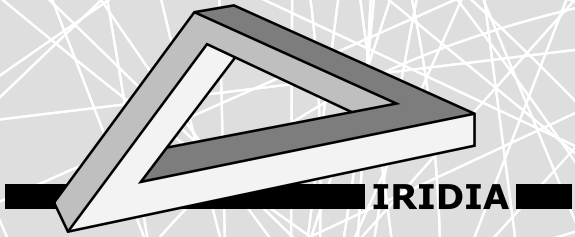

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

An Introduction to Ant Colony Optimization

Marco Dorigo and Krzysztof Socha

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2006-010

April 2006

Accepted for publication as a chapter in Approximation Algorithms and Metaheuristics, a book edited by T. F. Gonzalez.

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2006-010

An Introduction to Ant Colony Optimization

Marco Dorigo* and Krzysztof Socha†

IRIDIA, Université Libre de Bruxelles, CP 194/6,

Av. Franklin D. Roosevelt 50, 1050 Brussels, Belgium

http://iridia.ulb.ac.be

December 16, 2005

1 Introduction

This chapter presents an overview of ant colony optimization (ACO) – a metaheuristic inspired by the behavior of real ants. ACO was proposed by Dorigo and colleagues [18, 14, 19] as a method for solving hard combinatorial optimization problems.

ACO algorithms may be considered to be part of *swarm intelligence*, that is, the research field that studies algorithms inspired by the observation of the behavior of *swarms*. Swarm intelligence algorithms are made up of simple individuals that cooperate through self-organization, that is, without any form of central control over the swarm members. A detailed overview of the self-organization principles exploited by these algorithms, as well as examples from biology, can be found in [8]. Many swarm intelligence algorithms have been proposed in the literature. For an overview of the field of swarm intelligence, we refer the interested reader to [4].

This chapter, which is dedicated to present a concise overview of ACO, is organized as follows. Section 2 presents the biological phenomenon that provided the original inspiration. Section 3 presents a formal description of the ACO metaheuristic. Section 4 overviews the most popular variants of ACO and gives

*mdorigo@ulb.ac.be

†ksocha@ulb.ac.be

examples of their application. Section 5 shows current research directions, and Section 6 summarizes and concludes the chapter.

2 From Biology to Algorithms

Ant colony optimization was inspired by the observation of the behavior of real ants. In this section, we present a number of observations made in experiments with real ants, and then we show how these observations inspired the design of the ACO metaheuristic.

2.1 Ants

One of the first researchers to investigate the social behavior of insects was the French entomologist Pierre-Paul Grassé. In the forties and fifties of the 20-th century, he was observing the behavior of termites – in particular, the *Bellicositermes natalensis* and *Cubitermes* species. He discovered [26] that these insects are capable to react to what he called "significant stimuli", signals that activate a genetically encoded reaction. He observed [27] that the effects of these reactions can act as new significant stimuli for both the insect that produced them and for the other insects in the colony. Grassé used the term *stigmergy* [27] to describe this particular type of indirect communication in which the "workers are stimulated by the performance they have achieved".

The two main characteristics of stigmergy that differentiate it from other means of communication are:

- the physical, non-symbolic nature of the information released by the communicating insects, which corresponds to a modification of physical environmental states visited by the insects; and
- the local nature of the released information, which can only be accessed by those insects that visit the place where it was released (or its immediate neighborhood).

Examples of stigmergy can be observed in colonies of ants. In many ant species, ants walking to, and from, a food source deposit on the ground a substance called *pheromone*. Other ants are able to smell this pheromone, and its presence influences the choice of their path—i.e., they tend to follow strong pheromone

concentrations. The pheromone deposited on the ground forms a *pheromone trail*, which allows the ants to find good sources of food that have been previously identified by other ants.

Some researchers investigated experimentally this pheromone laying and following behavior in order to better understand it and to be able to quantify it. Deneubourg *et al.* [11] set up an experiment called a "binary bridge experiment". They used *Linepithema humile* ants (also known as Argentine ants). The ants' nest was connected to a food source by two bridges of equal length. The ants could freely choose which bridge to use when searching for food and bringing it back to the nest. Their behavior was then observed over a period of time.

In this experiment, initially there is no pheromone on the two bridges. The ants start exploring the surroundings of the nest and eventually cross one of the bridges and reach the food source. When walking to the food source and back, the ants deposit pheromone on the bridge they use. Initially, each ant randomly chooses one of the bridges. However, due to random fluctuations, after some time there will be more pheromone deposited on one of the bridges than on the other. Because ants tend to prefer in probability to follow a stronger pheromone trail, the bridge that has more pheromone will attract more ants. This in turn makes the pheromone trail grow stronger, until the colony of ants converges towards the use of a same bridge.[1]

This colony level behavior, based on autocatalysis, that is, on the exploitation of positive feedback, can be exploited by ants to find the shortest path between a food source and their nest. This was demonstrated in another experiment conducted by Goss *et al.* [25], in which the two bridges were not of the same length: one was significantly longer than the other. In this case, the stochastic fluctuations in the initial choice of a bridge were much reduced as a second mechanism played an important role: those ants choosing by chance the shorter bridge were also the first to reach the nest and when returning to the nest they chose the shorter bridge with higher probability as it had a stronger pheromone trail. Therefore, the ants—thanks to the pheromone following and depositing mechanism—quickly converged to the use of the shorter bridge.

In the next section we explain how these experiments and findings were used to develop optimization algorithms.

[1] Deneubourg *et al.* conducted several experiments, and results show that each of the two bridges was used in about 50% of the cases.

