MONACO - Multi-Objective Network Optimisation Based on an ACO

P. Cardoso*, M. Jesus† and A. Márquez‡

13th November 2003

Abstract

The Ant Colony Optimisation Algorithm (ACO) supports the development of a system for a multi-objective network optimisation problem. The ACO system bases itself on an agent's population and, in this case, uses a multi-level pheromone trail associated to a cost vector, which will be optimised.

1. Introduction

Evolutionary and adaptive computing algorithms emerged in the last decades, taking advantage of the computer age, namely: the processing speed and the amount of memory available. Those global algorithms, usually based on random meta-heuristic searches, almost always try to imitate some natural process like: the group insect behaviour usually called swarm algorithms (Ant Colony Optimisation algorithm - ACO) [3, 8], the evolutionary Darwin's theory (Genetic Algorithm)[9, 5], physics behaviours (Simulated Annealing algorithm) [5, 7]. In particular, the swarm algorithms are based in a collective intelligence, defined as the ability of a group to solve problems more efficiently than its individuals [6].

*Universidade do Algarve (EST). Dep. Eng. Electrotécnica. Email: pcardoso@ualg.pt
†Universidade do Algarve (EST). Dep. Eng. Civil. Email: mjesus@ualg.pt
‡Universidad de Sevilla (FIE). Dep. de Matemática Aplicada I. Email: almar@us.es

In this paper, we introduce an ACO based algorithm for a multi-objective network optimisation problem with a time discrete approach. In our method, the optimisation process uses a single colony with a multi-level pheromone trail and an heuristic that allowed us to introduce a random-heuristic formula to construct optimised paths.

Some of the traditional methods use an objective function that joins in a single expression the costs. This expression is frequently quite artificial due to the different nature of values included. Another approach suggest the use of a multi-colony solution with the exchange of information between colonies [4].

Therefore, this paper is divided in five sections. Next, we introduce some preliminaries necessary to define the problem, namely: the network definition and the ACO algorithm. In the third, we describe our optimisation method, followed with the presentation of the MONACO algorithm, some results and their discussion. In the last one, we draw some conclusions and further work.

2. Preliminaries

In this section, we will dedicate ourselves to introduce some notation, to present a concise description of the problem as well as a small introduction to the ACO.

§ 2.1. Networks.— Let us consider a network as

$$\mathcal{N} = (\mathcal{V}, \mathcal{E}, \mathcal{C}, \mathcal{O}) \tag{1}$$

where

- $\mathcal{V}$ is the set of nodes (points in the Euclidean plane).
- $\mathcal{E}$ is the set of edges, each defined by a pair of nodes, (u, v), and a certain amount of ticks necessary to throughout each one.
- $\mathcal{C} : \mathcal{E} \to (\mathbb{R}_0^+)^n$ is a vector function such that, for each $(u, v) \in \mathcal{E}$,

$$\mathcal{C}(u, v) = \left(c_{u,v}^{(1)}, c_{u,v}^{(2)}, \ldots, c_{u,v}^{(n)}\right) \tag{2}$$

  is the cost vector. We define the path final cost vector as the sum of the cost vectors associated to its edges. In other words, the path

$\pi_{s,t} = (u_0 = s, u_1, u_2, \ldots, u_k = t)$ as a total cost vector:

$$\mathcal{T}^{\pi_{s,t}} = \left(\sum_{i=0}^{k-1} c^{(1)}_{u_i,u_{i+1}}, \sum_{i=0}^{k-1} c^{(2)}_{u_i,u_{i+1}}, \ldots, \sum_{i=0}^{k-1} c^{(m)}_{u_i,u_{i+1}} \right). \tag{3}$$

- $$\mathcal{O} = \{O_t : t \in I\} \tag{4}$$

  is a set of origin/destination matrices such that $O_t = [o^{(t)}_{i,j}]$, where $o^{(t)}_{i,j}$ is the flow from node i to node j at tick $t \in I$ (I is our time window of observation).

