Ant Colony Optimization: A New Meta-Heuristic

Marco Dorigo
IRIDIA
Université Libre de Bruxelles
mdorigo@ulb.ac.be

Gianni Di Caro
IRIDIA
Université Libre de Bruxelles
gdicaro@iridia.ulb.ac.be

Abstract- Recently, a number of algorithms inspired by the foraging behavior of ant colonies have been applied to the solution of difficult discrete optimization problems. In this paper we put these algorithms in a common framework by defining the Ant Colony Optimization (ACO) meta-heuristic. A couple of paradigmatic examples of applications of these novel meta-heuristic are given, as well as a brief overview of existing applications.

1 Introduction

In the early nineties an algorithm called *ant system* was proposed as a novel heuristic approach for the solution of combinatorial optimization problems (Dorigo *et al.*, 1991; Dorigo, 1992; Dorigo *et al.*, 1996). Ant system (AS), which was first applied to the traveling salesman problem, was recently extended and/or modified both to improve its performance and to apply it to other optimization problems. Improved versions of AS include, among others, ACS (Dorigo & Gambardella, 1997), $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (Stützle & Hoos, 1998b), and AS_{rank} (Bullnheimer *et al.*, 1997b). All these algorithms have been applied to the TSP with varying degree of success, but always improving over AS performance. These improved versions, as well as the original AS, have also been applied to a diverse set of optimization problems. Examples are quadratic assignment (Maniezzo *et al.*, 1994; Stützle & Hoos, 1998a), vehicle routing (Bullnheimer *et al.*, 1997a; Gambardella *et al.*, 1999), connection oriented and connectionless network routing (Schoonderwoerd *et al.*, 1996; Di Caro & Dorigo, 1998; Di Caro & Dorigo, 1998), sequential ordering (Gambardella & Dorigo, 1997), graph coloring (Costa & Hertz, 1997), shortest common supersequence (Michel & Middendorf, 1998), single machine tardiness (Bauer *et al.*, 1999), multiple knapsack (Leguizamón & Michalewicz, 1999), etc. For many of these problems, the results obtained place the ant system based approaches among the best available heuristics.

In this paper we present the Ant Colony Optimization (ACO) meta-heuristic, that is, the result of an effort to define a common framework for all these versions of AS. The main motivations for defining the ACO meta-heuristic are the desire to provide a unitary view of the ongoing research in this growing field, as well as the hope that such a characterization will help to identify the most important aspects of these algorithms and therefore make the development of new applications easier.

2 The ACO meta-heuristic

The ACO meta-heuristic can be applied to discrete optimization problems characterized as follows.

- $C = \{c_1, c_2, \ldots, c_{N_C}\}$ is a finite set of *components*.
- $L = \{l_{c_i c_j} \mid (c_i, c_j) \in \tilde{C}\}$, $|L| \leq N_C^2$ is a finite set of possible *connections/transitions* among the elements of $\tilde{C}$, where $\tilde{C}$ is a subset of the Cartesian product $C \times C$.
- $J_{c_i c_j} \equiv J(l_{c_i c_j}, t)$ is a *connection cost* function associated to each $l_{c_i c_j} \in L$, possibly parameterized by some time measure t.
- $\Omega \equiv \Omega(C, L, t)$ is a finite set of *constraints* assigned over the elements of C and L.
- $s = \langle c_i, c_j, \ldots, c_k, \ldots \rangle$ is a sequence over the elements of C (or, equivalently, of L). A sequence s is also called a *state* of the problem. If S is the set of all possible sequences, the set $\tilde{S}$ of all the (sub)sequences that are feasible with respect to the constraints $\Omega(C, L, t)$, is a subset of S. The elements in $\tilde{S}$ define the problem's *feasible states*. The length of a sequence s, that is, the number of components in the sequence, is expressed by $|s|$.
- Given two states s_1 and s_2 a *neighborhood structure* is defined as follows: the state s_2 is said to be a neighbor of s_1 if both s_1 and s_2 are in S, and the state s_2 can be reached from s_1 in one logical step (that is, if c_1 is the last component in the sequence determining the state s_1, it must exists $c_2 \in C$ such that $l_{c_1 c_2} \in L$ and $s_2 \equiv \langle s_1, c_2 \rangle$). The neighborhood of a state s is denoted by $\mathcal{N}_s$.
- ψ is a *solution* if it is an element of $\tilde{S}$ and satisfies all the problem's requirements. A *multi-dimensional* solution is a solution defined in terms of multiple distinct sequences over the elements of C.
- $J_\psi(L, t)$ is a *cost* associated to each solution ψ. $J_\psi(L, t)$ is a function of all the costs $J_{c_i c_j}$ of all the connections belonging to the solution ψ.