2.2 Algorithms

Stimulated by the interesting results of the experiments described in the previous section, Goss *et al.* [25] developed a model to explain the behavior observed in the binary bridge experiment. Assuming that after t time units since the start of the experiment, m_1 ants had used the first bridge and m_2 the second one, the probability p_1 for the $(m+1)$-th ant to choose the first bridge can be given by:

$$p_{1(m+1)} = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \ , \tag{1.1}$$

where parameters k and h are needed to fit the model to the experimental data. The probability that the same $(m+1)$-th ant chooses the second bridge is $p_{2(m+1)} = 1 - p_{1(m+1)}$. Monte Carlo simulations, run to test how the model corresponds to the real data [44], showed very good fit for $k \approx 20$ and $h \approx 2$.

This basic model, which explains the behavior of real ants, may be used as an inspiration to design artificial ants that solve optimization problems defined in a similar way. In the above described *ant foraging behavior* example, stigmergic communication happens via the pheromone that ants deposit on the ground. Analogously, artificial ants may simulate pheromone laying by modifying appropriate pheromone variables associated with problem states they visit while building solutions to the optimization problem. Also, according to the stigmergic communication model, the artificial ants would have only local access to these pheromone variables.

Therefore, the main characteristics of stigmergy mentioned in the previous section can be extended to artificial agents by:

- associating state variables with different problem states; and
- giving the agents only local access to these variables.

Another important aspect of real ants' foraging behavior that may be exploited by artificial ants is the coupling between the autocatalytic mechanism and the *implicit evaluation* of solutions. By implicit solution evaluation, we mean the fact that shorter paths (which correspond to lower cost solutions in the case of artificial ants) are completed earlier than longer ones, and therefore they receive pheromone reinforcement quicker. Implicit solution evaluation coupled with autocatalysis can be very effective: the shorter the path,

the sooner the pheromone is deposited, and the more ants use the shorter path. If appropriately used, it can be a powerful mechanism in population-based optimization algorithms (e.g., in evolutionary algorithms [33, 21] autocatalysis is implemented by the selection/reproduction mechanism).

Stigmergy, together with implicit solution evaluation and autocatalytic behavior, gave rise to ACO. The basic idea of ACO follows very closely the biological inspiration. Therefore, there are many similarities between real and artificial ants. Both real and artificial ant colonies are composed of a population of individuals that work together to achieve a certain goal. A colony is a population of simple, independent, asynchronous agents that cooperate to find a good *solution* to the problem at hand. In the case of real ants, the problem is to find the food, while in the case of artificial ants, it is to find a good solution to a given optimization problem. A single ant (either a real or an artificial one) is able to find a solution to its problem, but only cooperation among many individuals through stigmergy enables them to find *good* solutions.

In the case of real ants, they deposit and react to a chemical substance called *pheromone*. Real ants simply deposit it on the ground while walking. Artificial ants live in a *virtual* world, hence they only modify numeric values (called for analogy *artificial pheromones*) associated with different problem states. A sequence of pheromone values associated with problem states is called *artificial pheromone trail*. In ACO, the artificial pheromone trails are the sole means of communication among the ants. A mechanism analogous to the evaporation of the physical pheromone in real ant colonies allows the artificial ants to *forget* the past history and focus on new promising search directions.

Just like real ants, artificial ants create their solutions sequentially by moving from one problem state to another. Real ants simply walk, choosing a direction based on local pheromone concentrations and a stochastic decision policy. Artificial ants also create solutions step-by-step, moving through available problem states and making stochastic decisions at each step.

There are however some important differences between real and artificial ants:

- Artificial ants live in a discrete world – they move sequentially through a finite set of problem states.
- The pheromone update (i.e., pheromone depositing and evaporation) is not accomplished in exactly the same way by artificial ants as by real ones. Sometimes the pheromone update is done only by some of the artificial ants, and often *only after* a solution has been constructed.

- Some implementations of artificial ants use additional mechanisms that do not exist in the case of real ants. Examples include look-ahead, local search, backtracking, etc.

3 The Ant Colony Optimization Metaheuristic

Ant colony optimization (ACO) has been formalized into a combinatorial optimization metaheuristic by Dorigo *et al.* [15, 16, 20] and has since been used to tackle many combinatorial optimization problems.

Given a combinatorial optimization problem (COP), the first step for the application of ACO to its solution consists in defining an adequate model. This is then used to define the central component of ACO: the pheromone model. The model of a COP may be defined as follows:

Definition 1.1 *A model $P = (\mathbf{S}, \mathbf{\Omega}, f)$ of a COP consists of:*

- *a search space $\mathbf{S}$ defined over a finite set of discrete decision variables and a set $\mathbf{\Omega}$ of constraints among the variables;*
- *an objective function $f : \mathbf{S} \to \mathbb{R}_0^+$ to be minimized.*[2]

The search space $\mathbf{S}$ is defined as follows: Given is a set of discrete variables $X_i,\ i = 1, ..., n$, with values $v_i^j \in \mathbf{D}_i = \{v_i^1, ..., v_i^{|\mathbf{D}_i|}\}$. A variable instantiation, that is, the assignment of a value v_i^j to a variable X_i, is denoted by $X_i \leftarrow v_i^j$. A solution $s \in \mathbf{S}$—i.e., a complete assignment in which each decision variable has a value assigned—that satisfies all the constraints in the set $\mathbf{\Omega}$, is a feasible solution of the given COP. If the set $\mathbf{\Omega}$ is empty, P is called an unconstrained problem model, otherwise it is said to be constrained. A solution $s^* \in \mathbf{S}$ is called a global optimum if and only if: $f(s^*) \leq f(s)\ \forall_{s \in \mathbf{S}}$. The set of all globally optimal solutions is denoted by $\mathbf{S}^* \subseteq \mathbf{S}$. Solving a COP requires finding at least one $s^* \in \mathbf{S}^*$.

