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 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 algorithm, successful applications of ACO algorithms to a wide range of computationally hard problems, and the theoretical understanding of important properties of ACO algorithms. This chapter reviews these developments and gives an overview of recent research trends in ACO.

1 Introduction

Ant Colony Optimization (ACO) [63, 65, 72] 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. By analogy with 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 distributed, numerical information, which is used by the ants to probabilistically construct solutions to the problem being

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

solved and which they adapt during the algorithm's execution to reflect their search experience.

The first example of such an algorithm is Ant System (AS) [61, 69, 70, 71], which was proposed using as example application the well known traveling salesman problem (TSP) [6, 110, 155]. 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 [84], scheduling [20], assembly line balancing [21], probabilistic TSP [8], 2D-HP protein folding [160], DNA sequencing [27], protein–ligand docking [107], packet-switched routing in Internet-like networks [52], and so on. The ACO metaheuristic provides a common framework for the existing applications and algorithmic variants [63, 65]. 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 2, we briefly overview construction heuristics and local search algorithms. In Section 3, we present a specific version of the ACO metaheuristic that focuses on applications to $\mathcal{NP}$-hard problems. Section 4 outlines the inspiring biological analogy and describes the historical developments leading to ACO. In Section 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 6 gives an overview of recent developments in ACO and Section 7 concludes the chapter.

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 [88, 150] 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 the instance size. Often, approximate algorithms

```
procedure Greedy Construction Heuristic
    s_p = empty solution
    while s_p not_a_complete_solution do
        e = GreedyComponent(s_p)
        s_p = s_p ⊗ e
    end
    return s_p
end Greedy Construction Heuristic
```

Fig. 1 Algorithmic skeleton of a greedy construction heuristic. The addition of component e to a partial solution s_p is denoted by the operator $\otimes$.

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

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 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 possibilities at later stages, often causing very poor moves in the final phases of the solution construction.

[1] Other approximate methods are also conceivable. For example, when stopping exact methods, like Branch & Bound, before completion [11, 104] (after some given time bound, or when some guarantee on the solution quality is obtained through the use of lower and upper bounds, for example), we can convert exact algorithms into approximate ones.

```
procedure IterativeImprovement (s ∈ S)
   s' = Improve(s)
   while s' ≠ s do
      s = s'
      s' = Improve(s)
   end
   return s
end IterativeImprovement
```

Fig. 2 Algorithmic skeleton of iterative improvement.

As an example, consider a greedy construction heuristic for the TSP. In the TSP we are given a complete weighted graph $G = (N, A)$ with N being the set of vertices, representing the cities, and A the set of edges fully connecting the vertices N. Each edge is assigned a value d_{ij}, which is the length of edge $(i, j) \in A$. The TSP is the problem of finding a minimum length Hamiltonian cycle of the graph, where an Hamiltonian cycle is a closed tour visiting exactly once each of the $n = |\mathcal{N}|$ vertices of G. For symmetric TSPs, the distances between the cities are independent of the 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 are often not of very high quality and they are not guaranteed to be optimal with respect to small changes; therefore, the results produced by constructive heuristics can often be improved by local search algorithms.

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 2. The procedure `Improve` returns a better neighbor solution if one exists, otherwise it returns the current solution, in which case the algorithm stops.

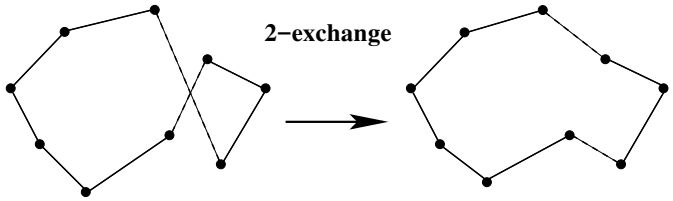

Fig. 3 Schematic illustration of a 2-exchange move. The proposed move reduces the total tour length if we consider the Euclidean distance between the points.

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. A neighborhood example for the TSP is the *k-exchange* neighborhood in which neighbor solutions differ by at most k edges. Figure 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 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* [188] 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.

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 run-time 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 towards the most promising solutions. More important, the ants' search experience can be used to influence, in a way reminiscent of reinforcement learning [179], 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 the remainder of this section, we first define a generic problem representation that the ants in ACO may exploit to construct solutions, and then we define the ACO metaheuristic.

3.1 Problem representation

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

Definition 1. A model $P = (S, \Omega, f)$ of a combinatorial optimization problem consists of

- a search space S that is defined by a finite set of decision variables, each with a finite domain, and a set Ω of constraints among the variables;
- an objective function $f : S \mapsto \mathbb{R}_0^+$ that is 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 in 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 Ω are satisfied. If Ω 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 $f(s^*) \leq f(s)\ \forall s \in S$. We denote by $S^* \subseteq S$ the set of all global minima. □

This model of a combinatorial optimization problem can be directly used to derive 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 solution 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 T_{ij}. We denote the set of all solution components by C and the set of all pheromone variables by T. Each pheromone variable T_{ij} has a pheromone value τ_{ij}; this value indicates the desirability of choosing

[2] The adaptation to a maximization problem is straightforward.

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

Fig. 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.

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 τ_{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 with each city. The domain of each variable X_i has then $n-1$ values, $j = 1, \ldots, n,\ j \neq i$. As a result, with each edge between a pair of cities is associated one pheromone value τ_{ij}. 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 possible 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.

3.2 The metaheuristic

A general outline of the ACO metaheuristic for applications to static combinatorial optimization problems[3] is given in Figure 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 im-

[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 run-time. 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 [52], in which traffic patterns change all the time.

proved 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.

`Initialization.` At the start of the algorithm, parameters are set and all pheromone variables are initialized to a value τ_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 $c_i^j \in \mathcal{N}(s_p) \subseteq C$ and adding it to its current partial solution. $\mathcal{N}(s_p)$ is the set of solution components that may be added while maintaining feasibility and it is defined implicitly by the solution construction process that the ants implement. If a partial solution cannot be extended while 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 depending on the degree of violation of the problem constraints.

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) [71]:

$$p(c_i^j|s_p) = \frac{\tau_{ij}^\alpha \cdot [\eta(c_i^j)]^\beta}{\sum_{c_i^l \in N(s_p)} \tau_{il}^\alpha \cdot [\eta(c_i^l)]^\beta}, \qquad \forall c_i^j \in N(s_p) \tag{1}$$

where $\eta(\cdot)$ is a function that assigns a heuristic value to each feasible solution component $c_i^j \in N(s_p)$, which is usually called the heuristic information. Parameters α and β 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 [72]. More generally, local search is one example of what have been called *daemon actions* [63, 65]. These are used to implement problem specific or centralized actions that cannot be performed by individual ants.

`GlobalUpdatePheromones.` The pheromone update is intended to make solution components belonging to good solutions more desirable for 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_{upd} of good solutions. The goal is to make these solution components more attractive for ants in the following iterations. 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 towards a sub-optimal 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:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s) \tag{2}$$

where S_{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 it is commonly called *evaluation function*.

ACO algorithms typically differ in the way pheromone update is implemented: different specifications of how to determine S_{upd} result in different instantiations of update rule 2. Typically, S_{upd} is a subset of $S_{iter} \cup \{s_{gb}\}$, where S_{iter} is the set of all solutions constructed in the current iteration of the main loop and s_{gb} is the best solution found since the start of the algorithm (gb stands for global-best).

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 [61, 71]. 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 ACO algorithms which show better performance than AS when applied, for example, to the TSP. In the remainder of this section, we first briefly introduce the biological metaphor on 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.

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 [48, 89]. 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 [48, 89] 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 of these experiments for our purposes 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 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 ca-

[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 [89].

