Ant Algorithms for Discrete Optimization

Marco Dorigo
Gianni Di Caro
IRIDIA CP 194/6
Université Libre de Bruxelles
Avenue Franklin Roosevelt 50
B-1050 Brussels
Belgium
mdorigo@ulb.ac.be
gdicaro@iridia.ulb.ac.be

Luca M. Gambardella
IDSIA
Corso Elvezia 36
CH-6900 Lugano
Switzerland
luca@idsia.ch

Abstract This article presents an overview of recent work on ant algorithms, that is, algorithms for discrete optimization that took inspiration from the observation of ant colonies' foraging behavior, and introduces the ant colony optimization (ACO) metaheuristic. In the first part of the article the basic biological findings on real ants are reviewed and their artificial counterparts as well as the ACO metaheuristic are defined. In the second part of the article a number of applications of ACO algorithms to combinatorial optimization and routing in communications networks are described. We conclude with a discussion of related work and of some of the most important aspects of the ACO metaheuristic.

Keywords
ant algorithms, ant colony optimization, swarm intelligence, metaheuristics, natural computation

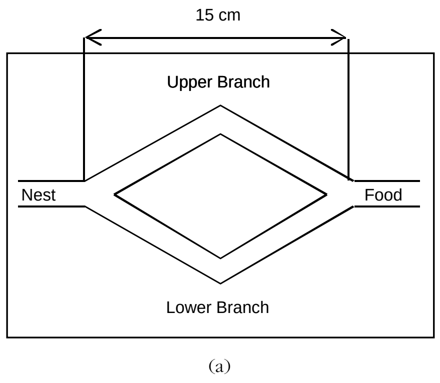
1 Introduction

Ant algorithms were first proposed by Dorigo and colleagues [33, 40] as a multi-agent approach to difficult combinatorial optimization problems such as the traveling salesman problem (TSP) and the quadratic assignment problem (QAP). There is currently much ongoing activity in the scientific community to extend and apply ant-based algorithms to many different discrete optimization problems [5, 21]. Recent applications cover problems such as vehicle routing, sequential ordering, graph coloring, routing in communications networks, and so on.

Ant algorithms were inspired by the observation of real ant colonies. Ants are social insects, that is, insects that live in colonies and whose behavior is directed more to the survival of the colony as a whole than to that of a single individual component of the colony. Social insects have captured the attention of many scientists because of the high structuration level their colonies can achieve, especially when compared to the relative simplicity of the colony's individuals. An important and interesting behavior of ant colonies is their foraging behavior, and, in particular, how ants can find the shortest paths between food sources and their nest.

While walking from food sources to the nest and vice versa, ants deposit on the ground a substance called *pheromone*, forming in this way a pheromone trail. Ants can smell pheromone, and when choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. The pheromone trail allows the ants to find their way back to the food source (or to the nest). Also, it can be used by other ants to find the location of the food sources found by their nestmates.

It has been shown experimentally that this pheromone trail following behavior can give rise, once employed by a colony of ants, to the emergence of shortest paths. That is, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants to discover the shortest path from the nest to the food source and back.



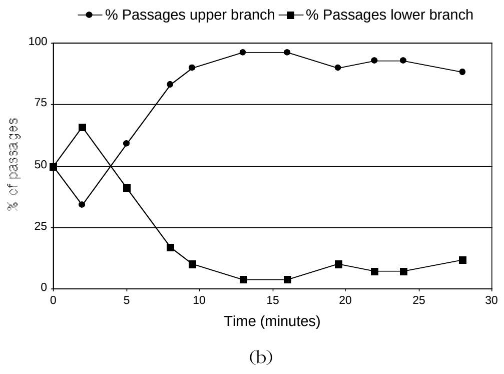


Figure 1. Single bridge experiment. (a) Experimental setup. (b) Results for a typical single trial, showing the percentage of passages on each of the two branches per unit of time as a function of time. Eventually, after an initial short transitory phase, the upper branch becomes the most used. After Deneubourg et al., 1990 [25].

To study in controlled conditions the ants' foraging behavior, the binary bridge experiment has been set up by Deneubourg et al. [25] (see Figure 1a). The nest of a colony of ants of the species *Linepithema humile* and a food source have been separated by a double bridge in which each branch has the same length. Ants are then left free to move between the nest and the food source and the percentage of ants that choose one or the other of the two branches is observed over time. The result (see Figure 1b) is that after an initial transitory phase in which some oscillations can appear, ants tend to converge on a same path.

In the above experiment initially there is no pheromone on the two branches, which are therefore selected by the ants with the same probability. Nevertheless, random fluctuations, after an initial transitory phase, cause a few more ants to select one branch randomly, the upper one in the experiment shown in Figure 1a, over the other. Because ants deposit pheromone while walking, the greater number of ants on the upper branch determines a greater amount of pheromone on it, which in turn stimulates more ants to choose it, and so on. The probabilistic model that describes this phenomenon, which closely matches the experimental observations, is the following [60]. We first make the assumption that the amount of pheromone on a branch is proportional to the number of ants that used the branch in the past. This assumption implies that pheromone evaporation is not taken into account. Given that an experiment typically lasts approximately one hour, it is plausible to assume that the amount of pheromone evaporated in this time period is negligible. In the model, the probability of choosing a branch at a certain time depends on the total amount of pheromone on the branch, which in turn is proportional to the number of ants that used the branch until that time. More precisely, let U_m and L_m be the numbers of ants that have used the upper and lower branch after m ants have crossed the bridge, with $U_m + L_m = m$. The probability $P_U(m)$ with which the $(m+1)$-th ant chooses the upper branch is

$$P_U(m) = \frac{(U_m + k)^h}{(U_m + k)^h + (L_m + k)^h} \tag{1}$$

while the probability $P_L(m)$ that it chooses the lower branch is $P_L(m) = 1 - P_U(m)$. This functional form of the probability of choosing one branch over the other was obtained from experiments on trail following [83]; the parameters h and k allow us to fit the model to experimental data. The ant choice dynamics follows from the above

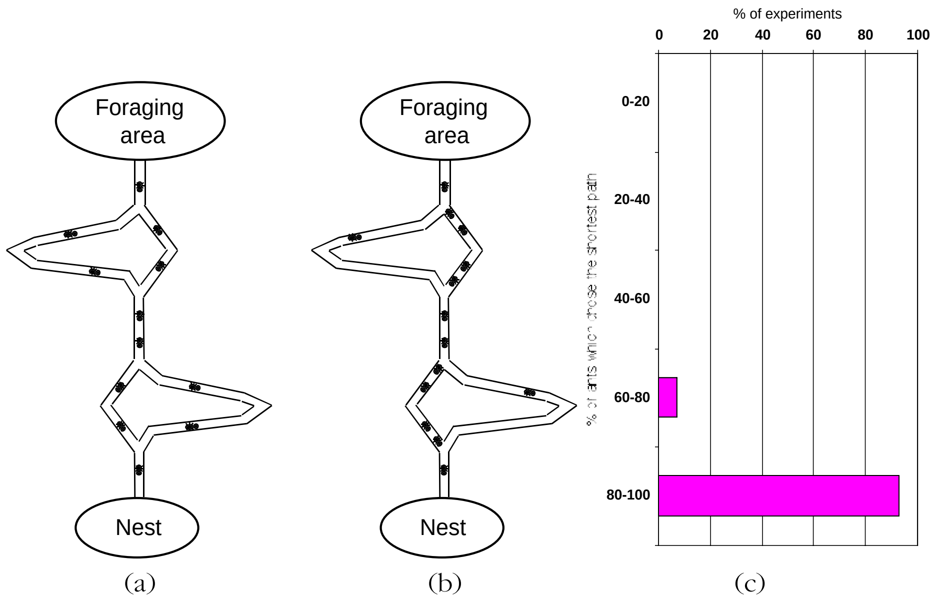


Figure 2. Double bridge experiment. (a) Ants start exploring the double bridge. (b) Eventually most of the ants choose the shortest path. (c) Distribution of the percentage of ants that selected the shorter path. After Goss et al. 1989 [60].

equation: $U_{m+1} = U_m + 1$, if $\psi \leq P_U$, $U_{m+1} = U_m$ otherwise, where ψ is a random variable uniformly distributed over the interval [0,1].

Monte Carlo simulations were run to test the correspondence between this model and the real data: Results of simulations were in agreement with the experiments with real ants when parameters were set to $k \approx 20$ and $h \approx 2$ [83].

It is easy to modify the experiment above to the case in which the bridge's branches are of different length [60], and to extend the model of Equation 1 so that it can describe this new situation. In this case, because of the same pheromone-laying mechanism as in the previous situation, the shortest branch is most often selected: The first ants to arrive at the food source are those that took the two shortest branches, so that, when these ants start their return trip, more pheromone is present on the short branch than on the long branch, stimulating successive ants to choose the short branch. In this case, the importance of initial random fluctuations is much reduced, and the stochastic pheromone trail following behavior of the ants coupled to differential branch length is the main mechanism at work. In Figure 2 are shown the experimental apparatus and the typical result of an experiment with a double bridge with branches of different lengths.

It is clear that what is going on in the above-described process is a kind of distributed optimization mechanism to which each single ant gives only a very small contribution. It is interesting that, although a single ant is in principle capable of building a solution (i.e., of finding a path between nest and food reservoir), it is only the ensemble of ants, that is, the ant colony, that presents the "shortest path-finding" behavior.[1] In a sense, this behavior is an emergent property of the ant colony. It is also interesting to note that ants can perform this specific behavior using a simple form of indirect communication mediated by pheromone laying, known as *stigmergy* [62].

1 The above-described experiments have been run in strongly constrained conditions. A formal proof of the pheromone-driven shortest path-finding behavior in the general case is missing. Bruckstein et al. [9, 10] consider the shortest path-finding problem in absence of obstacles for ants driven by visual clues and not by pheromones and prove the convergence of the ants' path to the straight line.

As defined by Grassé in his work on *Bellicositermes Natalensis* and *Cubitermes* [62], stigmergy is the "stimulation of the workers by the very performances they have achieved" (p. 79).[2]

In fact, Grassé [61] observed that insects are capable of responding to so-called "significant stimuli" that activate a genetically encoded reaction. In social insects, of which termites and ants are some of the best known examples, the effects of these reactions can act as new significant stimuli for both the insect that produced them and for other insects in the colony. The production of a new significant stimulus as a consequence of the reaction to a significant stimulus determines a form of coordination of the activities and can be interpreted as a form of indirect communication. For example, Grassé [62] observed that *Bellicositermes Natalensis* as well as *Cubitermes*, when building a new nest, start by a random, noncoordinated activity of earth pellet depositing. But once the earth pellets reach a certain density in a restricted area they become a new significant stimulus that causes more termites to add earth pellets so that pillar and arches, and eventually the whole nest, are built.

What characterizes stigmergy from other means of communication is (a) the physical nature of the information released by the communicating insects, which corresponds to a modification of physical environmental states visited by the insects, and (b) the local nature of the released information, which can only be accessed by insects that visit the state in which it was released (or some neighborhood of that state).

Accordingly, in this article we take the stance that it is possible to talk of stigmergetic communication whenever there is an *indirect communication mediated by physical modifications of environmental states which are only locally accessible by the communicating agents*.

One of the main tenets of this article is that the stigmergetic model of communication in general, and the one inspired by ants' foraging behavior in particular, is an interesting model for artificial multi-agent systems applied to the solution of difficult optimization problems. In fact, the above-mentioned characteristics of stigmergy can easily be extended to artificial agents by (a) associating with problem states appropriate state variables, and (b) giving the artificial agents only local access to these variables' values.

For example, in the above-described foraging behavior of ants, stigmergetic communication is at work via the pheromone that ants deposit on the ground while walking. Correspondingly, our artificial ants will simulate pheromone laying by modifying appropriate "pheromone variables" associated with problem states they visit while building solutions to the optimization problem to which they are applied. Also, according to the stigmergetic communication model, our artificial ants will have only local access to these pheromone variables.

Another important aspect of real ants' foraging behavior that is exploited by artificial ants is the coupling between the *autocatalytic* (*positive feedback*) mechanism [40] and the *implicit evaluation* of solutions. By implicit solution evaluation we mean the fact that shorter paths (which correspond to lower cost solutions in artificial ants) will be completed earlier than longer ones, and therefore they will receive pheromone reinforcement more quickly. Implicit solution evaluation coupled with autocatalysis can be very effective: The shorter the path, the sooner the pheromone is deposited by the ants, the more the ants that use the shorter path. If appropriately used, autocatalysis can be a powerful mechanism in population-based optimization algorithms (e.g., in evolutionary computation algorithms [45, 66, 85, 91] autocatalysis is implemented by the selection/reproduction mechanism). In fact, it quickly favors the best individuals,

2 Workers are one of the castes in termite colonies. Although Grassé introduced the term stigmergy to explain the behavior of termite societies, the same term has been used to describe indirect communication mediated by modifications of the environment that can also be observed in other social insects.

so that they can direct the search process. When using autocatalysis some care must be taken to avoid premature convergence (*stagnation*), that is, the situation in which some not very good individual takes over the population just because of a contingent situation (e.g., because of a local optimum, or just because of initial random fluctuations that caused a not very good individual to be much better than all the other individuals in the population) impeding further exploration of the search space. We will see that pheromone trail evaporation and stochastic state transitions are the needed complements to autocatalysis drawbacks.

In the remainder of this article we discuss a number of ant algorithms based on the above ideas. We start by defining, in Section 2, the characterizing aspects of ant algorithms and the ant colony optimization (ACO) metaheuristic.[3] Section 3 is an overview of most of the applications of ACO algorithms. In Section 4 we briefly discuss related work, and in Section 5 we discuss some of the characteristics of implemented ACO algorithms. Finally, we draw some conclusions in Section 6.

