

Theory and practice of Ant Colony Optimization for routing in dynamic telecommunications networks *

Gianni A. Di Caro, Frederick Ducatelle and Luca M. Gambardella

“Dalle Molle” Institute for Artificial Intelligence Studies (IDSIA)
Lugano, Switzerland

`{gianni,fred,luca}@idsia.ch`

1 Introduction

Telecommunications networks are becoming increasingly large, dynamic, and heterogeneous. The global Internet is rapidly evolving toward a highly complex system which comprises and integrates a number of wired and wireless networks covering the needs of different community of users and ranging from small body area networks to global satellite networks. It is just matter of time before the “*all the time, everywhere access*” foreseen in the view of pervasive computing [45] will be put into practice allowing a continual access to data, users, and services. Core features of current and forthcoming network scenarios are: the continual mobility and appearance/disappearance of users, the continual updating, migration and reorganization of available data and services, the tight interaction between wireless and wired communication nodes with totally different characteristics in terms of capacity and delivered quality of service. It is apparent that control and management of network systems with such characteristics is a highly complex task. Existing protocols and algorithms were conceived to deal with rather static and homogeneous network environments, and to resort on human intervention in case of major problems and tuning. Therefore, a sort of paradigm shift is necessary to deal with the challenges posed by extremely large, dynamic, and heterogeneous networks. Novel control and management protocols have to be designed which show the following essential properties:

- *adaptivity* to changes (in traffic, topology, services, etc.);
- *robustness* to component failures, transmission errors, and small perturbations;
- *self-organizing* and *decentralized* behavior;
- *scalability* in terms of performance versus overhead and usage of network resources;
- ability to work across (sub)networks which are *heterogeneous* in terms of transmission technology, and/or node and topology characteristics;

*Draft version of the chapter appeared in the book: Sala N., Orsucci F. (Eds.), *Reflecting Interfaces: the Complex Coevolution of Information Technology Ecosystems*, Idea Group, Hershey, PA, USA, 2008

- *self-** behavior, that is, being able to self-tune internal parameters, self-manage, self-configure, self-govern, etc.

The central idea contained in this “wish list” is that novel algorithms should continually and robustly *learn* about the current network status and user context, and accordingly adapt their internal characteristics and decision policies. In a broad sense, this is the approach advocated, with different perspectives, in the two recent frameworks of *traffic engineering* [1] and *autonomic communications* [37]. Where for the latter, the term “autonomic” precisely stresses the need for a design that lets the different network components being fully autonomous and control and manage the whole network through self-organizing social interaction. The central aim is to minimize human intervention while maximizing the overall performance and efficiency.

The design from scratch of novel protocols and algorithms with the listed properties is in general a very challenging task, also considering the rather large number of different network systems with different characteristics which are nowadays available (e.g., wired wide area networks, Wi-Fi networks, wireless mesh networks, sensor networks, etc.). On the other hand, we can draw some basic inspiration from *biological systems*. In fact, a number of systems that can be observed in Nature precisely show the general properties we wish to have in a network control system. As a matter of fact, biological systems have evolved the ability to effectively adapt both in terms of structure and behavior to constantly changing environments. Most of these systems can be seen as composed of a large number of dynamic, autonomous, and distributed units which generate a variety of useful effective behaviors at the system-level as a result of local interactions and self-organization. Moreover, biological systems are usually robust to internal perturbations or loss of units, and are able to survive and evolve over a wide range of different environments. Nature has already served as source of inspiration for a number of successful algorithms and frameworks. For instance, neural networks [35, 6] have been originally designed after brain’s neurons, evolutionary computation [33, 27] stems from the observation of the processes underlying evolution and natural selection, swarm intelligence [7] finds its roots in both the swarm and social behaviors of groups of animals.

In this chapter we show that taking basic inspiration from Nature’s complex adaptive systems to design novel *routing* algorithms to tackle the complexity of modern networks is a general and effective way to proceed. In particular, we discuss *Ant Colony Optimization (ACO)* [18, 17, 11, 20], a combinatorial optimization framework that reverse-engineers and formalizes the basic mechanisms at work in a *shortest-path behavior* observed in *ant colonies* [30, 10]. It has been observed that ants in a colony are able to converge on the shortest among multiple paths connecting their nest and a food source. The driving force behind this behavior is the use of a volatile chemical substance called *pheromone*. While moving, ants lay pheromone on the ground, and they also go in the direction of higher pheromone intensities. This mechanism allows to implicitly mark paths to guide subsequent ants, and let good paths arise from the overall behavior of the colony. Details about the mechanism are explained further on.

ACO is based on the *iterative construction of multiple solutions* for the optimization problem under consideration. The aim is to learn about the characteristics and the regularities of the search space, and recursively use this knowledge to direct the solution construction processes. These solution construction processes are defined in the terms of *sequential decision processes*, and are driven by a *parametric stochastic decision policy* π . ACO’s strategy consists in the *progressive learning of the parameters* used by this decision policy in order to eventually discover a global policy that can allow the generation of good/optimal solutions to the problem at hand. Solution

construction processes are designed after ant behavior and are termed *ants*, while the parameters that are the object of learning play the role of pheromone, and are termed *pheromone variables* (see Figure 1).

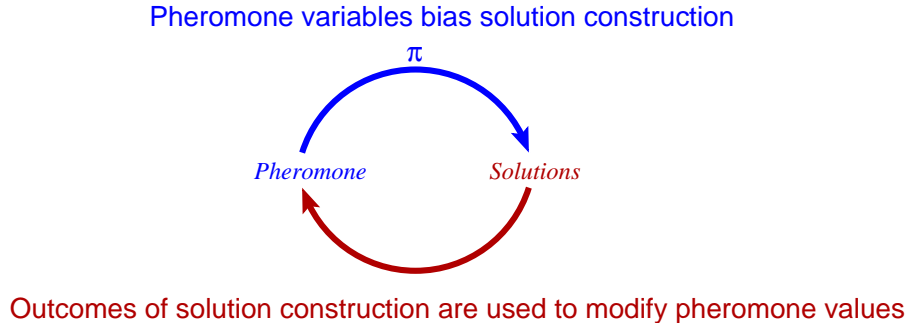


Figure 1: Circular relationship between repeated solution generation and policy adaptation.

ACO’s recipe has been applied with success to a number of combinatorial optimization problems. For many problems of both practical and theoretical interest ACO implementations represent the state-of-the-art. Its general application to *routing problems* is a relatively straightforward task due to a somehow natural mapping between the characteristics of ACO and those of typical network routing tasks. As a matter of fact most of ACO implementations for routing share common structure and strategies. Each ant is actually a lightweight mobile agent (usually implemented as an “intelligent” control packet) responsible for discovering a routing path between the generating source node and an assigned destination nodes. Ants actively explore the network, collect useful information along the followed path (e.g., about the congestion status), and use in turn this information to learn about the network dynamics and to continually update the node routing policy, encoded in pheromone variables, to track network changes.

Based on the common underlying structure and set of strategies that can be singled out from actual ACO-based routing algorithms, in this chapter we informally define the *Ant Colony Routing (ACR)* framework. ACR is a high-level distributed control architecture that specializes the core ideas of the ACO approach to the specific case of network routing and, at the same time, provides a generalization of these same ideas to define a fully distributed architecture for autonomic routing. We show by means of a general discussion and a concrete example of an ACR algorithm, that following the ACR guidelines it is relatively straightforward to design novel routing algorithms that possess most of the characteristics reported in the wish list made up earlier on. Moreover, we show that the same core ideas can be applied with success over a range of dynamic network scenarios obtaining state-of-the-art performance.

The rest of this chapter is organized as follows. We first provide some general background on the class of problems and on the methods discussed throughout the chapter. In particular, we provide a general discussion about the problem of routing in networks (Section 2), we describe the shortest path behavior in ant colonies in Nature and single out all the basic mechanisms at work (Section 3.1), and we report the definition of the ACO metaheuristic and a discussion of its properties (Section 3.2). Then, we concentrate on the application of ACO to routing in networks (Section 4). First, we provide the general definition of ACR and discuss its characteristics (Subsection 4.2). Next, we describe the practical implementation of *AntHocNet*, an ACO/ACR algorithm for routing in the extremely dynamic *mobile ad hoc networks* (Subsection 4.2). An-

tHocNets characteristics are described and a number of experimental results over several network scenarios are reported in order to provide a sound validation of the ACR approach. We conclude the chapter with a summary of its contents and a discussion of future developments (Section 5).

2 Generalities on network routing

In this section we discuss the general characteristics of network routing. The reader can find more detailed and insightful discussions on routing in [5, 50, 47, 53, 44, 11].

A telecommunication network can be represented as a directed weighted graph $G = (V, E)$, where each node in the set V represents a *processing and forwarding unit* and each edge in E is a *transmission system* with some capacity/bandwidth and propagation characteristics. Two nodes are adjacent in the graph if they can communicate directly by means of wired or wireless transmission, and are called *neighbors*. *Data traffic* originates from a *source* node $s \in V$ and is directed to a set $d \subseteq V$ of *destination* nodes. In this chapter we consider only the case of a unique destination node for each source, which is called *unicast traffic*. The routing component of a network control system addresses the basic issue of finding and setting up feasible paths to forward incoming data packets from source to destination nodes. In packet-switched networks (e.g., the Internet), data packets are independently forwarded at each node according to a local *routing decision policy* parametrized by a local data structure called *routing table* which holds routing information. In the general case, entries in the routing table associate for each destination of interest and for each neighbor a measure of the cost of reaching the destination through the neighbor. In this sense, a network routing system can be properly seen as a *distributed decision system*.

