VII, n.3 - Autumn 2002

Ant Colony Optimization

by Andrea Roli

Scientists have always been fascinated by the problem solving capabilities of Nature: the evolution of the species, the nest building activity of termites, the way crystal forms out of shapeless materials are only a few of the possible examples. For long, biologists, physicists, and chemists have studied and tried to understand these phenomena, building explicative models. More recently, also computer scientists have become interested in these phenomena: a wide set of nature inspired optimization algorithms have been developed: among others, we may mention genetic algorithms, neural networks and simulated annealing.

A recent and important class of nature inspired algorithms is that of ant algorithms [1]. These are algorithms inspired by the observation of social insect behavior, and in particular by the behavior of ant colonies. In these algorithms, the traditional emphasis on control, preprogramming, and centralization is replaced by an emphasis on autonomy, emergence, and distributed functioning. A particularly successful research direction in ant algorithms, known as Ant Colony Optimization [3,5], is concerned with applications to discrete optimization problems. The aim of this short note is to introduce and briefly describe origins and basic principles of Ant Colony optimization (ACO).

From Nature to models

ACO algorithms have been inspired by direct observation of a colony of real ants. In this experience, a laboratory colony of *Argentine ants* is given access to a food source in an area linked to the colony's nest by a bridge with two branches of different length (see Figure 1).

The branches are arranged in such a way that ants going in different directions (from nest to food or vice versa) must choose one of the two branches. The experimental observation is that, after a transient phase that can last a few minutes, most of the ants use the shortest branch. The selection of the shortest path can be seen as an emergent property of the system: it is the result of a number of probabilistic decisions made by the ants based on local information, without any individual ant having a plan for achieving a "shortest path behavior". Indeed, individual ants are not even aware of the fact that they are selecting the shortest path. Local information is provided by *pheromone*, a chemical substance deposited by the ants while they walk. Ants can also smell pheromone, and their probabilistic decisions are biased in favor of paths marked with higher amounts of pheromone.

The dynamics of the experiment are the following: at the beginning, no pheromone is laid on the branches and the ants do not have any bit of information about the branches length. However, since one branch is shorter than the other, while the process iterates, the shorter branch receives pheromone at a higher rate than the longer one and, eventually, it will be selected by almost all ants of the colony.

Andrea Roli

An important role is played here by positive feedback: if an ant chooses the left branch at time t, it will increase the amount of pheromone on that path, therefore, for a subsequent ant reaching the decision point at time $T>t$, the probability of choosing the left branch is increased. This mechanism is not sufficient to enable the colony to select the shortest branch, though. In fact, since positive feedback reinforces previously taken decisions, it might force the selection of the wrong path as a consequence of initial random fluctuations.

The so-called *differential path length effect* now comes into play: the longer a branch, the higher the time required to traverse it, thus the smaller the number of ants that will take that branch in a time interval, hence the smaller the amount of pheromone deposited on it. Additionally, another important element is pheromone evaporation that enables the system to forget possibly wrong past decisions. The effect of this mechanism is to prune the tree of possible paths, setting to a value greater than zero those that are more frequently chosen. In fact, in paths that rarely receive pheromone, or on which pheromone has been deposited just at the beginning, pheromone evaporates until disappearing completely. Conversely, on frequently traversed paths, the effect of evaporation is counterbalanced by new pheromone added by ants.

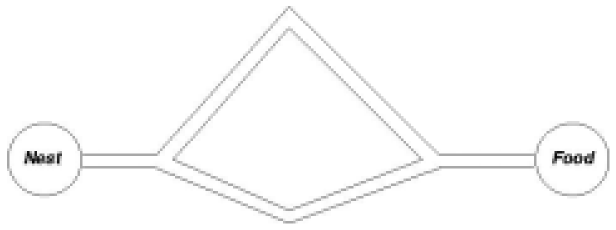

Fig. 1. - Experimental setting: nest and food are connected by two paths of different length.

From models to algorithms

Heuristic algorithms for discrete optimization can be designed by adapting the model of real ants' foraging behavior. *Ant System* (AS) is the earliest example of this kind of algorithms. AS was first applied to solve the *Traveling Salesman Problem* (TSP) and it achieved encouraging results, yet not competitive with the state of the art on large problem instances. AS has been further modified and extended, and several variants have been designed. In recent years, a general framework for ant algorithms (and their variants) applied to combinatorial optimization has been proposed. This is called *Ant Colony Optimization* (ACO) metaheuristic. In the following, we informally define the problem representation adopted in ACO and we outline the high level algorithm.

The main entities of ACO are artificial ants (hereafter called simply ants) that "walk" on a connected graph, called *construction graph* $G=(C,L)$, where arcs L (*connections*) fully connect nodes C (*components*). The combinatorial problem at hand is mapped onto G in such a way that feasible solutions to the original problem correspond to paths on G. Connections, components, or both, can have associated a pheromone trail and a heuristic value. Pheromone trails provide a kind of distributed long-term memory which encodes the history of the whole ants' search process. Heuristic values represent a priori information on the problem or dynamic heuristic information.