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

[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*.

pacity for depositing a quantity of pheromone proportional to the quality of the solution produced (a similar behavior is observed also in some real ants 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 [10]).

In the next section 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 better or at least 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 trail are present in all ACO algorithms.

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 [61] and first appeared in a technical report [69, 70] that was published a few years later in the IEEE Transactions on Systems, Man, and Cybernetics [71]. Other early publications are [36, 37].

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 the following subsection), 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 researchers, mostly in Europe, to develop extensions and improvements of its basic ideas so as to produce better performing, and often state-of-the-art, algorithms.

4.2.1 The first ACO algorithm: Ant System and the TSP

The TSP is a paradigmatic $\mathcal{NP}$-hard combinatorial optimization problem, which has attracted an enormous amount of research effort [6, 103, 110]. 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 [61, 69, 70, 71], 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, which stores the partial solution it has constructed so far (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 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 (1). The heuristic information is given by $\eta_{ij} = 1/d_{ij}$ and $\mathcal{N}(s_p)$ 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 (2), where we have

$$S_{upd} = S_{iter} \tag{3}$$

and

$$g(s) = 1/f(s), \tag{4}$$

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.

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 [61, 71]. It consists of giving the best tour since the start of the algorithm (called s_{gb}) a strong additional weight. In practice, each time the pheromone trails are updated by Equation (2), we have that $S_{upd} = S_{iter} \cup \{s_{gb}\}$ while $g(s), s \neq s_{gb}$, is given by Equation (4) and $g(s_{gb}) = e/f(s_{gb})$, where e is a positive integer. Note that this type of pheromone update is a first example of a daemon action as described in Section 3.2.

[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).

Other improvements reported in the literature rank-based Ant System (AS_{rank}), $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS), and Ant Colony System (ACS). AS_{rank} [32] 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 rth 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_{upd} and $g(s)$ in Equation (2).

$\mathcal{MM}$AS [172, 175, 176] introduces upper and lower bounds to the values of the pheromone trails, as well as a different initialization of their values. In practice, the allowed range of the pheromone trail strength in $\mathcal{MM}$AS 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 [172, 176] it is discussed how to set the upper and lower pheromone trail limits. 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_{ib}, the iteration-best solution, or s_{gb}. In fact, 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 [172, 176]. As an additional means for increasing the explorative behavior of $\mathcal{MM}$AS (and of ACO algorithms, in general), occasional pheromone trail reinitialization is used. $\mathcal{MM}$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 [66, 67, 83] 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 [67]: 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, $j = \arg\max_{c_i^j \in \mathcal{N}(s_p)}\{\tau_{ij} \cdot \eta_{ij}^{\beta}\}$, 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 (1)). The value q_0 is a parameter:

[8] ACS was an offspring of Ant-Q [82], an algorithm intended to create a link between reinforcement learning [179] 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.

ACO algorithm	*Main references*	*Year*	*TSP*
Ant System	[61, 69, 71]	1991	yes
Elitist AS	[61, 69, 71]	1992	yes
Ant-Q	[82]	1995	yes
Ant Colony System	[66, 67, 83]	1996	yes
$\mathcal{MM}$AS	[174, 175, 176]	1996	yes
Rank-based AS	[31, 32]	1997	yes
ANTS	[124, 125]	1998	no
Best-Worst AS	[38, 39]	2000	yes
Population-based ACO	[92]	2002	yes
Beam-ACO	[19, 20]	2004	no

Table 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.

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 $\mathcal{MM}$AS, only the best ant (the global-best or the iteration-best ant) is allowed to add pheromone after each iteration of ACS; 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 ρ is the pheromone evaporation.

Finally, ACS also differs from most ACO algorithms because ants update the pheromone trails while building solutions (as in *ant-quantity* and in *ant-density*). 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). Similarly to $\mathcal{MM}$AS, ACS also usually exploits local search to improve its performance.

We could continue by enumerating the modifications that have been proposed in various other ACO algorithms that have been reported in the literature. Instead, we give an overview of the various developments on ACO algorithms for $\mathcal{NP}$-hard problems in Table 1. There 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.

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 [159] was concerned with 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 [50, 51, 52, 53] and discussed in Section 5.3. AntNet was applied to routing in packet-switched networks (e.g., the Internet). 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 [74].

4.2.4 Towards 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 [63] made the synthesis effort that led to the definition of a first version of the ACO metaheuristic (see also [63, 65, 72]). 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 applications to "classical" $\mathcal{NP}$-hard COPs *and* to highly dynamic problems in network routing applications. As such, this early version of the ACO metaheuristic left very large freedom to the algorithm designer in the definition of the solution components, construction mechanism, pheromone update, and ants' behavior. This more comprehensive variant of the ACO metaheuristic is presented in many publications on this topic [63, 65, 72]. The version of the ACO metaheuristic described in Section 3 is targeted towards 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 solution construction procedure. It follows mainly the versions presented in Chapter 3 of [72] or [23, 24].

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 the previous section. 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 [52].

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 zero. 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 which 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 one variable X_i for each position i in the sequence 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 [47] 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: τ_{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 [9, 47, 136, 170]. Concerning the heuristic information, the use of three priority rules allowed to define three different types of heuristic information for the SMTWTP [47]. The in-

vestigated priority rules were: (i) the earliest due date rule, which puts the jobs in non-decreasing order of the due dates d_j, (ii) the modified due date rule which puts the jobs in non-decreasing order of the modified due dates given by $mdd_j = \max\{C + p_j, d_j\}$ [9], where C is the sum of the processing times of the already sequenced jobs, and (iii) the apparent urgency rule which puts the jobs in non-decreasing order of the apparent urgency [144], given by $au_j = (w_j/p_j) \cdot \exp(-(\max\{d_j - C_j, 0\})/k\bar{p})$, where k is a parameter. 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 4.2, where in the global pheromone update, $g(s_{gb})$ is the total weighted tardiness of the global best solution.

In [47], 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://www.ms.ic.ac.uk/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 [47].

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 τ_i and a heuristic information η_i that indicate 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

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

$$p_i(s_p) = \frac{\tau_i^\alpha \cdot [\eta_i(s_p)]^\beta}{\sum_{l \in \mathcal{N}(s_p)} \tau_l^\alpha \cdot [\eta_l(s_p)]^\beta}, \qquad \forall c_i^1 \in \mathcal{N}(s_p) \tag{5}$$

where $\mathcal{N}(s_p)$ 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 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, 42, 100, 112, 156]. The best results of these ACO algorithms are (still) obtained by the variants tested by Lessing et al. [112]. 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 [186]. 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.

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 [13]), 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 [52], 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 towards 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 τ_{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 $\eta_{ij} = 1 - q_{ij} / \sum_{l \in \mathcal{N}_i} q_{il}$, 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.

Ants choose their way probabilistically, using as probability a functional composition of the local pheromone trails τ_{ijd} and heuristic values η_{ij}. While building the path to their destinations, ants move using the same link queues as data packets and experience the same delays. 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:

$\tau_{ijd}(t) \leftarrow \tau_{ijd}(t)/(1 + \Delta\tau^k(t))$, $\forall j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the set of neighbors of vertex 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 [52].

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 succesful ACO applications to $\mathcal{NP}$-hard problems use local search algorithms to improve the ants' solutions. Another common feature of many successful ACO applications is that they use one of the advanced ACO algorithms such as ACS, $\mathcal{MM}$AS, etc. 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. For a complete overview of ACO applications until the year 2004 we refer to [72]. Pointers to some early, successful applications of ACO algorithms to challenging "static" optimization problems is also given in Table 2.