A good routing strategy is one that selects the feasible paths optimizing the overall network performance, which is usually expressed in terms of maximizing the number of delivered data packets while minimizing their end-to-end latency and inter-arrival jitter time. Optimized path selection is a complex task since it involves the concurrent optimization of multiple and often conflicting objectives and requires to take into account the dynamic evolution of the network in terms of input traffic patterns and network characteristics like topology and link transmission delays. When all this knowledge is available the problem of routing assignments can be in principle solved to optimality following *optimal routing* [28, 5] approaches, which look at routing as a multicommodity flow problems [41]. Unfortunately, in most of the practical cases of interest, a perfect/reliable knowledge about the dynamics of input traffic and network structure and characteristics (e.g., due to user mobility and hardware failures) is not available. Therefore, from one hand, the routing decision policy at the nodes should be dynamically updated to keep tracking of network changes. From the other hand, updating can only be based on some imperfect knowledge about current and future status of the network. In *dynamic* (or *adaptive*) routing strategies, this knowledge is collected online at the nodes (e.g., by monitoring the congestion level of the connected links) and possibly shared among the nodes, and it is used to *automatically adapt* the local routing policies to changing network conditions. On the other hand, in *static* (or *oblivious*) routing systems, routing paths are determined without regard to the current network state. Paths are usually chosen as the result of some offline optimization process based on prior knowledge and are statically assigned and used. Adaptive routers are, in principle, the most attractive ones. As a drawback, they can cause oscillations and inconsistencies in the selected paths, and, in turn, these can cause, circular paths, as well as large fluctuations in measured performance. Moreover, the general non stationarity of the network environment makes parameter setting a challenging task.

As a matter of fact, the great majority of the routing algorithms at work in deployed networks are basically static with respect to traffic variations and are adaptive with respect to topological variations. For instance, this is the case of popular intra-domain protocols like OSPF [39] and RIP [38] which are at the very core of the wired Internet. Other intra-domain protocols like CISCO's EIGRP come with only simple traffic adaptivity schemes. This is justified by the fact that *topological adaptivity* is an essential property to guarantee the basic functioning of any network. On the other hand, adaptivity to other aspects of the network environment can be more regarded as performance/resources optimization, and since in a sense it is a harder task than topological adaptivity, it is usually not included in the protocols used in real-world applications. Often, boost up of performance is obtained not by injecting "intelligence" into the network but rather at the expenses of adding more/redundant resources. However, this is way of proceeding by over-dimensioning is not always possible (e.g., like in the case of mobile ad hoc networks [44], which are generated on-the-fly), it can happen only on a slow time scale, and it requires a major economical investment.

The ACO-based approach which is the focus of this chapter shows that it is actually possible to obtain both traffic and topological adaptivity in a very efficient, robust, and scalable way through a totally distributed and self-organizing strategy.

3 The Ant Colony Optimization (ACO) metaheuristic

3.1 Biological inspiration: The shortest path behavior of ant colonies

A lot of species of ants have a trail-laying/trail-following behavior when *foraging* [34] for food. While moving, individual ants deposit on the ground a volatile chemical substance called *pheromone*, forming in this way *pheromone trails*. Ants can smell pheromone and, when choosing their way, they tend to choose, in probability, the paths marked by stronger pheromone concentrations. In this way, the pheromone distribution modifies the way the environment is perceived by the ants acting as a sort of attractive field. This evolved characteristics of ant colonies is useful to find the way back to nest and food sources, to find the location of food sources discovered by nest-mates, and to recruit other ants to carry out cooperative tasks where it is necessary. Between the end of the 1980's and the beginning of the 1990's, a group of researchers of the Université Libre de Bruxelles, in Brussels, Belgium (S. Aron, R. Beckers, J.-L. Deneubourg, S. Goss and J.-M. Pasteels), ran several experiments and obtained original theoretical results concerning the influence of the pheromone fields on the ant decision patterns. These works seemed to indicate that pheromone acts as a sort of *dynamic collective memory* of the colony, a repository of all the most recent "foraging experiences" of the ants belonging to the same colony. By continually updating and sensing this chemical repository the ants can indirectly communicate and influence each other through the environment. Indeed, this basic form of indirect communication termed *stigmergy* [31, 51, 18], coupled with a form of positive feedback, can be enough to allow the colony *as a whole* to discover, when only few alternative paths are possible, the *shortest path* connecting a source of food to the colony's nest.

This fact can be understood considering the following binary bridge experiment described in [30] (see Figure 2). The nest of a colony of Argentine ants *Linepithema humile* and a food source have been separated by a diamond-shaped double bridge with branches of different length. Ants are then left free to move between the nest and the food source. The percentage of ants which choose one or the other of the two branches is observed over time. The experimental observation is that, after a transitory phase which can last a few minutes, most of the ants use the shortest

branch. It is also observed that the colony's probability of selecting the shortest path increases with the difference in length between the long and the short branches. In fact, the first ants able to arrive at the food source are those that traveled along the shortest branch. Accordingly, the pheromone that these same ants have laid on the shortest branch while moving *forward* towards the food source makes this branch marked by more pheromone than the longest one. The higher levels of pheromone present on the shortest branch stimulate these same ants to probabilistically choose again the shortest branch when moving *backward* to their nest. This recursive behavior can be thoroughly described as a self-sustaining *positive feedback effect* because the very fact of choosing a path increases its probability of being chosen again in the near future. During the backward journey, additional pheromone is released on the shortest path. In this way, pheromone is laid on the shortest branch at a *higher rate* than on the longest branch. This reinforcement of the pheromone intensity on the shorter path is the result of a form of *implicit path evaluation*: the shorter path is completed earlier than the longer one, and therefore they receive pheromone reinforcement more quickly. Therefore, for a same number of ants choosing either the shortest or the longest branch at the beginning (Figure 2a), since the pheromone on the shortest branch is accumulated at a higher rate than on the longest one, the choice of the shortest branch becomes more and more attractive for the subsequent ants at both the decision points. The result (see Figure 2b and 2c) is that after an initial transitory phase lasting few minutes during which some oscillations can appear, ants tend to converge on the same shorter path.

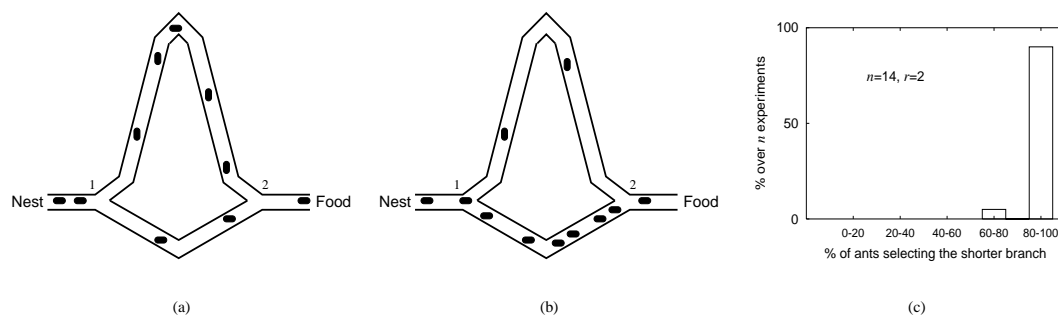


Figure 2: Effect of laying/following pheromone trails in a colony of Argentine ants *Linepithema humile* crossing an asymmetric bridge. Modified from [30].

According to the just described experiment, the ability of ant colonies to select shortest paths can be understood as the result of the synergistic interaction among a number of elements such as:

- a population (colony) of foraging ants,
- forward-backward path following,
- step-by-step laying and sensing of pheromone,
- sequence of stochastic decisions biased by local pheromone intensity,
- positive feedback,
- (implicit) path evaluation,
- iteration over time.

In more computational terms the ants in the colony can be seen as *minimalist autonomous agents* that act in a completely *asynchronous, concurrent* and *distributed* fashion to *collectively* solve a shortest path problem. Stigmergic communication makes the ant system *self-organizing*. Multiple paths (solutions) are repeatedly tried out moving back and forth and some information related to each followed path is released on the environment, encoded in the pheromone trails, representing a *shared distributed memory* of the system. In turn, the *local* content of this memory affects the stochastic decisions of the ants, such that, when there is a significant difference in the lengths of the possible paths, implicit *path evaluation* gets at work and, coupled with positive feedback, results in a distributed and collective *path optimization* mechanism. Given enough time (depending on the number of ants, length and relative length difference of the paths, and other factors), this can result in the convergence of all the ants in the colony on the shortest among the possible paths. Each ant gives a contribution to the overall behavior. But, although a single ant is capable of building a “solution” (i.e., finding a path between its nest and a food reservoir), is only the simultaneous presence and synergistic action of an ensemble of ants that makes possible the shortest path finding behavior (i.e., the convergence to the shortest path), which is a property of the colony and of the concurrent presence of all the discussed ingredients, and not a property of the single ant.

All these mechanisms have been abstracted, reverse-engineered, and put to work almost as they are in *ant colony optimization*, naturally resulting in a general framework for the design of *robust, distributed and adaptive multi-agent system* for the solution of *shortest path problems*. Indeed, this class of problems is a very important one and encompasses a vast number of other problems (e.g, see [4]). Graphs whose nodes represent possible alternatives/states and whose edges represent distances/losses/rewards/costs associated to node transitions are graphical models for a huge number of practical and theoretical decision and optimization problems. In general, almost any combinatorial optimization, network flow or sequential decision problem can modeled in the terms of a shortest path problem. Having in the hands an effective procedure to handle this class of problems opens endless opportunities for applications, especially considering the distributed, adaptive and self-organizing properties of the approach.

3.2 ACO: definition and general properties

ACO is a metaheuristic for combinatorial optimization problems. An instance of a *combinatorial optimization problem* is a pair (S, J) , where S is a finite set of feasible solutions and J is a function that associates a real *cost* to each feasible solution, $J : S \rightarrow \mathbf{R}$. The problem consists in finding the element $\sigma^* \in S$ which minimizes the function J :

$$\sigma^* = \arg \min_{\sigma \in S} J(\sigma). \quad (1)$$

It is common practice to define an instance of a combinatorial problem using a more compact definition of the type: $\langle C, \Omega, J \rangle$. C is a finite set of components, $\{c_0, c_1, \dots, c_n\}$, used to define a solution. Ω is a set of mathematical relations defining *constraints* on the way C 's elements have to be selected in order to produce a feasible solution $\sigma \in S$. For the class of problems we are interested in this chapter, namely routing in telecommunications networks, the set S is represented by the set of all possible node paths joining each pair (s, d) of source and destination nodes involved in data exchange. The component set C is the set of all nodes belonging to the network, while Ω defines that consecutive nodes in a solution path must be able to communicate (i.e., a wired or a wireless physical transmission channel must allow data exchange between the

two nodes), and that no infinite loops must be present. The cost of a path is the sum of the costs $\mathcal{J}(c_t, c_{t+1})$ (e.g., time delays) associated to the transmission between two adjacent nodes c_t and c_{t+1} belonging to the path. In the following we restrict the discussion to the class of combinatorial problems whose costs are additive in this way (e.g., this is true for important problems such as the traveling salesman problem [29]).