The model of a COP is used to derive the pheromone model used by ACO. First, an instantiated decision variable $X_i = v_i^j$ (i.e., a variable X_i with a value v_i^j assigned from its domain $\mathbf{D}_i$), is called a *solution component* and denoted by c_{ij}. The set of all possible solution components is denoted by $\mathbf{C}$. A pheromone trail parameter T_{ij} is then associated with each component c_{ij}. The set of all pheromone trail parameters

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

Algorithm 1 Ant colony optimization metaheuristic

```
Set parameters, initialize pheromone trails
while termination conditions not met do
  ConstructAntSolutions
  ApplyLocalSearch        {optional}
  UpdatePheromones
end while
```

is denoted by $\mathbf{T}$. The value of a pheromone trail parameter T_{ij} is denoted by τ_{ij} (and called pheromone value).[3] This pheromone value is then used and updated by the ACO algorithm during the search. It allows modeling the probability distribution of different components of the solution.

In ACO, artificial ants build a solution to a combinatorial optimization problem by traversing the so-called *construction graph*, $G_C(\mathbf{V}, \mathbf{E})$. The fully connected construction graph consists of a set of vertexes $\mathbf{V}$ and a set of edges $\mathbf{E}$. The set of components $\mathbf{C}$ may be associated either with the set of vertexes $\mathbf{V}$ of the graph G_C, or with the set of its edges $\mathbf{E}$. The ants move from vertex to vertex along the edges of the graph, incrementally building a *partial solution*. Additionally, the ants deposit a certain amount of pheromone on the components, that is, either on the vertexes or on the edges that they traverse. The amount $\Delta\tau$ of pheromone deposited may depend on the quality of the solution found. Subsequent ants utilize the pheromone information as a guide towards more promising regions of the search space.

The ACO metaheuristic is shown in Algorithm 1. It consists of an initialization step and a loop over three algorithmic components. A single iteration of the loop consists of constructing solutions by all ants, their (optional) improvement with the use of a local search algorithm, and an update of the pheromones. In the following, we explain these three algorithmic components in more detail.

ConstructAntSolutions: A set of m artificial ants construct solutions from elements of a finite set of available solution components $\mathbf{C} = \{c_{ij}\},\ i = 1, ..., n,\ j = 1, ..., |\mathbf{D}_i|$. A solution construction starts with an empty partial solution $s^p = \emptyset$. Then, at each construction step, the current partial solution s^p is extended by adding a feasible solution component from the set of feasible neighbors $\mathbf{N}(s^p) \subseteq \mathbf{C}$. The process of

[3] Note that pheromone values are in general a function of the algorithm's iteration $t : \tau_{ij} = \tau_{ij}(t)$.

constructing solutions can be regarded as a path on the construction graph $G_C = (\mathbf{V}, \mathbf{E})$. The allowed paths in G_C are hereby implicitly defined by the solution construction mechanism that defines the set $\mathbf{N}(s^p)$ with respect to a partial solution s^p.

The choice of a solution component from $\mathbf{N}(s^p)$ is done probabilistically at each construction step. The exact rules for the probabilistic choice of solution components vary across different ACO variants. The best known rule is the one of Ant System (AS) [19]:

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta(c_{ij})^{\beta}}{\sum_{c_{il} \in \mathbf{N}(s^p)} \tau_{il}^{\alpha} \cdot \eta(c_{il})^{\beta}}, \quad \forall c_{ij} \in \mathbf{N}(s^p), \tag{1.2}$$

where τ_{ij} is the pheromone value associated with the component c_{ij}, and $\eta(\cdot)$ is a function that assigns at each construction step a heuristic value to each feasible solution component $c_{ij} \in \mathbf{N}(s^p)$. The values that are given by this function are commonly called *heuristic information*. Furthermore, α and β are positive parameters, whose values determine the relative importance of pheromone versus heuristic information. Eq. 1.2 is a generalization of Eq. 1.1 presented in Sec. 2: ACO formalization follows closely the biological inspiration.

ApplyLocalSearch: Once solutions have been constructed, and before updating pheromones, often some optional actions may be required. These are often called *daemon actions*, and can be used to implement problem specific and/or centralized actions, which cannot be performed by single ants. The most used daemon action consists in the application of local search to the constructed solutions: the locally optimized solutions are then used to decide which pheromones to update.

UpdatePheromones: The aim of the pheromone update is to increase the pheromone values associated with good or promising solutions, and to decrease those that are associated with bad ones. Usually, this is achieved (i) by decreasing all the pheromone values through *pheromone evaporation*, and (ii) by increasing the pheromone levels associated with a chosen set of good solutions $\mathbf{S}_{upd}$:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \sum_{s \in \mathbf{S}_{upd} | c_{ij} \in s} F(s) \ , \tag{1.3}$$

where $\mathbf{S}_{upd}$ is the set of solutions that are used for the update, $\rho \in (0, 1]$ is a parameter called evaporation rate, and $F : \mathbf{S} \rightarrow \mathbb{R}_0^+$ is a function such that $f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in \mathbf{S}$. $F(\cdot)$ is commonly

called the *fitness function*.