2 The Ant Colony Optimization Approach

In the ant colony optimization (ACO) metaheuristic a colony of artificial ants cooperates in finding good solutions to difficult discrete optimization problems. Cooperation is a key design component of ACO algorithms: The choice is to allocate the computational resources to a set of relatively simple agents (artificial ants) that communicate indirectly by stigmergy. Good solutions are an emergent property of the agents' cooperative interaction.

Artificial ants have a double nature. On the one hand, they are an abstraction of those behavioral traits of real ants that seemed to be at the heart of the shortest path-finding behavior observed in real ant colonies. On the other hand, they have been enriched with some capabilities that do not find a natural counterpart. In fact, we want ant colony optimization to be an engineering approach to the design and implementation of software systems for the solution of difficult optimization problems. It is therefore reasonable to give artificial ants some capabilities that, although not corresponding to any capacity of their real ant counterparts, make them more effective and efficient. In the following we discuss first the nature-inspired characteristics of artificial ants, and then how they differ from real ants.

2.1 Similarities and Differences with Real Ants

Most of the ideas of ACO stem from real ants. In particular, the use of: (a) a colony of cooperating individuals, (b) an (artificial) pheromone trail for local stigmergetic communication, (c) a sequence of local moves to find shortest paths, and (d) a stochastic decision policy using local information and no lookahead.

Colony of cooperating individuals. As real ant colonies, ant algorithms are composed of a population, or colony, of concurrent and asynchronous entities globally cooperating to find a good "solution" to the task under consideration. Although the complexity of each artificial ant is such that it can build a feasible solution (as a real ant can somehow find a path between the nest and the food), high quality solutions are the result of the cooperation among the individuals of the whole colony. Ants cooperate by means of the information they concurrently read/write on the problem's states they visit, as explained in the next item.

3 It is important here to clarify briefly the terminology used. We talk of ACO metaheuristic to refer to the general procedure presented in Section 2. The term ACO algorithm will be used to indicate any generic instantiation of the ACO metaheuristic. Alternatively, we will also talk more informally of ant algorithms to indicate any algorithm that, while following the general guidelines set above, does not necessarily follow all the aspects of the ACO metaheuristic. Therefore, all ACO algorithms are also ant algorithms, though the converse is not true (e.g., we will see that HAS-QAP is an ant, but not strictly an ACO algorithm).

Pheromone trail and stigmergy. Artificial ants modify some aspects of their environment as the real ants do. While real ants deposit on the world's state they visit a chemical substance, the pheromone, artificial ants change some numeric information locally stored in the problem's state they visit. This information takes into account the ant's current history or performance and can be read/written by any ant accessing the state. By analogy, we call this numeric information *artificial pheromone trail*, pheromone trail for short in the following. In ACO algorithms local pheromone trails are the only communication channels among the ants. This stigmergetic form of communication plays a major role in the utilization of collective knowledge. Its main effect is to change the way the environment (the problem landscape) is locally perceived by the ants as a function of all the past history of the whole ant colony. Usually, in ACO algorithms an evaporation mechanism, similar to real pheromone evaporation, modifies pheromone information over time. Pheromone evaporation allows the ant colony slowly to forget its past history so that it can direct its search toward new directions without being over-constrained by past decisions.

Shortest path searching and local moves. Artificial and real ants share a common task: to find a shortest (minimum cost) path joining an origin (nest) to destination (food) sites. Real ants do not jump; they just walk through adjacent terrain's states, and so do artificial ants, moving step-by-step through "adjacent states" of the problem. Of course, exact definitions of state and adjacency are problem specific.

Stochastic and myopic state transition policy. Artificial ants, as real ones, build solutions applying a probabilistic decision policy to move through adjacent states. As for real ants, the artificial ants' policy makes use of local information only and it does not make use of lookahead to predict future states. Therefore, the applied policy is completely local, in space and time. The policy is a function of both the a priori information represented by the problem specifications (equivalent to the terrain's structure for real ants), and of the local modifications in the environment (pheromone trails) induced by past ants.

As we said, artificial ants also have some characteristics that do not find their counterpart in real ants.

- Artificial ants live in a *discrete world* and their moves consist of transitions from discrete states to discrete states.
- Artificial ants have an *internal state*. This private state contains the memory of the ants' past actions.
- Artificial ants deposit an amount of pheromone that is a function of the *quality of the solution* found.[4]
- Artificial ants' timing in pheromone laying is problem dependent and often does not reflect real ants' behavior. For example, in many cases artificial ants update pheromone trails only after having generated a solution.
- To improve overall system efficiency, ACO algorithms can be enriched with *extra capabilities* such as lookahead, local optimization, backtracking, and so on that cannot be found in real ants. In many implementations ants have been hybridized with local optimization procedures (see, e.g., [38, 51, 98]), while, so far, only Michel and Middendorf [78] have used a simple one-step lookahead function and there are no examples of backtracking procedures added to the basic ant capabilities, except for simple recovery procedures used by Di Caro and Dorigo [26, 29].[5]

4 In reality, some real ants have a similar behavior: They deposit more pheromone in case of richer food sources.

5 Usually, backtracking strategies are suitable to solve constraint satisfaction problems, (e.g., *n*-queens) and lookahead is very useful

In the following section we will show how artificial ants can be put to work in an algorithmic framework so that they can be applied to discrete optimization problems.

2.2 The ACO Metaheuristic

In ACO algorithms a finite-size colony of artificial ants with the above-described characteristics collectively searches for good-quality solutions to the optimization problem under consideration. Each ant builds a solution, or a component of it,[6] starting from an initial state selected according to some problem-dependent criteria. While building its own solution, each ant collects information on the problem characteristics and on its own performance and uses this information to modify the representation of the problem, as seen by the other ants. Ants can act concurrently and independently, showing a cooperative behavior. They do not use direct communication: It is the stigmergy paradigm that governs the information exchange among the ants.

An incremental constructive approach is used by the ants to search for a feasible solution. A solution is expressed as a minimum cost (shortest) path through the states of the problem in accordance with the problem's constraints. The complexity of each ant is such that even a single ant is able to find a (probably poor quality) solution. High-quality solutions are only found as the emergent result of the global cooperation among all the agents of the colony concurrently building different solutions.

According to the assigned notion of neighborhood (problem-dependent), each ant builds a solution by moving through a (finite) sequence of neighbor states. Moves are selected by applying a stochastic local search policy directed (a) by ant private information (the ant internal state, or memory) and (b) by publicly available pheromone trails and a priori problem-specific local information.

The ant's internal state stores information about the ant past history. It can be used to carry useful information to compute the value/goodness of the generated solution and/or the contribution of each executed move. Moreover it can play a fundamental role to manage the feasibility of the solutions. In some problems, in fact, typically in combinatorial optimization, some of the moves available to an ant in a state can take the ant to an infeasible state. This can be avoided by exploiting the ant's memory. Ants therefore can build feasible solutions using only knowledge about the local state and about the effects of actions that can be performed in the local state.

The local, public information comprises both some problem-specific heuristic information and the knowledge, coded in the pheromone trails, accumulated by all the ants from the beginning of the search process. This time-global pheromone knowledge built up by the ants is a shared local long-term memory that influences the ants' decisions. The decisions about when the ants should release pheromone on the "environment" and how much pheromone should be deposited depend on the characteristics of the problem and on the design of the implementation. Ants can release pheromone while building the solution (online step-by-step), or after a solution has been built, moving back to all the visited states (online delayed), or both. As we said, autocatalysis plays

when the cost of making a local prediction about the effect of future moves is much lower than the cost of the real execution of the move sequence (e.g., mobile robotics). To our knowledge, until now ACO algorithms have not been applied to these classes of problems.

6 To make more intuitive what we mean by component of a solution, we can consider, as an example, a transportation routing problem: Given a set of n cities, $\{c_i\}$, $i \in \{1, \ldots, n\}$, and a network of interconnection roads, we want to find all the shortest paths s_{ij} connecting each city pair $c_i c_j$. In this case, a complete solution is represented by the set of all the $n(n-1)$ shortest path pairs, while a component of a solution is a single path s_{ij}.

an important role in ACO algorithms functioning: The more ants choose a move, the more the move is rewarded (by adding pheromone) and the more interesting it becomes for the next ants. In general, the amount of pheromone deposited is made proportional to the goodness of the solution an ant has built (or is building). In this way, if a move contributed to generating a high-quality solution its goodness will be increased proportionally to its contribution.

A functional composition of the locally available pheromone and heuristic values defines *ant-decision tables*, that is, probabilistic tables used by the ants' decision policy to direct their search toward the most interesting regions of the search space. The stochastic component of the move choice decision policy and the previously discussed pheromone evaporation mechanism prevent a rapid drift of all the ants toward the same part of the search space. Of course, the level of stochasticity in the policy and the strength of the updates in the pheromone trail determine the balance between the exploration of new points in the state space and the exploitation of accumulated knowledge. If necessary and feasible, the ants' decision policy can be enriched with problem-specific components such as backtracking procedures or lookahead. Once an ant has accomplished its task, consisting of building a solution and depositing pheromone information, the ant "dies," that is, it is deleted from the system.

The overall ACO metaheuristic, besides the two above-described components acting from a local perspective (i.e., ants' generation and activity, and pheromone evaporation), can also comprise some extra components that use global information and that go under the name of `daemon_actions` in the algorithm reported in Figure 3. For example, a *daemon* can be allowed to observe the ants' behavior and to collect useful global information to deposit additional pheromone information, biasing, in this way, the ant search process from a nonlocal perspective. Or, it could, on the basis of the observation of all the solutions generated by the ants, apply problem-specific local optimization procedures and deposit additional pheromone "offline" with respect to the pheromone the ants deposited online.

The three main activities of an ACO algorithm (ant generation and activity, pheromone evaporation, and daemon actions) may need some kind of synchronization, performed by the `schedule_activities` construct of Figure 3. In general, a strictly sequential scheduling of the activities is particularly suitable for nondistributed problems, where the global knowledge is easily accessible at any instant and the operations can be conveniently synchronized. On the contrary, some form of parallelism can be easily and efficiently exploited in distributed problems such as routing in telecommunications networks, as will be discussed in Section 3.2.

In Figure 3, a high-level description of the ACO metaheuristic is reported in pseudocode. As pointed out above, some described components and behaviors are optional, such as daemon activities, or strictly implementation dependent, such as when and how the pheromone is deposited. In general, the online step-by-step pheromone update and the online delayed pheromone update components (respectively, lines 24–27 and 30–34 in the `new_active_ant()` procedure) are mutually exclusive and only in a few cases are they both present or both absent (when both components are absent, the pheromone is deposited by the daemon).

ACO algorithms, as a consequence of their concurrent and adaptive nature, are particularly suitable for distributed stochastic problems where the presence of exogenous sources determines a nonstationarity in the problem representation (in terms of costs and/or environment). For example, many problems related to communications or transportation networks are intrinsically distributed and nonstationary and it is often not possible to have an exact model of the underlying variability. On the contrary, because stigmergy is both the only inter-ant communication method and it is spatially

```
1  procedure ACO_Meta_heuristic()
2    while (termination_criterion_not_satisfied)
3      schedule_activities
4        ants_generation_and_activity();
5        pheromone_evaporation();
6        daemon_actions();   {optional}
7      end schedule_activities
8    end while
9  end procedure

10 procedure ants_generation_and_activity()
11   while (available_resources)
12     schedule_the_creation_of_a_new_ant();
13     new_active_ant();
14   end while
15 end procedure

16 procedure new_active_ant() {ant lifecycle}
17   initialize_ant();
18   $\mathcal{M}$ = update_ant_memory();
19   while (current_state $\neq$ target_state)
20     $\mathcal{A}$ = read_local_ant-routing_table();
21     $\mathcal{P}$ = compute_transition_probabilities($\mathcal{A}, \mathcal{M}$, problem_constraints);
22     next_state = apply_ant_decision_policy($\mathcal{P}$, problem_constraints);
23     move_to_next_state(next_state);
24     if (online_step-by-step_pheromone_update)
25       deposit_pheromone_on_the_visited_arc();
26       update_ant-routing_table();
27     end if
28     $\mathcal{M}$ = update_internal_state();
29   end while
30   if (online_delayed_pheromone_update)
31     evaluate_solution();
32     deposit_pheromone_on_all_visited_arcs();
33     update_ant-routing_table();
34   end if
35   die();
36 end procedure
```

Figure 3. The ACO metaheuristic in pseudo-code. Comments are enclosed in braces. All the procedures at the first level of indentation in the statement `in_parallel` are executed concurrently. The procedure `daemon_actions()` at line 6 is optional and refers to centralized actions executed by a daemon possessing global knowledge. The `target_state` (line 19) refers to a complete solution, or to a component of a complete solution, built by the ant. The step-by-step and delayed pheromone updating procedures at lines 24–27 and 30–34 are often mutually exclusive. When both of them are absent the pheromone is deposited by the daemon.

localized, ACO algorithms could perform not at their best in problems where each state has a large-sized neighborhood. In fact, an ant that visits a state with a large-sized neighborhood has a huge number of possible moves among which to choose. Therefore, the probability that many ants visit the same state is very small, and consequently there is little, if any, difference between using or not using pheromone trails. A more formal definition of the ACO metaheuristic, as well as of the class of problems to which it can be applied, can be found in [35].

3 Applications of ACO Algorithms

There are now available numerous successful implementations of the ACO metaheuristic (Figure 3) applied to a number of different combinatorial optimization problems. Looking at these implementations it is possible to distinguish among two classes of applications: those to static combinatorial optimization problems, and those to dynamic ones.

