [12]), it becomes extremely difficult when the costs on the edges are time-varying stochastic variables. This is the case of routing in packet-switched networks, the target application for AntNet.

Here we briefly describe a simplified version of AntNet (the interested reader should refer to [47], where the AntNet approach to routing is explained and evaluated in detail). As stated earlier, in AntNet each ant searches for a minimum cost path between a given pair of vertices of the network. To this end, ants are launched from each network vertex toward randomly selected destination vertices. Each ant has a source vertex $s$ and a destination vertex $d$ and moves from $s$ to $d$ hopping from one vertex to the next until vertex $d$ is reached. When ant $k$ is in vertex $i$, it chooses the next vertex $j$ to move to according to a probabilistic decision rule which is a function of the ant's memory and of local pheromone and heuristic information (very much like AS, for example).

Unlike AS, where pheromone trails are associated with edges, in AntNet pheromone trails are associated with edge-destination pairs. That is, each directed edge $(i, j)$ has $n-1$ associated trail values $\tau_{ijd} \in [0,1]$, where $n$ is the number of vertices in the graph associated with the routing problem. In other words, there is one trail value $\tau_{ijd}$ for each possible destination vertex $d$ an ant located in vertex $i$ can have. In general, it will hold that $\tau_{ijd} \neq \tau_{jid}$. Each edge also has an associated heuristic value $\eta_{ij} \in [0,1]$ independent of the final destination. The heuristic values can be set, for example, to the values $\eta_{ij} = 1 - q_{ij}/\sum_{l \in N_i} q_{il}$, where $q_{ij}$ is the length (in bits waiting to be sent) of the queue of the link connecting vertex $i$ with its neighbor $j$: links with a shorter queue have a higher heuristic value $N_i$ is the set of neighbors to vertex $i$.

Ants choose their way probabilistically, using as probability a functional composition of the local pheromone trails $\tau_{ijd}$ and heuristic values $\eta_{ij}$. While building the path to their destinations, ants move using the same link queues as data and experience the same delays as data packets. Therefore, the time $T_{sd}$ elapsed while moving from the source vertex $s$ to the destination vertex $d$ can be used as a measure of the quality of the path they built. The overall quality of a path is evaluated by a heuristic function of the trip time $T_{sd}$ and of a local adaptive statistical model maintained in each vertex. In fact, paths need to be evaluated relative to the network status because a trip time $T$ judged of low quality under low congestion conditions could be an excellent one under high traffic load. Once the generic ant $k$ has completed a path, it deposits on the visited vertices an amount of pheromone $\Delta \tau^k(t)$ proportional to the quality of the path. To deposit pheromone, after reaching its destination vertex, the ant moves back to its source vertex along the same path but backward and using high priority queues, to allow a fast propagation of the collected information. The pheromone trail intensity of each edge $l_{ij}$ used by the ant while it was moving from $s$ to $d$ is increased as follows: $\tau_{ijd}(t) \leftarrow \tau_{ijd}(t) + \Delta \tau^k(t)$. After the pheromone trail on the visited edges has been updated, the pheromone value of all the outgoing connections of the same vertex $i$, relative to destination $d$, evaporates in such a way that the pheromone values are normalized and can continue to be used as probabilities: $\tau_{ijd}(t) \leftarrow \tau_{ijd}(t)/(1 + \Delta \tau^k(t)), \forall j \in N_i$.

AntNet was compared with many state-of-the-art algorithms on a large set of benchmark problems under a variety of traffic conditions. It always compared favorably with competing approaches and it was shown to be very robust with respect to varying traffic conditions and parameter settings. More details on the experimental results can be found in [47].

## 8.5.4 Applications of the ACO Metaheuristic

ACO has raised a lot of interest in the scientific community. There are now hundreds of successful implementations of the ACO metaheuristic applied to a wide range of different combinatorial optimization problems. The vast majority of these applications concern $\mathcal{NP}$-hard combinatorial optimization problems.

Many successful ACO applications to $\mathcal{NP}$-hard problems use local search algorithms to improve the ants' solutions. Another common feature of most successful ACO applications is that they use one of the advanced ACO algorithms such as ACS and $\mathcal{MM}$AS. In fact, AS has been abandoned by now in favor of more performing variants. Finally, for problems for which ACO algorithms reach very high performance, the available ACO algorithms are fine-tuned to the problem under consideration. Apart from fine-tuning parameter settings, this typically involves the exploitation of problem knowledge, for example, through the use of appropriate heuristic information, informed choices for the construction mechanism, or the use of fine-tuned local search algorithms. An overview of some successful applications of ACO algorithms to challenging "static" optimization problems is given in Table 8.2. In fact, this overview is very incomplete since the currently available ACO applications

**Table 8.2** Some current applications of ACO algorithms. Applications are listed according to problem types.

| Problem type | Problem name | Authors | Year | References |
|---|---|---|---|---|
| Routing | Traveling salesman | Dorigo et al. | 1991, 1996 | [64, 65] |
| | | Dorigo and Gambardella | 1997 | [61] |
| | | Stützle and Hoos | 1997, 2000 | [147, 148] |
| | TSP with time windows | López Ibáñez et al. | 2009 | [102] |
| | Sequential ordering | Gambardella and Dorigo | 2000 | [76] |
| | Vehicle routing | Gambardella et al. | 1999 | [77] |
| | | Reimann et al. | 2004 | [127] |
| | | Favoretto et al. | 2007 | [72] |
| | | Fuellerer et al. | 2009 | [73] |
| | Multicasting | Hernández and Blum | 2009 | [91] |
| Assignment | Quadratic assignment | Maniezzo | 1999 | [105] |
| | | Stützle and Hoos | 2000 | [148] |
| | Frequency assignment | Maniezzo and Carbonaro | 2000 | [106] |
| | Course timetabling | Socha et al. | 2002, 2003 | [138, 139] |
| | Graph coloring | Costa and Hertz | 1997 | [39] |
| Scheduling | Project scheduling | Merkle et al. | 2002 | [111] |
| | Weighted tardiness | den Besten et al. | 2000 | [42] |
| | | Merkle and Middendorf | 2000 | [109] |
| | Flow shop | Stützle | 1997 | [142] |
| | | Rajendran, Ziegler | 2004 | [125] |
| | Open shop | Blum | 2005 | [18] |
| | Car sequencing | Solnon | 2008 | [140] |
| Subset | Set covering | Lessing et al. | 2004 | [101] |
| | *l*-cardinality trees | Blum and Blesa | 2005 | [20] |
| | Multiple knapsack | Leguizamón and Michalewicz | 1999 | [100] |
| | Maximum clique | Solnon and Fenet | 2006 | [141] |
| Machine learning | Classification rules | Parpinelli et al. | 2002 | [124] |
| | | Martens et al. | 2006 | [107] |
| | | Otero et al. | 2008 | [121] |
| | Bayesian networks | Campos, Fernández-Luna | 2002 | [40, 41] |
| | Neural networks | Socha, Blum | 2007 | [135] |
| Bioinformatics | Protein folding | Shmygelska and Hoos | 2005 | [132] |
| | Docking | Korb et al. | 2006 | [97, 98] |
| | DNA sequencing | Blum et al. | 2008 | [25] |
| | Haplotype inference | Benedettini et al. | 2008 | [11] |

go into the hundreds. For a more complete overview (covering, however, only applications until the year 2004), we refer to [66].