Consider the graph $G = (C, L)$ associated to a given discrete optimization problem instance as defined above. The solutions to the optimization problem can be expressed in terms of feasible paths on the graph G. ACO algorithms can be used to find minimum cost paths (sequences) feasible with respect to the constraints Ω. For example, in the traveling salesman problem, C is the set of cities, L is the set of arcs connecting

0-7803-5536-9/99/$10.00 ©1999 IEEE

cities, and a solution ψ is an Hamiltonian circuit.

ACO algorithms, that is, instances of the ACO meta-heuristic introduced in the following of this section, use a population of ants to collectively solve the optimization problem under consideration by using the above defined graph representation. Information collected by the ants during the search process is stored in *pheromone trails* τ_{ij} associated to connections l_{ij}. (Here and in the following, we simplify notation by referring to $l_{c_i c_j}$ as l_{ij}.) Pheromone trails encode a long-term memory about the whole ant search process. Depending on the problem representation chosen, pheromone trails can be associated to all problem's arcs, or only to some of them. Arcs can also have an associated *heuristic value* η_{ij} representing a priori information about the problem instance definition or run-time information provided by a source different from the ants.

Ants of the colony have the following properties:

- An ant searches for minimum cost feasible solutions $\hat{J}_\psi = \min_\psi J_\psi(L,t)$.
- An ant k has a memory $\mathcal{M}^k$ that it can use to store information on the path it followed so far. Memory can be used to build feasible solutions, to evaluate the solution found, and to retrace the path backward.
- An ant k in state $s_r = \langle s_{r-1}, i\rangle$ can move to any node j in its *feasible neighborhood* $\mathcal{N}_i^k$, defined as $\mathcal{N}_i^k = \{j \mid (j \in \mathcal{N}_i) \wedge (\langle s_r, j\rangle \in \tilde{S})\}$.
- An ant k can be assigned a *start state* s_s^k and one or more *termination conditions* e^k. Usually, the start state is expressed as a unit length sequence, that is, a single component.
- Ants start from the start state and move to feasible neighbor states, building the solution in an incremental way. The construction procedure stops when for at least one ant k at least one of the termination conditions e^k is satisfied.
- An ant k located on node i can move to a node j chosen in $\mathcal{N}_i^k$. The move is selected applying a probabilistic decision rule.
- The ants' probabilistic decision rule is a function of (i) the values stored in a node local data structure $\mathcal{A}_i = [a_{ij}]$ called *ant-routing table*, obtained by a functional composition of node locally available pheromone trails and heuristic values, (ii) the ant's private memory storing its past history, and (iii) the problem constraints.
- When moving from node i to neighbor node j the ant can update the pheromone trail τ_{ij} on the arc (i,j). This is called *online step-by-step pheromone update*.
- Once built a solution, the ant can retrace the same path backward and update the pheromone trails on the traversed arcs. This is called *online delayed pheromone update*.
- Once it has built a solution, and, if the case, after it has retraced the path back to the source node, the ant dies, freeing all the allocated resources.

Although each ant of the colony is complex enough to find a feasible solution to the problem under consideration, good quality solutions can only emerge as the result of the collective interaction among the ants. Each ant makes use only of private information and of information local to the node it is visiting (note that we use the terms node and component, as well as arc and connection/transition, as synonyms), and communication among ants is indirect and is mediated by the information they read/write in the variables storing pheromone trail values. Ants adaptively modify the way the problem is represented and perceived by other ants, but they are not adaptive themselves.

Informally, the behavior of ants in an ACO algorithm can be summarized as follows. A colony of ants concurrently and asynchronously move through adjacent states of the problem by moving through neighbor nodes of G. They move by applying a stochastic local decision policy that makes use of the information contained in the node-local ant-routing tables. By moving, ants incrementally build solutions to the optimization problem. Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution and deposits information about its goodness on the pheromone trails of the connections it used. This pheromone information will direct the search of the future ants.

Besides ants' activity, an ACO algorithm includes two more procedures: *pheromone trail evaporation* and *daemon actions*. Pheromone evaporation is the process by means of which the pheromone trail intensity on the connections automatically decreases over time. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm towards a sub-optimal region. It implements a useful form of *forgetting*, favoring the exploration of new areas of the search space.

Daemon actions, that are an optional component of the ACO meta-heuristic, can be used to implement centralized actions which cannot be performed by single ants. Examples are the activation of a local optimization procedure, or the collection of global information that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from a non-local perspective. As a practical example, the daemon can observe the path found by each ant in the colony and choose to deposit extra pheromone on the arcs used by the ant that made the shortest path. Pheromone updates performed by the daemon are called *off-line pheromone updates*.

