ANT COLONY OPTIMIZATION

MARCO DORIGO
MARCO A. MONTES DE OCA
SABRINA OLIVEIRA
THOMAS STÜTZLE
IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium

Ant colony optimization (ACO) [1–8] is a class of algorithms for tackling optimization problems that is inspired by the pheromone trail laying and following behavior of some ant species. While foraging, ants leave on the ground a chemical substance, called *pheromone*, that attracts other fellow nestmates [9]. The pheromone trail laying and following behavior of the ants induces a positive feedback process whereby trails with high concentration of pheromones become more and more attractive as more ants follow them [10–12]. As a result, whenever two paths to the same food source are discovered, the colony is more likely to select the shortest one because ants will traverse it faster and thus it will have a higher pheromone concentration than the longer one.

ACO algorithms exploit a mechanism analogous to the one that allows colonies of real ants to find shortest paths. In ACO, (artificial) ants construct candidate solutions to the problem instance under consideration. Their solution construction is stochastically biased by (artificial) pheromone trails, which are represented in the form of numerical information that is associated with appropriately defined solution components, and possibly by heuristic information based on the input data of the instance being solved. A key aspect of ACO algorithms is the use of a positive feedback loop implemented by iterative modifications of the artificial pheromone trails that are a function of the ants' search experience; the goal of this feedback loop is to bias the colony toward the most promising solutions.

The ACO metaheuristic is a high-level algorithmic framework for applying the above ideas to the approximate solution of optimization problems. When applied to a specific optimization problem, this ACO framework needs to be concretized by taking into account the specifics of the problem under consideration and possibly by adding additional techniques such as problem-specific solution improvement procedures. The development of effective ACO algorithm variants has been one of the most active research directions in ACO: this article gives an overview of the most important of these developments. For more information about successful applications of ACO, we refer to the article titled ***A Concise Overview of Applications of Ant Colony Optimization***, in this encyclopedia.

ACO EXAMPLE APPLICATIONS

Perhaps the easiest way to understand how ACO algorithms work is through examples. Here, we present two examples where the ACO algorithms use different solution representations. The first example shows how ACO is applied to solve the traveling salesman problem (TSP), which was the first optimization problem to which an ACO algorithm was applied. The second example concerns the set covering problem (SCP): it shows how ACO algorithms can be used to solve problems using a binary representation.

Example 1. Ant Colony Optimization for the Traveling Salesman Problem

The TSP is one of the most widely studied combinatorial optimization problems. It is also a problem to which the application of ACO algorithms is rather intuitive and straightforward. An instance of the TSP is determined by a set of locations (cities) and by the distances between them. The goal is to find a closed tour of minimal length that visits each city exactly once. A TSP instance can be represented by a fully connected graph $G = (V, E, d)$, V being the set of $n = |V|$ vertices (representing the cities), E being the set of edges that fully connects the vertices, and d being a distance function that assigns to each edge (i,j) a distance d_{ij}. Here, we assume that the distance function is symmetric, that is, we have $d_{ij} = d_{ji}$, meaning that the distance is the same whether one goes from i to j or in the opposite direction.

To tackle a TSP instance with an ACO algorithm, each edge $(i,j) \in E$ needs a pheromone value τ_{ij} associated with it. The pheromone values are represented by real numbers that are modified while running the algorithm; they reflect the *learned* desirability of choosing an edge: the higher the pheromone value τ_{ij}, the higher is the desirability of choosing edge (i,j) as a solution component. Additionally, each edge has an associated heuristic value $\eta_{ij} = 1/d_{ij}$. For the TSP, the value η_{ij} is a measure of the *heuristic* desirability of having edge (i,j) as a component of a tour: the shorter the distance, the higher is the heuristic desirability.

An intuitive approach for constructing a tour is to first choose a vertex randomly and then, at each step, to go from the current vertex to the closest one that has not yet been visited. This solution construction ends when all vertices have been visited and the round trip is closed by returning to the initial vertex. In all ACO algorithms that have been implemented so far, the ants follow a randomized version of this construction rule. In fact, at each construction step, they choose randomly a next vertex based on the pheromone trail information and the heuristic information. The probabilistic choice is biased by pheromone and heuristic values: the higher the pheromone and the heuristic values associated with an edge, the higher the probability that an ant will choose it. Once all ants have completed their tours, the pheromone on the edges is updated. First, all pheromone values are decreased by a constant factor, simulating the phenomenon of pheromone evaporation. Then, each edge receives an amount of pheromone proportional to the quality of the solutions to which it belongs (there is one solution per ant); that is, the shorter the associated tour, the more pheromone is deposited on the edges, making them more attractive in future iterations.