Pheromone evaporation is needed to avoid a too rapid convergence of the algorithm. It implements a useful form of *forgetting*, favoring the exploration of new areas in the search space. Different ACO algorithms, such as for example Ant Colony System (ACS) [17] or $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [53] differ in the way they update the pheromone.

Instantiations of the update rule presented in Eq. 1.3 are obtained by different specifications of $\mathbf{S}_{upd}$, which in many cases is a subset of $\mathbf{S}_{iter} \cup \{s_{bs}\}$, where $\mathbf{S}_{iter}$ is the set of solutions that were constructed in the current iteration, and s_{bs} is the *best-so-far* solution, that is, the best solution found since the first algorithm iteration. A well-known example is the AS-update rule, that is, the update rule of Ant System [19], where:

$$\mathbf{S}_{upd} \leftarrow \mathbf{S}_{iter} \ . \tag{1.4}$$

An example of a pheromone update rule that is more often used in practice is the IB-update rule (where IB stands for *iteration-best*):

$$\mathbf{S}_{upd} \leftarrow \arg \max_{s \in \mathbf{S}_{iter}} F(s) \ . \tag{1.5}$$

The IB-update rule introduces a much stronger bias towards the good solutions found than the AS-update rule. Although this increases the speed with which good solutions are found, it also increases the probability of premature convergence. An even stronger bias is introduced by the BS-update rule, where BS refers to the use of the best-so-far solution s_{bs}. In this case, $\mathbf{S}_{upd}$ is set to $\{s_{sb}\}$. In practice, ACO algorithms that use variations of the IB-update or the BS-update rules and that additionally include mechanisms to avoid premature convergence, achieve better results than those that use the AS-update rule.

3.1 Example: The Traveling Salesman Problem

One of the most popular ways to illustrate how the ACO metaheuristic works, is via its application to the traveling salesman problem (TSP). The TSP consists of a set of locations (cities) and a travelling salesman that has to visit all the locations once and only once. The distances between the locations are given and the task is to find a Hamiltonian tour of minimal length. The problem has been proven to be NP-hard [35].

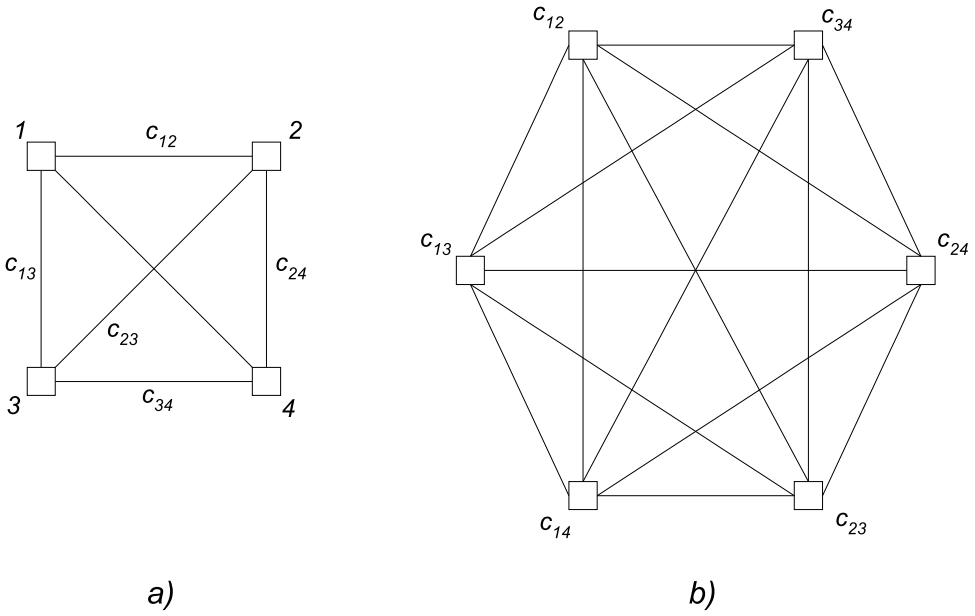

Figure 1.1: Example construction graphs for a 4-city TSP. a) - when components are associated with the edges of the graph, b) - when components are associated with the vertexes of the graph. Note that $c_{ij} \equiv c_{ji}$.

The application of ACO to the TSP is straightforward. The moves between the locations become the solution components—i.e, the move from city i to city j becomes a solution component $c_{ij} \equiv c_{ji}$. The construction graph $G_C = (\mathbf{V}, \mathbf{E})$ is defined by associating the set of locations with the set $\mathbf{V}$ of vertices of the graph. Since, in principle, it is possible to move from any city to any other one, the construction graph is fully connected and the number of vertices is equal to the number of locations defined by the problem instance. Furthermore, the lengths of the edges between the vertices are proportional to the distances between the locations represented by these vertices. The pheromone is associated with the set $\mathbf{E}$ of edges of the graph. An example of the resulting construction graph G_C is presented in Fig. 1.1a.

The ants construct the solutions as follows. Each ant starts from a randomly selected location (vertex of the graph G_C). Then, at each construction step it moves along the edges of the graph. Each ant keeps a memory of its path through the graph, and in subsequent steps it chooses among the edges that do not lead to vertexes that it has already visited. An ant has constructed a solution once it has visited all the vertexes of the graph. At each construction step an ant chooses probabilistically the edge to follow among the available ones (those that lead to yet unvisited vertices). The exact rule depends on the implementation,

an example being Eq. 1.2. Once all the ants have finished their tour, the pheromone on the edges is updated according to one of the possible implementations of Eq. 1.3. ACO has been shown to perform quite well on the TSP [51].