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 is time-varying. This is a common case in telecommunications networks. As said before, the first ACO applications have been to telephone like networks [159], which are circuit-switched, and to packet switched networks such as the Internet [52].

Ant-based algorithms have given rise to several other routing algorithms, enhancing performance in a variety of wired network scenarios, see [49, 161] for a survey. Later 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 high dynamics, link asymmetry) [190]. An ACO algorithm which 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. [54, 74]. For

Table 2 Some early 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	[70, 71]
		Dorigo & Gambardella	1997	[67]
		Stützle & Hoos	1997, 2000	[175, 176]
	TSP with time windows	López Ibáñez et al.	2009	[118]
	Sequential ordering	Gambardella & Dorigo	2000	[84]
	Vehicle routing	Gambardella et al.	1999	[86]
		Reimann et al.	2004	[154]
		Favoretto et al.	2007	[79]
		Fuellerer et al.	2009	[81]
	Multicasting	Hernández & Blum	2009	[101]
Assignment	Quadratic assignment	Maniezzo	1999	[125]
		Stützle & Hoos	2000	[176]
	Frequency assignment	Maniezzo & Carbonaro	2000	[126]
	Course timetabling	Socha et al.	2002,2003	[166, 167]
	Graph coloring	Costa & Hertz	1997	[41]
Scheduling	Project scheduling	Merkle et al.	2002	[137]
	Weighted tardiness	den Besten et al.	2000	[47]
		Merkle & Middendorf	2000	[135]
	Flow shop	Stützle	1997	[170]
		Rajendran, Ziegler	2004	[152]
	Open shop	Blum	2005	[20]
	Car sequencing	Solnon	2008	[168]
Subset	Set covering	Lessing et al.	2004	[112]
	l-cardinality trees	Blum & Blesa	2005	[22]
	Multiple knapsack	Leguizamón & Michalewicz	1999	[111]
	Maximum clique	Solnon, Fenet	2006	[169]
Machine learning	Classification rules	Parpinelli et al.	2002	[151]
		Martens et al.	2006	[127]
		Otero et al.	2008	[148]
	Bayesian networks	Campos, Fernández-Luna	2002	[44, 45]
	Neural networks	Socha, Blum	2007	[163]
Bioinformatics	Protein folding	Shmygelska & Hoos	2005	[160]
	Docking	Korb et al.	2006	[106, 107]
	DNA Sequencing	Blum et al.	2008	[27]
	Haplotype Inference	Benedettini et al.	2008	[12]

an exhaustive list of references on ACO applications for dynamic network routing problems we refer to [75, 78].

The above explicit applications are mainly early examples of successful ACO applications. They have motivated other researchers to either consider ACO-based algorithms for a wide range of different applications or to advance some aspects of ACO algorithms on widely studied benchmark problems. As a result, the number of ACO applications and, thus, also the number of articles focusing on ACO has increased a lot, reaching a level of several hundreds of articles listed annually in the Scopus database. In particular, Figure 5 gives

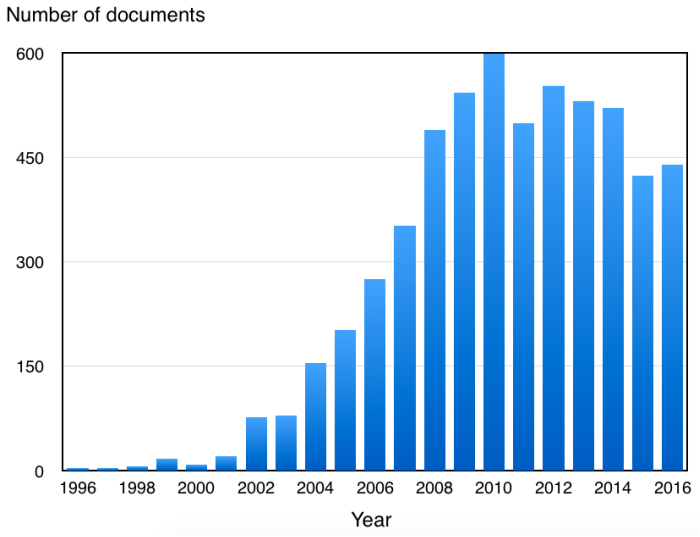

Fig. 5 Development of the number of publications containing the terms "ant system" or "ant colony system" or "ant colony optimization" in the title in the Scopus publication database from the year 1996 to 2016.

the number of articles that are published annually based on a search of the terms *ant system*, *ant colony system*, or *ant colony optimization* in article titles. In particular, since the publication of the 1996 journal article by Dorigo et al. [71], the number of articles published annually has increased strongly until ca. the year 2010 and since then has maintained a high level of about 400 to 600 articles each year.

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 adressed when attacking a new problem. These issues are discussed in the following.

5.5.1 Definition of solution components and pheromone trails

Of crucial importance in ACO applications is the definition of the solution components 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 position 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 components 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 contrary, in the SMTWTP (as well as in many other scheduling problems), π and π' 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 investigated in more depth. Blum and Sampels compare different ways of defining the pheromone model for shop scheduling problems [25]. In [24], Blum and Dorigo show that the choice of an inappropriate pheromone model can result in an undesirable performance degradation over time. Fortunately, in many applications the solution components used in high performing constructive algorithms, together with the correct choice of the pheromone model, typically result in high performing algorithms. 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 [140] or the application of ACO to protein–ligand docking [107].

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 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 α and β in Equation (1). Consider, for example, the influence of parameter α. (Parameter β has an analogous influence on the exploitation of the heuristic information). For $\alpha > 0$, the larger the value of α 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 α could be used to shift from exploration to exploitation and conversely.

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 since 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 [28, 67, 175]. Particularly good results are obtained when the integration of the local search in the ACO algorithm is well designed. To reach highest performance when very powerful local search algorithms are available or when problem instances are very large, modifications of the ACO algorithm may also be beneficial in some cases as shown by Gambardella et al. [85].

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 [52, 140].

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 $\mathcal{NP}$-hard problems) or at run-time (this is the typical situation in dynamic problems).

For most $\mathcal{NP}$-hard problems, the heuristic information η 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.

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.

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.

6.1.1 Multi-objective 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 [86]; an example of the latter is given by Doerner et al. [56] 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. [102], who discussed various alternatives to apply ACO to multi-objective 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, 120]. For a detailed overview of available multi-objective ACO algorithms we refer to the review articles by García-Martínez [87], which also contains an experimental evaluation of some proposed ACO approaches, and by Angus and Woodward [5].

A different approach to develop multi-objective ACO algorithms has been proposed by López-Ibáñez and Stützle [121, 122]. They have analyzed carefully the various existing ACO approaches to tackle multi-objective problems and proposed a generalized multi-objective ACO (MOACO) structure from which most of the then available approaches could be instantiated but also new variants be generated. Exploring the resulting design space of MOACO algorithms through a novel methodology for generating automatically multi-objective optimizers, they could generate new MOACO algorithms that clearly outperformed all previously proposed ACO algorithms for multi-objective optimization [121]. Such framework may also be further extended to consider more recent ACO approaches to many-objective problems such as those proposed by Falcón-Cardona and Coello Coello [77].

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 [52, 54]. 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 [76, 91, 92, 132]. More recent work in this area includes the explicit usage of local search algorithms to improve the ACO performance on dynamic problems [131]. Applications of ACO algorithms to dynamic vehicle routing problems are reported in [60, 133, 143], showing good results on both academic instances and real-world instances. For a recent review of swarm intelligence algorithms for dynamic optimization problems, including ACO, we refer to [130].