Static problems are those in which the characteristics of the problem are given once and for all when the problem is defined and do not change while the problem is being solved. A paradigmatic example of such problems is the classic traveling salesman problem [67, 71, 86], in which city locations and their relative distances are part of the problem definition and do not change at run time. On the contrary, dynamic problems are defined as a function of some quantities whose value is set by the dynamics of an underlying system. The problem changes therefore at run time and the optimization algorithm must be capable of adapting online to the changing environment. The paradigmatic example discussed in the remainder of this section is network routing.

Topological modifications (e.g., adding or removing a node), which are not considered by the above classification, can be seen as transitions between problems belonging to the same class.

Tables 1 and 2 list the available implementations of ACO algorithms. The main characteristics of the listed algorithms are discussed in the following two subsections. We then conclude with a brief review of existing parallel implementations of ACO algorithms.

3.1 Applications of ACO Algorithms to Static Combinatorial Optimization Problems

The application of the ACO metaheuristic to a static combinatorial optimization problem is relatively straightforward, once a mapping of the problem that allows the incremental construction of a solution, a neighborhood structure, and a stochastic state transition rule to be locally used to direct the constructive procedure is defined.

A strictly implementation-dependent aspect of the ACO metaheuristic regards the timing of pheromone updates (lines 24–27 and 30–34 of the algorithm in Figure 3). In ACO algorithms for static combinatorial optimization the way ants update pheromone trails changes across algorithms: Any combination of online step-by-step pheromone updates and online delayed pheromone updates is possible.

Another important implementation-dependent aspect concerns the `daemon_actions()` component of the ACO metaheuristic (line 6 of the algorithm in Figure 3). Daemon actions implement actions that require some kind of global knowledge about the problem. Examples are offline pheromone updates and local optimization of solutions built by ants.

Most of the ACO algorithms presented in this subsection are strongly inspired by Ant System (AS), the first work on ant colony optimization [33, 40]. Many of the successive applications of the original idea are relatively straightforward applications of AS to the specific problem under consideration. We therefore start the description of ACO algorithms with AS. Following AS, for each ACO algorithm listed in Table 1 we give a short description of the algorithm's main characteristics and of the results obtained.

3.1.1 Traveling Salesman Problem

The first application of an ant colony optimization algorithm was done using the traveling salesman problem as a test problem. The main reasons why the TSP, one of the most studied NP-hard [71, 86] problems in combinatorial optimization, was chosen are

Table 1. List of applications of ACO algorithms to static combinatorial optimization problems. Classification by application and chronologically ordered.

Problem name	Authors	Year	Main references	Algorithm name
Traveling salesman	Dorigo, Maniezzo, & Colorni	1991	[33, 40, 41]	AS
	Gambardella & Dorigo	1995	[49]	Ant-Q
	Dorigo & Gambardella	1996	[37, 38, 50]	ACS & ACS-3-opt
	Stützle & Hoos	1997	[98, 97]	$\mathcal{MM}$AS
	Bullnheimer, Hartl, & Strauss	1997	[12]	AS$_{rank}$
Quadratic assignment	Maniezzo, Colorni, & Dorigo	1994	[77]	AS-QAP
	Gambardella, Taillard, & Dorigo	1997	[53, 54]	HAS-QAP[a]
	Stützle & Hoos	1998	[99]	$\mathcal{MM}$AS-QAP
	Maniezzo & Colorni	1998	[76]	AS-QAP[b]
	Maniezzo	1998	[75]	ANTS-QAP
Job-shop scheduling	Colorni, Dorigo, & Maniezzo	1994	[20]	AS-JSP
Vehicle routing	Bullnheimer, Hartl, & Strauss	1996	[15, 11, 13]	AS-VRP
	Gambardella, Taillard, & Agazzi	1999	[52]	HAS-VRP
Sequential ordering	Gambardella & Dorigo	1997	[51]	HAS-SOP
Graph coloring	Costa & Hertz	1997	[22]	ANTCOL
Shortest common supersequence	Michel & Middendorf	1998	[78, 79]	AS-SCS

[a] HAS-QAP is an ant algorithm that does not follow all the aspects of the ACO metaheuristic.

[b] This is a variant of the original AS-QAP.

Table 2. List of applications of ACO algorithms to dynamic combinatorial optimization problems. Classification by application and chronologically ordered.

Problem name	Authors	Year	Main references	Algorithm name
Connection-oriented network routing	Schoonderwoerd, Holland, Bruten, & Rothkrantz	1996	[90, 89]	ABC
	White, Pagurek, & Oppacher	1998	[105]	ASGA
	Di Caro & Dorigo	1998	[30]	AntNet-FS
	Bonabeau, Henaux, Guérin, Snyers, Kuntz, & Theraulaz	1998	[6]	ABC-smart ants
Connectionless network routing	Di Caro & Dorigo	1997	[26, 29, 32]	AntNet & AntNet-FA
	Subramanian, Druschel, & Chen	1997	[100]	Regular ants
	Heusse, Guérin, Snyers, & Kuntz	1998	[64]	CAF
	van der Put & Rothkrantz	1998	[102, 103]	ABC-backward

that it is a shortest path problem to which the ant colony metaphor is easily adapted and that it is a didactic problem (i.e., it is very easy to understand and explanations of the algorithm behavior are not obscured by too many technicalities).

A general definition of the traveling salesman problem is the following. Consider a set N of nodes, representing cities, and a set E of arcs fully connecting the nodes N. Let d_{ij} be the length of the arc $(i, j) \in E$, that is, the distance between cities i and j, with $i, j \in N$. The TSP is the problem of finding a minimal length Hamiltonian circuit on the graph $G = (N, E)$, where a Hamiltonian circuit of graph G is a closed tour visiting once and only once all the $n = |N|$ nodes of G, and its length is given by the sum of the lengths of all the arcs of which it is composed.[7]

In the following we will briefly overview the ACO algorithms that have been proposed for the TSP, starting with Ant System. A more complete overview can be found in [96].

Ant System (AS). Ant System was the first (1991) [33, 40] ACO algorithm. Its importance resides mainly in being the prototype of a number of ant algorithms that have found many interesting and successful applications.

In AS, artificial ants build solutions (tours) of the TSP by moving on the problem graph from one city to another. The algorithm executes $t_{\max}$ iterations, in the following indexed by t. During each iteration m ants build a tour executing n steps in which a probabilistic decision (state transition) rule is applied. In practice, when in node i the ant chooses the node j to move to, and the arc (i, j) is added to the tour under construction. This step is repeated until the ant has completed its tour.

Three AS algorithms have been defined [19, 33, 40, 41] that differ by the way pheromone trails are updated. These algorithms are called *ant-density*, *ant-quantity*, and *ant-cycle*. In ant-density and ant-quantity ants deposit pheromone while building a solution,[8] while in ant-cycle ants deposit pheromone after they have built a complete tour.

Preliminary experiments run on a set of benchmark problems [33, 40, 41] have shown that ant-cycle's performance was much better than that of the other two algorithms. Consequently, research on AS was directed toward a better understanding of the characteristics of ant-cycle, which is now known as Ant System, while the other two algorithms were abandoned.

As we said, in AS after ants have built their tours, each ant deposits pheromone on pheromone trail variables associated to the visited arcs to make the visited arcs become more desirable for future ants (i.e., online delayed pheromone update is at work). Then the ants die. In AS no daemon activities are performed, while the pheromone evaporation procedure, which happens just before ants start to deposit pheromone, is interleaved with the ants' activity.

The amount of pheromone trail $\tau_{ij}(t)$ associated to arc (i, j) is intended to represent the learned desirability of choosing city j when in city i (which also corresponds to the desirability that the arc (i, j) belongs to the tour built by an ant). The pheromone trail information is changed during problem solution to reflect the experience acquired by ants during problem solving. Ants deposit an amount of pheromone proportional to the quality of the solutions they produced: The shorter the tour generated by an ant, the greater the amount of pheromone it deposits on the arcs that it used to generate the tour. This choice helps to direct search toward good solutions. The main role of pheromone evaporation is to avoid stagnation, that is, the situation in which all ants end up doing the same tour.

7 Note that distances need not be symmetric: In an asymmetric TSP (ATSP) $d_{ij} \neq d_{ji}$. Also, the graph need not be fully connected. If it is not, it suffices to add the missing arcs, giving them a very high length.

8 These two algorithms differ by the amount of pheromone ants deposit at each step: In *ant-density* ants deposit a constant amount of pheromone, while in *ant-quantity* they deposit an amount of pheromone inversely proportional to the length of the chosen arc.

The memory (or internal state) of each ant contains the already visited cities and is called *tabu list* (in the following we will continue to use the term tabu list[9] to indicate the ant's memory). The memory is used to define, for each ant k, the set of cities that an ant located on city i still has to visit. By exploiting the memory, therefore, an ant k can build feasible solutions by an implicit state-space graph generation (in the TSP this corresponds to visiting a city exactly once). Also, memory allows the ant to cover the same path to deposit online delayed pheromone on the visited arcs.

The ant-decision table $\mathcal{A}_i = [a_{ij}(t)]_{|\mathcal{N}_i|}$ of node i is obtained by the composition of the local pheromone trail values with the local heuristic values as follows:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum\limits_{l \in \mathcal{N}_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \qquad \forall j \in \mathcal{N}_i \tag{2}$$

where $\tau_{ij}(t)$ is the amount of pheromone trail on arc (i, j) at time t, $\eta_{ij} = 1/d_{ij}$ is the heuristic value of moving from node i to node j, $\mathcal{N}_i$ is the set of neighbors of node i, and α and β are two parameters that control the relative weight of pheromone trail and heuristic value.

The probability with which an ant k chooses to go from city i to city $j \in \mathcal{N}_i^k$ while building its tour at the t-th algorithm iteration is

$$p_{ij}^k(t) = \frac{a_{ij}(t)}{\sum\limits_{l \in \mathcal{N}_i^k} a_{il}(t)} \tag{3}$$

where $\mathcal{N}_i^k \subseteq \mathcal{N}_i$ is the set of nodes in the neighborhood of node i that ant k has not visited yet (nodes in $\mathcal{N}_i^k$ are selected from those in $\mathcal{N}_i$ by using the ant private memory $\mathcal{M}^k$).

The role of the parameters α and β is the following. If $\alpha = 0$, the closest cities are more likely to be selected: This corresponds to a classical stochastic greedy algorithm (with multiple starting points since ants are initially randomly distributed on the nodes). If, on the contrary, $\beta = 0$, only pheromone amplification is at work: This method will lead to the rapid emergence of a stagnation situation with the corresponding generation of tours that, in general, are strongly suboptimal [41]. A trade-off between heuristic value and trail intensity therefore appears to be necessary.

After all ants have completed their tour, pheromone evaporation on all arcs is triggered, and then each ant k deposits a quantity of pheromone $\Delta\tau_{ij}^k(t)$ on each arc that it has used:

$$\Delta\tau_{ij}^k(t) = \begin{cases} 1/L^k(t) & \text{if } (i, j) \in T^k(t) \\ 0 & \text{if } (i, j) \notin T^k(t) \end{cases} \tag{4}$$

where $T^k(t)$ is the tour done by ant k at iteration t, and $L^k(t)$ is its length. Note that in the symmetric TSP arcs are considered to be bidirectional so that arcs (i, j) and (j, i) are always updated contemporaneously (in fact, they are the same arc). Different is the case of the asymmetric TSP, where arcs have a directionality, and for which the pheromone trail level on the arcs (i, j) and (j, i) can be different. In this case, therefore, when an ant moves from node i to node j only arc (i, j), and not (j, i), is updated.

9 The term tabu list is used here to indicate a simple memory that contains the set of already visited cities and has no relation with tabu search [57, 58].

It is clear from Equation 4 that the value $\Delta\tau_{ij}^{k}(t)$ depends on how well the ant has performed: The shorter the tour done, the greater the amount of pheromone deposited.

In practice, the addition of new pheromone by ants and pheromone evaporation are implemented by the following rule applied to all the arcs:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5}$$

where $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$, m is the number of ants at each iteration (maintained constant), and $\rho \in (0, 1]$ is the pheromone trail decay coefficient. The initial amount of pheromone $\tau_{ij}(0)$ is set to a same small positive constant value τ_0 on all arcs, and the total number of ants is set to $m = n$, while α, β, and ρ are respectively set to 1, 5, and 0.5; these values were experimentally found to be good by Dorigo [33]. Dorigo et al. [40] introduced also *elitist ants*, that is, a daemon action by which the arcs used by the ant that generated the best tour from the beginning of the trial get extra pheromone.

Ant System was compared with other general purpose heuristics on some relatively small TSP problems (these were problems ranging from 30 to 75 cities). The results [40, 41] were very interesting and disappointing at the same time. AS was able to find and improve the best solution found by a genetic algorithm for Oliver30 [106], a 30-city problem, and it had a performance similar or better than that of some general-purpose heuristics with which it was compared. Unfortunately, for problems of growing dimensions AS never reached the best-known solutions within the allowed 3,000 iterations, although it exhibited quick convergence to good solutions. These encouraging, although not exceptional, results stimulated a number of researchers to study further the ACO approach to optimization. These efforts have resulted in numerous successful applications, listed in the following sections.

Others AS-like approaches. Stützle and Hoos [98, 97] have introduced Max-Min AS ($\mathcal{MM}$AS), which is the same as AS, but (a) pheromone trails are only updated offline by the daemon (the arcs that were used by the best ant in the current iteration receive additional pheromone), (b) pheromone trail values are restricted to an interval $[\tau_{\min}, \tau_{\max}]$, and (c) trails are initialized to their maximum value $\tau_{\max}$.