Another large class of applications of ACO algorithms is routing problems where some system properties such as the availability of links or the cost of traversing links are time varying. This is a common case in telecommunications networks. As said before, the first ACO applications have been to telephone-like networks [131], which are circuit-switched, and to packet-switched networks such as the Internet [47].

Ant-based algorithms have given rise to several other routing algorithms, enhancing performance in a variety of wired network scenarios, see [44, 133] for a survey.

More recent applications of these strategies involved the more challenging class of
mobile ad hoc networks (MANETs). Unfortunately, the straightforward application
of the ACO algorithms developed for wired networks has proven unsuccessful due to
the specific characteristics of MANETs (very fast dynamics, link asymmetry) [159].
An ACO algorithm that is competitive with state-of-the-art routing algorithms for
MANETs, while at the same time offering better scalability, has been proposed by
Ducatelle et al. [49, 68]. For an exhaustive list of references on ACO applications
for dynamic network routing problems we refer to [69, 71].

### 8.5.5 Main Application Principles

ACO algorithms have been applied to a large number of different combinatorial
optimization problems. Based on this experience, one can identify some basic issues
that need to be addressed when attacking a new problem. These issues are discussed
in the following.

#### 8.5.5.1 Definition of Solution Components and Pheromone Trails

Of crucial importance in ACO applications is the definition of the solution compo-
nents and of the pheromone model. Consider, for example, the differences in the
definition of solution components in the TSP and the SMTWTP. Although both
problems represent solutions as permutations, the definition of solution components
(and, hence, the interpretation of the pheromone trails), is very different. In the
TSP case, a solution component refers to the direct successor relationship between
elements, while in the SMTWTP it refers to the allocation of a job to a specific posi-
tion in the permutation. This is intuitively due to the different role that permutations
have in the two problems. In the TSP, only the relative order of the solution com-
ponents is important and a permutation $\pi = (1, 2, \ldots, n)$ has the same tour length
as the permutation $\pi' = (n, 1, 2, \ldots, n-1)$—it represents the same tour. On the con-
trary, in the SMTWTP (as well as in many other scheduling problems), $\pi$ and $\pi'$
would represent two different solutions with most probably very different costs; in
this case the position information is very important.

   In some applications, the role of the pheromone trail definition has been inves-
tigated in more depth. Blum and Sampels compare different ways of defining the
pheromone model for shop scheduling problems [23]. In [22], Blum and Dorigo
show that the choice of an inappropriate pheromone model can result in an undesir-
able performance degradation over time. Fortunately, in many applications the solu-
tion components used in high-performing constructive algorithms, together with the
correct choice of the pheromone model, typically result in high-performing algo-
rithms. However, finding the best pheromone model is not always a straightforward
task. Examples of some more complex or unusual choices are the ACO application
to the shortest common supersequence problem [114] or the application of ACO to
protein–ligand docking [98].

### 8.5.5.2 Balancing Exploration and Exploitation

Any effective metaheuristic algorithm has to achieve an appropriate balance between the exploitation of the search experience gathered so far and the exploration of unvisited or relatively unexplored search space regions. In ACO, several ways exist for achieving such a balance, typically through the management of the pheromone trails. In fact, the pheromone trails induce a probability distribution over the search space and determine which parts of the search space are effectively sampled, that is, in which part of the search space the constructed solutions are located with higher frequency.

The best performing ACO algorithms typically use an *elitist strategy* in which the best solutions found during the search contribute strongly to pheromone trail updating. A stronger exploitation of the "learned" pheromone trails can be achieved during solution construction by applying the pseudo-random proportional rule of ACS, as explained in Section 8.4.2.2. These exploitation features are then typically combined with some means to ensure enough search space exploration trying to avoid convergence of the ants to a single path, corresponding to a situation of search stagnation. There are several ways to try to avoid such stagnation situations. For example, in ACS the ants use a local pheromone update rule during solution construction to make the path they have taken less desirable for subsequent ants and, thus, to diversify the search. $\mathcal{MM}$AS introduces an explicit lower limit on the pheromone trail value so that a minimal level of exploration is always guaranteed. $\mathcal{MM}$AS also uses a reinitialization of the pheromone trails, which is a way of enforcing search space exploration. Finally, an important role in the balance of exploration and exploitation is played by the parameters $\alpha$ and $\beta$ in Equation (8.1). Consider, for example, the influence of parameter $\alpha$. (Parameter $\beta$ has an analogous influence on the exploitation of the heuristic information.) For $\alpha > 0$, the larger the value of $\alpha$ the stronger the exploitation of the search experience; for $\alpha = 0$ the pheromone trails are not taken into account at all; and for $\alpha < 0$ the most probable choices taken by the ants are those that are less desirable from the point of view of pheromone trails. Hence, varying $\alpha$ could be used to shift from exploration to exploitation and conversely.

### 8.5.5.3 ACO and Local Search

In many applications to $\mathcal{NP}$-hard combinatorial optimization problems, ACO algorithms perform best when coupled with local search algorithms. Local search algorithms locally optimize the ants' solutions and these locally optimized solutions are used in the pheromone update.

The use of local search in ACO algorithms can be very interesting as the two approaches are complementary. In fact, ACO algorithms perform a rather coarse-grained search, and the solutions they produce can then be locally fine-tuned by an adequate local search algorithm. On the other side, generating appropriate

initial solutions for local search algorithms is not an easy task. In practice, ants probabilistically combine solution components which are part of the best locally optimal solutions found so far and generate new, promising initial solutions for the local search. Experimentally, it has been found that such a combination of a probabilistic, adaptive construction heuristic with local search can yield excellent results [26, 61, 147].

Despite the fact that the use of local search algorithms has been shown to be crucial for achieving state-of-the-art performance in many ACO applications, it should be noted that ACO algorithms also show very good performance when local search algorithms cannot be applied easily [47, 114].

### 8.5.5.4 Heuristic Information

The possibility of using heuristic information to direct the ants' probabilistic solution construction is important because it gives the possibility of exploiting problem-specific knowledge. This knowledge can be available a priori (this is the most frequent situation in $\mathscr{NP}$-hard problems) or at runtime (this is the typical situation in dynamic problems).

For most $\mathscr{NP}$-hard problems, the heuristic information $\eta$ can be computed at initialization time and then it remains the same throughout the whole algorithm's run. An example is the use, in the TSP applications, of the length $d_{ij}$ of the edge connecting cities $i$ and $j$ to define the heuristic information $\eta_{ij} = 1/d_{ij}$. However, the heuristic information may also depend on the partial solution constructed so far and therefore be computed at each step of an ant's solution construction. This determines a higher computational cost that may be compensated by the higher accuracy of the computed heuristic values. For example, in the ACO applications to the SMTWTP and the SCP the use of such "adaptive" heuristic information was found to be crucial for reaching very high performance.

Finally, it should be noted that while the use of heuristic information is rather important for a generic ACO algorithm, its importance is strongly reduced if local search is used to improve solutions. This is due to the fact that local search takes into account information about the cost to improve solutions in a more direct way.

## 8.6 Developments

In this section, we review recent research trends in ACO. These include (i) the application of ACO algorithms to non-standard problems; (ii) the development of ACO algorithms that are hybridized with other metaheuristics or techniques from mathematical programming; (iii) the parallel implementation of ACO algorithms; and (iv) theoretical results on ACO algorithms.