§ 2.2. Ant Colony Optimisation.— In the evolutionary and adaptive algorithms one of the most recent is the Ant Colony Optimisation (ACO) computational paradigm introduced by Marco Dorigo [2, 3]. As the name suggest, this algorithm try to mimic the behaviour observed in an ant colony, where all the individuals work to the benefit of the colony, basing there options on a set of basic reactions (find food and take it to nest, leave pheromone trail, etc) despite having a superior ant.

In this manner, the basic idea of the ACO artificial system is to resemble the cooperative behaviour of the colony using: a colony of agents called artificial ants, an indirect communication between the ants, supported by an artificial pheromone trail reflecting the experience of past agents while solving the problem, and an heuristic to improve the search.

Algorithm 1 ACO - Ant Colony Optimisation Algorithm

```
1: Initialise the pheromone trail.
2: while stopping criteria is not met do
3:    for all ants do
4:       Construct a new solution using the current pheromone trail.
5:       Evaluate the solutions constructed.
6:    end for
7:    Update the pheromone trail.
8: end while
```

In [1] we used ACO algorithm to solve the k-Shortest Paths problem (ACO-KSPP). In each cycle of the ACO-KSPP, all the (artificial) ants try to find a shortest path between nodes s and t. To do it, each ant is placed in node s and constructs the path by the ordered addition of feasible (i.e., not yet visited) nodes, considering two measures: closeness to t and the

(artificial) pheromone trail in the edge between the current node and all the feasible nodes. The k shortest paths found are kept to establish the solution.

The ACO algorithm allows solving hard combinatorial problems like: Travelling Salesman Problem (TSP)[3], the Vehicle Routing Problem (VRP), the Quadratic Assignment Problem (QAP) [3] and the Job-shop Scheduling Problem (JSP) [3].

To improve the performance and the quality of the solutions, the algorithm must have a collection of capabilities, namely: memory to recognize instances in futures occurrences, *global vision* over the network allowing to recognize critical points in the network to find solutions and the capability to communicate.

In conclusion, some of the ACO main features are its versatility, its robustness [3] and the fact that it is a global method. These desirable properties counterbalance the fact that, for some applications, algorithms that are more specialized can outperform the ACO.

3. MONACO - Multi-Objective Network Optimisation using ACO

In this section, we introduce the MONACO method by describing the considered problem in more detail, followed by an explanation of the path construction strategy and ending with the pheromone update process.

§ 3.1. Problem Definition.— We consider the problem of determining paths that minimize the total costs of a vector defined in (3), for the flows generated between each pair of nodes. This values are given by the origin/destination matrices, $\mathcal{O}$ (4). The number of possible paths from a starting node s to a terminal node t, are usually very high [1] and should be chosen, somehow, trying to minimize several costs associated to the network.

Therefore, our problem is to find the paths that minimize a multi-objective cost vector, using the weights associated to each cost. Usually, when a multi-objective problem is proposed, it is normal to introduce an unit conversion and assemble an objective function that is the weighted sum of the converted costs. In this approach, we use a random-heuristic formula to construct the paths that only require the individual edge costs.

Furthermore, we will consider that there are a certain amount of ticks

necessary to throughout the edges, i.e., after the beginning of the process, there is a queue in each edge corresponding to the flow elements that have not yet arrived to its end.

§ 3.2. Paths Constructions.— Let us define an ant as a certain amount of flow with the same origin s, same destination t and leaving s at the same tick. Then, the path for that flow is constructed by the ant taking in account a multi-pheromone trail and an heuristic to improve the decision to choice a suitable route. Those pheromone trails, in the same number as the number of costs (m), represent the weight of an edge regarding the objective of guiding the ant to t. So, for each node t, there are m pheromone trails that catalyse the construction of the path toward the terminal node, i.e., the pheromone trails deposited in the edges, symbolize the value of the edge to achieve t.