Example 2. Ant Colony Optimization for the Set Covering Problem

The SCP is a problem in which a candidate solution is represented by a subset of elements from some other set subject to some feasibility constraints. In the SCP, one is given two sets A and B. Each element B_i of B is a subset of A and it has associated a cost c_i. The goal of the SCP is to find a subset of the set B of minimal cost such that A is covered, that is, every element of the set A occurs in at least one of the elements chosen from set B. To guarantee that such a solution exists, one necessary assumption to make is that the elements of B *cover* the set A, that is, $\bigcup_{i=1}^{n} B_i = A$. A candidate solution for the SCP can be represented by an n-dimensional binary vector $X = [x_i]$, where n is the cardinality of set B, $x_i = 1$ if B_i is selected to be part of the solution, otherwise $x_i = 0$.

For solving an instance of the SCP with an ACO algorithm, we define the solution components as the elements of the set B. Each set B_i has associated a pheromone trail τ_i. The pheromone trails represent, analogous to the TSP, the ants' cumulated experience in solving the problem. For the SCP, the pheromone τ_i gives the desirability for an ant to choose element B_i, that is, to set the decision variable $x_i = 1$. The heuristic information η_i can be defined in various ways. One possibility is to use $\eta_i = k_i/c_i$, where $k_i = |B_i|$ is the total number of elements covered by the subset B_i. Hence, the heuristic function gives the average cost for B_i of covering elements of A. In this case, the heuristic information makes use only of *a priori* available information. It is therefore possible to compute the heuristic information before running the algorithm and therefore to compute the values of $\tau_i \cdot \eta_i$ before each algorithm iteration, saving in this way computation time. However, it may be advantageous to make the heuristic information more accurate (but slower to compute) by taking into account an ant's partial solution. In the SCP case, η_i could then measure the unit cost of covering one additional, still uncovered element of set A. This can be done by using $\eta_i = e_i/c_i$, where e_i is the number of additional elements of set A covered when B_i is added to an ant's partial solution. Which of the two options—using the faster but less accurate precomputed heuristic information or adapting the heuristic information based on the ants' partial solutions—is preferable typically depends on the particular problem to which ACO is applied.

Ants construct solutions taking into account both the pheromone value and the heuristic information associated with each solution component. In the SCP case, an ant starts with an empty solution and chooses, at each construction step, one element of B until all elements of A are covered. In other words, an ant starts with all decision variables set to zero and at each construction step it sets one decision variable to one until all elements of A occur in at least one of the chosen elements of B. Note that in the SCP application, the number of construction steps to complete a solution may differ among the ants. Once each ant has terminated the construction of a candidate solution, it can remove subsets B_i that may have become redundant while constructing a solution before the pheromone trails are updated.

ACO METAHEURISTIC

ACO can be applied to any combinatorial optimization problem for which it is possible to devise an incremental solution construction procedure. Let us consider a general description of a combinatorial optimization problem that is modeled by the tuple (S, f, Ω), where

- S is the set of candidate solutions defined over a finite set of discrete decision variables $\mathbb{X}$. S is referred to as the *search space* of the problem being tackled;
- $f: S \to \mathbb{R}$ is an objective function to be minimized;
- Ω is a (possibly empty) set of constraints among the decision variables.

A decision variable $X_i \in \mathbb{X}$, with $i = 1, \ldots, n$, is said to be instantiated when a value v_i^j that belongs to its domain $D_i = \left\{v_i^1, \ldots, v_i^{|D_i|}\right\}$ is assigned to it. A solution $s \in S$ is called *feasible* if each decision variable has been instantiated satisfying all constraints in the set Ω. Solving the optimization problem requires finding a solution s^* such that $f(s^*) \le f(s)\ \forall s \in S$. Note that maximizing the value of an objective function f is the same as minimizing the value of $-f$; hence, every model of a combinatorial optimization problem can be described as a minimization problem.

```
procedure ACOMetaheuristic
    ScheduleActivities
        ConstructSolutions
        DaemonActions        //optional
        UpdatePheromones
    end-ScheduleActivities
end-procedure
```

Figure 1. ACO metaheuristic in pseudocode. It works by intertwining three high-level procedures: ConstructSolutions, DaemonActions, and UpdatePheromones.

ACO works by intertwining three high-level procedures: ConstructSolutions, DaemonActions, and UpdatePheromones as shown in Fig. 1. The ScheduleActivities construct does not specify how the three algorithmic components are scheduled and synchronized. However, in most applications, these procedures are executed in the depicted order.