### 8.6.1 Non-standard Applications of ACO

We review here applications of ACO to problems that involve complicating factors such as multiple objective functions, time-varying data, and stochastic information

about objective values or constraints. In addition, we review some recent applications of ACO to continuous optimization problems.

### 8.6.1.1 Multiobjective Optimization

Frequently, in real-world applications, various solutions are evaluated as a function of multiple, often conflicting objectives. In simple cases, objectives can be ordered with respect to their importance or they can be combined into a single-objective by using a weighted sum approach. An example of the former approach is the application of a two-colony ACS algorithm for the vehicle routing problem with time windows [77]; an example of the latter is given by Doerner et al. [51] for a bi-objective transportation problem.

If a priori preferences or weights are not available, the usual option is to approximate the set of Pareto-optimal solutions—a solution $s$ is Pareto optimal if no other solution has a better value than $s$ for at least one objective and is not worse than $s$ for the remaining objectives. The first general ACO approach targeted to such problems is due to Iredi et al. [93], who discussed various alternatives to apply ACO to multiobjective problems and presented results with a few variants for a bi-objective scheduling problem. Since then, several algorithmic studies have tested various alternative approaches. These possible approaches differ in whether they use one or several pheromone matrices (one for each objective), one or several heuristic information, how solutions are chosen for pheromone deposit, and whether one or several colonies of ants are used. Several combinations of these possibilities have been studied, for example, in [3, 92]. For a detailed overview of the available multiobjective ACO algorithms we refer to the review articles by García-Martínez [78], that also contains an experimental evaluation of some proposed ACO approaches, and by Angus and Woodward [5].

### 8.6.1.2 Dynamic Versions of $\mathcal{NP}$-Hard Problems

As said earlier, ACO algorithms have been applied with significant success to dynamic problems in the area of network routing [47, 49]. ACO algorithms have also been applied to dynamic versions of classical $\mathcal{NP}$-hard problems. Examples are the applications to dynamic versions of the TSP, where the distances between cities may change or where cities may appear or disappear [70, 81, 82]. More recently, applications of ACS to dynamic vehicle routing problems are reported in [54, 117], showing good results on academic instances of the problem as well as on real-world instances.

### 8.6.1.3 Stochastic Optimization Problems

In many optimization problems data are not known exactly before generating a solution. Rather, what is available is stochastic information on the objective function value(s), on the decision variable values, or on the constraint boundaries due to uncertainty, noise, approximation or other factors. ACO algorithms have been applied

to a few stochastic optimization problems. The first stochastic problem to which ACO was applied is the probabilistic TSP (PTSP), where for each city the probability that it requires a visit is known and the goal is to find an a priori tour of minimal expected length over all the cities. The first to apply ACO to the PTSP were Bianchi et al. [14], who used an adaptation of ACS. This algorithm was improved by Branke and Guntsch and, very recently, by Balaprakash et al. [7], resulting in a state-of-the-art algorithm for the PTSP. Other applications of ACO include vehicle routing problems with uncertain demands [13] and the selection of optimal screening policies for diabetic retinopathy [28]; the latter approach builds on the S-ACO algorithm proposed earlier by Gutjahr [85].

### 8.6.1.4 Continuous Optimization

Although ACO was proposed for combinatorial problems, researchers started to adapt it to continuous optimization problems.[10] The simplest approach for applying ACO to continuous problems would be to discretize the real-valued domain of the variables. This approach has been successfully followed when applying ACO to the protein–ligand docking problem [98], where it was combined with a local search that was, however, working on the continuous domain of the variables. Recently, ACO algorithms that handle continuous parameters natively have been proposed. An example is the work of Socha and Dorigo [137], where the probability density functions that are implicitly built by the pheromone model are explicitly represented by Gaussian kernel functions. Their approach has also been extended to mixed-variable—continuous and discrete—problems [136]. Other references on this subject are [134, 151, 153].

## *8.6.2 Algorithmic Developments*

In the early years of ACO research, the focus was in developing ACO variants that modify the pheromone update or the solution generation mechanism to improve the algorithmic performance. More recently, researchers have started to explore combinations of ACO with other algorithmic techniques. Here, we review some of the most noteworthy of these recent developments.

### 8.6.2.1 Hybridizations of ACO with Other Metaheuristics

The most straightforward hybridization of ACO is with local improvement heuristics, which are used to fine-tune the solutions constructed by the ants. Often simple iterative improvement algorithms are used. However, in various researches other

---

[10] There have been several proposals of ant-inspired algorithms for continuous optimization [15, 67, 116]. However, these differ strongly from the underlying ideas of ACO (for example, they use direct communication among ants) and therefore cannot be considered as algorithms falling into the framework of the ACO metaheuristic.

metaheuristic algorithms have been used as improvement methods. One example is the use of tabu search to improve ants' solutions for the quadratic assignment problem [148, 150]. Interestingly, also more sophisticated hybridizations have been proposed. A first one is to let the ants start the solution construction not from scratch but from partial solutions that are obtained either by removing solution components from an ants' complete solution [155, 158] or by taking partial solutions from other complete solutions [1, 2, 152]. Two important advantages of starting the solution construction from partial solutions are that (i) the solution construction process is much faster and (ii) good parts of solutions may be exploited directly. Probably the most straightforward of these proposals is the *iterated ants* [155], which uses ideas from the iterated greedy (IG) metaheuristic [130]. Once some initial solution has been generated, IG iterates over construction heuristics by first removing solution components of a complete solution $s$, resulting in a partial solution $s_p$. From $s_p$ a complete solution is then reconstructed using some construction mechanism. In the iterated ants algorithm, this mechanism is simply the standard solution construction of the underlying ACO algorithm. Computational results suggest that this idea is particularly useful if no effective local search is available.

### 8.6.2.2 Hybridizations of ACO with Branch-and-Bound Techniques

The integration of tree search techniques into constructive algorithms is an appealing possibility of hybridization since the probabilistic solution construction of ants can be seen as the stochastic exploration of a search tree. Particularly attractive are combinations of ACO with tree search techniques from mathematical programming such as branch-and-bound. A first algorithm is the approximate nondeterministic tree search (ANTS) algorithm by Maniezzo [105]. The most important innovation of ANTS is the use of lower bound estimates as the heuristic information for rating the attractiveness of adding specific solution components. Additionally, lower bound computations allow the method to prune feasible extensions of partial solutions if the estimated solution cost is larger than that of the best solution found so far. An additional innovation of ANTS consists in the use of an initial lower bound computation to influence the order in which solution components are considered in the solution construction. Computational results obtained with ANTS for the quadratic assignment and the frequency assignment problems are very promising [105, 106].

BeamACO, the combination of ACO algorithms with beam-search was proposed by Blum [18]. Beam-search is a derivative of branch-and-bound algorithms that keeps at each iteration a set of at most *fw* nodes in a search tree and expands each of them in at most *bw* directions according to a selection based on lower bounds [122]; this results in at most *fw·bw* partial candidate solutions. After each step of extending partial solutions, the best *fw* partial solutions are kept (where best is rated with respect to a lower bound). BeamACO takes from beam-search the parallel exploration of the search tree and replaces beam-search's deterministic solution extension mechanism with that of ACO. The results with BeamACO have been very good so far.

For example, it is a state-of-the-art algorithm for open shop scheduling [18] and for some variants of assembly line balancing [19].