It is worth noticing that it is also possible to associate the set of solution components of the TSP (or any other combinatorial optimization problem) with the set of vertices **V** rather than the set of edges **E** of the construction graph G_C. For the TSP, this would mean associating the moves between locations with the set **V** of vertices of the construction graph, and the locations with the set **E** of its edges. The corresponding example construction graph for a 4-city TSP is presented in Fig. 1.1b. When using this approach, the ants' solution construction process has to be also properly modified: the ants would have to move from vertex to vertex of the construction graph choosing thereby the *connections between the cities*.

It is important to note that both ways of defining the construction graph are correct and both may be used in practice. Depending on the problem at hand, one may be more intuitive than the other. For instance, for the University Course Timetabling Problem (UCTP) the second one seems better suited [48].

4 Main Variants of ACO

Several variants of ACO have been proposed in the literature. We present the main characteristics of the most successful ones together with a short list of their applications. We attempt to present them in chronological order as new variants are often based on ideas introduced earlier.

In the following sections we present Ant System—the first implementation of an ACO algorithm—followed by $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System and Ant Colony System. We mention also some others that are less popular but still quite interesting, such as hyper-cube ACO or population-based ACO. In order to illustrate the differences between them clearly, we use the example of the traveling salesman problem, as described in Sec. 3.1.

4.1 Ant System

Ant System (AS) was the first ACO algorithm to be proposed in the literature [18, 14, 19]. Its main characteristic is that the pheromone values are updated by *all* the ants that have completed the tour. The

pheromone update for τ_{ij}, that is, for edge joining cities i and j, is performed as follows:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \ , \tag{1.6}$$

where ρ is the evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone per unit length laid on edge (i,j) by the k-th ant:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour,} \\ 0 & \text{otherwise,} \end{cases} \tag{1.7}$$

where Q is a constant, and L_k is the tour length of the k-th ant.

When constructing the solutions, the ants in AS traverse a construction graph and make probabilistic decision at each vertex. The transitional probability p_{ij}^{k} of the k-th ant moving from city i to city j is given by:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{l\in\text{allowed}_k}\tau_{il}^{\alpha}\cdot\eta_{il}^{\beta}} & \text{if } j\in\text{allowed}_k, \\ 0 & \text{otherwise,} \end{cases} \tag{1.8}$$

where allowed_k is the list of cities not yet visited by the k-th ant, and α and β are parameters that control the relative importance of the pheromone versus the heuristic information η_{ij} given by:

$$\eta_{ij} = \frac{1}{d_{ij}} \ , \tag{1.9}$$

where d_{ij} is the length of edge (i,j).

Several implementations of the AS algorithm have been applied to different combinatorial optimization problems. The first and best known is the application to the TSP [18, 14, 19]. However, AS was also used successfully to tackle other combinatorial problems. The AS-QAP [39, 38] algorithm was used to tackle quadratic assignment problem (QAP), AS-JSP [9] for the job-shop scheduling problem (JSP), AS-VRP [5, 6] for the vehicle routing problem (VRP), and AS-SCS [42, 43] for the shortest common supersequence (SCS) problem.

4.2 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System

$\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) is an improvement over the original Ant System idea. $\mathcal{MM}$AS was proposed by Stützle and Hoos [53] and introduces the following two changes:

- only the best ant can update the pheromone trails, and
- the minimum and maximum values of the pheromone are limited.

Equation 1.6 takes hence the following new form:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{\text{best}} \ , \tag{1.10}$$

where $\Delta\tau_{ij}^{\text{best}}$ is the pheromone update value defined by:

$$\Delta\tau_{ij}^{\text{best}} = \begin{cases} \frac{Q}{L_{\text{best}}} & \text{if the best ant used edge } (i,j) \text{ in its tour,} \\ 0 & \text{otherwise.} \end{cases} \tag{1.11}$$

L_{best} is the length of the tour of the best ant. This may be (subject to the algorithm designer decision) either the best tour found in the current iteration—*iteration-best*, L_{ib}—or the best solution found since the start of the algorithm—*best-so-far*, L_{bs}—or a combination of both.

Concerning the limits on the minimal and maximal pheromone values allowed, respectively τ_{min} and τ_{max}, Stützle and Hoos suggest that they should be chosen experimentally based on the problem at hand. The maximum value τ_{max} may be calculated analytically provided that the optimum ant tour length is known. In the case of the TSP, τ_{max} is given by:

$$\tau_{max} = \frac{1}{\rho}\cdot\frac{1}{L^*} \ , \tag{1.12}$$

where L^* is the length of the optimal tour. The minimum pheromone value τ_{min} should be chosen with caution as it has a rather strong influence on the algorithm performance. They present an analytical approach to finding this value based on the probability p_{best} that an ant constructs the best tour found so far. This is done as follows. First, it is assumed that at each construction step an ant has a constant number k of options available. Therefore, the probability that an ant makes the *right* decision (i.e., the decision that

belongs to the sequence of decisions leading to the construction of the best tour found so far) at each of n steps is given by $p_{\text{dec}} = \sqrt[n-1]{p_{\text{best}}}$. The analytical formula they suggest for finding τ_{min} is:

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - p_{dec})}{k \cdot p_{dec}} \ . \tag{1.13}$$

For more details on how to choose τ_{max} and τ_{min}, we refer to [53]. It is important to mention here that it has been also shown [48] that for some problems the choice of an appropriate τ_{min} value is more easily done experimentally than analytically.