A combinatorial optimization problem can be approached with a variety of different solution methods, each having different properties in terms of the optimality of the produced solution, and in terms of the time required to reach this solution [29, 52]. For important classes of problems, like the NP-hard ones, the time required for an exact method to produce an optimal solution is prohibitive for large instances, and it is therefore often necessary in practice to rely on heuristics. These are solution methods that attempt to provide a near-optimal solution in reasonable time without any formal guarantee on performance. ACO is what is called a *metaheuristic*. That is a high-level strategy that guides the development of heuristics. A metaheuristic can be seen as a general algorithmic framework that can be applied to a possibly wide range of different optimization problems with relatively few modifications.

ACO's heuristic recipe is the following. Reverse-engineering the mechanisms at work in ant colonies, ACO is based on the *repeated sampling of multiple solutions* to the problem at hand and on the use of the outcomes of these solutions to update the value of variables, playing the role of pheromone in ant colonies, that bias in turn the process of solution generation. The aim is to progressively learn an assignment to these pheromone values that is able to capture regularities associated to good solutions and eventually allows the generation of optimal/near-optimal solutions to the problem under consideration.

The precise characterization of what pheromone variables represent is related to fact that in ACO the single solutions are *incrementally constructed*. That is, starting from an empty set $x_0 = \{\}$, a complete feasible solution $x_s \in S$ is built stepwise by adding one new component $c_i \in C$ at-a-time. This way of proceeding means that each solution is the result of a sequence of decisions about the next component to include into the current partial solution, represented by the sequence $x_t = [c_0, c_1, \dots, c_t]$. Therefore, the generation of a solution coincide with the output of a *sequential decision process*. This process actually mimics ant behavior building a path from the nest to a source of food. Accordingly, we speak of these processes in terms of ant-like agents, or, shortly, (artificial) *ants*.

As in the ant colony case, the single decisions are issued according to a *stochastic decision policy* π parametrized by a set of real-valued variables τ , called *pheromone* variables. A policy is a rule that associates the *state* of a process to an action, selected among those actions which are feasible in that state, that is, according to the constraints Ω in our case. A stochastic policy defines a distribution of probability over the feasible actions, assigning in practice a selection probability to each feasible action. Then an action is actually selected according to a chosen random scheme that takes into account the different probability values (e.g., a random proportional scheme selects proportionally to the probability values).

In many ACO implementations, the model used to represent the decision process is such that the states of the process are mapped one-to-one onto the components of the optimization problem. This means that at step t the decision policy selects the next component $c_{t+1} \in C$ conditionally to the fact of being in state c_t , and among the subset of components which are still feasible. After issuing the decision, the current state becomes c_{t+1} .

Following this approach, whereby optimized complete solutions result from a sequence of single decisions about movements between states (or problem components), it is important to have

a way to estimate the quality of each *state transition*. This is the role of the pheromone variables τ , which are used in ACO to associate a real-valued *quality* with each state transition. In practice, τ_{ij} represents a measure of the desirability of having the pair (c_i, c_j) in the solution sequence in the perspective of eventually building a good complete solution. For instance, in the case of routing, τ_{ij} represents the desirability of moving from c_i to c_j in order to reach destination d in the perspective of optimizing the final path from the source node s to d (e.g., obtaining a path with very low time latency for transmitting data packets from s to d). The τ values are the parameters of the decision policy. They are used to calculate the relative probability of each feasible decision by normalization: if $\mathcal{N}(c_i)$ is the set of components still feasible in state c_i given the current partial solution, then the probability of each $c_j \in \mathcal{N}(c_i)$ is calculated as:

$$P_{ij} = \frac{\tau_{ij}}{\sum_{c_j \in \mathcal{N}(c_i)} \tau_{ij}} . \quad (2)$$

The stochastic policy π takes in input the probabilities P_{ij} and use them to select the next component according to some random selection rule.

The ACO's fingerprint is that pheromone quality estimates biasing in this way solution construction are built-up and continually revised according to the outcomes of the solution generation process. The objective is to *adaptively learn*, through a continual sampling of solutions, a value assignment to the parameters of the policy that can eventually allow to generate good, possibly optimal, solutions to the combinatorial problem under consideration. Basically the idea is to reward those decisions that belonged to good complete solutions. The underlying assumption of the ACO approach is that good solutions posses common features that can be learned through repeated solution sampling. The single solutions are supposed to be constructed according to a *computationally light scheme*, in order to allow the sampling of a relatively large number of different solutions, realizing in this way an effective *directed exploration* of the search set.

Figure 3 shows in C-like pseudo-code the very general structure of the ACO metaheuristic. The algorithm is organized in three main logical blocks. The `daemon_actions()` block termed `daemon_actions()` refers to all those optional problem-specific activities which share no relationship with the biological context of inspiration of the metaheuristic and which do not really make use of pheromone information. For instance, it is common practice to interleave ants solution generation with local search procedures [19].

```

procedure ACO.metaheuristic()
  while ( $\neg$  stopping_criterion)
    schedule_activities
      construct_solutions_using_pheromone_and_stochastic_decisions();
      pheromone_updating();
      daemon_actions(); /* OPTIONAL */
    end schedule_activities
  end while
return best_solution_generated;

```

Figure 3: High-level description of the behavior of the ACO metaheuristic.

Precisely in the spirit of a metaheuristic, ACO does not specify the implementation details of its different components. This is clear from the pseudo-code, which does not specify how the ant

agents should construct the solutions, or how pheromone is updated following the evaluation of the generated solutions. Neither the characteristics of the pheromone mapping nor the modalities of the ant generation processes are specified. On the other hand, since a quite large number of ACO algorithms have been designed so far, there are quite consolidated and rather “standard” ways of designing new ACO algorithms. The reader is referred to the mentioned bibliography for loads of examples of ACO implementations, especially concerning classical combinatorial optimization problems (e.g., traveling salesman, quadratic assignment, task scheduling, etc.), which are static and can be solved in a fully centralized way. In the following we show how ACO’s recipe can be adopted almost as it is to design new state-of-the-art routing algorithms for modern dynamic networks.

4 ACO for routing in dynamic networks

In the Introduction we discussed the reasons behind the high complexity associated to the control of modern telecommunications networks. The main claim of this chapter is that the Nature-inspired ACO framework can provide basic guidelines to design in a rather straightforward way novel state-of-the-art routing algorithms showing those characteristics of adaptivity, robustness, scalability, self-organization, etc., reported in the Introduction wish list.

From the discussions of the previous sections it results that the fingerprints of the ACO approach are: repeated solution construction by means of lightweight agents, use of a stochastic decision policy to build the solutions, and progressive learning of the decision policy parameters from the observation of the outcomes of the solution generation process. All these characteristics are a natural match for the characteristics of routing and other network problems. This can be easily understood by considering the following simple facts. (i) Given the distributed nature of a network, the most natural way of proceeding to discover a feasible source-destination path consists in hopping from one node to the other until the final destination is reached. But this precisely means following a construction approach. (ii) Pheromone entries, which in ACO are the parameters of the construction policy, play exactly the same role of the entries in the node routing tables. (iii) Due to the dynamic and distributed nature of a network environment, it is intrinsically necessary to keep exploring the network and collecting useful information. That is, much alike in ant colonies, repeated sampling of full routing paths through the continual generation of ant agents (i.e., “smart” control packets) is expected to be an effective strategy to learn about network status and to consequently adapt the routing policy to it.

The good matching between the characteristics of ACO and those of network routing has attracted in recent years the interest of a relatively large number of researchers, resulting in a number of routing algorithms designed according to ACO guidelines (see [11, 26] for extensive references and reviews). The great majority of these algorithms are explicitly derived from the first two ACO implementations for routing problems: *ABC* [46] and *AntNet* [12, 11], which have addressed respectively routing in circuit-switched telephone networks and in packet-switched IP data networks. The empirical evidence is that well-designed ACO algorithms for routing perform comparably or much better than other/classical state-of-the-art approaches. This is the case of *AntNet* and *AntHocNet* (described later in the chapter) developed by the authors of this chapter.

In practice, a common underlying structure and set of strategies can be singled out from all these ACO-based routing algorithms. Here we use this evidence to informally define the *Ant Colony Routing (ACR)* framework, a high-level distributed control architecture that specializes the

general ideas of the ACO approach to the case of network routing and, at the same time, provides a generalization of the same ideas for the autonomic control of generic distributed systems. In the next two subsections, we first sketch the general underlying scheme common to most ACO implementations for routing, and then we show how we make at the same time a generalization and a specialization of it in order to provide an effective recipe for the design of novel routing algorithms based on ACO's principles. Finally, we describe the *AntHocNet* routing algorithm, as an example of how ACR can be adapted for a specific type of environment (in this case: *mobile ad hoc networks*).