### 8.6.2.3 Combinations of ACO with Constraint Programming Techniques

For problems that are highly constrained and for which it is difficult to find feasible solutions, an attractive possibility is to integrate constraint programming techniques into ACO. A first proposal in this direction has been made by Meyer and Ernst [113]. In particular, they integrate a constraint propagation mechanism into the solution construction of the ants to identify earlier in the construction process whether specific solutions extensions would lead to infeasible solutions. Computational tests on a highly constrained scheduling problem showed the high potential of this approach. More recently, Khichane et al. [96] have examined the integration of an ACO algorithm into a constraint solver.

## 8.6.3 Parallel Implementations

The very nature of ACO algorithms lends them to be parallelized in the data or population domains. In particular, many parallel models used in other population-based algorithms can be easily adapted to ACO. Most parallelization strategies can be classified into *fine-grained* and *coarse-grained* strategies. Characteristics of fine-grained parallelization are that very few individuals are assigned to one single processor and that frequent information exchange among the processors takes place. On the contrary, in coarse-grained approaches larger subpopulations or even full populations are assigned to single processors and information exchange is rather rare. We refer, for example, to [32] for an overview.

Fine-grained parallelization schemes have been investigated with parallel versions of AS for the TSP on the Connection Machine CM-2 by attributing a single processing unit to each ant [27]. Experimental results showed that communication overhead can be a major problem with this approach on fine-grained parallel machines, since ants end up spending most of their time communicating to other ants the modifications they have made to pheromone trails. Similar negative results have also been reported in [31, 126].

As shown by several researches [27, 31, 50, 103, 115, 143], coarse-grained parallelization schemes are much more promising for ACO. When applied to ACO, coarse-grained schemes run $p$ subcolonies in parallel, where $p$ is the number of available processors. Subcolonies exchange information according to some policy that defines the kind of information to be exchanged, how migrants between the subcolonies are selected, to which colonies the information is sent, when information is sent, and what is to be done with the received information. Middendorf et al. [115] investigate different ways of exchanging solutions among ant colonies. They consider an exchange of the global best solutions among all colonies and local exchanges based on a virtual neighborhood among subcolonies

which corresponds to a directed ring. Their main observation was that the best solutions, with respect to computing time and solution quality, were obtained by limiting the information exchange to a local neighborhood of the colonies. Twomey et al. [154] study communication strategies for multi-colony ACO algorithms. They show that the cooperation of multiple homogenous colonies becomes less effective for increasing search effort and stronger local search algorithms. Doerner et al. [50] study different frequencies of information exchange using a fixed-frequency policy. Chen and Zhang [33] propose the use of variable frequency schedules for the migration of ants. In the extreme case, there is no communication among the subcolonies, resulting in parallel independent runs. This is the easiest way to parallelize randomized algorithms and can be very effective as shown in the computational results of Stützle [143] and of Manfrin et al. [103].

### 8.6.4 Theoretical Results

The initial, experimentally driven research on ACO has established it as an interesting algorithmic technique. After this initial phase, researchers have started to obtain insights into fundamental properties of ACO algorithms.

The first question was whether an ACO algorithm, if given enough time, will eventually find an optimal solution. This is an interesting question, because the pheromone update could prevent ACO algorithms from ever reaching an optimum. The first convergence proofs were presented by Gutjahr in [83]. He proved convergence with probability $1 - \varepsilon$ to the optimal solution of Graph-Based Ant System (GBAS), an ACO algorithm whose empirical performance is unknown. Later, he proved convergence to any optimal solution [84] with probability one for two extended versions of GBAS. Interestingly, convergence proofs for two of the top performing ACO algorithms in practice, ACS and $MM$AS, could also be obtained [66, 145].

Unfortunately, these convergence proofs do not say anything about the speed with which the algorithms converge to the optimal solution. Recently, a number of results on the expected runtime when applying ACO algorithms to specific problems have been obtained. The first results in this direction were obtained by Gutjahr [86] and since then a number of additional results have been established [53, 88, 89, 119, 120]. For a recent review of this research direction, we refer to [87].

Other research in ACO theory has focused on establishing formal links between ACO and other techniques for learning and optimization. Birattari et al. [16] relate ACO to the fields of optimal control and reinforcement learning; Meuleau and Dorigo [112] examine the connections between ACO algorithms and probabilistic learning algorithms such as the stochastic gradient ascent and the cross-entropy method; finally, Zlochin et al. [160] have proposed a unifying framework for so-called model-based search algorithms that allows a better understanding of what are the important parts of an ACO algorithm and that will help to a better cross-fertilization among algorithms.

While convergence proofs give insight into some mathematically relevant properties of algorithms, they usually do not provide guidance to practitioners for the implementation of efficient algorithms. More relevant for practical applications are research efforts that aim at a better understanding of the behavior of ACO algorithms. Blum and Dorigo [22] have shown that ACO algorithms in general suffer from *first-order deception* in the same way as genetic algorithms suffer from deception. They further introduced the concept of *second-order deception*, which occurs, for example, in situations where some solution components receive updates from more solutions on average than others they compete with [24]. The first to study the behavior of ACO algorithms by analyzing the dynamics of the pheromone model were Merkle and Middendorf [108]. For idealized permutation problems, they showed that the bias introduced on decisions in the construction process (due to constraints on the feasibility of solutions) leads to a bias which they call *selection bias*.

A review paper on recent advances in ACO theory is [56].

## 8.7 Conclusions

Since the proposal of the first ACO algorithms in 1991, the field of ACO has attracted a large number of researchers and nowadays a large number of research results of both experimental and theoretical nature exist. By now ACO is a well-established metaheuristic. The importance of ACO is exemplified by (i) the biannual conference ANTS (International conference on Ant Colony Optimization and Swarm Intelligence; http://iridia.ulb.ac.be/~ants/), where researchers meet to discuss the properties of ACO and other ant algorithms, both theoretically and experimentally; (ii) the IEEE Swarm Intelligence Symposium series; (iii) various conferences on metaheuristics and evolutionary algorithms, where ACO is a central topic; and (iv) a number of journal special issues [38, 52, 58, 62]. More information on ACO can also be found on the Ant Colony Optimization web page: www.aco-metaheuristic.org. Additionally, a moderated mailing list dedicated to the exchange of information related to ACO is accessible at: www.aco-metaheuristic.org/mailinglist.html.

The majority of the currently published articles on ACO are clearly on its application to computationally challenging problems. While most researches here are on academic applications, it is noteworthy that companies have started to use ACO algorithms for real-world applications [129]. For example, the company AntOptima (www.antoptima.com) plays an important role in promoting the real-world application of ACO. In real-world applications, features such as time-varying data, multiple objectives, or the availability of stochastic information about events or data are rather common. Interestingly, applications of ACO to problems that show such characteristics are receiving increasing attention. In fact, we believe that ACO algorithms will show their greatest advantage when they will be systematically applied to such "ill-structured" problems for which it is not clear how to apply local search, or to highly dynamic domains where only local information is available.