The process of pheromone update in $\mathcal{MM}$AS is concluded by verifying that all pheromone values are within the imposed limits:

$$\tau_{ij} = \begin{cases} \tau_{max} & \text{if } \tau_{ij} > \tau_{max}, \\ \tau_{min} & \text{if } \tau_{ij} < \tau_{min}. \end{cases} \tag{1.14}$$

$\mathcal{MAX}$-$\mathcal{MIN}$ Ant System provided a significant improvement over the basic Ant System performance. While the first implementations focused on the TSP [53], it has been later applied to many other combinatorial optimization problems such as the QAP [52] or the university course timetabling problem (UCTP) [48], the generalized assignment problem (GAP) [37], and the set covering problem (SCP) [36].

4.3 Ant Colony System

Another improvement over the original Ant System was Ant Colony System (ACS) introduced by Gambardella and Dorigo [22, 17]. The most interesting contribution of ACS is the introduction of a *local pheromone update* in addition to the pheromone update performed at the end of the construction process (called here *offline* pheromone update).

The local pheromone update is performed by all the ants after each construction step. Each ant applies it only to the last edge traversed:

$$\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0 \ , \tag{1.15}$$

where $\varphi \in (0, 1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.

The main goal of the local update is to diversify the search performed by subsequent ants during one iteration. In fact, decreasing the pheromone concentration on the edges as they are traversed during one iteration encourages subsequent ants to choose other edges and hence to produce different solutions. This makes less likely that several ants produce identical solutions during one iteration.

The offline pheromone update, similarly to $\mathcal{MM}$AS, is applied at the end of each iteration by only one ant (the one that found the best solution in the iteration). However, the update formula is slightly different:

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij} & \text{if edge } (i,j) \text{ belongs to } T_{\text{best}}, \text{ the best tour found so far,} \\ \tau_{ij} & \text{otherwise,} \end{cases} \tag{1.16}$$

and in case of TSP, $\Delta\tau_{ij} = \frac{1}{L_{\text{best}}}$.

Another important difference between AS and ACS is in the decision rule used by the ants during the construction process. Ants in ACS use the so-called *pseudorandom proportional* rule: the probability for an ant to move from city i to city j depends on a random variable q uniformly distributed over $[0,1]$, and a parameter q_0; if $q \leq q_0$, then $j = \text{argmax}_{l\in N(s^p)}\{\tau_{il}\eta_{il}^{\beta}\}$, otherwise Eq. 1.8 is used.

ACS has been initially developed for the travelling salesman problem [22, 17], but it was later used to tackle various combinatorial optimization problems, including vehicle routing [1] and timetabling [49].

4.4 Others

In addition to the main variants of ACO just described, it is worth mentioning the hyper-cube ACO (HC-ACO) proposed by Blum [3], and population-based ACO (PB-ACO) proposed by Guntsch and Middendorf [29].

The main idea introduced by HC-ACO is the normalization of pheromone values used in the pheromone table. According to HC-ACO, the pheromone values should always be normalized in the interval $[0,1]$. It has been shown [3] that this makes the HC-ACO algorithm behavior independent of the scaling of the objective function, an issue for previous ACO algorithms.

Population-based ACO introduces a novel mechanism for pheromone updates. As in regular ACO, some of the good solutions found are used to increase the pheromone values. However, pheromone evaporation

is implemented differently. PB-ACO memorizes the solutions used to increase the pheromone values (the set of memorized solutions is called a "population", hence its name). Once the population has reached its maximum dimension (a parameter of the algorithm), the worst solutions in the population are removed to make room for the new ones. When a solution of the population is removed, the pheromone associated with it is also removed: this is obtained by applying a negative pheromone update.

5 Future Directions

Research in ant colony optimization is very active. It includes the application of ACO algorithms to new real-world optimization problems or new types of problems, such as dynamic optimization [28], multiobjective optimization [34], stochastic problems [32], or continuous and mixed-variable optimization [47]. Also, with an increasing popularity of parallel hardware architectures (multi-core processors and the grid technology), a lot of research is being done on creating parallel implementations of ACO that will be able to take advantage of the available hardware. In this section we shortly present current research in these new areas.

5.1 Other Types of Problems

One of the new areas of application of ACO is dynamic optimization. This type of problems are characterized by the fact that the search space dynamically changes. While an algorithm searches for good solutions, the conditions of the search as well as the quality of the solutions already found may change. This poses a whole new set of issues for designing successful algorithms that can deal with such situations. It becomes crucial for an algorithm to be able to adjust the search direction, following the changes of the problem being solved. Initial attempts to apply ACO to dynamic optimization problems have been quite successful [12, 31, 28] .

Multiobjective optimization is another area of application for metaheuristics that has received increasing attention over the past years. A multiobjective optimization problem involves solving simultaneously several optimization problems with potentially conflicting objectives. For each of the objectives, a different objective function is used to assess the quality of the solutions found. Algorithms usually aim at finding the so called *Pareto set*—i.e., a set of non-dominated solutions—based on the defined objective functions. In the Pareto set, no solution is worse than any other in the set, when evaluated over all the objective functions. Some

ACO algorithms designed to tackle multiobjective problems have been proposed in the literature [34, 30, 13].