Putting explicit limits on the trail strength restricts the range of possible values for the probability of choosing a specific arc according to Equation 3. This helps avoid stagnation, which was one of the reasons why AS performed poorly when an elitist strategy, such as allowing only the best ant to update pheromone trails, was used. To avoid stagnation, which may occur when some pheromone trails are close to $\tau_{\max}$ while most others are close to $\tau_{\min}$, Stützle and Hoos have added what they call a "trail smoothing mechanism"; that is, pheromone trails are updated using a proportional mechanism: $\Delta\tau_{ij} \propto (\tau_{\max} - \tau_{ij}(t))$. In this way the relative difference between the trail strengths gets smaller, which obviously favors the exploration of new paths. They found that, when applied to the TSP, $\mathcal{MM}$AS finds significantly better tours than AS, although comparable to those obtained with ACS. (ACS is an extension of AS discussed in the next subsection.)

Bullnheimer, Hartl, and Strauss [12] proposed yet another modification of AS, called AS$_{\text{rank}}$. In AS$_{\text{rank}}$, as was the case in $\mathcal{MM}$AS, the only pheromone updates are performed by the daemon, which implements the following activities: (a) the m ants are ranked by tour length $(L_1(t), L_2(t), \ldots, L_m(t))$ and the arcs that were visited by one of the first $\sigma - 1$ ants in the ranking receive an amount of pheromone proportional to the visiting ant rank, and (b) the arcs used by the ant that generated the best tour from the beginning of the trial also receive additional pheromone (this is equivalent to AS's elitist ants' pheromone updating). These are both forms of offline pheromone update. In their implementation the contribution of the best tour so far was multiplied by σ. The dynamics of the amount of pheromone $\tau_{ij}(t)$ is given by

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \sigma\Delta\tau_{ij}^{+}(t) + \Delta\tau_{ij}^{r}(t) \tag{6}$$

where $\Delta\tau_{ij}^{+}(t) = 1/L^{+}(t)$, L^{+} being the length of the best solution from beginning of the trial, and $\Delta\tau_{ij}^{r}(t) = \sum_{\mu=1}^{\sigma-1}\Delta\tau_{ij}^{\mu}(t)$, with $\Delta\tau_{ij}^{\mu}(t) = (\sigma-\mu)1/L^{\mu}(t)$ if the ant with rank μ has used arc (i, j) in its tour and $\Delta\tau_{ij}^{\mu}(t) = 0$ otherwise. $L^{\mu}(t)$ is the length of the tour performed by the ant with rank μ at iteration t. Equation 6 is applied to all arcs and implements therefore both pheromone evaporation and offline pheromone updating. They found that this new procedure improves significantly the quality of the results obtained with AS.

Ant Colony System (ACS), ACS-3-opt, and Ant-Q. The Ant Colony System (ACS) algorithm has been introduced by Dorigo and Gambardella [37, 38, 50] to improve the performance of AS, which was able to find good solutions within a reasonable time only for small problems. ACS is based on AS but presents some important differences.

First, the daemon updates pheromone trails offline: At the end of an iteration of the algorithm, once all the ants have built a solution, pheromone trail is added to the arcs used by the ant that found the best tour from the beginning of the trial. In ACS-3-opt the daemon first activates a local search procedure based on a variant of the 3-opt local search procedure [73] to improve the solutions generated by the ants and then performs offline pheromone trail update. The offline pheromone trail update rule is

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \tag{7}$$

where $\rho \in (0, 1]$ is a parameter governing pheromone decay, $\Delta\tau_{ij}(t) = 1/L^{+}$, and L^{+} is the length of T^{+}, the best tour since the beginning of the trial. Equation 7 is applied only to the arcs (i, j) belonging to T^{+}.

Second, ants use a different decision rule, called *pseudo-random-proportional* rule, in which an ant k on city i chooses the city $j \in \mathcal{N}_i^k$ to move to as follows. Let $\mathcal{A}_i = [a_{ij}(t)]_{|\mathcal{N}_i|}$ be the ant-decision table:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)][\eta_{ij}]^{\beta}}{\sum_{l\in\mathcal{N}_i}[\tau_{il}(t)][\eta_{il}]^{\beta}} \qquad \forall j \in \mathcal{N}_i \tag{8}$$

Let q be a random variable uniformly distributed over $[0, 1]$, and $q_0 \in [0, 1]$ be a tunable parameter. The pseudo-random-proportional rule, used by ant k located in node i to choose the next node $j \in \mathcal{N}_i^k$, is the following: If $q \le q_0$ then

$$p_{ij}^{k}(t) = \begin{cases} 1 & \text{if } j = \arg\max a_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

otherwise, when $q > q_0$

$$p_{ij}^{k}(t) = \frac{a_{ij}(t)}{\sum_{l\in\mathcal{N}_i^k} a_{il}(t)} \tag{10}$$

This decision rule has a double function: When $q \le q_0$ the decision rule exploits the knowledge available about the problem, that is, the heuristic knowledge about distances between cities and the learned knowledge memorized in the form of pheromone trails, while when $q > q_0$ it operates a biased exploration (equivalent to AS's Equation 3). Tuning q_0 allows us to modulate the degree of exploration and to choose whether to concentrate the activity of the system on the best solutions or to explore the search space.

Third, in ACS, ants perform only online step-by-step pheromone updates. These updates are performed to favor the emergence of other solutions than the best so far. The pheromone updates are performed by applying the rule

$$\tau_{ij}(t) \leftarrow (1-\varphi)\tau_{ij}(t) + \varphi\tau_0 \tag{11}$$

where $0 < \varphi \leq 1$.

Equation 11 says that an ant moving from city i to city $j \in \mathcal{N}_i^k$ updates the pheromone trail on arc (i, j). The value τ_0 is the same as the initial value of pheromone trails and it was experimentally found that setting $\tau_0 = (nL_{nn})^{-1}$, where n is the number of cities and L_{nn} is the length of a tour produced by the nearest neighbor heuristic [56], produces good results. When an ant moves from city i to city j, the application of the local update rule makes the corresponding pheromone trail τ_{ij} decrease. The rationale for decreasing the pheromone trail on the path an ant is using to build a solution is the following. Consider an ant k_2 starting in city 2 and moving to city 3, 4, and so on, and an ant k_1 starting in city 1 and choosing city 2 as the first city to move to. Then, there are good chances that ant k_1 will follow ant k_2 with one step delay. The trail-decreasing effect obtained by applying the local update rule reduces the risk of such a situation. In other words, ACS's local update rule has the effect of making the visited arcs less and less attractive as they are visited by ants, indirectly favoring the exploration of not-yet-visited arcs. As a consequence, ants tend not to converge to a common path. This fact, which was observed experimentally [38], is a desirable property given that if ants explore different paths, then there is a higher probability that one of them will find an improving solution than in the case when they all converge to the same tour (which would make the use of m ants pointless).

Last, ACS exploits a data structure called *candidate list*, which provides additional local heuristic information. A candidate list is a list of preferred cities to be visited from a given city. In ACS when an ant is in city i, instead of examining all the unvisited neighbors of i, it chooses the city to move to among those in the candidate list; only if no city in candidate list has unvisited status are other cities examined. The candidate list of a city contains cl cities ordered by increasing distance (cl is a parameter), and the list is scanned sequentially and according to the ant tabu list to avoid already visited cities.

ACS was tested (see [37, 38] for detailed results) on standard problems, both symmetric and asymmetric, of various sizes and compared with many other metaheuristics. In all cases, its performance, both in terms of quality of the solutions generated and of CPU time required to generate them, was the best one.

ACS-3-opt's performance was compared to that of the genetic algorithm (with local optimization) [47, 48] that won the First International Contest on Evolutionary Optimization [2]. The two algorithms showed similar performance with the genetic algorithm behaving slightly better on symmetric problems and ACS-3-opt on asymmetric ones.

To conclude, we mention that ACS was the direct successor of Ant-Q [36, 49], an algorithm that tried to merge AS and Q-learning [104] properties. In fact, Ant-Q differs from ACS only in the value τ_0 used by ants to perform online step-by-step pheromone updates. The idea was to update pheromone trails with a value that was a prediction of the value of the next state. In Ant-Q, an ant k implements online step-by-step pheromone updates by the following equation, which replaces Equation 11:

$$\tau_{ij}(t) \leftarrow (1-\varphi)\tau_{ij}(t) + \varphi\gamma \cdot \max_{l\in\mathcal{N}_j^k} \tau_{jl} \tag{12}$$

Unfortunately, it was later found that setting the complicate prediction term to a small constant value, as it is done in ACS, resulted in approximately the same performance. Therefore, although having a good performance, Ant-Q was abandoned for the equally good but simpler ACS.

Also, other versions of ACS were studied that differ from the one described above because of (a) the way online step-by-step pheromone updates were implemented (in [38] experiments were run disabling it, or by setting the update term in Equation 11 to the value $\tau_0 = 0$), (b) the way the decision rule was implemented (in [49] the pseudo-random-proportional rule of Equations 9 and 10 was compared to the *random-proportional* rule of Ant System, and to a *pseudo-random* rule that differs from the pseudo-random-proportional rule because random choices are done uniformly randomly), and (c) the type of solution used by the daemon to update the pheromone trails (in [38] the use of the best solution found in the current iteration was compared with ACS's use of the best solution found from the beginning of the trial). ACS as described above is the best performing of all the algorithms obtained by combinations of the above-mentioned choices.

3.1.2 Quadratic Assignment Problem

The quadratic assignment problem is the problem of assigning n facilities to n locations so that the cost of the assignment, which is a function of the way facilities have been assigned to locations, is minimized [69]. The QAP was, after the TSP, the first problem to be attacked by an AS-like algorithm. This was a reasonable choice, since the QAP is a generalization of the TSP.[10] Maniezzo, Colorni, and Dorigo [77] applied exactly the same algorithm as AS using the QAP-specific min-max heuristic to compute the η values used in Equation 3. The resulting algorithm, AS-QAP, was tested on a set of standard problems and turned out to be of the same quality as metaheuristic approaches such as simulated annealing and evolutionary computation. More recently, Maniezzo and Colorni [76] and Maniezzo [75] developed two variants of AS-QAP and added to them a local optimizer. The resulting algorithms were compared with some of the best heuristics available for the QAP with very good results: Their versions of AS-QAP gave the best results on all the tested problems.

Similar results were obtained by Stützle and Hoos with their $\mathcal{MM}$AS-QAP algorithm [99] ($\mathcal{MM}$AS-QAP is a straightforward application of $\mathcal{MM}$AS, see Section 3.1.1, to the QAP), and by Gambardella, Taillard, and Dorigo [53] with their HAS-QAP.[11] $\mathcal{MM}$AS-QAP and HAS-QAP were compared with some of the best heuristics available for the QAP on two classes of QAP problems: random and structured QAPs, where structured QAPs are instances of problems taken from real-world applications. These ant algorithms were found to be the best performing on structured problems [53, 54]. A detailed overview of applications of ACO algorithms to the QAP can be found in [95].

3.1.3 Job-Shop Scheduling Problem

Colorni, Dorigo, and Maniezzo (1994) [20] applied AS to the job-shop scheduling problem (JSP), which is formulated as follows. Given a set M of machines and a set J of jobs consisting of an ordered sequence of operations to be executed on these machines, the job-shop scheduling problem is that of assigning operations to machines so that the maximum of the completion times of all operations is minimized and no two jobs are processed at the same time on the same machine. JSP is NP-hard [55].

10 In fact, the TSP can be seen as the problem of assigning to each of n cities a different number chosen between 1 and n. QAP, as the TSP, is NP-hard [92].

11 HAS-QAP is an ant algorithm that, although initially inspired by AS, does not strictly belong to the ACO metaheuristic because of some peculiarities, such as ants that modify solutions as opposed to build them, and pheromone trail used to guide solution modifications and not as an aid to direct their construction.

The basic algorithm they applied was exactly the same as AS, where the η heuristic value was computed using the longest remaining processing time heuristic. Due to the different nature of the constraints with respect to the TSP they also defined a new way of building the ant's tabu list. AS-JSP was applied to problems of dimensions up to 15 machines and 15 jobs always finding solutions within 10% of the optimal value [20, 41]. These results, although not exceptional, are encouraging and suggest that further work could lead to a workable system. Also, a comparison with other approaches is necessary.

3.1.4 Vehicle Routing Problem

There are many types of vehicle routing problems (VRPs). Bullnheimer, Hartl, and Strauss [11, 13, 15] applied an AS-like algorithm to the following instance. Let $G = (N, A, d)$ be a complete weighted directed graph, where $N = (n_0, \dots, n_n)$ is the set of nodes, $A = \{(i, j)\colon\ i \neq j\}$ is the set of arcs, and each arc (i, j) has an associated weight $d_{ij} \geq 0$ that represents the distance between n_i and n_j. Node n_0 represents a depot, where M vehicles are located, each one of capacity D, while the other nodes represent customers' locations. A demand $d_i \geq 0$ and a service time $\delta_i \geq 0$ are associated to each customer n_i ($d_0 = 0$ and $\delta_0 = 0$). The objective is to find minimum cost vehicle routes such that (a) every customer is visited exactly once by exactly one vehicle, (b) for every vehicle the total demand does not exceed the vehicle capacity D, (c) the total tour length of each vehicle does not exceed a bound L, and (d) every vehicle starts and ends its tour in the depot.[12] AS-VRP, the algorithm defined by Bullnheimer, Hartl, and Strauss for the above problem, is a direct extension of AS based on their AS_{rank} algorithm discussed in Section 3.1.1. They used various standard heuristics for the VRP [18, 82] and added a simple local optimizer based on the 2-opt heuristic [24]. They also adapted the way the tabu list is built by taking into consideration the constraints on the maximum total tour length L of a vehicle and its maximum capacity D. Comparisons on a set of standard problems showed that AS-VRP performance is at least interesting: It outperforms simulated annealing and neural networks, while it has a slightly lower performance than tabu search.