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. [15], who used an adaptation of ACS. This algorithm was improved by Branke and Guntsch and by Balaprakash et al. [8], resulting in a state-of-the-art algorithm for the PTSP. Other applications of ACO include the vehicle routing problem with uncertain demands [14], the vehicle routing problem with uncertain demands and customers [7], and the selection of optimal screening policies for diabetic retinopathy [30], which builds on the S-ACO algorithm by Gutjahr [95]. For an overview of the application of metaheuristics, including ACO algorithms, to stochastic combinatorial optimization problems we refer to [16].

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 [107], where it was combined with a local search that was, however, working on the continuous domain of the variables. ACO algorithms that handle continuous parameters natively have been proposed [162]. An example is the ACO_R al-

[10] There have been several proposals of ant-inspired algorithms for continuous optimization [17, 73, 142]. 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.

gorithm by Socha and Dorigo [165], where the probability density functions that are implicitly built by the pheromone model in classic ACO algorithms are explicitly represented by Gaussian kernel functions. Other early references on this subject are [162, 181, 183]. ACO_R has been refined by Liao et al. using an increasing population-size and integrating powerful local search algorithms [113]; additional refinements are later reported by Kumar et al. [109]. A unified framework for ACO applications to continuous optimization is proposed by Liao et al. [114]. In their approach, many variants of ACO_R can be instantiated by choosing specific algorithm components and by setting freely a large number of algorithm parameters. Using the help of an automated algorithm configuration tool called irace [119], the unified framework proved to be able to generate continuous ACO algorithms superior to those previously proposed in the literature. An extension of ACO_R to multi-modal optimization is presented by Yang et al. [187]. Finally, the ACO_R approach has also been extended to mixed-variable—continuous and discrete–problems [115, 164].

6.2 Algorithmic developments

In the early years of ACO research, the focus was in developing ACO variants with modified pheromone update rules or solution generation mechanisms to improve the algorithmic performance. More recently, researchers have explored combinations of ACO with other algorithmic techniques. Here, we review some of the most noteworthy developments.

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 articles, other metaheuristic algorithms have been used as improvement methods. One example is the use of tabu search to improve the ants' solutions for the quadratic assignment problem [176, 180]. Interestingly, other, 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 ant's complete solution [185, 189] or by taking partial solutions from other complete solutions [1, 2, 182]. 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* [185], which uses ideas from the iterated greedy (IG) metaheuristic [158]. 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 rebuilt 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.

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 [125]. 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 of computing an initial lower bound 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 [125, 126].

BeamACO, the combination of ACO algorithms with beam-search, was proposed by Blum [20]. 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 [149]. At each extension step applied to the fw current partial solutions, $fw \cdot bw$ new partial solutions are generated and the fw best ones 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 the beam-search's deterministic solution extension mechanism by 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 [20], for some variants of assembly line balancing [21], and for the TSP with time windows [117].

6.2.3 Combinations of ACO with constraint and integer 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 can be found in [139]. In particular, the authors 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 have shown the high potential of this approach. More recently, Khichane et al. [105] have examined the integration of an ACO algorithm into a constraint solver. Massen et al. [128] have considered the usage of ACO mechanisms in a column generation approach to vehicle routing problems with black-box feasibility constraints. The ACO-based approach is used to generate heuristically candidate routes for the vehicles, which correspond to the columns in the integer programming model; an "optimal" combination of the generated candidate routes is then found by an integer programming technique. A further analysis of the parameters of this method is proposed by Massen et al. [129], which resulted in some improved solutions to various benchmark instances.

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 early 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 [34] for an overview.

Fine-grained parallelization schemes have been investigated early when multi-core CPUs and shared memory architectures were not available or not common. The first fine-grained parallelization schemes were studied with parallel versions of AS for the TSP on the Connection Machine CM-2 by attributing a single processing unit to each ant [29]. Experimental results showed that communication overhead can be a major problem, since ants ended up spending most of their time communicating the modifications they have made to pheromone trails. Similar negative results have also been reported in [33, 153].

As shown by several researches [29, 33, 123, 141, 171], coarse grained parallelization schemes are much more promising for ACO; such schemes are also still relevant in the context of modern architectures. When applied to ACO, coarse grained schemes run p subcolonies in parallel, where p is the number of available processors. Even though independent runs of the p subcolonies in parallel have shown to be effective [123, 171], often further improved performance may be obtained by a well-designed information exchange among the subcolonies. In this case a policy 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. We refer to Middendorf et al. [141] or Twomey et al. [184] for comprehensive studies on this subject. With the wide-spread availability of multi-core CPUs and shared memory architectures, thread-level parallelism is nowadays the option of choice to speed-up a single run of an ACO algorithm. Nevertheless, if high solution quality is desired, the above mentioned coarse-grained schemes can easily be implemented also on such architectures. Recent work on parallelization of ACO algorithms evaluates them on various platforms [90] and studies the exploitation of graphics processor units to speed-up them up [35, 43, 46].

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 [93]. He proved convergence with probability $1 - \epsilon$ 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 [94] with probability one for two extended versions of GBAS. Interestingly, convergence proofs for two of the top performing ACO algorithms in practice, ACS and $\mathcal{MM}$AS, could also be obtained [72, 173].

Unfortunately, these convergence proofs do not say anything about the speed with which the algorithms converge to the optimal solution. A more detailed analysis would therefore consider the expected runtime when applying ACO algorithms to specific problems. In fact, a number of results have been obtained in that direction. The first results can be found in [96] and since then a number of additional results have been obtained [58, 59, 98, 99, 145, 146]. Due to the difficulty of the theoretical analysis, most of these results, however, have been obtained considering idealized, polynomially solvable prob-

lems. While often these include simple pseudo-Boolean functions, in [147] a theoretical runtime analysis is carried out for a basic combinatorial problem, the minimum spanning tree problem, while Sudholt and Thyssen study the shortest path problem [178]. More recently, Lissov and Witt have considered the analysis of $\mathcal{MM}$AS for dynamic shortest path problems, studying, in particular, the impact of the population size on optimization performance as a function of the type of dynamic variations [116]. For an early review of this research direction, we refer to [97].

Other research in ACO theory has focused on establishing formal links between ACO and other techniques for learning and optimization. One example relates ACO to the fields of optimal control and reinforcement learning [18], while another examines the connections between ACO algorithms and probabilistic learning algorithms such as the stochastic gradient ascent and the cross-entropy method [138]. Zlochin et al. [191] have proposed a unifying framework for so-called *model-based search* algorithms. Among other advantages, this framework allows a better understanding of what are important parts of an algorithm and it could lead 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 aimed at a better understanding of the behavior of ACO algorithms. Blum and Dorigo [24] 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 [26]. The first to study the behavior of ACO algorithms by analyzing the dynamics of the pheromone model were Merkle and Middendorf [134]. 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 what they call a *selection bias*. When applying ACO to the TSP, the solution construction can be seen as a probabilistic version of the nearest neighbor heuristic. However, Kötzing et al. show that different construction rules result in better performance at least from a theoretical perspective [108].

A discussion of recent theoretical results on ACO including those on the expected run-time analysis is given in tutorials on the theory of swarm intelligence algorithms [177]. A review paper on early advancements in ACO theory is [62].

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 [40, 57, 64, 68]. 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/mailing-list.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 [157]. For example, the company AntOptima (`www.antoptima.com`) plays an important role in promoting the real-world application of ACO. Furthermore, the company Arcelor-Mittal uses ACO algorithms to solve several of the optimization problems arising in their production sites [55, 80]. 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 increased attention. In fact, we believe that ACO algorithms are particularly useful when they are 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.

Acknowledgements This work was supported by the COMEX project, P7/36, within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. Marco Dorigo and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which both are a Research Director.