Finally, recently researchers attempted to apply ACO algorithms to continuous optimization problems. When an algorithm designed for combinatorial optimization is used to tackle a continuous problem, the simplest approach is to divide the domain of each variable into a set of intervals. The set of intervals is finite and may be handled by the original discrete optimization algorithm. However, when the domain of the variables is large, and the required accuracy is high, this approach runs into problems. The problem size (i.e., the number of intervals) grows, and combinatorial optimization algorithms become less efficient. Also, this approach requires setting the number of intervals *a priori*—before the algorithm is run. In case of real-world problems, this is not always a sensible thing to do.

Due to these reasons, optimization algorithms able to handle continuous parameters natively have been developed. Recently, Socha [47] has extended ACO to continuous (and mixed-variable—continuous and discrete) problems. Research in this respect is ongoing and should result in new, efficient ACO implementations for continuous and mixed-variable problems.

5.2 Parallel ACO Implementations

Parallelization of algorithms becomes more and more an interesting and practical option for algorithm designers. ACO is particularly well suited for parallel implementations thanks to ants operating in an independent and asynchronous way. There have already been many attempts to propose parallel ACO algorithms. They are usually classified by their *parallel grain*, that is, the relationship between computation and communication. We can then distinguish between *coarse-grained* and *fine-grained* models. While the former are characterized by many ants using the same CPU and rare communication between the CPUs, in the latter only few ants use each CPU and there is a lot of communication going on. A review of the trends and strategies in designing parallel algorithms may be found in [10].

Randall and Lewis proposed a first reasonably complete classification of parallel ACO implementations [46]. Although many parallel ACO implementations have been proposed in the literature [40, 54, 24, 45, 7, 50], the results are fragmented and difficult to compare. Experiments are usually of limited scale and concern different optimization problems. Also, not all parallel implementations proposed are compared

with their sequential counterparts, which is an essential measure of their usefulness [50]. All this implies that more research is necessary in the area of parallelization of the ACO metaheuristic.

6 Conclusions

We have presented an introduction to ant colony optimization—a metaheuristic inspired by the foraging behavior of real ants. The central component of ACO is the pheromone model based on the underlying model of the problem being solved. The basic idea of ACO, which has been formalized into a metaheuristic framework, leaves many options and choices to the algorithm designer. Several variants of ACO have been already proposed, the most successful being $\mathcal{MM}$AS Ant System and Ant Colony System.

ACO is a relatively young metaheuristic, when compared to others such as evolutionary computation, tabu search, or simulated annealing. Yet, it has proven to be quite efficient and flexible. ACO algorithms are currently state-of-the-art for solving many combinatorial optimization problems including the sequential ordering problem (SOP) [23], the resource constraint project scheduling (RCPS) problem [41], and the open shop scheduling (OSS) problem [2]. For an in-depth overview of ACO, including applications, the interested reader should refer to [20].

Acknowledgments

Marco Dorigo acknowledges support from Belgian FNRS, of which he is a Research Director. This work was supported by the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

[1] L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Metaheuristics for the vehicle routing problem with stochastic demands. In X. Yao *et al.*, editor, *Proceedings of Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, volume 3242 of *LNCS*, pages 450–460. Springer-Verlag, Berlin, Germany, 2004.

[2] 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.

[3] 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.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[5] B. Bullnheimer, R. F. Hartl, and C. Strauss. Applying the ant system to the vehicle routing problem. In I. H. Osman, S. Voß, S. Martello, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 109–120. Kluwer Academic Publishers, Boston, MA, 1998.

[6] B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:312–328, 1999.

[7] B. Bullnheimer, G. Kotsis, and G. Strauß. Parallelization strategies for the ant system. Technical Report 8, Vienna University of Economics and Business Administration, Vienna, Austria, 1997.

[8] S. Camazine, J.-L. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ, 2003.

[9] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant System for job-shop scheduling. *JORBEL — Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.

[10] V.-D. Cung, S. L. Martins, C. C. Ribeiro, and C. Roucairol. Strategies for the parallel implementation of metaheuristics. In C. C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, volume 15 of *Operations Research/Computer Science Interfaces*, chapter 13. Kluwer Academic Publishers, Amsterdam, The Netherlands, 2001.

[11] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.

[12] G. Di Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.

[13] K. Doerner, W. Gutjahr, R. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1–4):79–99, 2004.

[14] M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[15] 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.

[16] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[17] 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.

[18] 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.

[19] 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.

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

[21] D. Fogel. *Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1995.

[22] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In T. Baeck, T. Fukuda, and Z. Michalewicz, editors, *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 622–627. IEEE Press, Piscataway, NJ, 1996.

[23] 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.

[24] L. M. Gambardella, E. 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, UK, 1999.

[25] S. Goss, S. Aron, J. Deneubourg, and J. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.

[26] P. P. Grassé. *Les insectes dans leur univers.* Ed. du Palais de la decouverte, Paris, 1946.

[27] P. P. Grasse. 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*, 6:41–81, 1959.

[28] M. Guntsch and M. Middendorf. Applying population based ACO to dynamic optimization problems. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 – Third International Workshop on Ant Algorithms*, volume 2463 of *LNCS*, pages 111–122. Springer-Verlag, Berlin, Germany, 2002.

[29] M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279 of *LNCS*, pages 71–80. Springer-Verlag, Berlin, Germany, 2002.

[30] M. Guntsch and M. Middendorf. Solving multi-criteria optimization problems with population-based ACO. In C. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Proceedings of Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003*, volume 2632 of *LNCS*, pages 464–478. Springer-Verlag, Berlin, Germany, 2003.

[31] M. Guntsch, M. Middendorf, and H. Schmeck. An ant colony optimization approach to dynamic TSP. In L. Spector *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 860–867. Morgan Kaufmann Publishers, San Francisco, CA, 2001.