4.1 Common characteristics of ACO implementations for routing

In the approach that is common to most ACO-based routing algorithms, nodes make use of ant packets to sample paths toward destination nodes of interest (e.g., end-points of frequent communications). Ants are usually generated according to simple periodic schemes. Each ant traveling *forward* to its destination constructs a solution, that is, a feasible path to its destination, in an incremental way by hopping between adjacent nodes. The task of an ant is to find a *good path* to the destination on the basis of the selected *metrics* of interest (e.g., time latency). Each node that can be chosen as a next hop has a *vector of pheromone variables* associated with it. These variables indicate how good it is to choose the given next hop, with respect to the selected optimization metrics, when traveling to the different possible destinations. Ants select their next hops according to a *stochastic policy* based on this pheromone information, and possibly also on further *local heuristic* information such as the length of the packet queues associated with the next hops. Once an ant has reached its destination, it evaluates the relative quality of the followed path and, carrying this information with it, retraces the path back to its source. At each of the visited nodes, the *backward* traveling ant provides this information to the node to let it update its pheromone information according to the fresh network information gathered by the ant during its journey.

This general scheme is independently and concurrently actuated by each node and by each generated ant. It is derived from the general ACO scheme consisting of the repeated pheromone-biased construction of solutions (paths), multiple successive solution evaluations, and recursive updating of the pheromone variables that directed the path-construction in the first place, with the general objective of dynamically learning properties of the problem at hand (here, the dynamically changing network environment). Clearly, different ACO instances show specific ways of implementing the different aspects of the general scheme, depending also on the specific characteristics of the network at hand. For instance, in some cases (e.g., cost-symmetric networks) the forward traveling ant can update pheromone information about the path leading back to its source, so that it is not necessary to travel backward to the source (this is the design choice made in ABC). Furthermore, in some cases using a periodic ant generation scheme is not appropriate. For example, in those cases in which bandwidth is scarce, like in wireless multihop ad hoc networks, ant generation is usually made according to on-demand schemes which use bandwidth in a much wiser way (e.g., see [32, 3, 13]).

4.2 The *Ant Colony Routing (ACR)* framework

The ACR framework inherits all the essential characteristics of ACO but at the same time introduces new basic types of agents, defines their hierarchical relationships, and points out the general characteristics and strategies that are expected to be part of a distributed control architecture based

on the ACO approach. In a sense, ACR's definition aims at make it clear what is an *ant-based* approach to network problems, and what is not.

In the ACR view the network is under the control of a system of a distributed society of adaptive autonomous agents, one for each node in the network. Each controller is called a *node manager*. Its internal status is defined by the values of the local pheromone tables and of possibly other additional data structures ad hoc for the network scenario at hand. Each entry in the pheromone table is associated to a different control action locally available to the node manager, and represents a statistical estimate of the goodness of taking that action. The controller adopts a stochastic decision policy which is parametrized by pheromone and possibly by other local heuristic variables. The target of each controller is to locally and autonomously learn a decision policy in terms of pheromone variables such that the distributed society of controllers can jointly learn a decision policy optimizing some global performance. Each controller is expected to learn good pheromone values by continual *monitoring* of the network environment and of the effect of its decisions on it, and by making use of the observed data to adaptively change the pheromone values as well as other parameters regulating its monitoring and control behavior. Local monitoring (also called *passive monitoring*) is realized directly by the node managers (e.g., monitoring of local traffic flows). Non-local network information is collected by explicit generation of lightweight *ant agents* (*active monitoring*) acting as *active long-range perceptions* of the node managers.

The task of such an ant agent is to explore the network with the aim of discovering a good path to an assigned destination, collect useful information about the quality and the status of the network along path, and communicate this information back to the node managers to let them updating their pheromone values accordingly. Typically an ant sample a full feasible network path, evaluate its quality, and retrace the path back to allow pheromone updates at the node managers along the path. Ants explore the network adopting a stochastic routing policy based on the node pheromone values. They can show a locally adaptive behavior for instance by *replicating* (or, *proliferating*) when there is more than one single equivalently alternative, or self-destroying in case of excessive local congestion. The term "active perception" is referred to the fact that both a non-local information gathering act is explicitly issued and each one of these perceptual acts can actually be generated with specific characteristics (e.g., different levels of exploratory attitude) in order to precisely get information about a specific area of the network by using a specific strategy.

Node managers act concurrently and without any form of global coordination. Therefore, they must act *socially* and possibly cooperate (e.g., exchange messages or even their decision policies) in order to get positive synergy (e.g., not injecting too many ants in case of congestion).

The design of an ACR algorithm involve several critical choices such as strategies for: (i) local monitoring, (ii) proactive and on-demand scheduling of the ant agents, (iii) definition of the internal characteristics for the generated ants, (iv) ant policies, and (v) use of the gathered information in order to learn effective decision policies for routing and tune other node internal parameters regulating for instance the rate of proactive ant generation.

4.2.1 General properties and innovative aspects of ACR

Writing a novel routing algorithm (or, more generally, a network control algorithm) according to ACR guidelines, results in an algorithm which is expected to enjoy one or more properties that can be seen as highly desirable for the control of modern dynamic networks. We discuss these expected properties one by one in the following of this subsection.

Self-organization and robustness

The control system is fully distributed and self-organizing. Each controller is an independent learner and exchange information with other controllers either by local message passing or through the generation of path sampling ants. Decision policies are learned in a strictly local way, which makes the whole self-organizing process of collective learning highly robust to network changes. This fact can be understood by considering that at each node manager the updating of pheromone estimates results from the reception of information from the passing by ants and is realized independently from the estimates held at the other neighbor nodes. That is, the estimated goodness values for routing decisions recorded at each node are the result of the direct experiences of the ant agents. This way of proceeding differs substantially from that followed in the most consolidated routing frameworks such as the distance-vector and link-state ones (e.g., see [5, 11]), from which are derived the most popular Internet routing protocols such as the already mentioned RIP and OSPF. Distance-vector strategies are based on *information bootstrapping*. Bootstrapping is a technique that characterizes dynamic programming [49]. Nodes estimate the cost-to-go of a path by combining the cost estimates made by neighboring nodes and the cost to go to those neighboring nodes, rather than by the direct sampling of a full path. Bootstrapping is a powerful technique that can save a lot of computation when the system is basically stationary. On the other hand, when the system is highly dynamic, bootstrapping can be harmful. In fact, since each local estimate is propagated step-by-step throughout the whole network and used in turn to build new estimates, if an estimate becomes incorrect due to local changes this fact will affect the entire network and the new correct estimate has to be re-propagated to all the network nodes. An analogous problem can arise in link-state algorithms, where each node holds a complete representation of the entire network in terms of a weighted graph. Each node builds an estimate of the cost of its local links (e.g., in terms of queue length) and periodically floods these estimates into the network. On reception of cost estimates, each node updates the weights of its network representation and re-calculates minimum cost paths to define its local routing policy. Again, the approach is very effective in the quasi-stationary case but suffers of the same problems of the bootstrapping-based ones in case of dynamic situations.

On the other hand, ACR's approach, which is termed *Monte Carlo learning* in reinforcement learning jargon [49] is expected to be less effective in the (uncommon) quasi-stationary case while is way more robust when the system is highly dynamic since no unnecessary dependencies among node policies are created. ACR is also robust to losses of control packets (ants). In fact, if an ant gets lost or has to be discarded because of hardware errors or buffer overflows, the impact on the system is minimal. It will likely result just in a slower update of the pheromone tables for a single specific source-destination pair.

Monitoring and adaptivity

Passive and active network monitoring is the key to obtain adaptive behavior. In ACR algorithms node managers are able to progressively adapt the used decision policies to network changes precisely through continual monitoring. Clearly, monitoring creates an overhead, especially in terms of repeated generation of ant agents. The challenge of any ACR implementation consists in finding the right tradeoff between the rate and the amount of gathered information and the resulting overhead in order to obtain good and scalable performance.

Active monitoring has been rarely implemented in routing strategies precisely due to its potential negative impact in terms of overhead (e.g., see the use of packet-probing techniques for capacity estimation [21]). On the other hand, in Subsection 4.3 we show that even in the challenging environment of mobile ad hoc networks, we could define in our AntHocNet algorithm an effective way to minimize the overhead due to active monitoring by ant agents while providing at the same time state-of-the-art performance.

Multiple paths, stochastic data routing, resources optimization, and scalability

Routing tables in ACR algorithms consist of pheromone entries. This means that for each destination of interest d and for each neighbor node n , an estimate of expected quality of the path that goes through n to eventually reach d is available at the nodes. In practice this means that a *bundle of paths*, each with an associated estimated quality, results from the ant path sampling activities. And both the paths and their estimates are continually updated to track network changes. This path bundle can be used to implement *multiple path routing of data packets* and/or as *backup paths* in case of sudden changes or failures that cause the main path(s) becoming unavailable. Moreover, pheromone values can be used to implement *probabilistic routing* schemes for data forwarding precisely as it happens for the ants, but with a strong preference for the best paths, since exploration must not be carried by user data packets. Concurrent probabilistic spreading of data packets across the best multiple paths results in automatic *load balancing* and performance optimization (e.g., throughput increase). On the other hand, when the path bundle is used for backup purposes, the adaptive response of the system to sudden changes is expected to be very robust and smooth. Both these two different uses of the available path bundle result also in a good scalability of the system since allow a thorough resource optimization.

Generally speaking, the use of multiple paths even if potentially appealing is by no means free of problems. Issues such as how many paths to use, paths selection criteria (which is particularly challenging in interfering wireless environments), and data distribution policy, are very complex to deal with at design time (e.g., see [40]) in order to really boost up performance. Again, in the next section, we show through experimental results that AntHocNet's design can deal in rather automatic way with these issues using the adaptive quality estimates associated to each available path by means of pheromone values.

4.3 AntHocNet: ACO for routing in mobile ad hoc networks

In the remainder of this section, we describe an example of how the ACR framework can be applied to develop an adaptive routing algorithm for a specific network environment. We focus on best-effort routing in *mobile ad hoc networks (MANETs)*. MANETs form a particularly challenging type of networks, in which all the properties of modern networks mentioned earlier in our wish list, such as adaptivity, robustness, decentralized working, etc., are essential. The ACR algorithm we describe is *AntHocNet*. This algorithm has been thoroughly discussed and evaluated in a number of recent papers [13, 14, 23, 24, 15, 16, 2, 22, 25]. Here we provide a summary description of its behavior. The interested reader can find more detailed descriptions and discussions in the given references.