Gambardella, Taillard, and Agazzi [52] have also attacked the VRP by means of an ACO algorithm. They first reformulate the problem by adding to the city set $M - 1$ depots, where M is the number of vehicles. Using this formulation, the VRP problem becomes a TSP with additional constraints. Therefore they can define an algorithm, called HAS-VRP, which is inspired by ACS: Each ant builds a complete tour without violating vehicle capacity constraints (each vehicle has associated a maximum transportable weight). A complete tour comprises many subtours connecting depots, and each subtour corresponds to the tour associated to one of the vehicles. Pheromone trail updates are done offline as in ACS. Also, a local optimization procedure based on edge exchanges is applied by the daemon. Results obtained with this approach are competitive with those of the best-known algorithms and new upper bounds have been computed for well-known problem instances. They also study the vehicle routing problem with time windows (VRPTW), which extends the VRP by introducing a time window $[b_i, e_i]$ within which a customer i must be served. Therefore, a vehicle visiting customer i before time b_i will have to wait. In the literature the VRPTW is solved considering two objectives functions: The first one is to minimize the number of vehicles and the second one is to minimize the total travel time. A solution with a lower number of vehicles is always preferred to a solution with a higher number of vehicles but lower travel time. To optimize both objectives simultaneously, a two-colony ant

12 Also in this case it can be easily seen that the VRP is closely related to the TSP: A VRP consists of the solution of many TSPs with common start and end cities. As such, VRP is an NP-hard problem.

algorithm has been designed. The first colony tries to minimize the number of vehicles, while the other one uses V vehicles, where V is the number of vehicles computed by the first colony, to minimize travel time. The two colonies work using different sets of pheromone trails, but the best ants are allowed to update the pheromone trails of the other colony. This approach has been proved to be competitive with the best-known methods in the literature.

3.1.5 Shortest Common Supersequence Problem

Given a set L of strings over an alphabet Σ, find a string of minimal length that is a supersequence of each string in L, where a string S is a supersequence of a string A if S can be obtained from A by inserting in A zero or more characters.[13] This is the problem known as the shortest common supersequence problem (SCS) that Michel and Middendorf [78, 79] attacked by means of AS-SCS. AS-SCS differs from AS in that it uses a *lookahead function* that takes into account the influence of the choice of the next symbol to append at the next iteration. The value returned by the lookahead function takes the place of the heuristic value η in the probabilistic decision rule (Equation 3). Also, in AS-SCS the value returned by a simple heuristic called LM [7] is factorized in the pheromone trail term. Michel and Middendorf [78,79] further improved their algorithm by the use of an island model of computation (i.e., different colonies of ants work on the same problem concurrently using private pheromone trail distributions; every fixed number of iterations they exchange the best solution found).

AS-SCS-LM (i.e., AS-SCS with LM heuristic, lookahead, and island model of computation) was compared in [78] with the MM [46] and LM heuristics, as well as with a recently proposed genetic algorithm specialized for the SCS problem. On the great majority of the test problems AS-SCS-LM turned out to be the best-performing algorithm.

3.1.6 Graph-Coloring Problem

Costa and Hertz [22] have proposed the AS-ATP algorithm for assignment type problems.[14] The AS-ATP algorithm they define is basically the same as AS except that ants need to make two choices: First they choose an item, then they choose a resource to assign to the previously chosen item. These two choices are made by means of two probabilistic rule functions of two distinct pheromone trails τ_1 and τ_2 and of two appropriate heuristic values η_1 and η_2. In fact, the use of two pheromone trails is the main novelty introduced by AS-ATP. They exemplify their approach by means of an application to the graph-coloring problem (GCP). Given a graph $G = (N, E)$, a q-coloring of G is a mapping c: $N \rightarrow \{1, \ldots, q\}$ such that $c(i) = c(j)$ if $(i, j) \in E$. The GCP is the problem of finding a coloring of the graph G so that the number q of colors used is minimum. The algorithm they propose, called ANTCOL, makes use of well-known graph-coloring heuristics such as *recursive large first* (RLF) [72] and DSATUR [8]. Costa and Hertz tested ANTCOL on a set of random graphs and compared it with some of the best available heuristics. Results have shown that ANTCOL performance is comparable to that obtained by the other heuristics: On 20 randomly generated graphs of 100 nodes with any two nodes connected with probability 0.5 the average number of colors used by ANTCOL was 15.05, whereas the best known result [23, 44] is 14.95. More research will be necessary to establish whether the proposed use of two pheromone trails can be a useful addition to ACO algorithms.

13 Consider for example the set $L = \{bcab, bccb, baab, acca\}$. The string *baccab* is a shortest supersequence. The shortest common supersequence (SCS) problem is NP-hard even for an alphabet of cardinality two [84].

14 Examples of assignment type problems are the QAP, the TSP, graph coloring, set covering, and so on.

3.1.7 Sequential Ordering Problem

The sequential ordering problem (SOP) [42] consists of finding a minimum weight Hamiltonian path on a directed graph with weights on the arcs and on the nodes, subject to precedence constraints among nodes. It is very similar to an asymmetric TSP in which the end city is not directly connected to the start city. The SOP, which is NP-hard, models real-world problems such as single-vehicle routing problems with pick-up and delivery constraints, production planning, and transportation problems in flexible manufacturing systems and is therefore an important problem from an applications point of view.

Gambardella and Dorigo [51] attacked the SOP by HAS-SOP, an extension of ACS. In fact, HAS-SOP is the same as ACS except for the set of feasible nodes, which is built taking into consideration the additional precedence constraints, and for the local optimizer, which was a specifically designed variant of the well-known 3-opt procedure. Results obtained with HAS-SOP are excellent. Tests were run on a great number of standard problems[15] and comparisons were done with the best available heuristic methods. In all cases HAS-SOP was the best-performing method in terms of solution quality and of computing time. Also, on the set of problems tested it improved many of the best known results.

3.2 Applications of ACO Algorithms to Dynamic Combinatorial Optimization Problems

Research on the applications of ACO algorithms to dynamic combinatorial optimization problems has focused on communications networks. This is mainly because network optimization problems have characteristics, such as inherent information and computation distribution, nonstationary stochastic dynamics, and asynchronous evolution of the network status, that well match those of the ACO metaheuristic. In particular, the ACO approach has been applied to routing problems.

Routing is one of the most critical components of network control and concerns the network-wide distributed activity of building and using *routing tables* to direct data traffic. The routing table of a generic node i is a data structure that tells data packets entering node i which should be the next node to move to among the set $\mathcal{N}_i$ of neighbors of i. In the applications presented in this section routing tables for data packets are obtained by some functional transformation of ant-decision tables.

Let $G = (N, A)$ be a directed weighted graph, where each node in the set N represents a network node with processing/queuing and forwarding capabilities, and each oriented arc in A is a transmission system (link) with an associated weight defined by its physical properties. Network applications generate data flows from source to destination nodes. For each node in the network, the local routing component uses the local routing table to choose the best outgoing link to direct incoming data toward their destination nodes.

The generic routing problem can be informally stated as the problem of building routing tables so that some measure of network performance is maximized.[16]

Most of the ACO implementations for routing problems well match the general guidelines of the metaheuristic shown in Figure 3. Ants are launched from each network node toward heuristically selected destination nodes (launching follows some random or problem-specific schedule). Ants, like data packets, travel across the network and

15 In fact, on all the problems registered in the TSPLIB, a well-known repository for TSP-related benchmark problems available on the Internet at http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html.

16 The choice of what particular measure of network performance to use depends on the type of network considered and on which aspects of the provided services are most interesting. For example, in a telephone network, performance can be measured by the percentage of accepted calls and by the mean waiting time to setup or refuse a call, while in an Internet-like network, performance can be scored by the amount of correctly delivered bits per time unit (throughput), and by the distribution of data packet delays.

build paths from source to destination nodes by applying a probabilistic transition rule that makes use of information maintained in pheromone trail variables associated to links and, in some cases, of additional local information. Algorithm-specific heuristics and information structures are used to score the discovered paths and to set the amount of pheromone ants deposit.

A common characteristic of ACO algorithms for routing is that the role of the daemon (line 6 of the ACO metaheuristic of Figure 3) is much reduced: In the majority of the implementations it simply does not perform any actions.

ACO implementations for communications networks are grouped in two classes, those for connection-oriented and those for connectionless networks. In connection-oriented networks all the packets of a same session follow a common path selected by a preliminary setup phase. On the contrary, in connectionless, or datagram, networks data packets of a same session can follow different paths. At each intermediate node along the route from the source to the destination node a packet-specific forwarding decision is taken by the local routing component. In both types of networks best-effort routing, that is, routing without any explicit network resource reservation, can be delivered. Moreover, in connection-oriented networks an explicit reservation (software or hardware) of the resources can be done. In this way, services requiring specific characteristics (in terms of bandwidth, delay, etc.) can be delivered.[17]

3.2.1 Connection-Oriented Network Routing

The work by Schoonderwoerd, Holland, Bruten, and Rothkrantz [89, 90] has been, to our knowledge, the first attempt to apply an ACO algorithm to a routing problem. Their algorithm, called ant-based control (ABC), was applied to a model of the British Telecom (BT) telephone network. The network is modeled by a graph $G = (N, A)$, where each node i has the same functionalities as a crossbar switch with limited connectivity (capacity) and links have infinite capacity (i.e., they can carry a potentially infinite number of connections). Each node i has a set $\mathcal{N}_i$ of neighbors and is characterized by a total capacity C_i, and a spare capacity S_i. C_i represents the maximum number of connections node i can establish, while S_i represents the percentage of capacity that is still available for new connections. Each link (i, j) connecting node i to node j has an associated vector of pheromone trail values τ_{ijd}, $d = 1, \ldots, i-1, i+1, \ldots, N$. The value τ_{ijd} represents a measure of the desirability of choosing link (i, j) when the destination node is d.

Because the algorithm does not make use of any additional local heuristics, only pheromone values are used to define the ant-decision table values: $a_{ind}(t) = \tau_{ind}(t)$. The ant stochastic decision policy uses the pheromone values as follows: $\tau_{ind}(t)$ gives the probability that a given ant, the destination of which is node d, be routed at time t from node i to neighbor node n. An exploration mechanism is added to the ant's decision policy: With some low probability ants can choose the neighbor to move to following a uniformly random scheme over all the current neighbors. The ant internal state keeps track only of the ant source node and launching time. No memory about the visited nodes is maintained to avoid ant cycles. Routing tables for calls are obtained using ant-decision tables in a deterministic way: At setup time a route from node s to node d is built by choosing sequentially and deterministically, starting from node s, the neighbor node with the highest probability value until node d is reached. Once the call is set up in this way, the spare capacity S_i of each node i on the selected route is decreased by a fixed amount. If at call set-up time any of the nodes along the route under construction has no spare capacity left, then the call is rejected. When a call

17 For example, telephone calls need connection-oriented networks able to guarantee the necessary bandwidth during the entire call time.

terminates, the corresponding reserved capacity of nodes on its route is made available again for other calls. Ants are launched at regular temporal intervals from all the nodes toward destination nodes selected in a uniformly random way. Ants deposit pheromone only online, step-by-step, on the links they visit (they do not deposit pheromone after the path has been built, that is, they do not implement the procedure of lines 30–34 of the ACO metaheuristic of Figure 3). An ant k originated in node s and arriving at time t in node j from node i adds an amount $\Delta\tau^k(t)$ of pheromone to the value $\tau_{jis}(t)$ stored on link (j, i). The updated value $\tau_{jis}(t)$, which represents the desirability to go from node j to destination node s via node i, will be used by ants moving in the opposite direction of the updating ant. This updating strategy can be applied when the network is (approximately) cost symmetric, that is, when it is reasonable to use an estimate of the cost from node i to node j as an estimate of the cost from node j to node i. In the network model used by Schoonderwoerd et al. [89, 90] cost symmetry is a direct consequence of the assumptions made on the switches and transmission link structure. The pheromone trail update formula is

$$\tau_{jis}(t) \leftarrow \tau_{jis}(t) + \Delta\tau^k(t) \tag{13}$$

After the visited entry has been updated, the pheromone value of all the entries relative to the destination s decays. Pheromone decay, as usual, corresponds to the evaporation of real pheromone.[18] In this case the decay factor is set to $1/(1 + \Delta\tau^k(t))$ so that it operates a normalization of the pheromone values, which continue therefore to be usable as probabilities:

$$\tau_{ins}(t) \leftarrow \frac{\tau_{ins}(t)}{(1 + \Delta\tau^k(t))}, \qquad \forall n \in \mathcal{N}_i \tag{14}$$

The value $\Delta\tau^k(t)$ is a function of the ant's age. Ants move over a control network isomorphic to the real one. They grow older after each node hop and they are virtually delayed in nodes as a function of the node spare capacity. By this simple mechanism, the amount of pheromone deposited by an ant is made inversely proportional to the length and to the degree of congestion of the selected path. Therefore, the overall effect of ants on pheromone trail values is such that routes that are visited frequently and by "young" ants will be favored when building paths to route new calls.

In ABC no daemon actions are included in the algorithm. Each new call is accepted or rejected on the basis of a setup packet that looks for a path with spare capacity by probing the deterministically best path as indicated by the routing tables.

ABC has been tested on the above-described model of the British Telecom telephone network (30 nodes) using a sequential discrete time simulator and compared, in terms of percentage of accepted calls, to an agent-based algorithm developed by BT researchers. Results were encouraging:[19] ABC always performed significantly better than its competitor on a variety of different traffic situations.