# References

1. Acan, A.: An external memory implementation in ant colony optimization. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 73–84. Springer, Berlin (2004)
2. Acan, A.: An external partial permutations memory for ant colony optimization. In: Raidl, G., Gottlieb, J. (eds.) Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science, vol. 3448, pp. 1–11. Springer, Berlin (2005)
3. Alaya, I., Solnon, C., Ghédira, K.: Ant colony optimization for multi-objective optimization problems. In: 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), vol. 1, pp. 450–457. IEEE Computer Society, Los Alamitos, CA (2007)
4. Alexandrov, D.A., Kochetov, Y.A.: The behavior of the ant colony algorithm for the set covering problem. In: Inderfurth, K., Schwödiauer, G., Domschke, W., Juhnke, F., Kleinschmidt, P., Wäscher, G. (eds.) Operations Research Proceedings 1999, pp. 255–260. Springer, Berlin (2000)
5. Angus, D., Woodward, C.: Multiple objective ant colony optimization. Swarm Intell. **3**(1), 69–85 (2009)
6. Applegate, D., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton, NJ (2006)
7. Balaprakash, P., Birattari, M., Stützle, T., Yuan, Z., Dorigo, M.: Estimation-based ant colony optimization algorithms for the probabilistic travelling salesman problem. Swarm Intell., **3**(3), 223–242 (2009)
8. Bauer, A., Bullnheimer, B., Hartl, R.F., Strauss, C.: An ant colony optimization approach for the single machine total tardiness problem. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), pp. 1445–1450. IEEE Press, Piscataway, NJ (1999)
9. Beckers, R., Deneubourg, J.-L., Goss, S.: Modulation of trail laying in the ant *Lasius niger* (hymenoptera: Formicidae) and its role in the collective selection of a food source. J. Insect Behav. **6**(6), 751–759 (1993)
10. Bellman, R., Esogbue, A.O., Nabeshima, I.: Mathematical Aspects of Scheduling and Applications. Pergamon Press, New York, NY (1982)
11. Benedettini, S., Roli, A., Di Gaspero, L.: Two-level ACO for haplotype inference under pure parsimony. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A. F. T. (eds.) Ant Colony Optimization and Swarm Intelligence, 6th International Workshop, ANTS 2008. Lecture Notes in Computer Science, vol. 5217, pp. 179–190. Springer, Berlin (2008)
12. Bertsekas, D.: Network Optimization: Continuous and Discrete Models. Athena Scientific, Belmont, MA (1998)
13. Bianchi, L., Birattari, M., Manfrin, M., Mastrolilli M., Paquete, L., Rossi-Doria, O., Schiavinotto, T.: Hybrid metaheuristics for the vehicle routing problem with stochastic demands. J. Math. Model. Algorithms **5**(1), 91–110 (2006)
14. Bianchi, L., Gambardella, L.M., Dorigo, M.: An ant colony optimization approach to the probabilistic traveling salesman problem. In: Merelo Guervós, J.J., Adamidis, P., Beyer, H.-G., Fernández-Villacanas, J.-L., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature – PPSN VII: 7th International Conference, Lecture Notes in Computer Science, vol. 2439, pp. 883–892. Springer, Berlin (2002)

15. Bilchev, G., Parmee, I.C.: The ant colony metaphor for searching continuous design spaces. In: Fogarty, T.C. (ed.) Evolutionary Computing, AISB Workshop, Lecture Notes in Computer Science, vol. 993, pp. 25–39. Springer, Berlin (1995)

16. Birattari, M., Di Caro, G., Dorigo, M.: Toward the formal foundation of ant programming. In: Dorigo, M., Di Caro, G., Sampels, M. (eds.) Ant Algorithms: Third International Workshop, ANTS 2002, Lecture Notes in Computer Science, vol. 2463, pp. 188–201. Springer, Berlin (2002)

17. Blum, C.: Theoretical and Practical Aspects of Ant Colony Optimization. PhD Thesis, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2004

18. Blum, C.: Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling. Comput. Oper. Res. **32**(6), 1565–1591 (2005)

19. Blum, C.: Beam-ACO for simple assembly line balancing. INFORMS J. Comput. **20**(4), 618–627 (2008)

20. Blum, C., Blesa, M. J.: New metaheuristic approaches for the edge-weighted k-cardinality tree problem. Comput. Oper. Res. **32**(6), 1355–1377 (2005)

21. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. IEEE Trans. Syst. Man Cybern. – Part B **34**(2), 1161–1172 (2004)

22. Blum, C., Dorigo, M.: Search bias in ant colony optimization: on the role of competition-balanced systems. IEEE Trans. Evol. Comput. **9**(2), 159–174 (2005)

23. Blum, C., Sampels, M.: Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), pp. 1558–1563. IEEE Press, Piscataway, NJ, 2002

24. Blum, C., Sampels, M., Zlochin, M.: On a particularity in model-based search. In: Langdon, W.B. et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002), pp. 35–42. Morgan Kaufmann, San Francisco, CA (2002)

25. Blum, C., Yabar, M., Blesa, M.J.: An ant colony optimization algorithm for DNA sequencing by hybridization. Comput. Oper. Res. **35**(11), 3620–3635 (2008)

26. Boese, K.D., Kahng, A.B., Muddu, S.: A new adaptive multi-start technique for combinatorial global optimization. Oper. Res. Lett. **16**, 101–113 (1994)

27. Bolondi, M., Bondanza, M.: Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore. Master's thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1993

28. Brailsford, S.C., Gutjahr, W.J., Rauner, M.S., Zeppelzauer, W.: Combined discrete-event simulation and ant colony optimisation approach for selecting optimal screening policies for diabetic retinopathy. Comput. Manage. Sci. **4**(1), 59–83 (2006)

29. Bullnheimer, B., Hartl, R.F., Strauss, C.: A new rank based version of the ant system—a computational study. Technical Report, Institute of Management Science, University of Vienna, 1997

30. Bullnheimer, B., Hartl, R.F., Strauss, C.: A new rank-based version of the ant system: A computational study. Cent. Eur. J. Oper. Res. Econ. **7**(1), 25–38 (1999)

31. Bullnheimer, B., Kotsis, G., Strauss, C.: Parallelization strategies for the ant system. In: De Leone, R., Murli, A., Pardalos, P., Toraldo, G. (eds.) High Performance Algorithms and Software in Nonlinear Optimization. Kluwer Series of Applied Optmization, vol. 24 pp. 87–100. Kluwer, The Netherlands (1998)

32. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer, Boston, MA (2000)

33. Chen, L., Zhang, C.: Adaptive parallel ant colony algorithm. In: Advances in Natural Computation, First International Conference, ICNC 2005. Lecture Notes in Computer Science, vol. 3611, pp. 1239–1249. Springer, Berlin (2005)

34. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Varela, F.J., Bourgine, P. (eds.) Proceedings of the First European Conference on Artificial Life, pp. 134–142. MIT Press, Cambridge, MA (1992)

35. Colorni, A., Dorigo, M., Maniezzo, V.: An investigation of some properties of an ant algorithm. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving from Nature – PPSN II, pp. 509–520. North-Holland, Amsterdam, The Netherlands (1992)

36. Cordón, O., Fernández de Viana, I., Herrera, F.: Analysis of the best-worst Ant System and its variants on the TSP. Math. Soft Comput. **9**(2–3), 177–192 (2002)

37. Cordón, O., Fernández de Viana, I., Herrera, F., Moreno, L.: A new ACO model integrating evolutionary computation concepts: The best-worst Ant System. In: Dorigo, M., Middendorf, M., Stützle, T. (eds.) Abstract proceedings of ANTS 2000 – From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms, pp. 22–29. IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2000)