In MANETs, all nodes are mobile and can enter and leave the network at any time. They communicate with each other via wireless connections. All nodes are equal and there is neither centralized control nor fixed infrastructure to rely on (e.g., ground antennas). There are no desig-

nated routers: all nodes can serve as routers for each other, and data packets are forwarded from node to node in a multi-hop fashion. The wireless channel is shared among the peer nodes and the access must be arbitrated according to same Medium Access Control (MAC) layer protocol. MANETs represent the likely most dynamic, constrained and complex example of a real-world network. MANET control definitely asks for algorithms meeting the requirements of the Introduction's wish list. MANET can find application in variety of scenarios needing the creation of a network on-the-fly due to the unavailability of a connecting infrastructure. This is a situation typical in battlefields, disaster or remote areas, or in geographically limited areas where specific communities of users must suddenly and temporarily interact.

Providing reliable data transport in MANETs is quite difficult, and a lot of research is being devoted to this. Especially the routing problem is very hard to deal with, due to constant changes in topology and traffic and to the limited bandwidth available from the shared wireless channel. In recent years a number of routing algorithms have been proposed (e.g., see [44, 8, 48]), but even current state-of-the-art protocols are quite unreliable in terms of data delivery and delay.

The main challenge of MANETs for ant-based routing schemes consists in finding the right balance between the *rates of ant generation* and the resulting *network overhead*. In fact, from one hand, repeated path sampling is at the very core of ACR algorithms: more ant agents means that an increased and more up-to-date amount of routing information is gathered, possibly resulting in a better adaptation of the routing policy at the nodes. On the other hand, an excessive generation of routing packets can heavily affect in negative sense large number of nodes at once determining a degradation rather than an increase in performance. This is due to the fact that the radio channel is a shared resource, such that multiple radio collisions can happen at the MAC layer in case of high traffic load in dense node areas, with consequent underutilization of the nominal bandwidth. In the final part of this section we show by reporting experimental results that AntHocNet's design is such that such a good balance can actually be achieved while providing at the same time extremely good adaptivity with respect to traffic and topology, as well as robustness and scalability.

4.3.1 Algorithm description

In MANET jargon AntHocNet is termed a *hybrid* algorithm since it makes use of both reactive and proactive strategies to establish routing paths. It is *reactive* in the sense that a node only starts gathering routing information for a specific destination when a local traffic session needs to communicate with the destination and no routing information is available. It is *proactive* because as long as the communication starts, and for the entire duration of the communication, the nodes proactively keep the routing information related to the ongoing flow up-to-date with network changes. In this way both the costs and the number of paths used by each running flow can reflect the actual status of the network, providing an optimized network response. The reactive component of the algorithm deals with the phase of *path setup* and is totally based on the generation of ant agents to find a good initial path for the new data session. The proactive component of the algorithm implements *path maintenance and improvement*. During the course of a session it proactively adapts to network changes the paths the session is using. Path maintenance and improvement is realized by a combination of ant path sampling and slow-rate *pheromone diffusion*: the routing information obtained via ant path sampling is spread between the nodes managers and is used to update the pheromone tables according to a bootstrapping scheme (see Section 4.2.1) that in turn provides main guidance for the exploration behavior of the ants. *Link failures* are dealt with using a local path repair process or via the generation of ant agents carrying explicit notifica-

tion information. *Stochastic decisions* are used both for ant exploration and to spread data packets over multiple paths.

In the following we provide a concise description of each of the components of the algorithm in a separate subsection.

Pheromone metrics used to define path quality

As in any ACR algorithm, paths are implicitly defined by the values of pheromone variables contained in so-called *pheromone table* playing the role of routing tables. An entry $\tau_{nd}^i \in \mathbb{R}$ in the pheromone table \mathcal{T}^i of node i contains a value indicating the estimated goodness of going from i over neighbor n to reach destination d . Since AntHocNet only maintains information about destinations which are active in a communication session, and the neighbors of a node change constantly, filling of the pheromone tables is sparse and dynamic. In principle in the complex MANET environment several aspects concur to define the quality of a path: *number of hops, end-to-end delay, signal quality, congestion level, battery usage, node speed, etc.*. AntHocNet defines the pheromone variables in terms of some combination of these metrics. In particular, the pheromone variables used in the experiments reported later makes use of a combination of number of hops and signal-to-noise ratio (see [22] for a discussion on the use of different combination of metrics and on their relative evaluation). This means that the algorithm tries to find paths characterized by minimal number of hops and good signal quality between adjacent nodes. The use of such composite pheromone values allows to optimize multiple objectives simultaneously, which can be very important in complex networks.

Reactive path setup

When a source node s starts a communication session with a destination node d , and no pheromone information is available about how to reach d , the node manager needs to gather long range information about possible paths. Therefore, it broadcasts a *reactive forward ant*. This constitutes a reactive approach to ant generation, aimed at improving the efficiency of the system by focusing on gathering information that is strictly necessary. Broadcast is equivalent to replicating the ant agent to all the neighbors having an equivalent (null, in this case) estimate of path quality toward the destination. Ants are sent over high priority queues. At each node, the ant is either unicast or broadcast, according to whether or not the current node has pheromone information for d . If pheromone information is available, the ant makes use of a random proportional rule to select its next hop assigning to each neighbor n a selection probability P_{nd} which depends on the relative goodness of n as a next hop as expressed by the value of the associated pheromone variable τ_{nd}^i :

$$P_{nd} = \frac{(\tau_{nd}^i)^\beta}{\sum_{j \in \mathcal{N}_d^i} (\tau_{jd}^i)^\beta}, \quad \beta \geq 1, \quad (3)$$

where \mathcal{N}_d^i is the set of neighbors of i over which a path to d is known and not yet visited by the ant, and β is a parameter which controls the exploratory behavior of the ants. The expression for P_{nd} is equivalent to the generic one for ACO described in (2). If no pheromone is available, the ant is broadcast. Since it is quite likely that somewhere along the path no pheromone is found, in the experiments we normally use a high value of β to avoid excessive ant proliferation. Due to subsequent broadcasts, many duplicate copies of the same ant travel to the destination. A node

manager which receives multiple copies of the same ant only accepts the first and discards the other. In this way, only one path is set up initially. During the course of the communication session, more paths are added via the proactive path maintenance and exploration mechanism discussed in the next subsection.

Each forward ant keeps a list of the nodes it has visited. Upon arrival at the destination d , it is converted into a *backward ant*, which travels back to the source *retracing* the path. This is the typical mode of operation of ACO algorithms. At each intermediate node i , coming from neighbor n , the ant information is used to update the entry τ_{nd}^i in the i 's pheromone table. The way the entry is updated depends on the path quality metrics used to define pheromone variables. For instance, in the simplest case when the pheromone is expressed using only the number of hops as a measure of path goodness, at each hop the backward ant increments an internal hop counter and the inverse of this value is used to locally assign the value τ_d^i which is used to update the pheromone variable τ_{nd}^i as follows:

$$\tau_{nd}^i = \gamma\tau_{nd}^i + (1 - \gamma)\tau_d^i, \quad \gamma \in [0, 1]. \quad (4)$$

For different metrics, the calculation of τ_d^i is more complex but follows the same logic. For instance, if delay is used, the ant needs to incrementally calculate at each node a robust estimate of the expected delay to reach the destination (see [14, 24]).

Once the backward ant has returned, a single path is made available between source and destination. During the course of the communication session, more paths are added via the proactive path maintenance and exploration mechanism discussed next.

Proactive path maintenance and exploration

During the course of a communication session, managers at source nodes periodically send out *proactive forward ants* to update the information about currently used paths and to try to find new and potentially better paths. They follow pheromone and update pheromone tables in the same way as reactive forward ants do. Such continual proactive sampling of paths is the basic mode of operation implemented in existing ACO routing algorithms, and corresponds to a proactive approach of ant generation, with the aim to keep monitoring the situation along the used paths. However, in MANETs the ant sending frequency which is needed to faithfully keep track of the constant network changes is in general too high for the available bandwidth. Moreover, to find entirely new paths, excessive blind exploration through random walks or broadcasts would be required. Therefore, the node managers we keep the ant sending rate low and integrate the ant sampling actions with a lightweight cooperative information bootstrapping process termed *pheromone diffusion*. This process provides a second way of updating pheromone on existing paths, while at the same time can provide useful information to guide exploratory ant behavior.

In the pheromone diffusion process, each node manager n periodically and asynchronously broadcasts a sort of beacon message containing a list of destinations it has information about, including for each destination d its best pheromone value $\tau_{m^*d}^n$. A node manager i receiving the message from n first of all registers that n is its neighbor. This is a form of local monitoring performed by the node managers. Then, for each destination d listed in the message, it derives an estimate of the goodness of going from i to d over n , combining the cost of hopping from i to n with the reported pheromone value $\tau_{m^*d}^n$. We call the obtained estimate b_{nd}^i *bootstrapped pheromone*, since it is built up bootstrapping on the value of the path quality estimate received from an adjacent node. The bootstrapped pheromone can be in turn forwarded in the next beacon message sent out by n . The concurrent iteration of this process by all nodes gives rise to a

bootstrapped pheromone field over the MANET. As already pointed out in Subsection 4.2.1, this is the typical way of calculating estimates implemented by the routing approaches based on dynamic programming, such the class of distributed Bellman-Ford routing algorithms [5] and the reinforcement learning algorithms derived from Q-learning [49]. Bootstrapping is not appropriate for highly dynamic environments, and also in our case, due to the slow multi-step forwarding, bootstrapped pheromone does not provide the most accurate view of the current situation and has difficulty to properly track the distributed changes in the network. We can say that the pheromone diffusion process is efficient and light-weight, but gives potentially unreliable information. The key element in AntHocNet is that pheromone diffusion is not the only source of routing information, but rather serves as a secondary process to complement the ant-based Monte Carlo path sampling process. It is important to take care when and how to use the bootstrapped pheromone information.