References

1. A. Acan. An external memory implementation in ant colony optimization. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Work-*

shop, ANTS 2004, volume 3172 of *Lecture Notes in Computer Science*, pages 73–84. Springer Verlag, Heidelberg, Germany, 2004.
2. A. Acan. An external partial permutations memory for ant colony optimization. In G. Raidl and J. Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3448 of *Lecture Notes in Computer Science*, pages 1–11. Springer Verlag, Heidelberg, Germany, 2005.
3. I. Alaya, C. Solnon, and K. Ghédira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 1, pages 450–457. IEEE Computer Society, Los Alamitos, CA, 2007.
4. D.A. Alexandrov and Y.A. Kochetov. The behavior of the ant colony algorithm for the set covering problem. In K. Inderfurth, G. Schwödiauer, W. Domschke, F. Juhnke, P. Kleinschmidt, and G. Wäscher, editors, *Operations Research Proceedings 1999*, pages 255–260. Springer Verlag, Berlin, Germany, 2000.
5. D. Angus and C. Woodward. Multiple objective ant colony optimization. *Swarm Intelligence*, 3(1):69–85, 2009.
6. D. Applegate, R.E. Bixby, V. Chvátal, and W.J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, 2006.
7. P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and M. Dorigo. Estimation-based metaheuristics for the single vehicle routing problem with stochastic demands and customers. *Computational Optimization and Applications*, 61(2):463–487, 2015.
8. P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and Marco Dorigo. Estimation-based ant colony optimization algorithms for the probabilistic travelling salesman problem. *Swarm Intelligence*, 3(3):223–242, 2009.
9. A. Bauer, B. Bullnheimer, R. F. Hartl, and C. Strauss. An ant colony optimization approach for the single machine total tardiness problem. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 1445–1450. IEEE Press, Piscataway, NJ, 1999.
10. R. Beckers, J.-L. Deneubourg, and S. Goss. Modulation of trail laying in the ant *Lasius niger* (hymenoptera: Formicidae) and its role in the collective selection of a food source. *Journal of Insect Behavior*, 6(6):751–759, 1993.
11. R. Bellman, A. O. Esogbue, and I. Nabeshima. *Mathematical Aspects of Scheduling and Applications*. Pergamon Press, New York, NY, 1982.
12. S. Benedettini, A. Roli, and L. Di Gaspero. Two-level ACO for haplotype inference under pure parsimony. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. F. T. Winfield, editors, *Ant Colony Optimization and Swarm Intelligence, 6th International Workshop, ANTS 2008*, volume 5217 of *Lecture Notes in Computer Science*, pages 179–190. Springer Verlag, Heidelberg, Germany, 2008.
13. D. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, MA, 1998.
14. L. Bianchi, M. Birattari, M. Manfrin, M. Mastrolilli L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms*, 5(1):91–110, 2006.
15. L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacanas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VII: 7th International Conference*, volume 2439 of *Lecture Notes in Computer Science*, pages 883–892. Springer Verlag, Heidelberg, Germany, 2002.
16. L. Bianchi, L. M. Gambardella, M. Dorigo, and W. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.
17. G. Bilchev and I. C. Parmee. The ant colony metaphor for searching continuous design spaces. In T. C. Fogarty, editor, *Evolutionary Computing, AISB Workshop*,

volume 993 of *Lecture Notes in Computer Science*, pages 25–39. Springer Verlag, Heidelberg, Germany, 1995.
18. M. Birattari, G. Di Caro, and M. Dorigo. Toward the formal foundation of ant programming. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Ant Algorithms: Third International Workshop, ANTS 2002*, volume 2463 of *Lecture Notes in Computer Science*, pages 188–201. Springer Verlag, Heidelberg, Germany, 2002.
19. C. Blum. *Theoretical and Practical Aspects of Ant Colony Optimization*. PhD thesis, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2004.
20. C. Blum. Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, 32(6):1565–1591, 2005.
21. C. Blum. Beam-ACO for simple assembly line balancing. *INFORMS Journal on Computing*, 20(4):618–627, 2008.
22. C. Blum and M. J. Blesa. New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *Computers & Operations Research*, 32(6):1355–1377, 2005.
23. C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 34(2):1161–1172, 2004.
24. C. Blum and M. Dorigo. Search bias in ant colony optimization: On the role of competition-balanced systems. *IEEE Transactions on Evolutionary Computation*, 9(2):159–174, 2005.
25. C. Blum and M. Sampels. 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)*, pages 1558–1563. IEEE Press, Piscataway, NJ, 2002.
26. C. Blum, M. Sampels, and M. Zlochin. On a particularity in model-based search. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 35–42. Morgan Kaufmann Publishers, San Francisco, CA, 2002.
27. C. Blum, M. Yabar, and M. J. Blesa. An ant colony optimization algorithm for DNA sequencing by hybridization. *Computers & Operations Research*, 35(11):3620–3635, 2008.
28. K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimization. *Operations Research Letters*, 16(2):101–113, 1994.
29. M. Bolondi and M. Bondanza. Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore. Master's thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1993.
30. S. C. Brailsford, W. J. Gutjahr, M. S. Rauner, and W. Zeppelzauer. Combined discrete-event simulation and ant colony optimisation approach for selecting optimal screening policies for diabetic retinopathy. *Computational Management Science*, 4(1):59–83, 2006.
31. B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank based version of the Ant System — a computational study. Technical report, Institute of Management Science, University of Vienna, 1997.
32. B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank-based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, 7(1):25–38, 1999.
33. B. Bullnheimer, G. Kotsis, and C. Strauss. Parallelization strategies for the Ant System. In R. De Leone, A. Murli, P. Pardalos, and G. Toraldo, editors, *High Performance Algorithms and Software in Nonlinear Optimization*, number 24 in Kluwer Series of Applied Optmization, pages 87–100. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
34. E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA, 2000.

35. J.M. Cecilia, J.M. García, A. Nisbet, M. Amos, and M. Ujaldón. Enhancing data parallelism for ant colony optimization on GPUs. *Journal of Parallel and Distributed Computing*, 73(1):52–61, 2013.
36. A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, pages 134–142. MIT Press, Cambridge, MA, 1992.
37. A. Colorni, M. Dorigo, and V. Maniezzo. An investigation of some properties of an ant algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature – PPSN II*, pages 509–520. North-Holland, Amsterdam, The Netherlands, 1992.
38. O. Cordón, I. Fernández de Viana, and F. Herrera. Analysis of the best-worst Ant System and its variants on the TSP. *Mathware and Soft Computing*, 9(2–3):177–192, 2002.
39. O. Cordón, I. Fernández de Viana, F. Herrera, and L. Moreno. A new ACO model integrating evolutionary computation concepts: The best-worst Ant System. In M. Dorigo, M. Middendorf, and T. Stützle, editors, *Abstract proceedings of ANTS 2000 – From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, pages 22–29. IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2000.
40. O. Cordón, F. Herrera, and T. Stützle. Special issue on Ant Colony Optimization: Models and applications. *Mathware and Soft Computing*, 9(2–3), 2003.
41. D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48(3):295–305, 1997.
42. B. Crawford, R. Soto, E. Monfroy, F. Paredes, and W. Palma. A hybrid ant algorithm for the set covering problem. *International Journal of Physical Sciences*, 6(19):4667–4673, 2011.
43. L. Dawson and I. A. Stewart. Improving ant colony optimization performance on the GPU using CUDA. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013*, pages 1901–1908. IEEE Press, Piscataway, NJ, 2013.
44. L. M. de Campos, J. M. Fernández-Luna, J. A. Gámez, and J. M. Puerta. Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311, 2002.
45. L. M. de Campos, J. A. Gamez, and J. M. Puerta. Learning Bayesian networks by ant colony optimisation: Searching in the space of orderings. *Mathware and Soft Computing*, 9(2–3):251–268, 2002.
46. A. Delvacq, P. Delisle, M. Gravel, and M. Krajecki. Parallel ant colony optimization on graphics processing units. *Journal of Parallel and Distributed Computing*, 73(1):52–61, 2013.
47. M. L. den Besten, T. Stützle, and M. Dorigo. Ant colony optimization for the total weighted tardiness problem. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 611–620. Springer Verlag, Heidelberg, Germany, 2000.
48. J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, 3(2):159–168, 1990.
49. G. Di Caro. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2004.
50. G. Di Caro and M. Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 1997.
51. G. Di Caro and M. Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel,