In ACO, ants are no longer reactive agents without memory. They are essentially stochastic solution construction procedures, equipped with memory to store the solution built (i.e., the path described in the graph) and heuristic information. Ants move on the basis of a construction policy that is a function of the problem constraints. They build paths by incrementally adding a node, among the feasible ones, to the current path by taking probabilistic decisions. A *state transition rule* returns the probability of adding a node to the current path. Once a solution is completed, the ant evaluates it and retraces the same path backward depositing on nodes (or arcs) an amount of pheromone proportional to the solution quality. This action is called online *delayed pheromone update*. It is also possible for the ants to add or remove pheromone during the path construction, executing the so called *step-by-step pheromone update*. The information provided by pheromone trails will guide the solution construction of future ants. Pheromone evaporation is achieved by executing a procedure called *pheromone trail evaporation*, which decreases pheromone on the whole graph (usually by decreasing each amount τ of a fraction $\rho\tau$, with $0<\rho<1$). ACO includes also optional activities called *daemon actions*, which are non local procedures, such as the application of local search to solutions, or the pheromone update on the path corresponding to the best solution found from the beginning of the run.

The high level scheme of ACO is the following:

```
procedure ACO metaheuristic
    ScheduleActivities
        ManageAntsActivity()
        EvaporatePheromone()
        DaemonActions() {optional}
    end ScheduleActivities
end ACO metaheuristic
```

The ScheduleActivities construct does not specify how the three inner activities are scheduled and synchronized and the designer is free to specify how these procedures interact.

A simple example: Ant System for the Traveling Salesman Problem

As an example of a specific implementation of ACO, we briefly describe Ant System applied to the TSP.
The TSP can be represented in ACO as follows:
- nodes of G (the components) are the cities to be visited;
- a solution is an Hamiltonian tour in the graph;
- constraints are used to avoid cycles (an ant cannot visit a city more than once).

The ant colony is composed of m ants that iterate the same sequence of actions until a termination condition is verified. At the beginning of the cycle ants are randomly put on the cities (nodes of the construction graph G). Starting from its start city, an ant moves from city to city to build an Hamiltonian tour. For the ant k, the probability of moving from city i to city j is given by the following *state transition rule*:

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} & \text{if } \quad j \in \mathcal{N}_i^k \\ 0 & \text{otherwise} \end{cases}$$

where τij is the pheromone laid on arc *(i,j)*, η_{ij}=1/distance*(i,j)* is the heuristic, the parameters α and β balance the relative influence of pheromone and heuristic, and $\mathcal{N}_i^k$ is the set of cities not yet visited by ant *k*.

The *pheromone trail evaporation* is ruled by the following formula:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} \qquad \forall (i,j)$$

where ρ (0<ρ<1) is the evaporation parameter.

Finally, the *delayed pheromone update rule* adjusts pheromone so as to make more preferable arcs belonging to short tours:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \qquad \forall (i,j)$$

$$\Delta\tau_{ij}^k = \begin{cases} 1/L^k & \text{if arc } (i,j) \text{ is used by ant } k \\ 0 & \text{otherwise,} \end{cases}$$

where L^k is the length of the tour built by ant *k*. In Ant System no daemon rules are applied (e.g., every constructed solution can be used as initial solution for the application of local search, in a similar way to hybrid genetic algorithms).
It is important to note that, besides the natural metaphor, ACO can be seen from an operative standpoint as an adaptive stochastic constructive procedure, possibly joint with a solution improvement phase (e.g., local search).

Applications and current research trends

ACO algorithms have been applied successfully to a large number of NP-hard problems, including the Quadratic Assignment Problem, the Job-Shop Scheduling Problem, the Vehicle Routing Problem, and the Timetabling Problem [4]. It is important to note that the stochastic constructive procedure alone is often not enough to obtain good results, and in most of the cases it is necessary to improve the solutions constructed by the artificial ants via a local search procedure, as mentioned in the previous section.

One of the strengths of ACO is its capacity of adapting to changing conditions: therefore, time-varying problems (e.g., dynamic TSP or routing in telecommunications networks), where topology and costs may change dynamically, represent a particularly successful setting for ACO. For instance, AntNet [2] has been developed to solve the network routing problem in a packet-switched telecommunications network under stochastic and time-varying traffic conditions.

Current research is focused on the theoretical foundations of ACO (convergence theorems have recently been proven [7]) and the investigation of relations between ACO and other search methods such as gradient descent and Monte Carlo algorithms [6]. An additional direction concerns the design of efficient implementations of ACO, both for sequential and parallel systems..

Further and up-to-date information about ACO is available at the ACO webpage:
http://www.aco-metaheuristic.org

Andrea Roli
aroli@deis.unibo.it

Acknowledgments
I thank Marco Dorigo for valuable comments and suggestions during the preparation of this note.

References

[1] Bonabeau E., Dorigo M., Theraulaz T., *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY: Oxford University Press, 1999.
[2] Di Caro G., Dorigo M., (1998). AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research (JAIR)*, 9: 317-365, 1998.
[3] Dorigo M., Di Caro G., *The Ant Colony Optimization Meta-Heuristic*. In D. Corne, M. Dorigo and F. Glover, editors, New Ideas in Optimization, McGraw-Hill, 11-32, 1999.
[4] Dorigo M., Di Caro G., Gambardella L. M., Ant Algorithms for Discrete Optimization, *Artificial Life*, 5(2):137-172, 1999.
[5] Dorigo M., Stützle T., *Ant Colony Optimization*. Cambridge, MA: MIT Press/Bradford Books, 2003.
[6] Meuleau N., Dorigo M., Ant Colony Optimization and Stochastic Gradient Descent, *Artificial Life,* 8 (2): 103–121, 2002.
[7] Stützle T., Dorigo M., A Short Convergence Proof for a Class of ACO Algorithms, *IEEE Transactions on Evolutionary Computation*, 6 (4): 358-365, 2002.