For path maintenance, bootstrapped pheromone is used directly. If i already has a pheromone entry τ_{nd}^i in its routing table for destination d going over neighbor n , we know that the path from i to d over n has in the past been sampled by ants. This path can therefore be considered reliable. b_{nd}^i is then just treated as an update of the goodness estimate of this reliable path, and is used directly to replace τ_{nd}^i . This way, pheromone on current paths is kept up-to-date.

On the other hand, *for path exploration*, bootstrapped pheromone is used indirectly, only after an explicit checking. If i does not have yet a value for τ_{nd}^i in its pheromone table, b_{nd}^i could indicate a possible new path from i to d over n . However, this path has never been explicitly tried out by an ant, such that due to the slow multi-step process it could have disappeared, or it can contain undetected loops or dangling links. It is therefore not safe to use for data forwarding before being checked. This is the task assigned to proactive forward ants, which make use both regular and bootstrapped pheromone on their way to the destination. This way, promising pheromone is checked out, and if the associated path is there and has the expected good quality, it can be turned into a regular path available for data. This increases the number of paths available for data routing, which grows to a full mesh, and allows the algorithm to exploit new opportunities in the ever changing topology.

The approach followed in this proactive path maintenance and exploration process and the earlier described reactive path setup process illustrates the generality of ACR node managers. Rather than mere ant generators, they are general learning agents, which can use different monitoring and learning strategies to obtain their goals. In the case of AntHocNet, reactive and proactive ant-based Monte Carlo sampling is combined with information bootstrapping to obtain adaptivity, reliability and efficiency.

Stochastic data routing

The path setup phase together with the proactive path improvement actions create a mesh of multiple paths between source and destination. Data are forwarded according to a *stochastic policy* depending onto the pheromone values. When a node has multiple next hops for the destination d of the data, it randomly selects one of them, with probability P_{nd} , which is calculated in the same way as for reactive forward ants in (3), but with a much higher exponent, in order to be greedy with respect to the better paths.

In the experiments, β_2 was set to 20. Setting the routing exponent so high means that if several paths have similar quality, data will be spread over them. However, if one path is clearly better than another, it will almost always be preferred. According to this strategy, we do not have to

choose a priori how many paths to use: their number will be automatically selected in function of their quality.

If estimates are kept up-to-date (which is done using the bootstrapped pheromone), the dynamic availability of multiple paths leads to *automatic load balancing*, optimization of resources utilization, and increase in throughput. This is a common characteristic in ACR implementations.

Link failures

Nodes can detect link failures (e.g., a neighbor has moved away) through *local monitoring*. Specifically, link failures are assumed to have taken place when unicast transmissions (of data packets or ants) fail, or when expected pheromone diffusion messages were not received. When a neighbor is assumed to have disappeared, the node manager takes a number of actions. In the first place, it removes the neighbor from its neighbor list and all the associated entries from its routing table.

Further actions depend on the event which was associated with the discovered disappearance. If the event was a failed transmission of a control packet, the node broadcasts a *link failure notification* message. Such a message contains a list of the destinations to which the node lost its best path, and the new best estimated end-to-end delay and number of hops to this destination (if it still has entries for the destination). All its neighbors receive the notification and update their pheromone table using the new estimates. If they in turn lost their best or their only path to a destination due to the failure, they will broadcast the notification further, until all concerned nodes are notified of the new situation.

If the event was the failed transmission of a data packet, the node does not include the destination of the data packet in question in the link failure notification. For this destination, the node starts a *local route repair* process. The node manager broadcasts a *route repair ant* that travels to the involved destination like a reactive forward ant: it follows available routing information when it can, and is broadcast otherwise. One important difference is that it has a maximum number of broadcasts (which we set to 2 in our experiments), so that its proliferation is limited. The node waits for a certain time (in the experiments set to 5 times the estimated end-to-end delay of the lost path), and if no backward repair ant is received by then, it concludes that it was not possible to find an alternative path to the destination. Packets which were in the meantime buffered for this destination are discarded, and the node sends a new link failure notification about the lost destination.

4.3.2 Experimental results

AntHocNet's performance has been extensively evaluated against state-of-the-art algorithms under a number of different MANET scenarios for both open space and realistic urban conditions. We studied the behavior of the algorithm under different conditions for network size, connectivity, change rate, data traffic patterns, and node mobility. Performance was measured in terms of data delivery ratio, end-to-end packet delay and delay jitter as measures of *effectiveness*, and routing overhead in number of control packets per successfully delivered data packet as measure of *efficiency*. Here we report only a small subset of these results with the aim of supporting the claim that AntHocNet shows superior performance in terms of general efficacy and efficiency, and in terms of adaptivity, robustness and scalability. The reader is referred to the mentioned AntHocNet references for the full set of results.

All experiments are carried out in simulation using *QualNet* [43], a commercial state-of-the-art simulator for telecommunications networks. The *open space* scenarios used in the tests reported on here were all derived from the same base scenario. In this scenario, 100 nodes are randomly placed in an open space area of $2400 \times 800 \text{ m}^2$. Each experiment is run for 900 seconds. Data traffic is generated by 20 constant bit rate (CBR) sources sending four 64-byte packets per second using UDP at the transport layer. Each source starts sending at a random time between 0 and 180 seconds after the start of the simulation, and keeps sending until the end. A two-ray path-loss model is used in the radio propagation model. The radio range of the nodes is 250 meters, and the data rate is 2 Mbit/s. At the MAC layer we use the IEEE 802.11b DCF protocol as is common practice in MANET research. The nodes move according to the random waypoint (RWP) mobility model [36]: they choose a random destination point and a random speed, move to the chosen point with the chosen speed, and rest there for a fixed amount of pause time before they choose a new destination and speed. The speed is chosen between 0 and 10 m/s, and the pause time is 30 seconds.

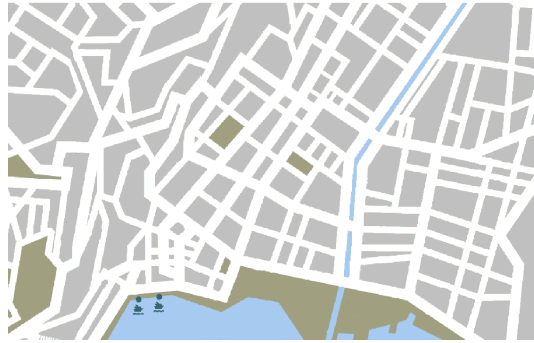


Figure 4: The map of the city of Lugano, Switzerland, used for the experiments in the realistic urban scenario.

The *urban* scenario is supposed to model a realistic urban environment where a MANET could be actually deployed and used. The scenario is derived from the street organization of downtown Lugano, Switzerland. Figure 4 shows the city map we considered as reference. It corresponds to an area of $1561 \times 997 \text{ m}^2$, which covers most of downtown Lugano. Streets define the open spaces where nodes are free to move. Buildings are inaccessible to the nodes and basically play the role of obstacles that put constraints on agent movements and shield radio signal propagation. Three classes of nodes live in the scenario: nodes moving at typical urban car speed, nodes simulating walking people, and non-mobile users. Most of the basic settings used in the open space scenarios have been also used in the urban one. The characteristics and the complexity of the urban scenario are quite different from those of the open space ones. Here we report just few results for the urban case with the aim of showing that the ACR approach is effective over radically different classes of network scenarios. The interested reader can find loads of experimental results for the urban scenario in [16, 25].

To assess the performance of our algorithm relative to the state-of-the-art in the field, we compare each time to *Ad-hoc On-demand Distance Vector routing (AODV)* [42], and *Optimized Link State Routing (OLSR)* [9], two important reference algorithms in the field. AODV is a reactive algorithm, while OLSR is based on a proactive strategy. All points reported in the data plots are the average over 15 simulation runs with different random placement of the nodes.

Scalability: Varying the number of nodes in open space

We increase the number of nodes, from 100 to 800 nodes. The MANET area was increased accordingly, to keep the node density constant. The results are presented in Figures 5 and 6. For OLSR we report only results up to 500 nodes, as simulation run times became prohibitively large beyond that, and performance very low. We can see that AntHocNet's advantage over both other algorithms grows for all measures of effectiveness for larger networks. This is an indication that it is a scalable algorithm. Also in terms of efficiency, AntHocNet seems to be scalable: while its overhead is comparable to that of the other algorithms for small networks, it increases less quickly and is much lower for the larger networks.

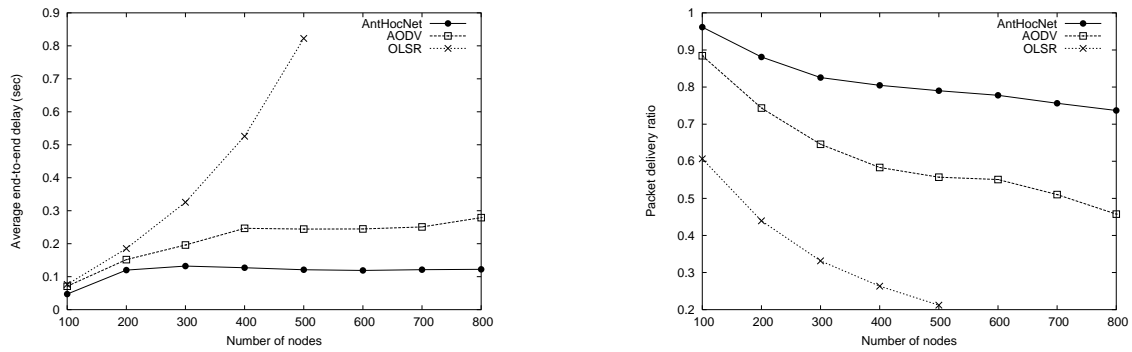


Figure 5: Average delay (left) and delivery ratio (right) for an increasing number of nodes.