- *ConstructSolutions.* This procedure implements the artificial ants' incremental construction of candidate solutions. In ACO, an instantiated decision variable $X_i \leftarrow v_i^j$ is called a *solution component* $c_{ij} \in C$, where C denotes the set of solution components. A pheromone trail value τ_{ij} is associated with each component $c_{ij} \in C$. (More formally, each solution component has an associated pheromone variable that can take a value, the pheromone trail value, in a specific range.) A solution construction starts from an initially empty partial solution s^p. At each construction step, s^p is extended by appending to it a feasible solution component from the set of its feasible neighbors $N(s^p) \subseteq C$ that satisfies the constraints in Ω. The choice of a solution component is guided by a stochastic decision policy, which is biased by both the pheromone trail and the heuristic values associated with c_{ij}. The exact rules for the probabilistic choice of solution components vary across different variants of ACO. The best known rule is the one used first in the ant system algorithm [4]

$$p_{c_{ij}|s^p} = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{c_{il} \in N(s^p)} [\tau_{il}]^\alpha \cdot [\eta_{il}]^\beta}, \tag{1}$$

where τ_{ij} and η_{ij} are, respectively, the pheromone trail value and the heuristic value associated with the component c_{ij}. The parameters $\alpha > 0$ and $\beta > 0$ determine the relative importance of pheromone versus heuristic information.

- *DeamonActions.* This procedure, although optional, is important when state-of-the-art results are sought [7]. It allows the execution of problem-specific operations, such as the use of local search procedures, or of centralized actions that cannot be performed by artificial ants. It is usually executed before the update of pheromone values in order to bias the ants' search toward high quality solutions.
- *UpdatePheromones.* This procedure updates the pheromone trail values associated with the solution components in the set C. The modification of the pheromone trail values is performed in two stages: (i) *pheromone evaporation*, which decreases the pheromone values of all components by a constant factor ρ (called *evaporation rate*) in order to avoid premature convergence, and (ii) *pheromone deposit*, which increases the pheromone trail values associated with components of a set of promising solutions S_{upd}. The general form of the pheromone update rule is as follows:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \sum_{s \in S_{upd} | c_{ij} \in s} F(s), \tag{2}$$

where $\rho \in (0, 1]$ is the evaporation rate, and $F : S \to \mathbb{R}^+$ is a function such that $f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in S$. $F(\cdot)$ is called the *fitness function*. Different definitions for the set S_{upd} exist. Two common choices are $S_{upd} = s_{bsf}$, and $S_{upd} = s_{ib}$, where s_{bsf} is the best-so-far solution, that is, the best solution found since the start of the algorithm, and s_{ib} is the best solution of the current iteration. The specific implementation of the pheromone update mechanism differs across ACO variants [1,4,13–15].

When applying the ACO metaheuristic to a specific problem, the definition of solution components and, hence, the definition of the interpretation of the pheromone trails is decisive for the final performance of the ACO algorithm. In fact, even when restricting to problems where candidate solutions can be represented by a same representation (e.g., permutations), different interpretations for solution components and pheromone trails may be useful. For example, while in the TSP case (see Example 1 in the previous section), the successor relationship is important, that is, τ_{ij} should refer to the desirability of visiting city j directly after city i, in scheduling applications, it is often preferable to interpret a pheromone trail τ_{ij} as the desirability of assigning a job j to position i. When facing problems for which several alternative definitions of pheromone are reasonable, which one would be the best choice has to be determined experimentally.

ACO ALGORITHMS

The ACO metaheuristic is a general algorithmic framework. Various specific ACO algorithms, which all follow the high-level rules of the ACO metaheuristic, have been proposed in the literature. In fact, these variants are obtained by various instantiations of the three main procedures that build the ACO metaheuristic. Some of the most noteworthy variants are described below.

Ant System

Ant System (AS) was the first ACO algorithm reported in the literature [2–4]. In AS, the pheromone values are updated at each iteration by all m ants of the colony. All pheromone trail values τ_{ij} are updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{3}$$

where ρ is the evaporation rate, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on c_{ij} by ant k. $\Delta\tau_{ij}^{k}$ is defined as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} F(s_k) & \text{if component } (i,j) \text{ is in the} \\ & \text{solution constructed by ant } k, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where the value of $F(s_k)$ is a function of the quality of the solution constructed by ant k. Normally, the better the solution, the higher is the amount of pheromone deposited.

In the solution construction, ants select solution components according to a stochastic mechanism, following Equation (1).

AS is mainly of historical interest because it was the first ACO algorithm proposed in the literature. Initial computational results have been interesting in the sense that they showed that the underlying mechanism works and allows to find good quality solutions. However, the performance of AS was still quite far from state-of-the-art methods. The main importance of AS is that it has seeded follow-up work by various researchers on better performing algorithmic variants, such as the two presented next.