editors, *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 673–682. Springer Verlag, Heidelberg, Germany, 1998.
52. G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.
53. G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In H. El-Rewini, editor, *Proceedings of the 31st International Conference on System Sciences (HICSS-31)*, pages 74–83. IEEE Computer Society Press, Los Alamitos, CA, 1998.
54. G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455, 2005.
55. D. Díaz, P. Valledor, P. Areces, J. Rodil, and M. Suárez. An ACO algorithm to solve an extended cutting stock problem for scrap minimization in a bar mill. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle, editors, *Swarm Intelligence, 9th International Conference, ANTS 2014*, volume 8667 of *Lecture Notes in Computer Science*, pages 13–24. Springer Verlag, Heidelberg, Germany, 2014.
56. K. F. Doerner, R. F. Hartl, and M. Reimann. Are CompetAnts more competent for problem solving? The case of a multiple objective transportation problem. *Central European Journal for Operations Research and Economics*, 11(2):115–141, 2003.
57. K. F. Doerner, D. Merkle, and T. Stützle. Special issue on ant colony optimization. *Swarm Intelligence*, 3(1), 2009.
58. B. Doerr, F. Neumann, D. Sudholt, and C. Witt. On the runtime analysis of the 1-ANT ACO algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings*, pages 33–40. ACM press, New York, NY, 2007.
59. B. Doerr, F. Neumann, D. Sudholt, and C. Witt. Runtime analysis of the 1-ant ant colony optimizer. *Theoretical Computer Science*, 412(17):1629 – 1644, 2011.
60. A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185(3):1174–1191, 2008.
61. M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
62. M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005.
63. M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw Hill, London, UK, 1999.
64. M. Dorigo, G. Di Caro, and T. Stützle (Editors). Special issue on "Ant Algorithms". *Future Generation Computer Systems*, 16(8), 2000.
65. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
66. M. Dorigo and L. M. Gambardella. Ant colonies for the traveling salesman problem. *BioSystems*, 43(2):73–81, 1997.
67. M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
68. M. Dorigo, L. M. Gambardella, M. Middendorf, and T. Stützle (Editors). Special issue on "Ant Algorithms and Swarm Intelligence". *IEEE Transactions on Evolutionary Computation*, 6(4), 2002.
69. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: An autocatalytic optimizing process. Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.
70. M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

71. M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.
72. M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.
73. J. Dréo and P. Siarry. Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Computer Systems*, 20(5):841–856, 2004.
74. F. Ducatelle, G. Di Caro, and L. M. Gambardella. Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications*, 5(2):169–184, 2005.
75. F. Ducatelle, G. Di Caro, and L. M. Gambardella. Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. *Swarm Intelligence*, 2009.
76. C. J. Eyckelhof and M. Snoek. Ant systems for a dynamic TSP: Ants caught in a traffic jam. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Ant Algorithms: Third International Workshop, ANTS 2002*, volume 2463 of *Lecture Notes in Computer Science*, pages 88–99. Springer Verlag, Heidelberg, Germany, 2002.
77. J. G. Falcón-Cardona and C. A. Coello Coello. A new indicator-based many-objective ant colony optimizer for continuous search spaces. *Swarm Intelligence*, 11(1):71–100, 2017.
78. M. Farooq and G. Di Caro. Routing protocols for next-generation intelligent networks inspired by collective behaviors of insect societies. In C. Blum and D. Merkle, editors, *Swarm Intelligence: Introduction and Applications*, Natural Computing Series, pages 101–160. Springer Verlag, Berlin, Germany, 2008.
79. D. Favaretto, E. Moretti, and P. Pellegrini. Ant colony system for a VRP with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2):263–284, 2007.
80. S. Fernández, S. Álvarez, D. Díaz, M. Iglesias, and B. Ena. Scheduling a galvanizing line by ant colony optimization. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle, editors, *Swarm Intelligence, 9th International Conference, ANTS 2014*, volume 8667 of *Lecture Notes in Computer Science*, pages 146–157. Springer Verlag, Heidelberg, Germany, 2014.
81. G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 36(3):655–673, 2009.
82. L. M. Gambardella and M. Dorigo. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*, pages 252–260. Morgan Kaufmann Publishers, Palo Alto, CA, 1995.
83. L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 622–627. IEEE Press, Piscataway, NJ, 1996.
84. L. M. Gambardella and M. Dorigo. Ant Colony System hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.
85. L. M. Gambardella, R. Montemanni, and D. Weyland. Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, 220(3):831–843, 2012.
86. L. M. Gambardella, É. D. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw Hill, London, UK, 1999.
87. C. García-Martínez, O. Cordón, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1):116–148, 2007.

88. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, San Francisco, CA, 1979.
89. S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76(12):579–58, 1989.
90. G.D. Guerrero, J.M. Cecilia, A. Llanes, J.M. García, M. Amos, and M. Ujaldón. Comparative evaluation of platforms for parallel ant colony optimization. *Journal of Supercomputing*, 69(1):318–329, 2014.
91. M. Guntsch and M. Middendorf. Pheromone modification strategies for ant algorithms applied to dynamic TSP. In E. J. W. Boers, J. Gottlieb, P. L. Lanzi, R. E. Smith, S. Cagnoni, E. Hart, G. R. Raidl, and H. Tijink, editors, *Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2001*, volume 2037 of *Lecture Notes in Computer Science*, pages 213–222. Springer Verlag, Heidelberg, Germany, 2001.
92. M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279 of *Lecture Notes in Computer Science*, pages 71–80. Springer Verlag, Heidelberg, Germany, 2002.
93. W. J. Gutjahr. A Graph-based Ant System and its convergence. *Future Generation Computer Systems*, 16(8):873–888, 2000.
94. W. J. Gutjahr. ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, 82(3):145–153, 2002.
95. W. J. Gutjahr. S-ACO: An ant-based approach to combinatorial optimization under uncertainty. In M. Dorigo, L. Gambardella, F. Mondada, T. Stützle, M. Birratari, and C. Blum, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 238–249. Springer Verlag, Heidelberg, Germany, 2004.
96. W. J. Gutjahr. On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability*, 8(1):105–133, 2006.
97. W. J. Gutjahr. Mathematical runtime analysis of ACO algorithms: survey on an emerging issue. *Swarm Intelligence*, 1(1):59–79, 2007.
98. W. J. Gutjahr. First steps to the runtime complexity analysis of ant colony optimization. *Computers & OR*, 35(9):2711–2727, 2008.
99. W. J. Gutjahr and G. Sebastiani. Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability*, 10(3):409–433, 2008.
100. R. Hadji, M. Rahoual, E. Talbi, and V. Bachelet. Ant colonies for the set covering problem. In M. Dorigo, M. Middendorf, and T. Stützle, editors, *Abstract proceedings of ANTS 2000 – From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, pages 63–66. Université Libre de Bruxelles, Brussels, Belgium, 2000.
101. H. Hernández and C. Blum. Ant colony optimization for multicasting in static wireless ad-hoc networks. *Swarm Intelligence*, 3(2):125–148, 2009.
102. S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization, (EMO'01)*, volume 1993 of *Lecture Notes in Computer Science*, pages 359–372. Springer Verlag, Heidelberg, Germany, 2001.
103. D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, Chichester, UK, 1997.
104. M. Jünger, G. Reinelt, and S. Thienel. Provably good solutions for the traveling salesman problem. *Zeitschrift für Operations Research*, 40(2):183–217, 1994.