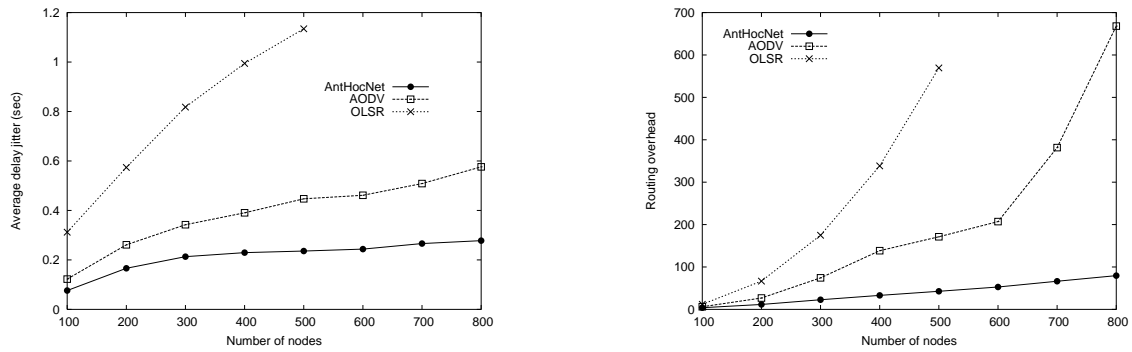


Figure 6: Average jitter (left) and routing overhead (right) for an increasing number of nodes.

Adaptivity to topological changes: Varying the pause time in open space

In this set of experiment we study the effect of increased mobility by changing the RWP pause time. This has a direct effect on the changing rate of the network topology. Results are reported in Figures 7 and 8. AntHocNet's average end-to-end delay is about half that of AODV for low pause times, and around one third for high pause times. In terms of delivery ratio, the difference is less striking but still significant. OLSR is always performing very bad. Also in terms of jitter we can see a large advantage of AntHocNet over the two other algorithms. Again, the better AntHocNet's performance is not payed back in terms of higher routing overhead. These results seem to indicate

the effectiveness of AntHocNet to adapt robustly to dramatically different rates of topological changes inside the network.

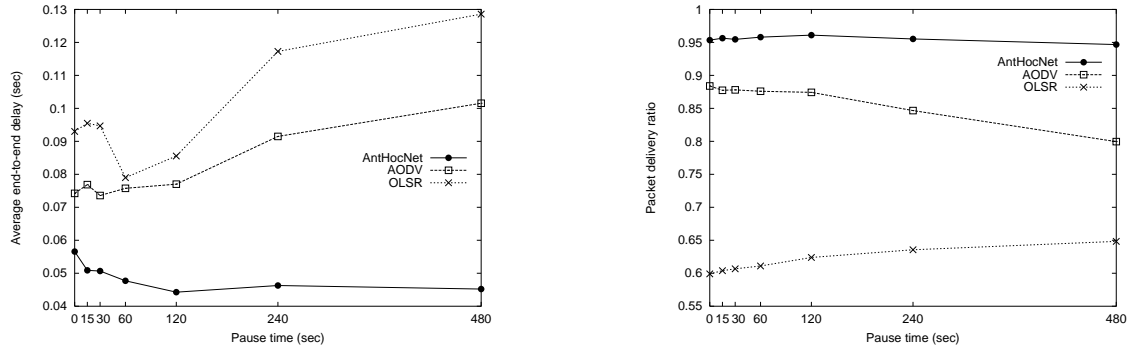


Figure 7: Average delay (left) and delivery ratio (right) for increasing pause time.

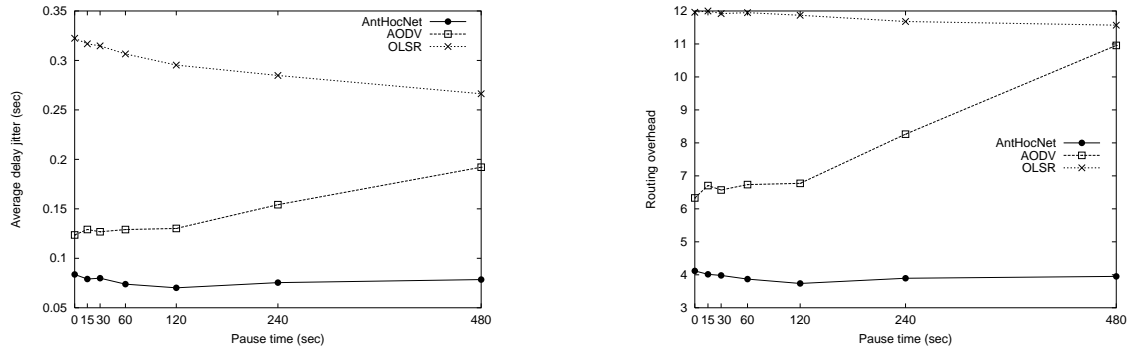


Figure 8: Average jitter (left) and routing overhead (right) for increasing pause time.

Scalability and adaptivity to traffic loads: Varying the number of data sessions in open space

Here we report the results for varying the total number of active data traffic sessions (Figures 9 and 10). In this way we aim to study the response of the algorithms to load increase. This gives important indications about both the traffic scalability of the algorithms and their ability to cope with a very dynamic situation at the local level caused by the continual transmissions of large amounts of data packets. Also this set of experiments shows a significantly superior performance of AntHocNet with respect to both AODV and OLSR for the considered metrics. These results provide a further validation of the characteristics of adaptivity and scalability of the approach.

Scalability: Varying the number of nodes in the urban scenario

We increase the number of nodes, from 200 to 500 nodes. The total number of active CBR traffic sessions is kept constant at 30, as well as the number of nodes moving at car speed, which is fixed at 25. 30% of the nodes are non-mobile. Basically increasing the number of nodes (up to a certain threshold) in the radio-constrained urban scenario makes the task easier, since it brings more local connectivity. However, the algorithm has to show to ability to profit for the increase in

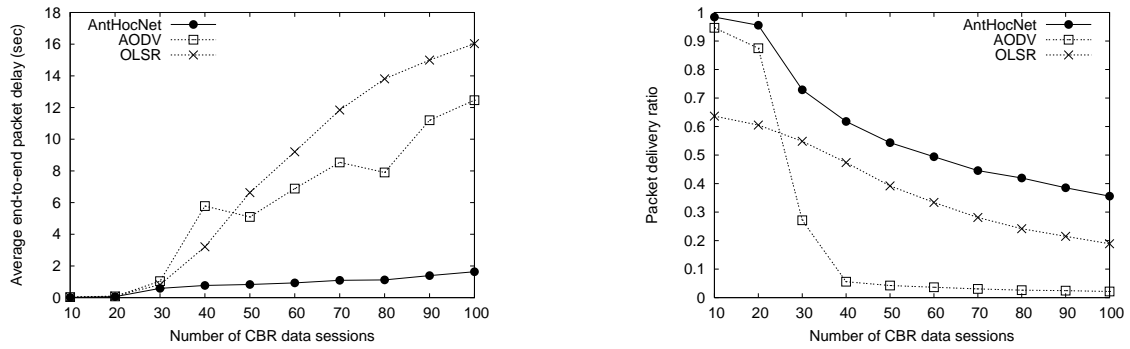


Figure 9: Average delay (left) and delivery ratio (right) for increasing the number of sessions.

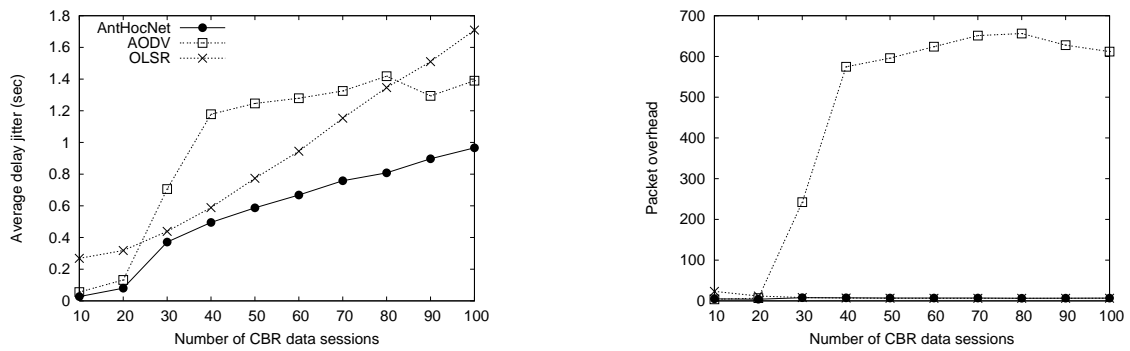


Figure 10: Average jitter (left) and routing overhead (right) for increasing the number of sessions.

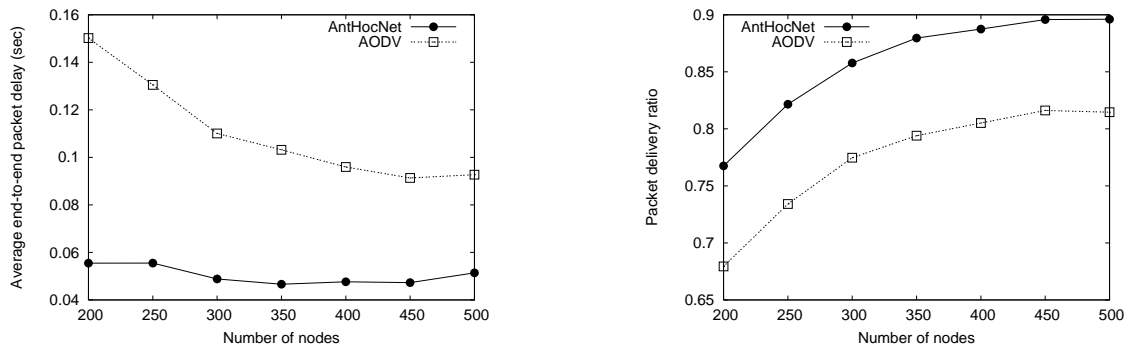


Figure 11: Average delay (left) and delivery ratio (right) for an increasing number of nodes in the urban scenario.