In Figure 1 the ACO meta-heuristic is described in pseudo-code. The main procedure of the ACO meta-heuristic manages, via the `schedule_activities` construct, the scheduling of the three above discussed components of an ACO algorithm: ants' generation and activity, pheromone evaporation, and daemon actions. It is important to note that the `schedule_activities` construct does not specify how these three activities are scheduled and synchronized and, in particular, whether they should be executed in a com-

```
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

1  procedure ants_generation_and_activity()
2    while (available_resources)
3      schedule_the_creation_of_a_new_ant();
4      new_active_ant();
5    end while
6  end procedure

1  procedure new_active_ant() {ant lifecycle}
2    initialize_ant();
3    M = update_ant_memory();
4    while (current_state ≠ target_state)
5      A = read_local_ant-routing_table();
6      P = compute_transition_probabilities(A, M, Ω);
7      next_state = apply_ant_decision_policy(P, Ω);
8      move_to_next_state(next_state);
       if (online_step-by-step_pheromone_update)
9        deposit_pheromone_on_the_visited_arc();
10       update_ant-routing_table();
       end if
11     M = update_internal_state();
12   end while
     if (online_delayed_pheromone_update)
13     foreach visited_arc ∈ ψ do
14       deposit_pheromone_on_the_visited_arc();
15       update_ant-routing_table();
16     end foreach
     end if
17   die();
18 end procedure
```

Figure 1: The ACO meta-heuristic. In the `ACO_meta_heuristic()` procedure, the procedure `daemon_actions()` (line 6) is optional. When implemented, it refers to centralized actions executed by a daemon possessing global knowledge. In the `new_active_ant()` procedure, the `target_state` (line 4) refers to a complete solution built by the ant, while the step-by-step and delayed pheromone updating procedures at lines 9-10 and 14-15 are often mutually exclusive. When both of them are absent the pheromone is deposited by the daemon. Comments are enclosed in braces.

pletely parallel and independent way, or if some kind of synchronization among them is necessary. This leaves the designer the freedom to specify the way these three procedures should interact.

Although ACO algorithms are suitable to find minimum cost (shortest) paths on a graph in general, it is important to note that they are an interesting approach only for those shortest path problems to which more classical algorithms like dynamic programming or label correcting methods (Bertsekas, 1995) cannot be efficiently applied. This is the case, for example, for the following types of shortest path problems:

- NP-hard problems, for which the dimension of the full state-space graph is exponential in the dimension of the problem representation. In this case, ants make use of the much smaller graph G, built from the problem's components, and use their memory to generate feasible solutions which in most ACO implementations are then taken to a local optimum by a problem specific local optimizer.
- Those shortest path problems in which the properties of the problem's graph representation change over time concurrently with the optimization process, that has to adapt to the problem's dynamics. In this case, the problem's graph can even be physically available (like in networks problems), but its properties, like the value of connection costs $J_{c_i c_j}(t)$, can change over time. In this case we conjecture that the use of ACO algorithms becomes more and more appropriate as the variation rate of costs $J_{c_i c_j}(t)$ increases and/or the knowledge about the variation process diminishes.
- Those problems in which the computational architecture is spatially distributed, as in the case of parallel and/or network processing. Here ACO algorithms, due to their intrinsically distributed and multi-agent nature that well matches these types of architectures, can be very effective.

In the following section we will consider the application of the ACO-meta-heuristic to two paradigmatic problems belonging to the above defined classes of problems: the traveling salesman problem (TSP) and adaptive routing in communications networks. TSP is the prototypical representative of NP-hard combinatorial optimization problems (Garey & Johnson, 1979) where the problem instance is statically assigned and the information is globally available. On the contrary, in the problem of adaptive routing in communications networks an exogenous process (the incoming data traffic) makes the problem instance change over time, and temporal constraints impose to solve the problem in a distributed way.

3 ACO for the traveling salesman problem

Using the terminology introduced in the previous section, the traveling salesman problem can be defined as follows. Let C be a set of components, representing cities, L be a set of connections fully connecting the elements in C, and $J_{c_i c_j}$ be the cost (length) of the connection between c_i and c_j, that is, the distance between cities i and j. The TSP is the problem of finding a minimal length Hamiltonian circuit on the graph $G = (C, L)$. An Hamiltonian circuit of graph G is a closed tour ψ visiting once and only once all the N_C nodes of G. Its length is given by the sum of the lengths $J_{c_i c_j}$ of all the arcs of which it is composed. Distances may be asymmetric: we have then an asymmetric TSP in which $J_{c_i c_j}$ can be different from $J_{c_j c_i}$. Also, the graph need not be fully connected: if it is not, the missing arcs can be added giving them a very high length.

The traveling salesman problem plays a central role in ant colony optimization because it was the first problem to be attacked by these methods (see (Dorigo, 1992; Dorigo *et al.*, 1991; Dorigo *et al.*, 1996)). Among the reasons the TSP was chosen we find the following ones: (i) it is relatively easy to adapt the ant colony metaphor to it, (ii) it is a very difficult problem (*NP*-hard), (iii) it is one of the most studied problems in combinatorial optimization (Lawler *et al.*, 1985;

Reinelt, 1994), and (iv) it is very easy to state and explain it, so that the algorithm behavior is not obscured by too many technicalities.