38. Cordón, O., Herrera, F., Stützle, T.: Special issue on ant colony optimization: models and applications. Mathw. Soft Comput. **9**(2–3), 137–268 (2003)

39. Costa, D., Hertz, A.: Ants can colour graphs. J. Oper. Res. Soc. **48**, 295–305 (1997)

40. de Campos, L.M., Fernández-Luna, J.M., Gámez, J.A., Puerta, J.M.: Ant colony optimization for learning Bayesian networks. Int. J. Approx. Reasoning **31**(3), 291–311 (2002)

41. de Campos, L.M., Gamez, J.A., Puerta, J.M.: Learning Bayesian networks by ant colony optimisation: searching in the space of orderings. Mathw. Soft Comput. **9**(2–3), 251–268 (2002)

42. den Besten, M.L., Stützle, T., Dorigo, M.: Ant colony optimization for the total weighted tardiness problem. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (eds.) Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol. 1917, pp. 611–620. Springer, Berlin (2000)

43. Deneubourg, J.-L., Aron, S., Goss, S., Pasteels, J.-M.: The self-organizing exploratory pattern of the Argentine ant. J. Insect Behav. **3**, 159–168 (1990)

44. Di Caro, G.: Ant Colony Optimization and its application to adaptive routing in telecommunication networks. PhD thesis, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2004

45. Di Caro, G., Dorigo, M.: AntNet: a mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 1997

46. Di Caro, G., Dorigo, M.: Ant colonies for adaptive routing in packet-switched communications networks. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol. 1498, pp. 673–682. Springer, Berlin (1998)

47. Di Caro, G., Dorigo, M.: AntNet: distributed stigmergetic control for communications networks. J. Artif. Intell. Res. **9**, 317–365 (1998)

48. Di Caro, G., Dorigo, M.: Mobile agents for adaptive routing. In: El-Rewini, H. (ed.) Proceedings of the 31st International Conference on System Sciences (HICSS-31), pp. 74–83. IEEE Computer Society Press, Los Alamitos, CA (1998)

49. Di Caro, G., Ducatelle, F., Gambardella, L.M.: AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. Eur. Trans. Telecomm. **16**(5), 443–455 (2005)

50. Doerner, K.F., Hartl, R.F., Benkner, S., Lucka, M.: Parallel cooperative saving based ant colony optimization - multiple search and decomposition approaches. Parallel Process. Lett. **16**(3), 351–369 (2006)

51. Doerner, K.F., Hartl, R.F., Reimann, M.: Are CompetAnts more competent for problem solving? The case of a multiple objective transportation problem. Cent. Eur. J. Oper. Res. Econ. **11**(2), 115–141 (2003)

52. Doerner, K.F., Merkle, D., Stützle, T.: Special issue on ant colony optimization. Swarm Intell. **3**(1), 1–85 (2009)

53. Doerr, B., Neumann, F., Sudholt, D., Witt, C.: On the runtime analysis of the 1-ANT ACO algorithm. In: Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, pp. 33–40. ACM press, New York, NY (2007)

54. Donati, A.V., Montemanni, R., Casagrande, N., Rizzoli, A.E., Gambardella, L.M.: Time dependent vehicle routing problem with a multi ant colony system. Euro. J. Oper. Res. **185**(3), 1174–1191 (2008)

55. Dorigo, M.: Optimization, learning and natural algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992

56. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theor. Comput. Sci. **344**(2–3), 243–278 (2005)

57. Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 11–32. McGraw Hill, London, UK (1999)

58. Dorigo, M., Di Caro, G., Stützle T. (eds.): Special issue on "Ant Algorithms". Future Gen. Comput. Syst. **16**(8), 851–946 (2000)

59. Dorigo, M., Di Caro, G., Gambardella, L. M. Ant algorithms for discrete optimization. Artif. Life **5**(2), 137–172 (1999)

60. Dorigo, M., Gambardella, L.M.: Ant colonies for the traveling salesman problem. BioSystems **43**, 73–81 (1997)

61. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)

62. Dorigo, M., Gambardella, L.M., Middendorf, M., Stützle, T. (eds.): Special section on "Ant Colony Optimization". IEEE Trans. Evol. Comput. **6**(4), 317–365 (2002)

63. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: an autocatalytic optimizing process. Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991

64. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991

65. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. – Part B **26**(1), 29–41 (1996)

66. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge, MA (2004)

67. Dréo, J., Siarry, P.: Continuous interacting ant colony algorithm based on dense heterarchy. Future Gen. Comput. Syst. **20**(5), 841–856 (2004)

68. Ducatelle, F., Di Caro, G., Gambardella, L.M.: Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. Int. J. Comput. Intell. Appl. **5**(2), 169–184 (2005)

69. Ducatelle, F., Di Caro, G., Gambardella, L.M.: Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. Swarm Intell. (2009)

70. Eyckelhof, C.J., Snoek, M.: Ant systems for a dynamic TSP: ants caught in a traffic jam. In: Dorigo, M., Di Caro, G., Sampels, M. (eds.) Ant Algorithms: Third International Workshop, ANTS 2002. Lecture Notes in Computer Science, vol. 2463 pp. 88–99. Springer, Berlin (2002)

71. Farooq, M., Di Caro, G.: Routing protocols for next-generation intelligent networks inspired by collective behaviors of insect societies. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence: Introduction and Applications, Natural Computing Series, pp. 101–160. Springer, Berlin (2008)

72. Favaretto, D., Moretti, E., Pellegrini, P.: Ant colony system for a VRP with multiple time windows and multiple visits. J. Interdiscip. Math. **10**(2), 263–284 (2007)

73. Fuellerer, G., Doerner, K.F., Hartl, R.F., Iori, M.: Ant colony optimization for the two-dimensional loading vehicle routing problem. Comput. Oper. Res. **36**(3), 655–673 (2009)

74. Gambardella, L.M., Dorigo, M.: Ant-Q: a reinforcement learning approach to the traveling salesman problem. In: Prieditis, A., Russell, S. (eds.) Proceedings of the Twelfth International Conference on Machine Learning (ML-95), pp. 252–260. Morgan Kaufmann Publishers, Palo Alto, CA (1995)

75. Gambardella, L.M., Dorigo, M.: Solving symmetric and asymmetric TSPs by ant colonies. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), pp. 622–627. IEEE Press, Piscataway, NJ (1996)

76. Gambardella, L.M., Dorigo, M.: Ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS J. Comput. **12**(3), 237–255 (2000)

77. Gambardella, L.M., Taillard, É.D., Agazzi, G. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 63–76. McGraw Hill, London, UK (1999)

78. García-Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Euro. J. Oper. Res. **180**(1), 116–148 (2007)

79. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness. Freeman, San Francisco, CA (1979)

80. Goss, S., Aron, S., Deneubourg, J.L., Pasteels, J.M.: Self-organized shortcuts in the Argentine ant. Naturwissenschaften **76**, 579–581 (1989)

81. Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2001, Lecture Notes in Computer Science, vol. 2037, pp. 213–222. Springer, Berlin (2001)

82. Guntsch, M., Middendorf, M.: A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G. R. editors, Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim. Lecture Notes in Computer Science, vol. 2279, pp. 71–80. Springer, Berlin (2002)

83. Gutjahr, W.J.: A graph-based ant system and its convergence. Future Gen. Comput. Syst. **16**(8), 873–888 (2000)

84. Gutjahr, W.J.: ACO algorithms with guaranteed convergence to the optimal solution. Inf. Process. Lett. **82**(3), 145–153 (2002)