White, Pagurek, and Oppacher [105] use an ACO algorithm for routing in connection-oriented point-to-point and point-to-multipoint networks. The algorithm follows a scheme very similar to AS (see Section 3.1.1): The ant-decision table has the same form as AS's (Equation 2) and the decision rule (Equation 3) is identical. The heuristic information η is locally estimated by link costs. From the source node of each incoming connection, a group of ants is launched to search for a path. At the beginning of the

18 Schoonderwoerd et al. [89, 90] developed ABC independently from previous ant colony optimization work. Therefore, they do not explicitly speak of pheromone evaporation, even if their probability renormalization mechanism plays the same role.

19 In the following we use the word "encouraging" whenever interesting results were obtained but the algorithm was compared only on simple problems, or with no state-of-the-art algorithms, or under limited experimental conditions.

trip each ant k sets to 0 a cost variable C_k associated to its path, and after each link crossing the path cost is incremented by the link cost l_{ij}: $C_k \leftarrow C_k + l_{ij}$. When arrived at their destination, ants move backward to their source node and at each node they use a simple additive rule to deposit pheromone on the visited links. The amount of deposited pheromone is a function of the whole cost C_k of the path, and only this online step-by-step updating is used to update pheromone. When all the spooled ants arrive back at the source node, a simple local daemon algorithm decides whether to allocate a path, based on the percentage of ants that followed a same path. Moreover, during all the connection lifetime, the local daemon launches and coordinates exploring ants to re-route the connection paths in case of network congestion or failures. A genetic algorithm [59, 66] is used online to evolve the parameters α and β of the transition rule formula, which determine the relative weight of pheromone and link costs (because of this mechanism the algorithm is called ASGA, ant system plus genetic algorithm). Some preliminary results were obtained testing the algorithm on several networks and using several link cost functions. Results are promising: The algorithm is able to compute shortest paths and the genetic adaptation of the rule parameters considerably improves the algorithm's performance.

Bonabeau, Henaux, Guérin, Snyers, Kuntz, and Theraulaz [6] improved the ABC algorithm by the introduction of a dynamic programming mechanism. They update the pheromone trail values of all the links on an ant path not only with respect to the ant's origin node, but also with respect to all the intermediate nodes on the subpath between the origin and the ant's current node.

Di Caro and Dorigo [30] are currently investigating the use of AntNet-FS to manage fair-share best-effort routing in high-speed connection-oriented networks. AntNet-FS is a derivation of AntNet, an algorithm for best-effort routing in connectionless networks the same authors developed. Therefore, we postpone the description of AntNet-FS to the next sub-subsection, where AntNet and its extensions are described in detail.

3.2.2 Connectionless Network Routing

Several ACO algorithms have been developed for routing in connectionless networks taking inspiration both from AS (Section 3.1.1) in general and from ABC (Section 3.2.1) in particular.

Di Caro and Dorigo [26–29, 31] developed several versions of AntNet, an ACO algorithm for distributed adaptive routing in best-effort connectionless (Internet-like) data networks. The main differences between ABC and AntNet are that (a) in AntNet real trip times experienced by ants (ants move over the same, real network as data packets) and local statistical models are used to evaluate paths goodness, (b) pheromone is deposited once a complete path is built (this is a choice dictated by a more general assumption of cost asymmetry made on the network), and (c) the ant decision rule makes use of local heuristic information η about the current traffic status.

In AntNet the ant-decision table $\mathcal{A}_i = [a_{ind}(t)]_{|\mathcal{N}_i|,|N|-1}$ of node i is obtained by the composition of the local pheromone trail values with the local heuristic values as follows:

$$a_{ind}(t) = \frac{\omega \tau_{ind}(t) + (1-\omega)\eta_n(t)}{\omega + (1-\omega)(|\mathcal{N}_i| - 1)} \tag{15}$$

where $\mathcal{N}_i$ is the set of neighbors of node i, $n \in \mathcal{N}_i$, d is the destination node, η_n is a [0,1]-normalized heuristic value inversely proportional to the length of the local link queue toward neighbor n, $\omega \in [0, 1]$ is a weighting factor and the denominator is a normalization term. The decision rule of the ant located at node i and directed toward destination node d (at time t) uses the entry $a_{ind}(t)$ of the ant-decision table as follows:

$a_{ind}(t)$ is simply the probability of choosing neighbor n. This probabilistic selection is applied over all the not-yet-visited neighbors by the ant, or over all the neighbors if all the neighbors have already been visited by the ant. While building the path to the destination, ants move using the same link queues as data. In this way, ants experience the same delays as data packets and the time T_{sd} elapsed while moving from the source node s to the destination node d can be used as a measure of the path quality. The overall "goodness" of a path is evaluated by a heuristic function of the trip time T_{sd} and of local adaptive statistical models. In fact, paths need to be evaluated relative to the network status because a trip time T judged of low quality under low congestion conditions could be an excellent one under high traffic load. Once a path has been completed ants deposit on the visited nodes an amount of pheromone proportional to the goodness of the path they built. AntNet's ants use only this online delayed way to update pheromone, different from ABC, which uses only the online step-by-step strategy (lines 30–34 and 24–27, respectively, of the ACO metaheuristic of Figure 3). To this purpose, after reaching their destination nodes, ants move back to their source nodes along the same path but backward and using high-priority queues, to allow a fast propagation of the collected information (in AntNet the term "forward ant" is used for ants moving from source to destination nodes, and the term "backward ant" for ants going back to their source nodes). During the backward path, the pheromone value of each visited link is updated with a rule similar to ABC's. AntNet differs from ABC also in a number of minor details, the most important of which are (a) ants are launched from each node toward destinations chosen to match probabilistically the traffic patterns, (b) all the pheromone values on an ant path are updated with respect to all the successor nodes of the (forward) path (as is done also in [6]), (c) cycles are removed online from the ants' paths,[20] and (d) data packets are routed probabilistically using routing tables obtained by means of a simple functional transformation of the ant-decision tables. AntNet was tested using a continuous time discrete events network simulator, on a wide variety of different spatial and temporal traffic conditions, and on several real and randomly generated network configurations (ranging from 8 to 150 nodes). State-of-the-art static and adaptive routing algorithms have been used for comparison. Results were excellent: AntNet showed striking superior performance in terms of both throughput and packet delays. Moreover, it appears to be very robust to the ant production rate and its impact on network resources is almost negligible.

Di Caro and Dorigo [27, 30] recently developed an enhanced version of AntNet, called AntNet-FA.[21] AntNet-FA is the same as AntNet except for the following two aspects. First, forward ants are substituted by so-called "flying ants": While building a path from source to destination node, flying ants make use of high-priority queues and do not store the trip times T. Second, each node maintains a simple local model of the local link queue depletion process. By using this model, rough estimates of the missing forward ant trip times are built. Backward ants read these estimates online and use them to evaluate the path quality and consequently to compute the amount of pheromone to deposit. AntNet-FA appears to be more reactive; the information collected by ants is more up-to-date and is propagated faster than in the original AntNet. AntNet-FA has been observed to perform much better than the original AntNet in best-effort connectionless networks [32].

Starting from AntNet-FA, Di Caro and Dorigo [30] are currently developing AntNet-FS, a routing and flow control system to manage multipath adaptive fair-share routing in connection-oriented high-speed networks. In AntNet-FS, some ants have some extra

20 This is a simple form of backtracking.

21 In the original paper [32] the same algorithm was called AntNet-CO, because the algorithm was developed in the perspective of connection-oriented routing, while here and in the future the authors choose to use the name AntNet-FA, to emphasize the "flying ant" nature of forward ants.

functionalities to support the search and allocation of multipaths for each new incoming user session. Forward setup ants fork to search for convenient multipaths (virtual circuits) to allocate the session. A daemon component local to the session end-points decides whether or not to accept the virtual circuits discovered by the forward setup ants. Accepted virtual circuits are allocated by the backward setup ants, reserving at the same time the session's bandwidth in a fair-share [74] fashion over the circuit nodes. The allocated bandwidth is dynamically redistributed and adapted after the arrival of a new session or the departure of an old one. The AntNet-FS approach looks very promising for high-speed networks (such as ATM), but needs more testing.

Subramanian, Druschel, and Chen [100] proposed the *regular ants* algorithm, which essentially is an application of Schoonderwoerd et al.'s ABC algorithm [89, 90] to packet-switched networks, where the only difference is the use of link costs instead of ants' age. The way their ants use link costs requires the network to be (approximately) cost-symmetric. They also propose *uniform ants*, that is, ants without a precise destination, which live for a fixed amount of time in the network and explore it by using a uniform probability scheme over the node neighbors. Uniform ants do not use the autocatalytic mechanism that characterizes all ACO algorithms and therefore do not belong to the ACO metaheuristic.

Heusse, Guérin, Snyers, and Kuntz [64] developed a new algorithm for general cost-asymmetric networks, called Co-operative Asymmetric Forward (CAF). In CAF, each data packet, after going from node i to node j, releases on node j the information c_{ij} about the sum of the experienced waiting and crossing times from node i. This information is used as an estimate of the time distance to go from i to j and is read by the ants traveling in the opposite direction to perform online step-by-step pheromone updating (no online delayed pheromone updating is used in this case). The algorithm's authors tested CAF under some static and dynamic conditions, using the average number of packets waiting in the queues and the average packet delay as performance measures. They compared CAF to an algorithm very similar to an earlier version of AntNet. Results were encouraging; under all the test situations CAF performed better than its competitors.

Van der Put and Rothkrantz [102, 103] designed ABC-backward, an extension of the ABC algorithm applicable to cost-asymmetric networks. They use the same forward-backward ant mechanism as in AntNet: Forward ants, while moving from the source to the destination node, collect information on the status of the network, and backward ants use this information to update the pheromone trails of the visited links during their journey back from the destination to the source node. In ABC-backward, backward ants update the pheromone trails using an updating formula identical to that used in ABC, except for the fact that the ants' age is replaced by the trip times experienced by the ants in their forward journey. Van der Put and Rothkrantz have shown experimentally that ABC-backward has a better performance than ABC on both cost-symmetric—because backward ants can avoid depositing pheromone on cycles—and cost-asymmetric networks. They apply ABC-backward to a fax distribution problem proposed by the largest Dutch telephone company (KPN Telecom).

3.3 Parallel Implementations

The very nature of ACO algorithms lends them to be naturally parallelized in the data or population domains. In particular, many parallel models used in other population-based algorithms can be easily adapted to the ACO structure (e.g., migration and diffusion models adopted in the field of parallel genetic algorithms; see, for example, reviews in [16, 39]).

Early experiments with parallel versions of AS for the TSP on the Connection Machine CM-2 [65] adopted the approach of attributing a single processing unit to each ant [4].

Experimental results showed that communication overhead can be a major problem with this approach on fine-grained parallel machines, since ants end up spending most of their time communicating to other ants the modifications they made to pheromone trails. In fact, the algorithm's behavior was not impressive and scaled up very badly when increasing the problem dimensions. Better results were obtained on a coarse-grained parallel network of 16 transputers [4, 34]. The idea here was to divide the colony into p subcolonies, where p is the number of available processors. Each subcolony acts as a complete colony and therefore implements a standard AS algorithm. After each subcolony has completed an iteration of the algorithm, a hierarchical broadcast communication process collects the information about the tours of all the ants in all the subcolonies and then broadcasts this information to all the p processors so that a concurrent update of the pheromone trails can be done. In this case the speed-up was nearly linear when increasing the number of processors, and this behavior did not change significatively for increasing problem dimensions.

More recently, Bullnheimer, Kotsis, and Strauss [14] proposed two coarse-grained parallel versions of AS. The first one, called Synchronous Parallel Implementation (SPI), is basically the same as the one implemented by Bolondi and Bondanza [4], while the second one, called Partially Asynchronous Parallel Implementation (PAPI), exchanges pheromone information among subcolonies every fixed number of iterations done by each subcolony. The two algorithms have been evaluated by simulation. The findings show that the reduced communication due to the less frequent exchange of pheromone trail information among subcolonies determines a better performance of the PAPI approach with respect to running time and speed-up. More experimentation is necessary to compare the quality of the results produced by the SPI and the PAPI implementations.

Krüger, Merkle, and Middendorf [70] investigated which (pheromone trail) information should be exchanged between the m subcolonies and how this information should be used to update the subcolony's trail information. They compared an exchange of (a) the global best solution (every subcolony uses the global best solution to choose where to add pheromone trail), (b) the local best solutions (every subcolony receives the local best solution from all other subcolonies and updates pheromone trail on the corresponding arcs), and (c) the total trail information (every colony computes the average over the trail information of all colonies); that is, if $\tau^l = [\tau_{ij}^l]$ is the trail information of subcolony l, $1 \leq l \leq m$ then every colony l sends τ^l to the other colonies and afterward computes $\tau_{ij}^l = \sum_{h=1}^{m} \tau_{ij}^h$, $1 \leq i, j \leq n$. The results indicate that methods (a) and (b) are faster and give better solutions than method (c), but further investigations are necessary.

Last, Stützle [94] presents computational results for the execution of parallel independent runs on up to 10 processors of his $\mathcal{MM}$AS algorithm [98, 97]. The execution of parallel independent runs is the easiest way to obtain a parallel algorithm and, obviously, it is a reasonable approach only if the underlying algorithm, as is the case with ACO algorithms, is randomized. Stützle's results show that the performance of $\mathcal{MM}$AS grows with the number of processors.

4 Related Work

ACO algorithms show similarities with some optimization, learning, and simulation approaches such as heuristic graph search, Monte Carlo simulation, neural networks, and evolutionary computation. These similarities are now briefly discussed.

Heuristic graph search. In ACO algorithms each ant performs a heuristic graph search in the space of the components of a solution: Ants take biased probabilistic decisions to choose the next component to move to, where the bias is given by a heuristic evaluation function that favors components that are perceived as more promising. It is interesting to note that this is different from what happens, for example, in *stochastic hillclimbers* [81] or in *simulated annealing* [68], where (a) an acceptance criteria is defined and only those randomly generated moves that satisfy the criteria are executed, and (b) the search is usually performed in the space of the solutions.