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System

$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [15] is an improvement over the original Ant System. Its main features are (i) only one of the best ants deposits pheromone, and (ii) the range of the allowed pheromone trail values is bounded. The pheromone update is implemented as follows:

$$\tau_{ij} \leftarrow \begin{cases} \tau_{\min} & \text{if } \tau_{ij} < \tau_{\min}, \\ (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{\text{best}} & \text{if } \tau_{\min} \le \tau_{ij} \le \tau_{\max}, \\ \tau_{\max} & \text{if } \tau_{ij} > \tau_{\max}, \end{cases} \tag{5}$$

where $\tau_{\max}$ and $\tau_{\min}$ are, respectively, the upper and lower bounds imposed on the pheromone, and $\Delta\tau_{ij}^{\text{best}}$ is defined as

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} F(s_{\text{best}}) & \text{if solution component } (i,j) \\ & \text{is part of } s_{\text{best}}, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where the value of $F(s_{\text{best}})$ is a function of the quality of the best solution found. This solution can be s_{ib}, s_{bsf}, a combination of them, or possibly some other high-quality solution.

Concerning the lower and upper bounds on the pheromone values, a bound on the maximum value may be calculated analytically as $\tau_{\max}^{b} = F(s^{*})/\rho$, if the optimal solution s^{*} is known [16]. If s^{*} is not known, it can be approximated by s_{bsf}. Usually, setting $\tau_{\max} = \tau_{\max}^{b}$ (or to its approximation) results in good behavior of $\mathcal{MM}$AS. The initial value of the trails is set to $\tau_{\max}$ to increase the diversification of the search at the start of the algorithm. Some heuristic considerations for defining the setting of $\tau_{\min}$ have been proposed [15,17]. Finally, $\mathcal{MM}$AS was the first ACO algorithm to use additional mechanisms for increasing the diversification of the search such as a reinitialization of the pheromone trails or a smoothing of the pheromone trail values when no improvement is observed for a given number of iterations. For a detailed description of these mechanisms, a general overview of $\mathcal{MM}$AS, and some variants of $\mathcal{MM}$AS, we refer the reader to [17].

Ant Colony System

Ant Colony System (ACS) [13] differs in some key aspects from other ACO algorithms. The first is that it uses a different decision rule in the ants' solution construction, which is known as the *pseudorandom proportional rule*. In this rule, with probability q_0 the next solution component j is the one that maximizes the product of the pheromone and heuristic values, that is, $j = \arg\max_{c_{il}\in N(s^p)}\{\tau_{il}\eta_{il}^{\beta}\}$. With probability $1-q_0$ the probabilistic choice is made using Equation (1).

Similar to $\mathcal{MMAS}$, a pheromone update is applied at the end of each iteration by only one ant. The ACS pheromone update formula is as follows:

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij}^{\text{best}} & \text{if solution} \\ \quad \text{component } (i,j) \text{ is part of } s_{\text{best}}, \\ \tau_{ij} \quad \text{otherwise.} \end{cases} \tag{7}$$

As in $\mathcal{MMAS}$, $\Delta\tau_{ij}^{\text{best}} = F(s_{\text{best}})$, where s_{best} can be either s_{ib} or s_{bsf}. It is noteworthy that in ACS only the pheromone values of solution components associated with the best solution are updated.

To avoid search stagnation, in ACS a local pheromone update is performed by each ant after each construction step. This update decreases the pheromone trail value of the solution component that has been chosen in the previous step. The goal is to diversify the search performed by subsequent ants during an iteration: by decreasing the pheromone concentration for chosen components, these get less desirable for subsequent ants, thus increasing the chances of producing different solutions. Each ant applies the local pheromone update only to the pheromone trail of the last solution component added

$$\tau_{ij} \leftarrow (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_0, \tag{8}$$

where $\varphi \in (0,1)$ is a parameter called *pheromone decay coefficient*, and τ_0 is a parameter that determines the initial value for the pheromone trails. A good value for τ_0 was found to be $F(s^h)/n$, where n is the size of the instance and s^h is a solution constructed using a problem-specific heuristic [7].