105. M. Khichane, P. Albert, and C. Solnon. Integration of ACO in a constraint programming language. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A.F.T. Winfield, editors, *Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008*, volume 5217 of *Lecture Notes in Computer Science*, pages 84–95. Springer Verlag, Heidelberg, Germany, 2008.
106. O. Korb, T. Stützle, and T. E. Exner. Application of ant colony optimization to structure-based drug design. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A.F.T. Winfield, editors, *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006*, volume 4150 of *Lecture Notes in Computer Science*, pages 247–258. Springer Verlag, Heidelberg, Germany, 2006.
107. O. Korb, T. Stützle, and T. E. Exner. An ant colony optimization approach to flexible protein-ligand docking. *Swarm Intelligence*, 1(2):115–134, 2007.
108. T. Kötzing, F. Neumann, H. Röglin, and C. Witt. Theoretical analysis of two ACO approaches for the traveling salesman problem. *Swarm Intelligence*, 6(1):1–21, 2012.
109. U. Kumar, Jayadeva, and S. Soman. Enhancing IACOR local search by Mtsls1-BFGS for continuous global optimization. In S. Silva and A. I. Esparcia-Alcázar, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015*, pages 33–40. ACM Press, New York, NY, 2015.
110. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Travelling Salesman Problem*. John Wiley & Sons, Chichester, UK, 1985.
111. G. Leguizamón and Z. Michalewicz. A new version of Ant System for subset problems. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 1459–1464. IEEE Press, Piscataway, NJ, 1999.
112. L. Lessing, I. Dumitrescu, and T. Stützle. A comparison between ACO algorithms for the set covering problem. In M. Dorigo, L. Gambardella, F. Mondada, T. Stützle, M. Birratari, and C. Blum, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, Heidelberg, Germany, 2004.
113. T. Liao, M. Montes de Oca, D. Aydin, T. Stützle, and M. Dorigo. An incremental ant colony algorithm with local search for continuous optimization. In N. Krasnogor and P. L. Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 125–132. ACM Press, New York, NY, 2011.
114. T. Liao, M. Montes de Oca, T. Stützle, and M. Dorigo. A unified ant colony optimization algorithm for continuous optimization. *European Journal of Operational Research*, 234(3):597–609, 2014.
115. T. Liao, K. Socha, M. Montes de Oca, T. Stützle, and M. Dorigo. Ant colony optimization for mixed-variable optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(4):503–518, 2014.
116. A. Lissovoi and C. Witt. Runtime analysis of ant colony optimization on dynamic shortest path problems. *Theoretical Computer Science*, 561:73–85, 2015.
117. M. López-Ibáñez and C. Blum. Beam-ACO for the travelling salesman problem with time windows. *Computers & Operations Research*, 37(9):1570–1583, 2010.
118. M. López-Ibáñez, C. Blum, D. Thiruvady, A. T. Ernst, and B. Meyer. Beam-ACO based on stochastic sampling for makespan optimization concerning the TSP with time windows. In C. Cotta and P. Cowling, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 5482 of *Lecture Notes in Computer Science*, pages 97–108. Springer Verlag, Heidelberg, Germany, 2009.
119. M. López-Ibáñez, J. Dubois-Lacoste, L. Perez Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.
120. M. López-Ibáñez, L. Paquete, and T. Stützle. On the design of ACO for the biobjective quadratic assignment problem. In M. Dorigo, L. Gambardella, F. Mondada, T. Stützle, M. Birratari, and C. Blum, editors, *ANTS'2004, Fourth International Workshop on Ant Algorithms and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 214–225. Springer Verlag, Heidelberg, Germany, 2004.

121. M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6):861–875, 2012.
122. M. López-Ibáñez and T. Stützle. An experimental analysis of design choices of multi-objective ant colony optimization algorithms. *Swarm Intelligence*, 6(3):207–232, 2012.
123. M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo. Parallel ant colony optimization for the traveling salesman problem. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006*, volume 4150 of *Lecture Notes in Computer Science*, pages 224–234. Springer Verlag, Heidelberg, Germany, 2006.
124. V. Maniezzo. 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.
125. V. Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, 11(4):358–369, 1999.
126. V. Maniezzo and A. Carbonaro. An ANTS heuristic for the frequency assignment problem. *Future Generation Computer Systems*, 16(8):927–935, 2000.
127. D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens. Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 11(5):651–665, 2007.
128. F. Massen, Y. Deville, and P. van Hentenryck. Pheromone-based heuristic column generation for vehicle routing problems with black box feasibility. In N. Beldiceanu, N. Jussien, and E. Pinson, editors, *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimization Problems, CPAIOR 2012*, volume 7298 of *Lecture Notes in Computer Science*, pages 260–274. Springer Verlag, 2012.
129. F. Massen, M. López-Ibá nez, T. Stützle, and Y. Deville. Experimental analysis of pheromone-based heuristic column generation using irace. In M. J. Blesa, C. Blum, P. Festa, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 7919 of *Lecture Notes in Computer Science*, pages 92–106. Springer Verlag, 2013.
130. M. Mavrovouniotis, C. Li, and S. Yang. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation*, 33:1–17, 2017.
131. M. Mavrovouniotis, F. Martins Müller, and S. Yang. Ant colony optimization with local search for dynamic traveling salesman problems. *IEEE Transactions on Cybernetics*, 47(7):1743–1756, 2017.
132. M. Mavrovouniotis and S. Yang. Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *Applied Soft Computing*, 13(10):4023–4037, 2013.
133. M. Mavrovouniotis and S. Yang. Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. *Information Sciences*, 294:456–477, 2015.
134. D. Merkle and M. Middendorf. Modeling the dynamics of ant colony optimization. *Evolutionary Computation*, 10(3):235–262, 2002.
135. D. Merkle and M. Middendorf. Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence*, 18(1):105–111, 2003.
136. D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 893–900. Morgan Kaufmann Publishers, San Francisco, CA, 2000.
137. D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4):333–346, 2002.