connectivity, as well as the ability to deal with the radio and movement constraints imposed by the urban structure. Results are shown in Figures 11 and 12. OLSR's performance are not reported since in all the experiments we carried on the urban environment OLSR was always performing very poorly. We can see that the difference in performance between AODV and AntHocNet is this time approximately constant but still always clearly in favor of AntHocNet. Again, also this set of experiments ran in a scenario showing quite different characteristics from the previous

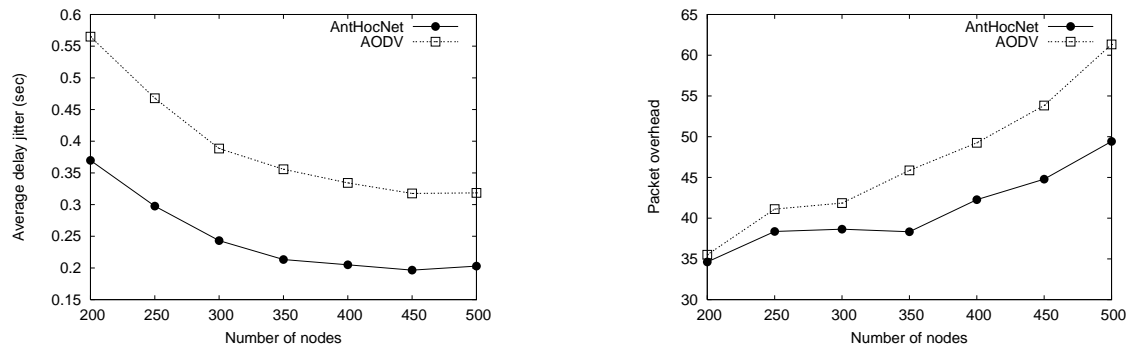


Figure 12: Average jitter (left) and routing overhead (right) for an increasing number of nodes in the urban scenario.

ones confirm the good scalability and adaptivity of approach, as well as its overall robustness in providing superior performance over a wide range of different scenarios.

5 Conclusions and Future Trends

In this chapter we considered the problem of designing novel routing algorithms for the very complex, dynamic, and heterogeneous modern networks. In the Introduction we stated a sort of wish list of desirable properties a network control algorithm should have to cope with this high complexity. In practice we need algorithms which are as much as possible *adaptive*, *robust*, *scalable*, and *self-organizing*. Our claim is that a *biologically-inspired* framework like *Ant Colony Optimization* can provide the basic guidelines to implement in a relatively straightforward way novel routing algorithms possessing precisely these characteristics. We described the biological process from which ACO was derived, that is, the pheromone-mediated ability of the ants in ant colonies to find in a totally distributed and dynamic way shortest paths joining their nest with sources of food. We defined ACO after this ant *shortest path behavior*. We described it as a general recipe to solve combinatorial optimization problems by repeated construction of multiple solutions according to a step-by-step stochastic decision policy depending on real-valued variables counterpart of the pheromone in ant colonies. The outcomes of the solution generation process are used in turn to adaptively modify the value of these pheromone variable. The aim is *learning a decision policy* that would eventually allow the generation of good solutions for the optimization problem at hand. We showed that this way of proceeding is particularly suitable to attack routing problems in network environments. Then starting from ACO, *Ant Colony Routing (ACR)*, a specialized framework for the design of autonomic routing systems based on adaptive learning components. We discussed the general properties expected from a properly designed ACR algorithm, and we pointed out that it would be relatively easy to obtain the desirable properties stated in the wish list. To show this in practice, we have described and thoroughly evaluated *AntHocNet*, a routing algorithm for *mobile ad hoc networks*, which form a very complex and dynamic class of networks. AntHocNet's performance was evaluated in both *open space* and *urban scenarios*. The experimental results seem to strongly suggest that the algorithm is able to provide superior performance and to show those characteristics of adaptivity, scalability, robustness and self-organization that we were looking for.

In this chapter, we have not provided any formal justification or theoretical analysis of our

claims. Therefore, we can only speak on the basis of empirical evidence. However, the amount of experiments that we and other researchers in the field have carried out so far over a large number of different network scenarios provides a significant statistical evidence of the overall efficacy and effectiveness of the ACR general guidelines for the design of state-of-the-art routing algorithms showing autonomic properties.

In terms of future work, we can see two important research directions. In the first place, it is important to carry out theoretical work in the direction of understanding under which dynamic conditions an adaptive learning approach like ACR is really useful. In fact, if the system dynamics are too fast or hectic to be tracked and in some sense understood in a distributed way, it is probably better to rely on simple purely reactive schemes instead of trying to collect information that becomes too rapidly out of date. On the other hand, if the system is rather static, non-adaptive approaches might prove to perform better. The second important research direction is of practical nature. Most of the protocols that are currently in use, like TCP, were conceived for much more static environments and are not really suited to deal properly with adaptive or probabilistic schemes. Therefore, to really bring adaptivity in modern networks, it will likely be necessary to rewrite or radically modify many existing network protocols. This is at the moment a major impediment for the real-world implementation and use of novel autonomic network controllers like AntHocNet.

References

- [1] D.O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X.P. Xiao. A framework for Internet traffic engineering. Internet Draft draft-ietf-tewg-framework-05, Internet Engineering Task Force - Traffic Engineering Working Group, January 2000.
- [2] O. Babaoglu, G. Canright, A. Deutsch, G.A. Di Caro, F. Ducatelle, L.M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montessor, and T. Urnes. Design patterns from biology for distributed computing. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1), September 2006.
- [3] J.S. Baras and H. Mehta. A probabilistic emergent routing algorithm for mobile ad hoc networks. In *WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
- [4] D. Bertsekas. *Dynamic Programming and Optimal Control*, volume I–II. Athena Scientific, 1995.
- [5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice–Hall, 1992.
- [6] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [7] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.
- [8] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom98)*, 1998.

- [9] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of IEEE INMIC*, 2001.
- [10] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.
- [11] G.A. Di Caro. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, November 2004.
- [12] G.A. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.
- [13] G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In *Proceedings of Parallel Problem Solving from Nature (PPSN) VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 461–470. Springer-Verlag, 2004. (Conference best paper award).
- [14] G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455, 2005.
- [15] G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Swarm intelligence for routing in mobile ad hoc networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*, Pasadena, USA, June 8–10, 2005.
- [16] G.A. Di Caro, F. Ducatelle, and L.M. Gambardella. Studies of routing performance in a city-like testbed for mobile ad hoc networks. Technical Report 07-06, IDSIA, Lugano (Switzerland), March 2006.
- [17] M. Dorigo and G.A. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.
- [18] M. Dorigo, G.A. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
- [19] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
- [20] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.
- [21] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions on Networking*, 12(6):963–977, 2004.
- [22] F. Ducatelle, G.A. Di Caro, and L. M. Gambardella. An analysis of the different components of the AntHocNet routing algorithm. In *Proceedings of ANTS 2006, 5th International Workshop on Ant Algorithms and Swarm Intelligence*, volume 4150 of *LNCS*, pages 37–48. Springer, 2006.

- [23] F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. Ant agents for hybrid multipath routing in mobile ad hoc networks. In *Proceedings of the Second Annual Conference on Wireless On demand Network Systems and Services (WONS)*, St. Moritz, Switzerland, January 18–19, 2005.
- [24] F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications*, Special Issue on Nature-Inspired Approaches to Networks and Telecommunications, 5(2):169–184, June 2005.
- [25] F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. An evaluation of two swarm intelligence manet routing algorithms in an urban environment. In *Proceedings of the 5th IEEE Swarm Intelligence Symposium*, St. Louis, USA, September 21–23, 2008.
- [26] M. Farooq and G.A. Di Caro. Routing protocols for next-generation intelligent networks inspired by collective behaviors of insect societies. In C. Blum and D. Merckle, editors, *Swarm Intelligence: Introduction and Applications*. Springer, 2008.
- [27] D. B. Fogel. *Evolutionary Computation*. IEEE Press, 1995.
- [28] R. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, 25:73–84, January 1977.
- [29] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W.H. Freeman and Company, 1979.
- [30] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.
- [31] P. P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes sp.* La théorie de la stigmergie: essai d’interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.
- [32] M. Günes, U. Sorges, and I. Bouazizi. ARA - the ant-colony based routing algorithm for MANETS. In *Proceedings of the 2002 ICPP International Workshop on Ad Hoc Networks (IWAHN 2002)*, pages 79–85, August 2002.
- [33] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [34] B. Hölldobler and E.O. Wilson. *The Ants*. Springer-Verlag, Berlin, Germany, 1990.
- [35] J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52, 1985.
- [36] D. B. Johnson and D. A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer, 1996.
- [37] J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer magazine*, pages 41–50, January 2003.

- [38] G. S. Malkin. *RIP: An Intra-Domain Routing Protocol*. Addison-Wesley, 1999.
- [39] J. Moy. *OSPF Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.
- [40] S. Nelakuditi and Z.-L. Zhang. On selection of paths for multipath routing. In *Proceedings of the Interational Workshop on QoS (IWQoS)*, volume 2092 of *Lecture Notes in Computer Science*, pages 170–182, 2001.
- [41] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, New Jersey, 1982.
- [42] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.
- [43] Qualnet Simulator, Version 3.9. Scalable Network Technologies, Inc., Culver City, CA, USA, 2005. <http://www.scalable-networks.com>.
- [44] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.
- [45] D. Saha and A. Mukherjee. Pervasive computing: A paradigm for the 21st century. *IEEE Computer*, 36(3):25–31, March 2003.
- [46] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.
- [47] M. E. Steenstrup, editor. *Routing in Communications Networks*. Prentice-Hall, 1995.
- [48] Ivan Stojmenović, editor. *Mobile Ad-Hoc Networks*. John Wiley and Sons, January 2002.
- [49] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
- [50] A. Tanenbaum. *Computer Networks*. Prentice-Hall, 4th edition, 2002.
- [51] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, Special Issue on Stigmergy, 5:97–116, 1999.
- [52] V. Vazirani. *Approximation algorithms*. Springer-Verlag, 2001.
- [53] J. Walrand and P. Varaiya. *High-performance Communication Networks*. Morgan Kaufmann, 1996.