Other Variants

In addition to the variants described above, there are others that have been reported in the literature. Table 1 summarizes the main ACO variants, including those discussed in the previous sections, which have been proposed in the literature for the approximate solution of NP-hard problems. For each of these variants, we give the main references and the year in which they were proposed. The main characteristics of the ACO algorithms that were not discussed so far are the following. Elitist AS is a direct variant of AS that gives a strong additional feedback to the best solution constructed since the start of the algorithm. Ant-Q is a predecessor of ACS that is inspired by the well-known Q-learning method from reinforcement learning. Rank-based AS extends Elitist AS by allowing not only the best-so-far ant, but also the r best ranked ants of the current iteration to deposit pheromone; the weight given to each ant in the pheromone update is inversely proportional to its rank, the highest weight being given to the best-so-far ant. ANTS is an ACO algorithm that exploits the connection with tree-search procedures by including elements from branch-and-bound techniques, such as lower bound information, into an ACO algorithm. Best-worst AS is an AS variant where the worst ant of the current iteration is used to subtract pheromone from solution components that are part of this worst ant but that do not occur in the best-so-far solution. Population-based ACO uses a set of elite solutions to define, at each iteration, the pheromone trail matrix. The set of elite solutions is managed by a population management mechanism that is responsible for updating the pheromone matrix each time a solution is added or removed from the elite set. Finally, Beam-ACO incorporates a heuristic derived from branch-and-bound algorithms called *beam search*.

Table 1. Overview of the main ACO algorithms for NP-hard problems that have been proposed in the literature

ACO Algorithm	Main References	Year
Ant System (AS)	[2–4]	1991
Elitist AS	[2–4]	1992
Ant-Q	[18]	1995
Ant Colony System	[13,14]	1996
$\mathcal{MAX}$–$\mathcal{MIN}$ AS	[15,19,20]	1996
Rank-based AS	[21,22]	1997
ANTS	[23,24]	1998
Best-worst AS	[25,26]	2000
Population-based ACO	[27]	2002
Beam-ACO	[28,29]	2004

Given are the ACO algorithm names, the main references where these algorithms are described, and the year in which they were first published.

Hybrid ACO Algorithms

Currently, it is a well-established fact that ACO algorithms reach best performance for most combinatorial optimization problems when they are combined with either iterative improvement algorithms or more complex local search methods such as tabu search or simulated annealing [7]. In these hybrid algorithms, the local improvement methods are used to improve the solutions constructed by one or more ants after each iteration. The usage of local search algorithms is also one example of the daemon actions that have been mentioned in the description of the ACO metaheuristic (see Fig. 1).

Various alternative ways of hybridizing ACO algorithms with other techniques have been studied. The ANTS and Beam-ACO algorithms, mentioned in the previous section, were explicitly designed as hybrid algorithms that integrate features from branch-and-bound techniques into ACO algorithms. Another active area is the integration of constraint programming techniques into ACO algorithms [30,31], which is particularly attractive for problems where, due to the problem constraints, it is difficult for the ants to generate feasible candidate solutions. Other hybrid techniques exploit the idea of using partial candidate solutions to seed an ant's solution construction. Examples of these hybrid methods are the use of external memory in ACO algorithms [32] or the extensions called *iterated ants* [33] and *cunning ants* [34].

The investigation of hybrid ACO algorithms is currently one of the most active areas in the research on ACO.

ACO APPLICATIONS

ACO algorithms have been successfully applied to a large variety of important problems from both the academic and industrial worlds (see the article ***A Concise Overview of Applications of Ant Colony Optimization*** for more information). The main application areas are the following:

NP-Hard Problems. The best known algorithms that are guaranteed to find an optimal solution to this kind of problems have exponential time complexity in the worst case [35]. However, heuristic methods such as ACO can be used to find high-quality solutions in a reasonable amount of time. Some examples of NP-hard problems for which ACO algorithms have been successful are *routing problems* [36–38], in which the goal is to find the shortest route that visits a set of locations; *assignment problems* [15], where a set of items (objects, activities, etc.) has to be assigned to a given number of resources (locations, agents, etc.) subject to some constraints; *subset problems* [39], where a solution to a problem is considered to be a selection of a subset of available items; and *scheduling problems* [29], in which the main concern is to optimally allocate scarce resources to tasks over time.

Rich Academic and Industrial Problems. After initial encouraging results on classic academic problems, ACO started to be applied to real industrial problems such as those arising in the food or in manufacturing industry [38,40]. As a result, richer versions of the academic problems started to be studied. Among the features of these problems are time-varying data [41], stochasticity [42,43], the presence of multiple objectives [44,45], continuous variables [46], mixed variables [47], and so on. Practically relevant dynamic problems are those found in the domain of telecommunication networks because some important properties, such as the cost of using links or the availability of nodes, vary over time. Some ACO algorithms have been shown to be very effective at solving these types of problems [48–50].