In the following we will present Ant System, a paradigmatic example of how ACO algorithms can be applied to the TSP. Extensions of Ant System can be found, for example, in (Bullnheimer *et al.*, 1997b; Dorigo & Gambardella, 1997; Stützle & Hoos, 1998b).

Ant System (AS) can be informally described as follows. A number m of ants is positioned in parallel on m cities. The ants' start state, that is, the start city, can be chosen randomly, and the memory $\mathcal{M}^k$ of each ant k is initialized by adding the current start city to the set of already visited cities (initially empty). Ants then enter a cycle (Figure 1, lines 4 $\rightarrow$ 12 of the `new_active_ant()` procedure) which lasts N_C iterations, that is, until each ant has completed a tour.

During each step an ant located on node i considers the feasible neighborhood, reads the entries a_{ij}'s of the ant-routing table $\mathcal{A}_i$ of node i (Figure 1, line 5 of the `new_active_ant()` procedure), computes the transition probabilities (line 6), and then applies its decision rule to choose the city to move to (line 7), moves to the new city (line 8), and updates its memory (line 11). Note that, because in AS there is no online step-by-step pheromone update, instructions at lines 9 and 10 of the `new_active_ant()` procedure of Figure 1 are not executed.

Once ants have completed a tour (which happens synchronously, given that during each iteration of the while loop each ant adds a new city to the tour under construction), they use their memory to evaluate the built solution and to retrace the same tour backward and increase the intensity of the pheromone trails τ_{ij} of visited connections l_{ij} (Figure 1, lines 13 $\rightarrow$ 16). This has the effect of making the visited connections become more desirable for future ants. Then the ants die, freeing all the allocated resources. In AS all the ants deposit pheromone and no problem-specific daemon actions are performed. The triggering of pheromone evaporation happens after all ants have completed their tours. Of course, it would be easy to add a local optimization daemon action, like a 3-opt procedure (Lin, 1965); this has been done in most of the ACO algorithms for the TSP that have been developed as extensions of AS (see for example (Dorigo & Gambardella, 1997; Stützle & Hoos, 1998a)).

The amount of pheromone trail $\tau_{ij}(t)$ maintained on connection l_{ij} is intended to represent the learned desirability of choosing city j when in city i (which also corresponds to the desirability that arc l_{ij} 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 ψ they produced: the shorter the tour generated by an ant, the greater the amount of pheromone it deposits on the arcs which it used to generate the tour. This choice helps to direct search towards good solutions.

The memory (or internal state) $\mathcal{M}^k$ of each ant contains the already visited cities. The memory $\mathcal{M}^k$ 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 $\mathcal{M}^k$ an ant k can build feasible solutions, that is, it can avoid to visit a city twice. Also, memory allows the ant to compute the length of the tour generated and to cover the same path backward to deposit pheromone on the visited arcs.

The ant-routing table $\mathcal{A}_i = [a_{ij}(t)]$ of node i, where $\mathcal{N}_i$ is the set of all the neighbor nodes of node i, is obtained by the following functional composition of pheromone trails $\tau_{ij}(t)$ and local heuristic values η_{ij}:

$$a_{ij} = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} \quad \forall j \in \mathcal{N}_i \tag{1}$$

where α and β are two parameters that control the relative weight of pheromone trail and heuristic value. The heuristic values used are $\eta_{ij} = 1/J_{c_i c_j}$, where $J_{c_i c_j}$ is the distance between cities i and j. In other words, the shorter the distance between two cities i and j, the higher the heuristic value η_{ij}.

The probability $p_{ij}^k(t)$ with which at the t-th algorithm iteration an ant k located in city i chooses the city $j \in \mathcal{N}_i^k$ to move to is given by the following probabilistic decision rule:

$$p_{ij}^k(t) = \frac{a_{ij}(t)}{\sum_{l \in \mathcal{N}_i^k} a_{il}(t)} \tag{2}$$

where $\mathcal{N}_i^k \subseteq \mathcal{N}_i$ is the feasible neighborhood of node i for ant k (that is, the set of cities ant k has not yet visited) as defined by using the ant private memory $\mathcal{M}^k$ and the problem constraints.

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*, that is, a situation in which all ants make the same tour which, in general, is strongly sub-optimal (Dorigo *et al.*, 1996). An appropriate trade-off has to be set between heuristic value and trail intensity.

After all ants have completed their tour, each ant k deposits a quantity of pheromone $\Delta\tau^k(t) = 1/J_\psi^k(t)$ on each connection l_{ij} that it has used, where $J_\psi^k(t)$ is the length of tour $\psi^k(t)$ done by ant k at iteration t:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau^k(t), \quad \forall\, l_{ij} \in \psi^k(t), \;\; k = 1, \ldots, m \tag{3}$$

where the number m of ants at each iteration is maintained constant) and is set to $m = N_C$. These parameters settings, as well as the settings $\alpha = 1$, $\beta = 5$ and $\rho = 0.5$, were experimentally found to be good by Dorigo (Dorigo, 1992).