Monte Carlo simulation. ACO algorithms can be interpreted as parallel replicated Monte Carlo systems [93]. Monte Carlo systems [87] are general stochastic simulation systems, that is, techniques performing repeated sampling experiments on the model of the system under consideration by making use of a stochastic component in the state sampling and/or transition rules. Experiment results are used to update some statistical knowledge about the problem, as well as the estimate of the variables the researcher is interested in. In turn, this knowledge can also be iteratively used to reduce the variance in the estimation of the desired variables, directing the simulation process toward the most interesting regions of the state space. Analogously, in ACO algorithms the ants sample the problem's solution space by repeatedly applying a stochastic decision policy until a feasible solution of the considered problem is built. The sampling is realized concurrently by a collection of differently instantiated replicas of the same ant type. Each ant "experiment" allows adaptive modification of the local statistical knowledge on the problem structure (i.e., the pheromone trails). The recursive transmission of such knowledge by means of stigmergy determines a reduction in the variance of the whole search process: The so-far most interesting explored transitions probabilistically bias future search, preventing ants from wasting resources in unpromising regions of the search space.

Neural networks. Ant colonies, being composed of numerous concurrently and locally interacting units, can be seen as "connectionist" systems [43], the most famous examples of which are neural networks [3, 63, 88]. From a structural point of view, the parallel between the ACO metaheuristic and a generic neural network is obtained by putting each state i visited by ants in correspondence with a neuron i, and the problem-specific neighborhood structure of state i in correspondence with the set of synaptic-like links exiting neuron i. The ants themselves can be seen as input signals concurrently propagating through the neural network and modifying the strength of the synaptic-like interneuron connections. Signals (ants) are locally propagated by means of a stochastic transfer function and the more a synapse is used, the more the connection between its two end neurons is reinforced. The ACO-synaptic learning rule can be interpreted as an a posteriori rule: Signals related to good examples, that is, ants that discovered a good quality solution, reinforce the synaptic connections they traverse more than signals related to poor examples. It is interesting to note that the ACO-neural network algorithm does not correspond to any existing neural network model.

The ACO-neural network is also reminiscent of networks solving *reinforcement learning* problems [101]. In reinforcement learning the only feedback available to the learner is a numeric signal (the *reinforcement*) that scores the result of actions. This is also the case in the ACO metaheuristic: The signals (ants) fed into the network can be seen as input examples with an associated approximate score measure. The strength of pheromone updates and the level of stochasticity in signal propagation play the role of a learning rate, controlling the balance between exploration and exploitation.

Finally, it is worth making a reference to the work of Chen [17], who proposed a neural network approach to TSP that bears important similarities with the ACO approach. Like in ACO algorithms, Chen builds a tour in an incremental way, according to synaptic strengths. It also makes use of candidate lists and 2-opt local optimization. The strengths of the synapses of the current tour and of all previous tours are updated according to a Boltzmann-like rule and a learning rate playing the role of an evaporation coefficient. Although there are some differences, the common features are, in this case, striking.

Evolutionary computation. There are some general similarities between the ACO metaheuristic and evolutionary computation (EC) [45]. Both approaches use a population of individuals that represent problem solutions, and in both approaches the knowledge about the problem collected by the population is used to generate stochastically a new population of individuals. A main difference is that in EC algorithms all the knowledge about the problem is contained in the current population, while in ACO a memory of past performance is maintained under the form of pheromone trails.

An EC algorithm that is very similar to ACO algorithms in general and to AS in particular is Baluja and Caruana's *Population Based Incremental Learning* (PBIL) [1]. PBIL maintains a vector of real numbers, the generating vector, which plays a role similar to that of the population in genetic algorithms [59, 66]. Starting from this vector, a population of binary strings is randomly generated: Each string in the population will have the ith bit set to 1 with a probability that is a function of the ith value in the generating vector. Once a population of solutions is created, the generated solutions are evaluated and this evaluation is used to increase (or decrease) the probabilities of each separate component in the generating vector so that good (bad) solutions in the future generations will be produced with higher (lower) probability. It is clear that in ACO algorithms the pheromone trail values play a role similar to Baluja and Caruana's generating vector, and pheromone updating has the same goal as updating the probabilities in the generating vector. A main difference between ACO algorithms and PBIL consists in the fact that in PBIL all the probability vector components are evaluated independently, making the approach work well only when the solution is separable into its components.

The $(1, \lambda)$ *evolution strategy* is another EC algorithm that is related to ACO algorithms, and in particular to ACS. In fact, in the $(1, \lambda)$ evolution strategy the following steps are iteratively repeated: (a) a population of λ solutions (ants) is initially generated, then (b) the best individual of the population is saved for the next generation, while all the other solutions are discarded, and (c) starting from the best individual, $\lambda - 1$ new solutions are stochastically generated by mutation, and finally (d) the process is iterated going back to step (b). The similitude with ACS is striking.

Stochastic learning automata. This is one of the oldest approaches to machine learning (see [80] for a review). An automaton is defined by a set of possible actions and a vector of associated probabilities, a continuous set of inputs and a learning algorithm to learn input-output associations. Automata are connected in a feedback configuration with the environment, and a set of penalty signals from the environment to the actions is defined. The similarity of stochastic learning automata and ACO approaches can be made clear as follows. The set of pheromone trails available on each arc/link is seen as a set of concurrent stochastic learning automata. Ants play the role of the environment signals, while the pheromone update rule is the automaton learning rule. The main difference lies in the fact that in ACO the "environment signals" (i.e., the ants) are stochastically biased, by means of their probabilistic transition rule, to direct the learning process toward the most interesting regions of the search space. That is, the whole environment plays a key, active role in learning good state-action pairs.

5 Discussion

The ACO metaheuristic was defined a posteriori, that is, it is the result of a synthesis effort effectuated on a set of algorithms inspired by a common natural process. Such a synthesis can be very useful because (a) it is a first attempt to characterize this new class of algorithms, and (b) it can be used as a reference to design new instances of ACO algorithms. On the other hand, the a posteriori character of the synthesis determines a great variety in the way some aspects of the metaheuristic are implemented, as discussed in the following.

Role of the local heuristic. Most of the ACO algorithms presented combine pheromone trails with local heuristic values to obtain ant-decision tables. An exception are the Schoonderwoerd et al. [89, 90] ABC algorithm and all the derived algorithms (ABC-smart ants, ABC-backward, regular ants, and CAF) in which ant-decision tables are obtained using only pheromone trail values. Current wisdom indicates that the use of a heuristic value, whenever possible, improves ACO performance considerably. The nature of the heuristic information differs between static and dynamic problems. In all static problems attacked by ACO a simple heuristic value directly obtainable from the problem definition was used. On the contrary, in dynamic problems the heuristic value must be estimated by local statistical sampling of the dynamic system.

Step-by-step versus delayed online solution evaluation. In dynamic combinatorial optimization problems some of the proposed ACO algorithms use step-by-step online solution evaluation: ABC, ABC-smart, and regular ants, which take advantage of the cost-symmetric nature of the network model, and CAF, which, although applied to a cost-asymmetric network, can apply a step-by-step solution evaluation by exploiting information deposited on nodes by data packets traveling in the opposite direction of ants. The other algorithms applied to cost-asymmetric networks, AntNet and ABC-backward, use delayed online solution evaluation. For the static optimization problems considered, the use of a step-by-step online solution evaluation would be misleading because problem constraints make the quality of partial solutions not a good estimate of the quality of complete solutions.

Pheromone trail directionality. The pheromone trail can have directional properties. This is true for all dynamic optimization problems considered. Differently, in all the applications to static problems (Section 3.1) pheromone trail is not directional: An ant using arc (i, j) will see the same pheromone trail values independent of the node it is coming from. (It is the decision policy that can take into consideration the node, or the series of nodes, the ant is coming from.)

Implicit solution evaluation. One of the interesting aspects of real ants' shortest path-finding behavior is that they exploit *implicit solution evaluation*: If an ant takes a shorter path it will arrive at the destination before any other ant that took a longer path. Therefore, shorter paths will receive pheromone earlier and they start to attract new ants before longer paths. This implicit solution evaluation property is exploited by the ACO algorithms applied to routing, and not by those applied to static optimization problems. The reason for this is that implicit solution evaluation is obtained for free whenever the speed with which ants move on the problem representation is inversely proportional to the cost of each state transition during solution construction. While this is the most natural way to implement artificial ants for network applications, it is not an efficient choice for the considered static problems. In fact, in this case it would be necessary to implement an extra algorithm component to manage each ant's speed, which would require extra computation resources without any guarantee of improved performance.

6 Conclusions

In this article we have introduced the ant colony optimization (ACO) metaheuristic and we have given an overview of ACO algorithms. ACO is a novel and very promising research field situated at the crossing between artificial life and operations research. The ant colony optimization metaheuristic belongs to the relatively new wave of stochastic metaheuristics such as evolutionary computation [45], simulated annealing [68], tabu search [57, 58], neural computation [3, 63, 88], and so on, which are built around some basic principles taken from the observation of a particular natural phenomenon. As is very common in the practical usage of these heuristics, ACO algorithms often end up at some distance from their inspiring natural metaphor. Often ACO algorithms are enriched with capacities that do not find a counterpart in real ants, such as local search and global-knowledge-based actions, so that they can compete with more application-specific approaches. ACO algorithms so enriched are very competitive and in some applications they have reached world-class performance. For example, on structured quadratic assignment problems AS-QAP, HAS-QAP, and $\mathcal{MM}$AS-QAP are currently the best available heuristics. Other very successful applications are those to the sequential ordering problem, for which HAS-SOP is by far the best available heuristic, and to data network routing, where AntNet resulted in being superior to a whole set of state-of-the-art algorithms.

Within the artificial life field, ant algorithms represent one of the most successful applications of swarm intelligence.[22] One of the most characterizing aspects of swarm intelligent algorithms, shared by ACO algorithms, is the use of the stigmergetic model of communication. We have seen that this form of indirect distributed communication plays an important role in making ACO algorithms successful. There are, however, examples of applications of stigmergy based on social insect behaviors other than ants' foraging behavior. For example, the stigmergy-mediated allocation of work in ant colonies has inspired models of task allocation in a distributed mail retrieval system; dead body aggregation and brood sorting, again in ant colonies, have inspired a data clustering algorithm; and models of collective transport by ants have inspired transport control strategies for groups of robot.[23]

In conclusion, we hope this paper has achieved its goal: To convince the reader that ACO, and more generally the stigmergetic model of communication, are worth further research.

Acknowledgments

We are grateful to Nick Bradshaw, Bernd Bullnheimer, Martin Heusse, Owen Holland, Vittorio Maniezzo, Martin Middendorf, Ruud Schoonderwoerd, and Dominique Snyers for critical reading of a draft version of this article. We also wish to thank Eric Bonabeau and the two referees for their valuable comments. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Research Associate. This work was supported by a Marie Curie Fellowship awarded to Gianni Di Caro (CEC-TMR Contract N. ERBFMBICT 961153).

References

1. Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In A. Prieditis & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning, ML-95* (pp. 38–46). Palo Alto, CA: Morgan Kaufmann.

2. Bersini, H., Dorigo, M., Langerman, S., Seront, G., & Gambardella, L. M. (1996). Results of the first international contest on evolutionary optimisation (1st ICEO). In *Proceedings of IEEE International Conference on Evolutionary Computation, IEEE-EC 96* (pp. 611–615). Piscataway, NJ: IEEE Press.

3. Bishop, C. M. (1995). *Neural networks for pattern recognition.* Oxford: Oxford University Press.

22 Swarm intelligence can be defined as the field that covers "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies" ([5], p. 7).

23 These and other examples of the possible applications of stigmergetic systems are discussed in detail by Bonabeau, Dorigo, and Theraulaz [5].

4. Bolondi, M., & Bondanza, M. (1993). *Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore*. Unpublished master's thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy.

5. Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *From natural to artificial swarm intelligence*. New York: Oxford University Press.

6. Bonabeau, E., Hénaux, F., Guérin, S., Snyers, D., Kuntz, P., & Théraulaz, G. (1998). Routing in telecommunications networks with "smart" ant-like agents. In S. Albayrak & F. J. Garijo (Eds.), *Proceedings of IATA '98, Second International Workshop on Intelligent Agents for Telecommunication Applications* (pp. 60–71). Lecture Notes in AI vol. 1437, Springer-Verlag.

7. Branke, J., Middendorf, M., & Schneider, F. (1998). Improved heuristics and a genetic algorithm for finding short supersequences. *OR-Spektrum, 20*, 39–46.

8. Brelaz, D. (1979). New methods to color vertices of a graph. *Communications of the ACM, 22*, 251–256.

9. Bruckstein, A. M. (1993). Why the ant trails look so straight and nice. *The Mathematical Intelligencer, 15*(2), 59–62.

10. Bruckstein, A. M., Mallows, C. L., & Wagner, I. A. (1997). Probabilistic pursuits on the grid. *American Mathematical Monthly, 104*(4), 323–343.

11. Bullnheimer, B., Hartl, R. F., & Strauss, C. (1997). *An improved Ant System algorithm for the vehicle routing problem*. (Tech. Rep. POM-10/97). Vienna, Austria: University of Vienna, Institute of Management Science. To appear in Dawid, Feichtinger, & Hartl (Eds.), *Annals of Operations Research: Nonlinear Economic Dynamics and Control*.

12. Bullnheimer, B., Hartl, R. F., & Strauss, C. (1997). *A new rank-based version of the Ant System: A computational study*. (Tech. Rep. POM-03/97). Vienna, Austria: University of Vienna, Institute of Management Science. Also available in (1999). *Central European Journal for Operations Research and Economics, 7*(1), 25–38.