ACO THEORY

Most of the research results on metaheuristics, in general, and on ACO, in particular, are of experimental nature. However, there is also a significant interest in more fundamental properties of ACO algorithms.

A first question that is usually asked is whether, given enough time, the algorithm will eventually find an optimal solution. An initial answer to this question was given by Gutjahr, who proved for an ACO algorithm called *graph based ant system* (GBAS) the convergence with probability $1 - \epsilon$ to the optimal solution [51]. In a later paper [52], convergence with probability 1 was proven for two variants of GBAS. While GBAS has not been studied in practical applications, it is remarkable that convergence proofs for two of the practically most successful ACO algorithms, ACS and $\mathcal{MM}$AS, have also been obtained [53]. More recently, the focus of research has shifted to studies of the expected runtime to find optimal solutions in ACO applications to specific problems. An overview of proof techniques and some results are given by Gutjahr [54]; recent publications in this direction can be found in [55–57].

Other contributions on theoretical aspects of ACO have focused on establishing connections to other methods. Zlochin *et al*. [58] have defined the framework of model-based search algorithms, of which ACO is one representative. Connections of ACO to stochastic gradient descent, an algorithm used, for example, for learning weights in neural networks, have been studied in [59]. Of more practical interest are studies on the behavior of ACO algorithms. Merkle and Middendorf were the first to analyze the dynamic behavior of the pheromone model in ACO algorithms [60]. Search bias in ACO algorithms is studied in [61], where the authors show that ACO algorithms may suffer from the same type of deception as evolutionary algorithms do. In addition, they show that ACO algorithms may suffer from what they call a second order deceptive behavior, where, due to an interaction between the pheromone update and the pheromone model chosen, the quality of the solutions generated by an ACO algorithm can decrease over time.

A more detailed discussion of theoretical results about ACO algorithms is given in [7,62].

CONCLUSIONS

ACO is now one of the main metaheuristics and an active area of research. Early research on ACO focused mainly on the development of effective ACO algorithm variants and a common framework for these algorithmic developments is given by the ACO metaheuristic. Currently, the main active research directions in ACO concern applications to computationally challenging problems, the hybridization of ACO algorithms with other search techniques, and the theoretical study of the behavior of specific ACO algorithms.

Evidence of the success of ACO algorithms is the number of specialized meetings, where researchers can discuss their research results on ACO algorithms and their applications. ACO is one of the main subjects of the biannual conference ANTS (International Conference on Swarm Intelligence; http://iridia.ulb.ac.be/ants/) and of the IEEE Swarm Intelligence Symposium series. In addition, ACO is a central topic at various conferences on metaheuristics and evolutionary algorithms. Finally, research on ACO has frequently been featured in journal special issues [63–66] and is a fundamental subject of the journal *Swarm Intelligence*. Information on ACO and related topics can be obtained through the moderated mailing list `aco-list`, and the ACO web page (www.aco-metaheuristic.org).

Acknowledgments

This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium and the E-SWARM ERC Advanced Grant. Marco Dorigo and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Director and a Research Associate, respectively.

REFERENCES

1. Dorigo M, Maniezzo V, Colorni A. Positive feedback as a search strategy. Italy: Dipartimento di Elettronica, Politecnico di Milano; 1991. pp. 91–016.