This way of setting the value $\Delta\tau^k(t)$ makes it a function of the ant's performance: the shorter the tour done, the greater the amount of pheromone deposited.

After pheromone updating has been performed by the ants[1], pheromone evaporation is triggered: the following rule is applied to all the arcs l_{ij} of the graph G

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) \tag{4}$$

where $\rho \in (0,1]$ is the pheromone trail decay coefficient (the initial amount of pheromone $\tau_{ij}(0)$ is set to a small positive constant value τ_0 on all arcs).

4 ACO for routing in communications networks

The generic routing problem in communications networks can be informally stated as the problem of building and using *routing tables* to direct data traffic so that some measure of network performance, function of the network type and of the services provided, is maximized.

We can use the terminology introduced in Section 2 to give a formal definition of the routing problem. Let the sets C and L correspond respectively to the sets of processing nodes and of communication links of the real network. Let $G = (C, L)$ be a directed graph, where each node in the set C represents a network node with processing/queuing and forwarding capabilities, and each oriented arc in L is a directional transmission system (link). Each link has associated a cost measure defined by its physical properties and crossing traffic flow. 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 towards their destination nodes. The routing table $\mathcal{R}_i = [r_{ijd}]$ of a generic node i, where $\mathcal{N}_i$ is the set of neighbors of i, says to data packets entering node i and directed towards destination node d which should be the next node $j \in \mathcal{N}_i$ to move to. Routing tables are bi-dimensional because the choice of the neighbor node to which a data packet entering a generic node i should be forwarded is a function of the packet destination node d. Ant-routing tables possess the same bi-dimensional structure: pheromone trails associated to each connection are vectors of $N_C - 1$ values. In fact, ant-routing tables, in all the ACO implementations for routing, are used to build the routing tables by means of implementation-dependent transformations. These vectorial pheromone trails are the natural extension of the scalar trails used for the TSP.

Other important differences with the TSP implementation arise from the different nature of the two problems: (i) each ant is assigned a defined pair (s, d) of start-destination nodes and, discovering a path between s and d, the ant builds only a part of the whole problem solution, defined in terms of paths between all the pairs (i, j) in the network, (ii) the costs associated to the connections are not statically assigned: they depend on the connection's physical properties and on the traffic crossing the connection, that interacts recursively with the routing decisions.

In the following subsection we present a simplified version of the AntNet algorithm. A full description of AntNet can be found in (Di Caro & Dorigo, 1998).

4.1 AntNet

In AntNet, each ant searches for a minimum cost path between a pair of nodes of the network. Ants are launched from each network node towards destination nodes randomly selected to match the traffic patterns. Each ant has a source node s and a destination node d, and moves from s to d hopping from one node to the next till node d is reached. When ant k is in node i, it chooses the next node j to move to according to a probabilistic decision rule which is a function of the ant's memory $\mathcal{M}^k$ and of the local ant-routing table $\mathcal{A}_i$.

Pheromone trails are still connected to arcs, but are memorized in variables associated to arc-destination pairs. That is, each directed arc (i, j) has $N_C - 1$ trail values $\tau_{ijd} \in [0,1]$ associated, one for each possible destination node d an ant located in node i can have (therefore, in general, $\tau_{ijd} \neq \tau_{jid}$). Each arc has also associated an heuristic value $\eta_{ij} \in [0,1]$ independent of the final destination. The heuristic values are set to the following values:

$$\eta_{ij} = 1 - \frac{q_{ij}}{\sum_{l \in \mathcal{N}_i} q_{il}} \tag{5}$$

where q_{ij} is the length (in bits waiting to be sent) of the queue of the link connecting node i with its neighbor j.

In AntNet, as well as in most other implementations of ACO algorithms for routing problems, the daemon component (line 6 of the ACO meta-heuristic of Figure 1) is not present.

The local ant-routing table $\mathcal{A}_i$ is obtained by a functional composition of the local pheromone trails τ_{ijd} and heuristic values η_{ij}. 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 can be evaluated by an heuristic function of the trip time T_{sd} and of a local adaptive statistical model maintained in each node. In fact, paths need to be evaluated relative to the network status because a trip time T judged of low quality under low congestion conditions could be an excellent one under high traffic load. Once the generic ant k has completed a path, it deposits on the visited nodes an amount of pheromone $\Delta\tau^k(t)$ proportional to the goodness of the path it built. To this purpose, after reaching its destination node, the ant moves back to its source nodes along the

[1] It should be noted that in the original ant system (Dorigo, 1992; Dorigo *et al.*, 1991) pheromone evaporation was performed before pheromone updating. The algorithm presented here and the original one are exactly the same if the values $\Delta\tau^k(t)$ used in Equation 3 are set to $\Delta\tau^k(t) = 1/((1-\rho) \cdot J^k_\psi(t))$.

same path but backward and using high priority queues, to allow a fast propagation of the collected information.