85. Gutjahr, W.J.: S-ACO: an ant-based approach to combinatorial optimization under uncertainty. In: Dorigo, M., Gambardella, L., Mondada, F., Stützle, T., Birratari, M., Blum, C. (eds.) Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 238–249. Springer, Berlin (2004)

86. Gutjahr, W.J.: On the finite-time dynamics of ant colony optimization. Methodol. Comput. Appl. Probability **8**(1), 105–133 (2006)

87. Gutjahr, W.J.: Mathematical runtime analysis of ACO algorithms: survey on an emerging issue. Swarm Intell. **1**(1), 59–79 (2007)

88. Gutjahr, W.J.: First steps to the runtime complexity analysis of ant colony optimization. Comput. OR **35**(9), 2711–2727 (2008)

89. Gutjahr, W.J., Sebastiani, G.: Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodol. Comput. Appl. Probability **10**, 409–433 (2008)

90. Hadji, R., Rahoual, M., Talbi, E., Bachelet, V.: Ant colonies for the set covering problem. In: Dorigo, M., Middendorf, M., Stützle, T. (eds.) Abstract proceedings of ANTS 2000 – From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms, pp. 63–66. Université Libre de Bruxelles, Brussels, Belgium (2000)

91. Hernández, H., Blum, C.: Ant colony optimization for multicasting in static wireless ad-hoc networks. Swarm Intell. **3**(2), 125–148 (2009)

92. López Ibáñez, M., Paquete, L., Stützle, T.: On the design of ACO for the biobjective quadratic assignment problem. In: Dorigo, M., Gambardella, L., Mondada, F., Stützle, T., Birratari, M., Blum, C. (eds.) ANTS'2004, Fourth International Workshop on Ant Algorithms and Swarm Intelligence, Lecture Notes in Computer Science, vol. 3172, pp. 214–225. Springer, Berlin (2004)

93. Iredi, S., Merkle, D., Middendorf, M.: Bi-criterion optimization with multi colony ant algorithms. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D. (eds.) First International Conference on Evolutionary Multi-Criterion Optimization, (EMO'01). Lecture Notes in Computer Science, vol. 1993, pp. 359–372. Springer, Berlin (2001)

94. Johnson, D.S., McGeoch, L.A.: The travelling salesman problem: a case study in local optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. Wiley, Chichester, UK (1997)

95. Jünger, M., Reinelt, G., Thienel, S.: Provably good solutions for the traveling salesman problem. Zeitschrift für Oper. Res. **40**, 183–217 (1994)

96. Khichane, M., Albert, P., Solnon, C.: Integration of ACO in a constraint programming language. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008. Lecture Notes in Computer Science, vol. 5217, pp. 84–95. Springer, Berlin (2008)

97. Korb, O., Stützle, T., Exner, T.E.: Application of ant colony optimization to structure-based drug design. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006. Lecture Notes in Computer Science, vol. 4150, pp. 247–258. Springer, Berlin (2006)

98. Korb, O., Stützle, T., Exner, T.E.: An ant colony optimization approach to flexible protein-ligand docking. Swarm Intelli. **1**(2), 115–134 (2007)

99. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: The Travelling Salesman Problem. Wiley, Chichester, UK (1985)

100. Leguizamón, G., Michalewicz, Z.: A new version of ant system for subset problems. In: Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99), pp. 1459–1464. IEEE Press, Piscataway, NJ (1999)

101. Lessing, L., Dumitrescu, I., Stützle, T.: A comparison between ACO algorithms for the set covering problem. In: Dorigo, M., Gambardella, L., Mondada, F., Stützle, T., Birratari, M., Blum, C. (eds.) Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 1–12. Springer, Berlin (2004)

102. López-Ibáñez, M., Blum, C., Thiruvady, D., Ernst, A.T., Meyer, B.: Beam-ACO based on stochastic sampling for makespan optimization concerning the TSP with time windows. In: Cotta, C., Cowling, P. (eds.) Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science, vol. 5482 pp. 97–108. Springer, Berlin (2009)

103. Manfrin, M., Birattari, M., Stützle, T., Dorigo, M.: Parallel ant colony optimization for the traveling salesman problem. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, *ANTS 2006*. Lecture Notes in Computer Science, vol. 4150, pp. 224–234. Springer, Berlin (2006)

104. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Technical Report CSR 98-1, Scienze dell'Informazione, Universitá di Bologna, Sede di Cesena, Italy, 1998

105. Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS J. Comput. **11**(4), 358–369 (1999)

106. Maniezzo, V., Carbonaro, A.: An ANTS heuristic for the frequency assignment problem. Future Gen. Comput. Syst. **16**(8), 927–935 (2000)

107. Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Trans. Evol. Comput. **11**(5), 651–665 (2007)

108. Merkle, D., Middendorf, M.: Modeling the dynamics of ant colony optimization. Evol. Comput. **10**(3), 235–262 (2002)

109. Merkle, D., Middendorf, M.: Ant colony optimization with global pheromone evaluation for scheduling a single machine. Appl. Intell. **18**(1), 105–111 (2003)

110. Merkle, D., Middendorf, M., Schmeck, H.: Ant colony optimization for resource-constrained project scheduling. In: Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.-G. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), pp. 893–900. Morgan Kaufmann, San Francisco, CA (2000)

111. Merkle, D., Middendorf, M., Schmeck, H.: Ant colony optimization for resource-constrained project scheduling. IEEE Trans. Evol. Comput. **6**(4), 333–346 (2002)

112. Meuleau, N., Dorigo, M.: Ant colony optimization and stochastic gradient descent. Artif. Life **8**(2), 103–121 (2002)
113. Meyer, B., Ernst, A.: Integrating ACO and constraint propagation. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 166–177. Springer, Berlin (2004)
114. Michel, R., Middendorf, M.: An ACO algorithm for the shortest supersequence problem. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 51–61. McGraw Hill, London, UK (1999)
115. Middendorf, M., Reischle, F., Schmeck, H.: Multi colony ant algorithms. J. Heuristics **8**(3), 305–320 (2002)
116. Monmarché, N., Venturini, G.: On how Pachycondyla apicalis ants suggest a new search algorithm. Future Gen. Comput. Syst. **16**(8), 937–946 (2000)
117. Montemanni, R., Gambardella, L.M., Rizzoli, A.E., Donati, A.V.: Ant system for a dynamic vehicle routing problem. J. Comb. Optimization **10**, 327–343 (2005)
118. Morton, T.E., Rachamadugu, R.M., Vepsalainen, A.: Accurate myopic heuristics for tardiness scheduling. GSIA Working Paper 36-83-84, Carnegie Mellon University, Pittsburgh, PA, 1984
119. Neumann, F., Sudholt, D., Witt, C.: Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intell. **3**(1), 35–68 (2009)
120. Neumann, F., Witt, C.: Runtime analysis of a simple ant colony optimization algorithm. Electron. Colloq. Comput. Complexity (ECCC) **13**(084) (2006)
121. Otero, F.E.B., Freitas, A.A., Johnson, C.G.: cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) Ant Colony Optimization and Swarm Intelligence, 6th International Workshop, ANTS 2008. Lecture Notes in Computer Science, vol. 5217, pp. 48–59. Springer, Berlin (2008)
122. Ow, P.S., Morton, T.E.: Filtered beam search in scheduling. Int. J. Prod. Res., **26**, 297–307 (1988)
123. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading, MA (1994)
124. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
125. Rajendran, C., Ziegler, H.: Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur. J. Oper. Res. **155**(2), 426–438 (2004)
126. Randall, M., Lewis, A.: A parallel implementation of ant colony optimization. J. Parallel Distr. Comput. **62**(9), 1421–1432 (2002)
127. Reimann, M., Doerner, K., Hartl, R.F.: D-ants: savings based ants divide and conquer the vehicle routing problems. Comput. Oper. Res. **31**(4), 563–591 (2004)
128. Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science, vol. 840, Springer, Berlin (1994)
129. Rizzoli, A.E., Montemanni, R., Lucibello, E., Gambardella, L.M.: Ant colony optimization for real-world vehicle routing problems. From theory to applications. Swarm Intell. **1**(2), 135–151 (2007)
130. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Euro. J. Oper. Res. **177**(3), 2033–2049 (2007)
131. Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: Ant-based load balancing in telecommunications networks. Adaptive Behav. **5**(2), 169–207 (1996)
132. Shmygelska, A., Hoos, H.H.: An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC Bioinformat. **6**, 30 (2005)
133. Sim, K.M., Sun, W.H.: Ant colony optimization for routing and load-balancing: survey and new directions. IEEE Trans. Syst. Man Cyber.-Part A: Syst. Hum. **33**(5), 560–572 (2003)
134. Socha, K.: ACO for continuous and mixed-variable optimization. In: Dorigo, M., Gambardella, L., Mondada, F., Stützle, T., Birratari, M., Blum, C. (eds.) Ant Colony Optimization