13. Bullnheimer, B., Hartl, R. F., & Strauss, C. (1998). Applying the Ant System to the vehicle routing problem. In S. Voss., S. Martello, I. H. Osman, & C. Roucairol (Eds.), *Meta-heuristics: Advances and trends in local search paradigms for optimization* (pp. 109–120). Boston: Kluwer.

14. Bullnheimer, B., Kotsis, G., & Strauss, C. (1997). *Parallelization strategies for the Ant System*. (Tech. Rep. POM-9-97). Vienna, Austria: University of Vienna, Institute of Management Science. Also available in (1998). R. De Leone, A. Murli, P. Pardalos, & G. Toraldo (Eds.), *High performance algorithms and software in nonlinear optimization* (pp. 87–100). (Series: Applied Optimization, vol. 24). Dordrecht: Kluwer.

15. Bullnheimer, B., & Strauss, C. (1996). *Tourenplanung mit dem Ant System*. (Tech. Rep. 6). Vienna, Austria: Instituts für Betriebwirtschaftslehre, Universität Wien.

16. Campanini, R., Di Caro, G., Villani, M., D'Antone, I., & Giusti, G. (1994). Parallel architectures and intrinsically parallel algorithms: Genetic algorithms. *International Journal of Modern Physics C, 5*(1), 95–112.

17. Chen, K. (1997). A simple learning algorithm for the traveling salesman problem. *Physical Review E, 55*, 7809–7812.

18. Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*, 568–581.

19. Colorni, A., Dorigo, M., & Maniezzo, V. (1992). Distributed optimization by ant colonies. In F. J. Varela & P. Bourgina (Eds.), *Proceedings of the First European Conference on Artificial Life* (pp. 134–142). Cambridge, MA: MIT Press/Bradford Books.

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

21. Corne, D., Dorigo, M., & Glover, F. (Eds.). (1999). *New ideas in optimization*. Maidenhead, UK: McGraw-Hill.

22. Costa, D., & Hertz, A. (1997). Ants can colour graphs. *Journal of the Operational Research Society, 48*, 295–305.

23. Costa, D., Hertz, A., & Dubuis, O. (1995). Embedding of a sequential algorithm within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics, 1*, 105–128.

24. Croes, G. A. (1958). A method for solving traveling salesman problems. *Operations Research, 6*, 791–812.

25. Deneubourg, J.-L., Aron, S., Goss, S., & Pasteels, J.-M. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior, 3*, 159–168.

26. Di Caro, G., & Dorigo, M. (1997). *AntNet: A mobile agents approach to adaptive routing.* (Tech. Rep. 97-12). Université Libre de Bruxelles, IRIDIA.

27. Di Caro, G., & Dorigo, M. (1998). An adaptive multi-agent routing algorithm inspired by ants behavior. In *Proceedings of PART98—Fifth Annual Australasian Conference on Parallel and Real-Time Systems* (pp. 261–272). Springer-Verlag.

28. Di Caro, G., & Dorigo, M. (1998). Ant colonies for adaptive routing in packet-switched communications networks. In A. E. Eiben, T. Bäck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature* (pp. 673–682). Berlin: Springer-Verlag.

29. Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research* (*JAIR*), *9*, 317–365. Available at http://www.jair.org/abstracts/dicaro98a.html.

30. Di Caro, G., & Dorigo, M. (1998). Extending AntNet for best-effort Quality-of-Service routing. Unpublished presentation at *ANTS '98—From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimization.* http://iridia.ulb.ac.be/ants98/ants98.html.

31. Di Caro, G., & Dorigo, M. (1998). Mobile agents for adaptive routing. In H. El-Rewini (Ed.), *Proceedings of the 31st International Conference on System Sciences* (*HICSS-31*) (Vol. 7, pp. 74–83). Los Alamitos, CA: IEEE Computer Society Press.

32. Di Caro, G., & Dorigo, M. (1998). Two ant colony algorithms for best-effort routing in datagram networks. In *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems* (*PDCS '98*) (pp. 541–546). IASTED/ACTA Press.

33. Dorigo, M. (1992). *Optimization, learning and natural algorithms* (in Italian). Unpublished doctoral dissertation, Politecnico di Milano, Dipartimento di Elettronica, Italy.

34. Dorigo, M. (1993). *Parallel Ant System: An experimental study.* Unpublished manuscript.

35. Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization.* Maidenhead, UK: McGraw-Hill.

36. Dorigo, M., & Gambardella, L. M. (1996). A study of some properties of Ant-Q. In H.-M. Voight, W. Ebeling, I. Rechenberg, & H.-P. Schwefel (Eds.), *Proceedings of PPSN-IV, Fourth International Conference on Parallel Problem Solving from Nature* (pp. 656–665). Berlin: Springer-Verlag.

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

38. Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation, 1*(1), 53–66.

39. Dorigo, M., & Maniezzo, V. (1993). Parallel genetic algorithms: Introduction and overview of the current research. In J. Stender (Ed.), *Parallel genetic algorithms: Theory & applications* (pp. 5–42). Amsterdam: IOS Press.

40. Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Positive feedback as a search strategy.* (Tech. Rep. 91-016). Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica.

41. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics—Part B, 26*(1), 29–41.

42. Escudero, L. F. (1988). An inexact algorithm for the sequential ordering problem. *European Journal of Operations Research, 37*, 232–253.

43. Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science, 6*, 205–254.

44. Fleurent, C., & Ferland, J. (1996). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research, 63*, 437–461.

45. Fogel, D. B. (1995). *Evolutionary computation*. Piscataway, NJ: IEEE Press.

46. Foulser, D. E., Li, M., & Yang, Q. (1992). Theory and algorithms for plan merging. *Artificial Intelligence, 57*, 143–181.

47. Freisleben, B., & Merz, P. (1996). Genetic local search algorithms for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of IEEE International Conference on Evolutionary Computation, IEEE-EC '96* (pp. 616–621). Piscataway, NJ: IEEE Press.

48. Freisleben, B., & Merz, P. (1996). New genetic local search operators for the traveling salesman problem. In H.-M. Voigt, W. Ebeling, I. Rechenberg, & H.-P. Schwefel (Eds.), *Proceedings of PPSN-IV, Fourth International Conference on Parallel Problem Solving from Nature* (pp. 890–899). Berlin: Springer-Verlag.

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

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

51. Gambardella, L. M., & Dorigo, M. (1997). *HAS-SOP: An hybrid ant system for the sequential ordering problem*. (Tech. Rep. 11-97). Lugano, Switzerland: IDSIA.

52. Gambardella, L. M., Taillard, É., & Agazzi, G. (1999). MACS-VRPTW: A multiple Ant Colony System for vehicle routing problems with time windows. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization*. Maidenhead, UK: McGraw-Hill.

53. Gambardella, L. M., Taillard, E. D., & Dorigo, M. (1997). *Ant colonies for the QAP*. (Tech. Rep. 4-97). Lugano, Switzerland: IDSIA.

54. Gambardella, L. M., Taillard, E. D., & Dorigo, M. (1999). Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society, 50*(2), 167–176.

55. Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 2*(2), 117–129.

56. Gavett, J. (1965). Three heuristic rules for sequencing jobs to a single production facility. *Management Science, 11*, 166–176.

57. Glover, F. (1989). Tabu search, part I. *ORSA Journal on Computing, 1*, 190–206.

58. Glover, F. (1989). Tabu search, part II. *ORSA Journal on Computing, 2*, 4–32.

59. Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.

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

61. Grassé, P. P. (1946). *Les insects dans leur univers*. Paris: Éditions du Palais de la découverte.

62. Grassé, P. P. (1959). 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.

63. Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation.* Redwood City, CA: Addison-Wesley.

64. Heusse, M., Guérin, S., Snyers, D., & Kuntz, P. (1998). Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems, 1*, 237–254.

65. Hillis, W. D. (1982). *The connection machine.* Cambridge, MA: MIT Press.

66. Holland, J. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

67. Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study. In E. H. Aarts & J. K. Lenstra (Eds.), *Local search in combinatorial optimization* (pp. 215–310). Chichester, UK: Wiley.

68. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

69. Koopmans, T. C., & Beckmann, M. J. (1957). Assignment problems and the location of econonomic activities. *Econometrica, 25*, 53–76.

70. Krüger, F., Merkle, D., & Middendorf, M. (1998). *Studies on a parallel Ant System for the BSP model.* Unpublished manuscript.

71. Lawler, E. L., Lenstra, J. K., Rinnooy-Kan, A. H. G., & Shmoys, D. B. (Eds.). (1985). *The travelling salesman problem.* Chichester, UK: Wiley.

72. Leighton, F. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards, 84*, 489–505.

73. Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell Systems Journal, 44*, 2245–2269.

74. Ma, Q., Steenkiste, P., & Zhang, H. (1996). Routing in high-bandwidth traffic in max-min fair share networks. *ACM Computer Communication Review* (*SIGCOMM '96*), *26*(4), 206–217.

75. Maniezzo, V. (1998). *Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem.* (Tech. Rep. CSR 98-1). In Scienze dell'Informazione, Universitá di Bologna, sede di Cesena, Italy.

76. Maniezzo, V., & Colorni, A. (in press). The Ant System applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering.*

77. Maniezzo, V., Colorni, A., & Dorigo, M. (1994). *The Ant System applied to the quadratic assignment problem.* (Tech. Rep. IRIDIA/94-28). Belgium: Université Libre de Bruxelles.

78. Michel, R., & Middendorf, M. (1998). An island model based Ant System with lookahead for the shortest supersequence problem. In A. E. Eiben, T. Bäck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature* (pp. 692–701). Berlin: Springer-Verlag.

79. Michel, R., & Middendorf, M. (1999). An ACO algorithm for the shortest common supersequence problem. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization.* Maidenhead, UK: McGraw-Hill.

80. Narendra, K., & Thathachar, M. (1989). *Learning automata: An introduction.* Prentice-Hall.

81. Nilsson, N. J. (1998). *Artificial intelligence: A new synthesis.* Morgan Kaufmann.

82. Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research, 34*, 336–344.

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

84. Räihä, K.-J., & Ukkonen, E. (1981). The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science, 16*, 187–198.

85. Rechenberg, R. I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Stuttgart, Germany: Frommann-Holzboog.

86. Reinelt, G. (1994). *The traveling salesman problem: Computational solutions for TSP applications.* Berlin: Springer-Verlag.

87. Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo method.* New York: Wiley.

88. Rumelhart, D. E., McClelland, J. L., & the PDP Research Group (Eds.). (1986). *Parallel distributed processing.* Cambridge, MA: MIT Press.

89. Schoonderwoerd, R., Holland, O., & Bruten, J. (1997). Ant-like agents for load balancing in telecommunications networks. In J. Müller (Ed.), *Proceedings of the First International Conference on Autonomous Agents* (pp. 209–216). ACM Press.

90. Schoonderwoerd, R., Holland, O., Bruten, J., & Rothkrantz, L. (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior, 5*(2), 169–207.

91. Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* Basel, Switzerland: Birkauser.

92. Shani, S., & Gonzales, T. (1976). P-complete approximation problems. *Journal of ACM, 23*, 555–565.

93. Streltsov, S., & Vakili, P. (1996). Variance reduction algorithms for parallel replicated simulation of uniformized Markov chains. *Discrete Event Dynamic Systems: Theory and Applications, 6*, 159–180.

94. Stützle, T. (1998). Parallelization strategies for ant colony optimization. In A. E. Eiben, T. Bäck, M. Schoenauer, & H.-P. Schwefel (Eds.), *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature* (pp. 722–731). Berlin: Springer-Verlag.

95. Stützle, T., & Dorigo, M. (1999). ACO algorithms for the quadratic assignment problem. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization.* Maidenhead, UK: McGraw-Hill.

96. Stützle, T., & Dorigo, M. (1999). ACO algorithms for the traveling salesman problem. In K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, & J. Periaux (Eds.), *Evolutionary algorithms in engineering and computer science.* Chichester, UK: Wiley.

97. Stützle, T., & Hoos, H. (1997). Improvements on the Ant System: Introducing $\mathcal{MAX}-\mathcal{MIN}$ ant system. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms* (pp. 245–249). Springer-Verlag.

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

99. Stützle, T., & Hoos, H. (1998). $\mathcal{MAX}-\mathcal{MIN}$ Ant System and local search for combinatorial optimization problems. In S. Voss, S. Martello, I. H. Osman, & C. Roucairol (Eds.), *Meta-heuristics: Advances and trends in local search paradigms for optimization* (pp. 137–154).

100. Subramanian, D., Druschel, P., & Chen, J. (1997). Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of IJCAI-97, International Joint Conference on Artifical Intelligence* (pp. 832–838). Palo Alto, CA: Morgan Kauffman.

101. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

102. van der Put, R. (1998). *Routing in the faxfactory using mobile agents.* (Tech. Rep. R&D-SV-98-276). Leidschendam, The Netherlands: KPN Research.

103. van der Put, R., & Rothkrantz, L. (in press). Routing in packet switched networks using agents. *Simulation Practice and Theory.*

104. Watkins, C. J. (1989). *Learning with delayed rewards*. Unpublished doctoral dissertation, Psychology Department, University of Cambridge, UK.

105. White, T., Pagurek, B., & Oppacher, F. (1998). Connection management using adaptive mobile agents. In H. R. Arabnia (Ed.), *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)* (pp. 802–809). Athens, GA: CSREA Press.

106. Whitley, D., Starkweather, T., & Fuquay, D. (1989). Scheduling problems and travelling salesman: The genetic edge recombination operator. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 133–140). Palo Alto, CA: Morgan Kaufmann.