During this backward path from d to s the ant k increases the pheromone trail value $\tau_{ijd}(t)$ of each connection l_{ij} previously used while it was moving from s to d. The pheromone trail intensity is increased by applying the following rule:

$$\tau_{ijd}(t) \leftarrow \tau_{ijd}(t) + \Delta\tau^k(t) \tag{6}$$

The reason the ant updates the pheromone trails during its backward trip is that, before it can compute the amount of pheromone $\Delta\tau^k(t)$ to deposit on the visited arcs it needs to complete a path from source to destination to evaluate it.

After the pheromone trail on the visited arc has been updated, the pheromone value of all the outgoing connections of the same node i, relative to the destination d, evaporates:[2]

$$\tau_{ijd}(t) \leftarrow \frac{\tau_{ijd}(t)}{(1 + \Delta\tau^k(t))}, \ \forall j \in \mathcal{N}_i \tag{7}$$

where $\mathcal{N}_i$ is the set of neighbors of node i.

As we said, AntNet's ant-routing table $\mathcal{A}_i = [a_{ijd}(t)]$ of node i is obtained, as usual, by the composition of the pheromone trail values with the local heuristic values. This is done as follows:

$$a_{ijd}(t) = \frac{w\tau_{ijd}(t) + (1-w)\eta_{ij}}{w + (1-w)(|\mathcal{N}_i| - 1)} \tag{8}$$

where $j \in \mathcal{N}_i$, d is the destination node, $w \in [0,1]$ is a weighting factor and the denominator is a normalization term.

The ants decision rule is then defined as follows. Let, at time t, ant k be located on node i and be directed towards node d. If $\mathcal{N}_i \not\subseteq \mathcal{M}^k$, that is, if there is at least one city in the ant's current location neighborhood that ant k has not visited yet, then the ant chooses the next node $j \in \mathcal{N}_i$ with probability

$$p^k_{ijd}(t) = \begin{cases} a_{ijd}(t) & \text{if } j \notin \mathcal{M}^k \\ 0 & \text{if } j \in \mathcal{M}^k \end{cases} \tag{9}$$

otherwise, the ant chooses a city $j \in \mathcal{N}_i$ with uniform probability: $p^k_{ijd}(t) = 1/(|\mathcal{N}_i|)$. (Note that in AntNet, differently from what happens in Ant System, the neighborhood and the feasible neighborhood are the same, that is, $\mathcal{N}^k_i \equiv \mathcal{N}_i$.)

In other words, ants try to avoid cycles (Equation 9) but, in the case all the nodes in i's neighborhood have already been visited by the ant, the ant has no choice and it has to re-visit a node, generating in this way a cycle. In this case the generated cycle is deleted from the ant memory, that forgets completely about it. Considering the stochasticity of the decision rule and the evolution in the traffic conditions, it is very unlikely that the ant repeats the same cycle over and over again.

[2] In this case the decay factor is chosen so that it operates a normalization of the pheromone values which continue therefore to be usable as probabilities.

5 Other ACO meta-heuristic applications

There are now available numerous successful implementations of the ACO meta-heuristic applied to a number of different combinatorial optimization problems.

The most widely studied problems using ACO algorithms are the traveling salesman problem and the quadratic assignment problem (QAP).

Results obtained by the application of ACO algorithms to the TSP are very encouraging (see (Stützle & Dorigo, 1999b) for an overview of applications of ACO algorithms to the TSP): they are often better than those obtained using other general purpose heuristics like evolutionary computation or simulated annealing. Also, when adding to ACO algorithms rather unsophisticated local search procedures based on 3-opt (Lin, 1965), the quality of the results obtained (Dorigo & Gambardella, 1997) is close to that obtainable by more sophisticated methods. More research will be necessary to assess whether ACO algorithms can reach the performance of state-of-the-art algorithms like Iterated Lin-Kernighan (Johnson & McGeoch, 1997).

As in the TSP case, the first application of an ACO algorithm to the QAP has been that of Ant System (Maniezzo *et al.*, 1994). In the last two years several ACO algorithms for the QAP have been presented by Maniezzo and Colorni (Maniezzo & Colorni, 1999, in press), Maniezzo (Maniezzo, 1998), and Stützle (Stützle & Hoos, 1998a). Currently, ACO algorithms are among the best heuristics for attacking real-life QAP instances[3]. We refer to (Stützle & Dorigo, 1999a) for an overview of these approaches.

The sequential ordering problem is closely related to the asymmetric TSP, but additional precedence constraints between the nodes have to be satisfied. Gambardella and Dorigo have tackled this problem by an extension of ACS enhanced by a local search algorithm (Gambardella & Dorigo, 1997). They obtained excellent results and were able to improve the best known solutions for many benchmark instances.