and Swarm Intelligence: 4th International Workshop, ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 25–36. Springer, Berlin (2004)

135. Socha, K., Blum, C.: An ant colony optimization algorithm for continuous optimization: An application to feed-forward neural network training. Neural Comput. Appl. **16**(3), 235–248 (2007)

136. Socha, K., Dorigo, M.: Ant colony optimization for mixed-variable optimization problems. Technical Report TR/IRIDIA/2007-019, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, October 2007

137. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. Eur. J. Oper. Res. **185**(3), 1155–1173 (2008)

138. Socha, K., Knowles, J., Sampels, M.: A $\mathcal{MAX}-\mathcal{MIN}$ ant system for the university course timetabling problem. In: Dorigo, M., Di Caro, G., Sampels, M. (eds.) Ant Algorithms: Third International Workshop, ANTS 2002. Lecture Notes in Computer Science, vol. 2463, pp. 1–13. Springer, Berlin (2002)

139. Socha, K., Sampels, M., Manfrin, M.: Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In: Raidl, G.R., Meyer, J.-A., Middendorf, M., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E. (eds.) Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2003. Lecture Notes in Computer Science, vol. 2611, pp. 334–345. Springer, Berlin (2003)

140. Solnon, C.: Combining two pheromone structures for solving the car sequencing problem with ant colony optimization. Eur. J. Oper. Res. **191**(3), 1043–1055 (2008)

141. Solnon, C., Fenet, S.: A study of ACO capabilities for solving the maximum clique problem. J. Heuristics **12**(3), 155–180 (2006)

142. Stützle, T.: An ant approach to the flow shop problem. In: Proceedings of the Sixth European Congress on Intelligent Techniques & Soft Computing (EUFIT'98), vol. 3, pp. 1560–1564. Verlag Mainz, Wissenschaftsverlag, Aachen, Germany, 1998

143. Stützle, T.: Parallelization strategies for ant colony optimization. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol. 1498, pp. 722–731. Springer, Berlin (1998)

144. Stützle, T.: Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications, DISKI, vol. 220, Infix, Sankt Augustin, Germany, 1999

145. Stützle, T., Dorigo, M.: A short convergence proof for a class of ACO algorithms. IEEE Trans. Evol. Comput. **6**(4), 358–365 (2002)

146. Stützle, T., Hoos, H.H.: Improving the ant system: a detailed report on the $\mathcal{MAX}-\mathcal{MIN}$ Ant System. Technical Report AIDA–96–12, FG Intellektik, FB Informatik, TU Darmstadt, Germany, August 1996

147. Stützle, T., Hoos, H.H.: The $\mathcal{MAX}-\mathcal{MIN}$ Ant System and local search for the traveling salesman problem. In: Bäck, T., Michalewicz, Z., Yao, X. (eds.) Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), pp. 309–314. IEEE Press, Piscataway, NJ (1997)

148. Stützle, T., Hoos, H.H.: $\mathcal{MAX}-\mathcal{MIN}$ ant system. Future Gen. Comput. Syst. **16**(8), 889–914 (2000)

149. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)

150. Talbi, E.-G., Roux, O.H., Fonlupt, C., Robillard, D.: Parallel ant colonies for the quadratic assignment problem. Future Gen. Comput. Syst. **17**(4), 441–449 (2001)

151. Tsutsui, S.: Ant colony optimisation for continuous domains with aggregation pheromones metaphor. In: Proceedings of the The 5th International Conference on Recent Advances in Soft Computing (RASC-04), pp. 207–212, Nottingham, UK (2004)

152. Tsutsui, S.: cAS: Ant colony optimization with cunning ants. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo Guervós, J.J., Whitley, L.D., Yao, X. (eds.) Parallel Problem Solving from Nature–PPSN IX, 9th International Conference. Lecture Notes in Computer Science, vol. 4193, pp. 162–171. Springer, Berlin (2006)

153. Tsutsui, S.: An enhanced aggregation pheromone system for real-parameter optimization in the ACO metaphor. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006. Lecture Notes in Computer Science, vol. 4150, pp. 60–71. Springer, Berlin (2006)
154. Twomey, C., Stützle, T., Dorigo, M., Manfrin, M., Birattari, M.: An analysis of communication policies for homogeneous multi-colony ACO algorithms. Technical Report TR/IRIDIA/2009-012, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, May 2009
155. Wiesemann, W., Stützle, T.: Iterated ants: an experimental study for the quadratic assignment problem. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006. Lecture Notes in Computer Science, vol. 4150, pp. 179–190. Springer, Berlin (2006)
156. Yagiura, M., Kishida, M., Ibaraki, T.: A 3-flip neighborhood local search for the set covering problem. Eur. J. Oper. Res. **172**, 472–499 (2006)
157. Yannakakis, M.: Computational complexity. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 19–55. Wiley, Chichester, UK (1997)
158. Yuan, Z., Fügenschuh, A., Homfeld, H., Balaprakash, P., Stützle, T., Schoch, M.: Iterated greedy algorithms for a real-world cyclic train scheduling problem. In: Blesa, M.J., Blum, C., Cotta, C., Fernández, A.J., Gallardo, J.E., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics, 5th International Workshop, HM 2008. Lecture Notes in Computer Science, vol. 5296, pp. 102–116. Springer, Berlin (2008)
159. Zhang, Y., Kuhn, L.D., Fromherz, M.P.J.: Improvements on ant routing for sensor networks. In: Dorigo, M., Gambardella, L.M., Mondada, F., Stützle, T., Birattari, M., Blum, C. (eds.) Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004. Lecture Notes in Computer Science, vol. 3172, pp. 154–165. Springer, Berlin (2004)
160. Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: a critical survey. Ann. Oper. Res. **131**(1–4), 373–395 (2004)