2. Dorigo M, Maniezzo V, Colorni A. The ant system: an autocatalytic optimizing process. Italy: Dipartimento di Elettronica, Politecnico di Milano; 1991. pp. 91–016. Revised.
3. Dorigo M. Optimization, learning and natural algorithms (in Italian). Italy: Dipartimento di Elettronica, Politecnico de Milano; 1992.
4. Dorigo M, Maniezzo V, Colorni A. Ant System: optimization by a colony of cooperating agents. IEEE Trans Syst Man and Cybern-Part B 1996;26(1):29–41.
5. Dorigo M, Di Caro G. The ant colony optimization meta-heuristic. In: Corne D, editor. New ideas in optimization. London: McGraw Hill; 1999. pp. 11–32.
6. Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behaviour. Nature 2000;406(6791):39–42.
7. Dorigo M, Stützle T. Ant Colony Optimization. Cambridge (MA): MIT Press; 2004.
8. Dorigo M. Ant colony optimization. Scholarpedia 2007;2(3):1461.
9. Grassé PP. La reconstruction du nid et les coordinations interindividuelles chez *Bellicositermes natalensis et Cubitermes sp.* La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insectes Sociaux 1959;6:41–81.
10. Pasteels JM, Deneubourg JL, Goss S. Self-organization mechanisms in ant societies (I): trail recruitment to newly discovered food sources. Experientia Suppl 1987; 54:155–175.
11. Goss S, Aron S, Deneubourg JL, *et al*. Self-organized shortcuts in the Argentine ant. Naturwissenschaften 1989;76:579–581.
12. Deneubourg JL, Aron S, Goss S, *et al*. The self-organizing exploratory pattern of the Argentine ant. J Insect Behav 1990;3(2):159–168.
13. Dorigo M, Gambardella LM. Ant Colony System: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1997;1(1):53–66.
14. Gambardella LM, Dorigo M. Solving symmetric and asymmetric TSPs by ant colonies. In: ICEC'96 Proceedings of the IEEE Conference on Evolutionary Computation. Piscataway (NJ): IEEE Press; 1996. pp. 622–627.
15. Stützle T, Hoos HH. $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. Future Gener Comput Syst 2000;16(8): 889–914.
16. Stützle T, Dorigo M. A short convergence proof for a class of ant colony optimization algorithms. IEEE Trans Evol Comput 2002; 6(4):358–365.
17. Stützle T. Volume 220, Local search algorithms for combinatorial problems: analysis, improvements, and new applications, DISKI. Germany: Infix, Sankt Augustin; 1999.
18. Gambardella LM, Dorigo M. Ant-Q: a reinforcement learning approach to the traveling salesman problem. In: Prieditis A, Russell S, editors. Proceedings of the 12th International Conference on Machine Learning (ML-95). Palo Alto (CA): Morgan Kaufmann Publishers; 1995. pp. 252–260.
19. Stützle T, Hoos HH. Improving the Ant System: a detailed report on the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. Germany: FG Intellektik, FB Informatik, TU Darmstadt; 1996. AIDA-96-12.
20. Stützle T, Hoos HH. The $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System and local search for the traveling salesman problem. In: Bäck T, Michalewicz Z, Yao X, editors. Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97). Piscataway (NJ): IEEE Press; 1997. pp. 309–314.
21. Bullnheimer B, Hartl RF, Strauss C. A new rank based version of the Ant System—a computational study. Vienna: Institute of Management Science, University of Vienna; 1997.
22. Bullnheimer B, Hartl RF, Strauss C. A new rank-based version of the Ant System: a computational study. Cent Eur J Oper Res Econ 1999;7(1):25–38.
23. Maniezzo V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Italy: Scienze dell'Informazione, Universitá di Bologna, Sede di Cesena; 1998. CSR 98-1.
24. Maniezzo V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. INFORMS J Comput 1999;11(4):358–369.
25. Cordón O, de Viana IF, Herrera F. Analysis of the best-worst Ant System and its variants on the TSP. Mathware Soft Comput 2002;9(2–3):177–192.
26. Cordón O, de Viana IF, Herrera F, *et al*. A new ACO model integrating evolutionary computation concepts: the best-worst Ant System. In: Dorigo M, Middendorf M, Stützle T, editors. Abstract proceedings of ANTS 2000-From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms. Brussels: IRIDIA, Université Libre de Bruxelles; 2000. pp. 22–29.