A first application of AS to the job-shop scheduling problem has been presented in (Colorni *et al.*, 1994). Despite showing the viability of the approach, the computational results are not competitive with state-of-the-art algorithms. More recently, $\mathcal{MM}$AS has been applied to the flow shop scheduling problem (FSP) in (Stützle, 1998). The computational results have shown that $\mathcal{MM}$AS outperforms earlier proposed Simulated Annealing algorithms, while it performs comparably to Tabu Search. Another very recent application is that to the single machine total tardiness problem, discussed in a paper published in these same proceedings (Bauer *et al.*, 1999). Again, the results obtained are very promising.

Costa and Herz (Costa & Hertz, 1997) have proposed an extension of AS to assignment type problems and present an application of their approach to the graph coloring prob-

[3] Another very good algorithm inspired by the ant colony metaphor is HAS-QAP (Gambardella *et al.*, 1997). It is not listed with the others above because it doesn't follow the ACO meta-heuristic.

lem obtaining results comparable to those obtained by other heuristic approaches.

An application of AS_{rank} to the capacitated vehicle routing problem is presented by Bullnheimer, Hartl, and Strauss (Bullnheimer *et al.*, 1997a). They obtained good computational results for standard benchmark instances, slightly worse than the best performing Tabu Search algorithms. A recent application by Gambardella, Taillard, and Agazzi to vehicle routing problems with time windows improves the best known solutions for some benchmark instances (Gambardella *et al.*, 1999).

Michel and Middendorf (Michel & Middendorf, 1998; Michel & Middendorf, 1999) have proposed an application of AS to the shortest common supersequence. They introduce the use of a lookahead function during the solution construction by the ants. They also present a parallel version of their algorithm based on the island model typical of many parallel genetic algorithms implementations.

$\mathcal{MM}$AS has recently been applied to the generalized assignment problem by Ramalhinho Lourençou and Serra (Lourenço & Serra, 1998), obtaining very promising computational results. In particular, their algorithm is shown to find optimal and near optimal solutions faster than a GRASP algorithm which was used for comparison.

Another very recent application of the ACO meta-heuristic is the one to the multiple knapsack problem proposed in one of the articles published in these same proceedings (Leguizamón & Michalewicz, 1999). Also in this case the obtained results are very promising.

Perhaps the currently most active research area in ACO algorithms is the study of their application to telecommunication network problems, in particular to network routing (Schoonderwoerd *et al.*, 1996; Di Caro & Dorigo, 1998). The application of ACO algorithms to network optimization problems is appealing because these problems have characteristics like distributed information, non-stationary stochastic dynamics, and asynchronous evolution of the network status which well match some of the properties of ACO algorithms like the use of local information to generate solutions, indirect communication via the pheromone trails and stochastic state transitions. A detailed overview of routing applications of ACO algorithms can be found in (Dorigo *et al.*, 1998).

6 Conclusions

In this paper we have given a formal description of the Ant Colony Optimization meta-heuristic, as well as of the class of problems to which it can be applied. We have then described two paradigmatic applications of ACO algorithms to the traveling salesman problem and to routing in packet-switched networks, and concluded with a short overview on the currently available applications. Ongoing work follows three main directions: the study of the formal properties of a simplified version of ant system, the development of AntNet for Quality of Service applications, and the development of further applications to combinatorial optimization problems.

7 Acknowledgments

Marco Dorigo is a Research Associate of the Belgian FNRS. Gianni Di Caro was supported by a Madame Curie Fellowship (CEC-TMR Contract N. ERBFMBICT 961153).

References

A. Bauer, B. Bullnheimer, R. F. Hartl, & C. Strauss. 1999. An Ant Colony Optimization Approach for the Single Machine Total Tardiness Problem. *In: Proc. of 1999 Congress on Evolutionary Computation.* IEEE Press.

D. Bertsekas. 1995. *Dynamic Programming and Optimal Control.* Athena Scientific.

B. Bullnheimer, R. F. Hartl, & C. Strauss. 1997a. *An Improved Ant System Algorithm for the Vehicle Routing Problem.* Tech. rept. POM-10/97. Institute of Management Science, University of Vienna, Austria. Accepted for publication in the *Annals of Operations Research.*

B. Bullnheimer, R. F. Hartl, & C. Strauss. 1997b. *A New Rank-Based Version of the Ant System: A Computational Study.* Tech. rept. POM-03/97. Institute of Management Science, University of Vienna, Austria. Accepted for publication in the *Central European Journal for Operations Research and Economics.*

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

D. Costa, & A. Hertz. 1997. Ants Can Colour Graphs. *Journal of the Operational Research Society*, **48**, 295–305.

G. Di Caro, & M. Dorigo. 1998. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research (JAIR)*, **9**, 317–365.

G. Di Caro, & M. Dorigo. 1998. Two Ant Colony Algorithms for Best-Effort Routing in Datagram Networks. *Pages 541–546 of: Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98).* IASTED/ACTA Press.