[32] W. J. Gutjahr. S-ACO: An ant-based approach to combinatorial optimization under uncertainty. In M. Dorigo, L. Gambardella, F. Mondada, T. Sttzle, M. Birratari, and C. Blum, editors, *ANTS'2004, Fourth Internatinal Workshop on Ant Algorithms and Swarm Intelligence*, volume 3172 of *LNCS*, pages 238–249. Springer-Verlag, Berlin, Germany, 2004.

[33] J. Holland. *Adaptation in Natural and Artiffcial Systems.* University of Michigan Press, MI, 1975.

[34] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler *et al.*, editor, *Proceedings of the Evolutionary Multi-Criterion Optimization, First International Conference (EMO'01)*, volume 1993 of *LNCS*, pages 359–372. Springer-Verlag, Berlin, Germany, 2001.

[35] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys. *The Travelling Salesman Problem.* John Wiley & Sons, New York, NY, 1985.

[36] L. Lessing, I. Dumitrescu, and T. Sttzle. A comparison between ACO algorithms for the set covering problem. In M. Dorigo, L. Gambardella, F. Mondada, T. Sttzle, M. Birratari, and C. Blum, editors, *ANTS'2004, Fourth Internatinal Workshop on Ant Algorithms and Swarm Intelligence*, volume 3172 of *LNCS*, pages 1–12. Springer-Verlag, Berlin, Germany, 2004.

[37] H. R. Lourenço and D. Serra. Adaptive approach heuristics for the generalized assignment problem. Technical Report Economic Working Papers Series No.304, Universitat Pompeu Fabra, Dept. of Economics and Management, Barcelona, Spain, 1998.

[38] V. Maniezzo and A. Colorni. The Ant System applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):769–778, 1999.

[39] V. Maniezzo, A. Colorni, and M. Dorigo. The Ant System applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, IRIDIA, Université Libre de Bruxelles, Belgium, 1994.

[40] D. Merkle and M. Middendorf. Fast ant colony optimization on runtime reconfigurable processor arrays. *Genetic Programming and Evolvable Machines*, 3(4):345–361, 2002.

[41] 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.

[42] R. Michel and M. Middendorf. An island model based ant system with lookahead for the shortest supersequence problem. In A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, pages 692–701. Springer-Verlag, Berlin, Germany, 1998.

[43] R. Michel and M. Middendorf. An ACO algorithm for the shortest common supersequence problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Methods in Optimisation*. McGraw Hill, Boston, MA, 1999.

[44] J. M. Pasteels, J.-L. Deneubourg, and S. Goss. Self-organization mechanisms in ant societies (i): Trail recruitment to newly discovered food sources. *Experientia Supplementum*, 54:155–175, 1987.

[45] M. Rahoual, R. Hadji, and V. Bachelet. Parallel ant system for the set covering problem. In M. Dorigo, G. D. Caro, and M. Sampels, editors, *Proceedings of Ant Algorithms - Third International Workshop, ANTS 2002*, volume 2463 of *LNCS*, pages 262–267. Springer-Verlag, Berlin, Germany, 2002.

[46] M. Randall and A. Lewis. A parallel implementation of ant colony optimization. *Journal of Parallel and Distributed Computing*, 62(9):1421–1432, 2002.

[47] K. Socha. ACO for continuous and mixed-variable 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 Workshop, ANTS 2004*, volume 3172 of *LNCS*, pages 25–36. Springer-Verlag, Berlin, Germany, 2004.

[48] K. Socha, J. Knowles, and M. Sampels. A $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for the university timetabling problem. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 – Third International Workshop on Ant Algorithms*, volume 2463 of *LNCS*, pages 1–13. Springer-Verlag, Berlin, Germany, 2002.

[49] 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. Raidl *et al.*, editor, *Proceedings of EvoCOP 2003 – 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization*, volume 2611 of *LNCS*, pages 334–345. Springer-Verlag, Berlin, Germany, 2003.

[50] T. Stützle. Parallelization strategies for ant colony optimization. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of Parallel Problem Solving from Nature - PPSN V: 5th International Conference*, pages 722–731. Springer-Verlag, Berlin, Germany, 1998.

[51] T. Stützle and M. Dorigo. ACO algorithms for the traveling salesman problem. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, pages 163–183. John Wiley & Sons, Chichester, UK, 1999.

[52] T. Stützle and H. Hoos. The $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System and Local Search for Combinatorial Optimization Problems: Towards Adaptive Tools for Combinatorial Global Optimisation. In S. Voss, S.Martello, I.H.Ossmann, and C.Roucairol, editors, *Meta-Heuristic, Advances and Trends in Local Search Paradigma for Optimization*, pages 313–329. Kluwer Academic Publishers, 1998.

[53] T. Stützle and H. H. Hoos. $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[54] E.-G. Talbi, O. Roux, C. Fonlupt, and D. Robillard. Parallel ant colonies for combinatorial optimization problems. In *Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pages 239–247. Springer-Verlag, Berlin, Germany, 1999.

Index

$\mathcal{MAX}$-$\mathcal{MIN}$ Ant System, 13

ant colony optimization, 1

Ant Colony System, 14

ant foraging behavior, 4

Ant System, 12

construction graph, 7

fitness function, 9

heuristic information, 8

implicit evaluation, 4

pheromone, 2

- evaporation, 8
- trail, 3, 5
- update, 8

pseudorandom proportional rule, 15

solution

- best-so-far, 9
- iteration-best, 9

stigmergy, 2