Therefore, knowing the necessary ticks to cross an edge, the problem resumes to choose next edge whenever the ant arrives into a node that is not t (the ant stops when it reaches t). Mathematically, the probability of going from node u to node v is given by the formula

$$p_{uv} = \begin{cases} \dfrac{d_{v,t}^{-\alpha_0} \prod_{k=1}^{m} \left(\tau_{u,v}^{(k,t)}\right)^{\alpha_k}}{\sum_{\{w:(u,w)\in\mathcal{E}\}} \left[d_{w,t}^{-\alpha_0} \prod_{k=1}^{m} \left(\tau_{u,w}^{(k,t)}\right)^{\alpha_k}\right]} & \text{if } (u,v) \in \mathcal{E} \\ 0 & \text{if } (u,v) \notin \mathcal{E} \end{cases}, \tag{5}$$

where $\tau_{u,v}^{(k,t)}$ is the of quantity k-pheromone to node t in edge (u,v) (analogous for $\tau_{u,w}^{(k,t)}$), $d_{v,t}$ is the Euclidean distance between nodes v and t (analogous for $d_{w,t}$) and $\alpha_k \in \mathbb{R}_0^+$ $(k = 0, 1, \ldots, m)$ are parameters that emphasis the heuristic $(k = 0)$ and the relative importance of the k-pheromone trail $(k = 1, 2, \ldots, m)$, i.e., the relative weight of the k costs in the final value.

So, supposing that the ant is in a node u and wants to go to t, a random-heuristic selection of the next node is taken from the adjacent nodes, considering their ability in the construction of a good path. The heuristic used give preference to nodes that are closer to t by placing, in the formula, the inverse of the Euclidean distance from those nodes to t. This can be seen a guidance ability.

In Figure 1 is presented a section of a network, where two costs are taken into account considering the values given in Table 1. Suppose also that the ant is on node a and t is its terminal node.

Then using (5) with $\alpha_0 = \alpha_1 = \alpha_2 = 1$ the probabilities are $p_{ab} = 0.03$, $p_{ac} = 0.43$, $p_{ad} = 0.29$ and $p_{ae} = 0.26$. If we take $\alpha_0 = \alpha_2 = 1$ and $\alpha_1 = 2$ then $p_{ab} = 0.02$, $p_{ac} = 0.36$, $p_{ad} = 0.40$ and $p_{ae} = 0.22$. We can also see that the variation of the probabilities is closely linked to the values of the parameters.

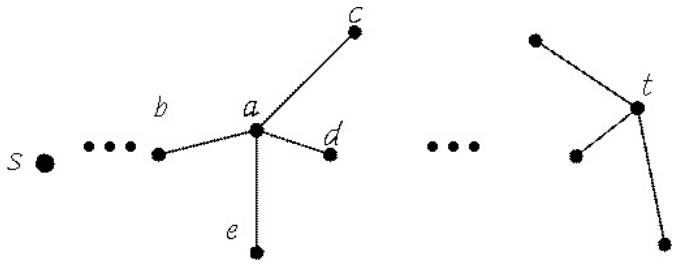

Figure 1: Random path determination example

node (x)	distance from x to t	$\tau_{Ax}^{(1,t)}$	$\tau_{Ax}^{(2,t)}$
B	4	1.0	1.0
C	1	1.5	2
D	1.5	2.5	1.5
E	2	1.5	3

Table 1: Random path determination example.

§ 3.3. k-Pheromones.— Relatively to the pheromones trails, when the process starts, they are all set equal. The construction of a pheromone trail is achieved considering a set of cycles. In each cycle, a pre-determined number of ticks are used and an ant arriving to a terminal node contributes to the variation of the pheromone trails. After each cycle the k-pheromones for node t are updated using the formula:

$$\tau_{uv}^{(k,t)} = \rho_k \tau_{uv}^{(k,t)} + \Delta\tau_{uv}^{(k,t)}, \quad k \in \{1, 2, \ldots, m\},\ t \in \mathcal{V} \tag{6}$$