M. Dorigo. 1992. *Optimization, Learning and Natural Algorithms* (in Italian). Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.

M. Dorigo, & L. M. Gambardella. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, **1**(1), 53–66.

M. Dorigo, G. Di Caro, & L. M. Gambardella. 1998. *Ant Algorithms for Discrete Optimization.* Tech. rept.

IRIDIA/98-10. Université Libre de Bruxelles, Belgium. Accepted for publication in *Artificial Life*.

M. Dorigo, V. Maniezzo, & A. Colorni. 1991. *Positive feedback as a search strategy*. Tech. rept. 91-016. Dipartimento di Elettronica, Politecnico di Milano, Italy.

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

L. M. Gambardella, & M. Dorigo. 1997. *HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem*. Tech. rept. IDSIA-11-97. IDSIA, Lugano, Switzerland.

L. M. Gambardella, E. D. Taillard, & M. Dorigo. 1997. *Ant Colonies for the QAP*. Tech. rept. IDSIA-4-97. IDSIA, Lugano, Switzerland. Accepted for publication in the *Journal of the Operational Research Society (JORS)*.

L. M. Gambardella, E. D. Taillard, & G. Agazzi. 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*. McGraw-Hill.

M. R. Garey, & D. S. Johnson. 1979. *Computers and Intractability*. W.H. Freeman and Company.

D. S. Johnson, & L. A. McGeoch. 1997. The Traveling Salesman Problem: A Case Study. *Pages 215–310 of:* E.H. Aarts, & J. K. Lenstra (eds), *In Local Search in Combinatorial Optimization*. Chichester: John Wiley & Sons.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, & D. B. Shmoys (eds). 1985. *The Travelling Salesman Problem*. Wiley.

G. Leguizamón, & Z. Michalewicz. 1999. A New Version of Ant System for Subset Problems. *In: Proc. of 1999 Congress on Evolutionary Computation*. IEEE Press.

S. Lin. 1965. Computer Solutions of the Traveling Salesman Problem. *Bell Systems Journal*, **44**, 2245–2269.

H. Ramalhinho Lourenço, & D. Serra. 1998 (May). *Adaptive Approach Heuristics for the Generalized Assignment Problem*. Economic Working Papers Series 304. Universitat Pompeu Fabra, Dept. of Economics and Management, Barcelona, Spain.

V. Maniezzo. 1998. *Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem*. Tech. rept. CSR 98-1. C. L. in Scienze dell'Informazione, Universitá di Bologna, Italy.

V. Maniezzo, & A. Colorni. 1999, in press. The Ant System Applied to the Quadratic Assignment Problem. *IEEE Trans. Knowledge and Data Engineering*.

V. Maniezzo, A. Colorni, & M. Dorigo. 1994. *The Ant System Applied to the Quadratic Assignment Problem*. Tech. rept. IRIDIA/94-28. Université Libre de Bruxelles, Belgium.

R. Michel, & M. Middendorf. 1998. An Island Model based Ant System with Lookahead for the Shortest Supersequence Problem. *Pages 692–701 of:* A. E. Eiben, T. Back, M. Schoenauer, & H.-P. Schwefel (eds), *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*. Springer-Verlag.

R. Michel, & M. Middendorf. 1999. An ACO Algorithm for the Shortest Common Supersequence Problem. *In:* D. Corne, M. Dorigo, & F. Glover (eds), *New Ideas in Optimization*. McGraw-Hill.

G. Reinelt. 1994. *The Traveling Salesman Problem: Computational Solutions for TSP Applications*. Berlin: Springer-Verlag.

R. Schoonderwoerd, O. Holland, J. Bruten, & L. Rothkrantz. 1996. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, **5**(2), 169–207.

T. Stützle. 1998. Parallelization Strategies for Ant Colony Optimization. *Pages 722–731 of:* A. E. Eiben, T. Back, M. Schoenauer, & H.-P. Schwefel (eds), *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*. Springer-Verlag.

T. Stützle, & M. Dorigo. 1999a. ACO Algorithms for the Quadratic Assignment Problem. *In:* D. Corne, M. Dorigo, & F. Glover (eds), *New Ideas in Optimization*. McGraw-Hill.

T. Stützle, & M. Dorigo. 1999b. 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: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*. John Wiley & Sons.

T. Stützle, & H. Hoos. 1998a. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System and Local Search for Combinatorial Optimization Problems. *Pages 313–329 of:* S. Voß, S. Martello, I.H. Osman, & C. Roucairol (eds), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academics, Boston.

T. Stützle, & Holger Hoos. 1998b. Improvements on the Ant System: Introducing the $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Pages 245–249 of:* R.F. Albrecht G.D. Smith, N.C. Steele (ed), *Artificial Neural Networks and Genetic Algorithms*. Springer Verlag, Wien New York.