27. Guntsch M, Middendorf M. A population based approach for ACO. In: Cagnoni S, *et al*., editors. Volume 2279, Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002, LNCS. Berlin: Springer; 2002. pp. 71–80.
28. Blum C. Theoretical and practical aspects of ant colony optimization. Brussels: IRIDIA, Université Libre de Bruxelles; 2004.
29. Blum C. Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling. Comput Oper Res 2005;32(6):1565–1591.
30. Meyer B, Ernst A. Integrating ACO and constraint propagation. In: Dorigo M, *et al*., editors. Volume 3172, Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, LNCS. Berlin: Springer; 2004. pp. 166–177.
31. Khichane M, Albert P, Solnon C. Integration of ACO in a constraint programming language. In: Dorigo M, *et al*., editors. Volume 5217, Ant Colony Optimization and Swarm Intelligence, 6th International Conference, ANTS 2008, LNCS. Berlin: Springer; 2008. pp. 84–95.
32. Acan A. An external memory implementation in ant colony optimization. In: Dorigo M, *et al*., editors. Volume 3172, Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004, LNCS. Berlin: Springer; 2004. pp. 73–84.
33. Wiesemann W, Stützle T. Iterated ants: an experimental study for the quadratic assignment problem. In: Dorigo M, *et al*., editors. Volume 4150, Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, LNCS. Berlin: Springer; 2006. pp. 179–190.
34. Tsutsui S. cAS: ant colony optimization with cunning ants. In: Runarsson TP, *et al*., editors. Volume 4193, Parallel Problem Solving from Nature-PPSN IX, 9th International Conference, LNCS. Berlin: Springer; 2006. pp. 162–171.
35. Garey MR, Johnson DS. Computers and Intractability: a Guide to the Theory of NP-Completeness. New York: W.H. Freeman & Co.; 1979.
36. Gambardella LM, Dorigo M. Ant Colony System hybridized with a new local search for the sequential ordering problem. INFORMS J Comput 2000;12(3):237–255.
37. Reimann M, Doerner K, Hartl RF. D-Ants: savings based ants divide and conquer the vehicle routing problem. Comput Oper Res 2004;31(4):563–591.
38. Rizzoli AE, Montemanni R, Lucibello E, Gambardella LM. Ant colony optimization for real-world vehicle routing problems: from theory to applications. Swarm Intell 2007; 1(2):135–151.
39. Blum C, Blesa MJ. New metaheuristic approaches for the edge-weighted k-cardinality tree problem. Comput Oper Res 2005; 32(6):1355–1377.
40. Dorigo M, Birattari M, Stützle T. Ant colony optimization: artificial ants as a computational intelligence technique. IEEE Comput Intell Mag 2006;1(4):28–39.
41. Montemanni R, Gambardella LM, Rizzoli AE, *et al*. Ant Colony System for a dynamic vehicle routing problem. J Comb Optim 2005; 10(4):327–343.
42. Bianchi L, Gambardella LM, Dorigo M. An ant colony optimization approach to the probabilistic traveling salesman problem. In: Merelo JJ, *et al*., editors. Volume 2439, Parallel Problem Solving from Nature–PPSN VII, 7th International Conference, LNCS. Berlin: Springer; 2002. pp. 883–892.
43. Balaprakash P, Birattari M, Stützle T, *et al*. Estimation-based ant colony optimization algorithms for the probabilistic travelling salesman problem. Sist Intell 2009;3(3): 223–242.
44. Doerner K, Gutjahr WJ, Hartl RF, *et al*. Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection. Ann Oper Res 2004;131(1–4):79–99.
45. Angus D, Woodward C. Multiple objective ant colony optimisation. Swarm Intell 2007; 3(1):69–85.
46. Socha K, Dorigo M. Ant colony optimization for continuous domains. Eur J Oper Res 2008;185(3):1155–1173.
47. Socha K. ACO for continuous and mixed-variable optimization. In: Dorigo M, *et al*., editors. Volume 3172, Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004, LNCS. Berlin: Springer; 2004. pp. 25–36.
48. Di Caro G, Dorigo M. AntNet: distributed stigmergetic control for communications networks. J Artif Intell Res 1998;9:317–365.
49. Mong Sim K, Hong Sun W. Ant colony optimization for routing and load-balancing: survey and new directions. IEEE Trans Syst Man Cybern Part A: Syst Humans 2003; 33(5):560–572.

50. Di Caro G, Ducatelle F, Gambardella LM. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. Eur Trans Telecommun 2005;16(5):443–455.

51. Gutjahr WJ. A Graph-based Ant System and its convergence. Future Gener Comput Syst 2000;16(8):873–888.

52. Gutjahr WJ. ACO algorithms with guaranteed convergence to the optimal solution. Inf Process Lett 2002;82(3):145–153.

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

54. Gutjahr WJ. Mathematical runtime analysis of ACO algorithms: survey on an emerging issue. Swarm Intell 2007;1(1):59–79.

55. Gutjahr WJ, Sebastiani G. Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodol Comput Appl Probab 2008;10:409–433.

56. Neumann F, Sudholt D, Witt C. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intell 2009;3(1):35–68.

57. Neumann F, Witt C. Runtime analysis of a simple ant colony optimization algorithm. Algorithmica 2009;54(2):243–255.

58. Zlochin M, Birattari M, Meuleau N, *et al*. Model-based search for combinatorial optimization: a critical survey. Ann Oper Res 2004;131(1–4):373–395.

59. Meuleau N, Dorigo M. Ant colony optimization and stochastic gradient descent. Artif Life 2002;8(2):103–121.

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

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

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

63. Cordón O, Herrera F, Stützle T. Special issue on Ant Colony Optimization: Models and applications. Mathware Soft Comput 2003;9(2–3):137–268.

64. Doerner KF, Merkle D, Stützle T. Special issue on ant colony optimization. Swarm Intell. 2009;3(1):1–85.

65. Dorigo M, Di Caro G, Stützle T. Special issue on “Ant Algorithms”. Future Gener Comput Syst 2000;16(8):851–946.

66. Dorigo M, Gambardella LM, Middendorf M, *et al*. Special issue on “Ant Algorithms and Swarm Intelligence”. IEEE Trans Evol Comput 2002;6(4):317–365.