where $\tau_{uv}^{(k,t)}$ is the quantity of k-pheromone to node t in edge (u, v), $0 < \rho_k \leq 1$ $(k = 0, 1, \ldots, m)$ is the persistence of the trail ($1 - \rho_k$ the is trail evaporation) and $\Delta\tau_{uv}^{(k,t)}$ is the quantity of k-pheromone leaved by the ants that went through (u, v) with destination t, in this cycle. That quantity is usually the inverse of the k cost of the path determined by the ant. That is, if $\mathcal{W}$ is the set of all ants that went to t through (u, v) and $\pi_{s,t}^{(a)}$ represents the path of $a \in \mathcal{W}$, then

$$\Delta\tau_{uv}^{(k,t)} = \sum_{a \in \mathcal{W}} \frac{Q}{\mathcal{T}_k^{\pi_{s,t}^{(a)}}} \tag{7}$$

where $\mathcal{T}_k^{\pi_{s,t}^{(a)}}$ is the k component of the cost vector defined in (3) and Q is a constant related to the amount of pheromone laid by ants.

4. Computational Experiences

Within this section a computational model named MONACO will be described using previous theoretical concepts, as reported in Section 3. Algorithm 2 globally sketches the MONACO computational model and Algorithm 3 presents a more detailed fase, focused on the ants update.

Algorithm 2 MONACO Algorithm

for all $t \in \mathcal{V}$ **do**
  for all $k \in \{1, 2, \ldots, m\}$ **do**
    Initialise the k-pheromone trails $\left(\tau^{(k,t)}\right)$
  end for
end for
Load the Network
repeat
  for $T \in I$ **do**
    Update packets all ready in the network (Algorithm 3).
    Add new packets to the networks (based on the origin/destination matrix for tick T).
  end for
  for all $t \in \mathcal{V}$ **do**
    for all $k \in \{1, 2, \ldots, m\}$ **do**
      Update the k-pheromone trails to t using (6).
    end for
  end for
  Remove all remaining ants from network
until Stopping criteria

Our computational model was implemented using *C++* programming language and a set of tests were used to verify its robustness, liability and accuracy. Those tests were executed on a *Pentium 4 - 2.0Ghz PC* with *256 Mb* of *RAM*.

In Figure 2 we present two examples of networks used in our tests: network 1 with 18 nodes and 27 edges and network 2 with 25 nodes and 40 edges. For each problem, we considered a cost vector with two components: number of ticks and Euclidean distance. We also have considered 100 ticks per cycle, 20 cycles as a stopping criteria for Algorithm 2, persistence parameteres $\rho_d = \rho_t = 0.5$ and $\alpha_d, \alpha_t, \alpha_h \in \{1, 2\}$ (were α_d, α_t and α_h are the parameters present in formula (5), associated, in this case, to the distance cost, the ticks cost and the heuristic, respectively). We used 122400 and

Algorithm 3 Ants update

```
1: for all ants in network do
2:   if ant arrived at the end of an edge then
3:     if ant arrived to t then
4:       for all k ∈ {1, 2, ..., m} do
5:         Update Δτ^(k,t) using (7)
6:       end for
7:       Remove ant from network
8:     else {not yet in t}
9:       Use (5) to random-heuristically determine next edge and put ant
         there.
10:    end if
11:  end if
12: end for
```

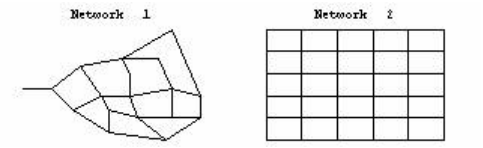

Figure 2: Example of tested networks

240000 ants/cycle for network 1 and network 2, respectively.

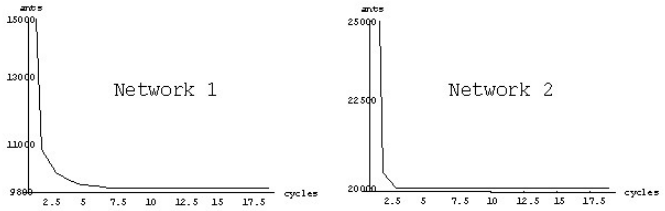

Figure 3: Number of ants present in the network at the end of the cycles - $\alpha_d = 2$ and $\alpha_t = \alpha_h = 1$.