138. N. Meuleau and M. Dorigo. Ant colony optimization and stochastic gradient descent. *Artificial Life*, 8(2):103–121, 2002.
139. B. Meyer and A. Ernst. Integrating ACO and constraint propagation. In M. Dorigo, M. Birattari, C. Blum, L.M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 166–177. Springer Verlag, Heidelberg, Germany, 2004.
140. R. Michel and M. Middendorf. An ACO algorithm for the shortest supersequence problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 51–61. McGraw Hill, London, UK, 1999.
141. M. Middendorf, F. Reischle, and H. Schmeck. Multi colony ant algorithms. *Journal of Heuristics*, 8(3):305–320, 2002.
142. N. Monmarché and G. Venturini and. On how *Pachycondyla apicalis* ants suggest a new search algorithm. *Future Generation Computer Systems*, 16(8):937–946, 2000.
143. R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343, 2005.
144. T. E. Morton, R. M. Rachamadugu, and A. Vepsalainen. Accurate myopic heuristics for tardiness scheduling. GSIA Working Paper 36-83-84, Carnegie Mellon University, Pittsburgh, PA, 1984.
145. F. Neumann, D. Sudholt, and C. Witt. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence*, 3(1):35–68, 2009.
146. F. Neumann and C. Witt. Runtime analysis of a simple ant colony optimization algorithm. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(084), 2006.
147. F. Neumann and C. Witt. Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science*, 411(25):2406 – 2413, 2010.
148. F. E. B. Otero, A. A. Freitas, and C. G. Johnson. cAnt-Miner: An ant colony classification algorithm to cope with continuous attributes. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. F. T. Winfield, editors, *Ant Colony Optimization and Swarm Intelligence, 6th International Workshop, ANTS 2008*, volume 5217 of *Lecture Notes in Computer Science*, pages 48–59. Springer Verlag, Heidelberg, Germany, 2008.
149. P.S. Ow and T.E. Morton. Filtered beam search in scheduling. *International Journal of Production Research*, 26(1):297–307, 1988.
150. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.
151. R. S. Parpinelli, H. S. Lopes, and A. A. Freitas. Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321–332, 2002.
152. C. Rajendran and H. Ziegler. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 155(2):426–438, 2004.
153. M. Randall and A. Lewis. A parallel implementation of ant colony optimization. *Journal of Parallel and Distributed Computing*, 62(9):1421–1432, 2002.
154. M. Reimann, K. Doerner, and R. F. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problems. *Computers & Operations Research*, 31(4):563–591, 2004.
155. G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*, volume 840 of *Lecture Notes in Computer Science*. Springer Verlag, Heidelberg, Germany, 1994.
156. Z.-G. Ren, Z.-R. Feng, L.-J. Ke, and Z.-J. Zhang. New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4):774–784, 2010.

157. A.E. Rizzoli, R. Montemanni, E. Lucibello, and L.M. Gambardella. Ant colony optimization for real-world vehicle routing problems. From theory to applications. *Swarm Intelligence*, 1(2):135–151, 2007.
158. R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.
159. R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.
160. A. Shmygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6:30, 2005.
161. K. M. Sim and W H. Sun. Ant colony optimization for routing and load-balancing: Survey and new directions. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(5):560–572, 2003.
162. K. Socha. ACO for continuous and mixed-variable optimization. In M. Dorigo, L. Gambardella, F. Mondada, T. Stützle, M. Birratari, and C. Blum, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 25–36. Springer Verlag, Heidelberg, Germany, 2004.
163. K. Socha and C. Blum. An ant colony optimization algorithm for continuous optimization: An application to feed-forward neural network training. *Neural Computing & Applications*, 16(3):235–248, 2007.
164. K. Socha and M. Dorigo. Ant colony optimization for mixed-variable optimization problems. Technical Report TR/IRIDIA/2007-019, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, October 2007.
165. K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.
166. K. Socha, J. Knowles, and M. Sampels. A $\mathcal{MAX} - \mathcal{MIN}$ Ant System for the university course timetabling problem. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Ant Algorithms: Third International Workshop, ANTS 2002*, volume 2463 of *Lecture Notes in Computer Science*, pages 1–13. Springer Verlag, Heidelberg, Germany, 2002.
167. K. Socha, M. Sampels, and M. Manfrin. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In G. R. Raidl, J.-A. Meyer, M. Middendorf, S. Cagnoni, J. J. R. Cardalda, D. W. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, and E. Marchiori, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2003*, volume 2611 of *Lecture Notes in Computer Science*, pages 334–345. Springer Verlag, Heidelberg, Germany, 2003.
168. C. Solnon. Combining two pheromone structures for solving the car sequencing problem with ant colony optimization. *European Journal of Operational Research*, 191(3):1043–1055, 2008.
169. C. Solnon and S. Fenet. A study of ACO capabilities for solving the maximum clique problem. *Journal of Heuristics*, 12(3):155–180, 2006.
170. T. Stützle. An ant approach to the flow shop problem. In *Proceedings of the Sixth European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564. Verlag Mainz, Wissenschaftsverlag, Aachen, Germany, 1998.
171. T. Stützle. Parallelization strategies for ant colony optimization. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, pages 722–731. Springer Verlag, Heidelberg, Germany, 1998.
172. T. Stützle. *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*, volume 220 of *DISKI*. Infix, Sankt Augustin, Germany, 1999.
173. T. Stützle and M. Dorigo. A short convergence proof for a class of ACO algorithms. *IEEE Transactions on Evolutionary Computation*, 6(4):358–365, 2002.

174. T. Stützle and H. H. Hoos. 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.
175. T. Stützle and H. H. Hoos. The $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and local search for the traveling salesman problem. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 309–314. IEEE Press, Piscataway, NJ, 1997.
176. T. Stützle and H. H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.
177. D. Sudholt. Theory of swarm intelligence: Tutorial at GECCO 2017. In P. A. N. Bosman, editor, *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 902–921. ACM Press, New York, NY, 2017.
178. D. Sudholt and C. Thyssen. Running time analysis of ant colony optimization for shortest path problems. *Journal of Discrete Algorithms*, 10:165–180, 2012.
179. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
180. E.-G. Talbi, O. H. Roux, C. Fonlupt, and D. Robillard. Parallel ant colonies for the quadratic assignment problem. *Future Generation Computing Systems*, 17(4):441–449, 2001.
181. S. Tsutsui. Ant colony optimisation for continuous domains with aggregation pheromones metaphor. In *Proceedings of the 5th International Conference on Recent Advances in Soft Computing (RASC-04)*, pages 207–212, Nottingham, UK, 2004.
182. S. Tsutsui. cAS: Ant colony optimization with cunning ants. In T. P. Runarsson, H.-G. Beyer, E. K. Burke, J. J. Merelo Guervós, L. D. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature–PPSN IX, 9th International Conference*, volume 4193 of *Lecture Notes in Computer Science*, pages 162–171. Springer Verlag, Heidelberg, Germany, 2006.
183. S. Tsutsui. An enhanced aggregation pheromone system for real-parameter optimization in the ACO metaphor. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006*, volume 4150 of *Lecture Notes in Computer Science*, pages 60–71. Springer Verlag, Berlin, Germany, 2006.
184. Colin Twomey, T. Stützle, M. Dorigo, M. Manfrin, and M. Birattari. An analysis of communication policies for homogeneous multi-colony ACO algorithms. *Information Sciences*, 180(12):2390–2404, 2010.
185. W. Wiesemann and T. Stützle. Iterated ants: An experimental study for the quadratic assignment problem. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006*, volume 4150 of *Lecture Notes in Computer Science*, pages 179–190. Springer Verlag, Heidelberg, Germany, 2006.
186. M. Yagiura, M. Kishida, and T. Ibaraki. A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172(2):472–499, 2006.
187. Q. Yang, W.-N. Chen, Z. Yu, T. Gu, Y. Li, H. Zhang, and J. Zhang. Adaptive multimodal continuous ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 21(2):191–205, 2017.
188. M. Yannakakis. Computational complexity. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–55. John Wiley & Sons, Chichester, UK, 1997.
189. Z. Yuan, A. Fügenschuh, H. Homfeld, P. Balaprakash, T. Stützle, and M. Schoch. Iterated greedy algorithms for a real-world cyclic train scheduling problem. In M. J. Blesa, C. Blum, C. Cotta, A. J. Fernández, J. E. Gallardo, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics, 5th International Workshop, HM 2008*, volume 5296

of *Lecture Notes in Computer Science*, pages 102–116. Springer Verlag, Heidelberg, Germany, 2008.

190. Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz. Improvements on ant routing for sensor networks. In M. Dorigo, L. M. Gambardella, F. Mondada, T. Stützle, M. Birattari, and C. Blum, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 154–165. Springer Verlag, Heidelberg, Germany, 2004.
191. M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: A critical survey. *Annals of Operations Research*, 131(1–4):373–395, 2004.