Figure 3 presents the variation of the number of ants in the network per cycle. Straight forward we can conclude that in few cycles a good trail of pheromone is created, thus implying an important decrease in the number of ants that have not reached t, i.e., less ants are 'lost' as the number of cycles increases. That number suggest a stabilization above the number of ants created in the last cycles that have not enough ticks to reach t.

In Figure 4 we can see the variation on the thickness of the edges in both networks. That thickness represents the pheromone levels in the edges and

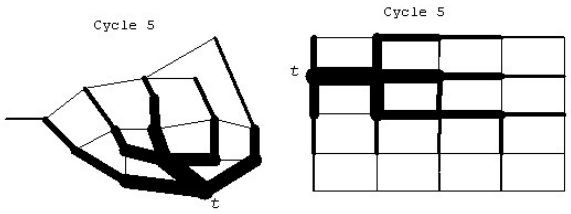

Figure 4: Pheromone trails for node t - the thickness of the edges corresponds to the quantity of pheromone present (relative to distance cost) in cycle 5 ($\alpha_d = 2$ and $\alpha_t = \alpha_h = 1$).

gives us an idea of the possible choices made by the ants.

Relatively to the influence of the parameters, if we compare the first network in Figure 4 and Figure 5, we can see that, due to the fact that in the second case $\alpha_t > \alpha_d$, in the edges were the number of ticks necessary to cross were lower (higher velocity), the line is thicker. Depending on the alpha parameters, the pheromone trails reflect the weight of the costs associated to those factors.

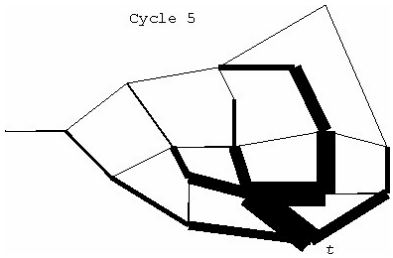

Figure 5: Pheromone trails for node t - the thickness of the edges corresponds to the quantity of pheromone present (relative to ticks cost) in cycle 5 ($\alpha_t = 2$ and $\alpha_d = \alpha_h = 1$).

5. Conclusions and Further Work

The practical use of MONACO requires many improvements that can be summarised in the following item:

- There should be some restriction to the network capacities, namely: node and edge capacities.

- The ability to assign dynamism to the network, namely: variations in the edge and node capacities.
- Modelling some real problem requires RAM and computational capabilities needed to solve and represent it. A possible solution may be in the construction of heuristics that are more efficient, some possible clusterisation or a distributed/parallel algorithm.

References

[1] Pedro Cardoso, Mário Jesus, and Álberto Márquez. Optimization system in networks using aco. In *Actas de las III Jornadas de Matemática Discreta y Algorítmica*, pages 89–94. Universidade de Sevilla, September 2002.

[2] Marco Dorigo. http://iridia.ulb.ac.be/ mdorigo/aco/aco.html, March 2003.

[3] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transaction on Systems, Man and Cybernetics*, 26(1):29–41, 1996.

[4] Martin Middendorf, Frank Reischle, and Hartmut Schmeck. Multi colony ant algorithms. *Journal of Heuristics*, (8):305–320, 2002.

[5] Ian Parmee. *Evolutionary and Adaptative Computing in Engineering Design*. Springer-Verlag, 2001.

[6] Mitchel Resnic. *Turtles, Termites, ans Traffic Jams: Exploration in Massively Parallel Microworlds*. MIT Press, 1999.

[7] Sadiq Sait and Habib Youseff. *Iterative Computer Algorithms with Applications in Engineering*. IEEE - Computer Society, 1997.

[8] Peter Tarasewich and Patrick McMullen. Swarm intelligence - power in the numbers. *Communications of the ACM*, 5(8):149–159, 2002.

[9] Michael Vose. *The Simple Genetic Algoritm: Foundations and Theory*. MIT Press, 1999.