GLOBAL MEMORY STRUCTURE FOR ANT COLONY OPTIMIZATION ALGORITHMS

Ángel Goñi, Paula Cordero

Abstract: Ant Colony Optimization (ACO) is a computer emergence model to solve problems by swarm intelligence. The aim of this paper is to provide ant colonies with a global memory structure (GMS) so that the time needed to resolve a problem decreases drastically. In the past few years ACO have become a strong alternative to classic algorithms. However, the biggest disadvantage of ACO is the lack of structures which can provide every individual of the population with simple memory mechanisms. The GMS presented in this paper is applied to cleaner robots that must search a gallery looking for piles of marks and clean them. It is based on the variation of the common map by deleting all the superfluous nodes which appears as the resolution of the problem progresses. A node is considered as superfluous when it is useless for every ant and it delays every route of the colony. All the robots must share and update the GMS when they find any superfluous node. The extra process charge needed for executing the memory in parallel to ant's activities is absorbed by the time saved in the resolution of the problem which depends on the characteristics of the map and its abstracted graph.

Keywords: Ant Colony Optimization, Swarm Intelligence, Memory.

ACM Classification Keywords: I.6. Simulation and Modelling, B.7.1 Advanced Technologies, I.2.8 Problem Solving, I.2.11 Distributed Artificial Intelligence

Conference: The paper is selected from Seventh International Conference on Information Research and Applications – i.Tech 2009, Varna, Bulgaria, June-July 2009

Introduction

The algorithms known as Ant Systems (AS) are based on the observed behavior of the ants of a colony during the food search they carry out for their survival (also called ant colony optimization or ACO). This work focuses its attention in AS although the research area in Swarm Intelligence is developing several algorithmic theories derived from other insects. The reason for turning the research efforts towards the natural behaviors seen on nature it is of relative common sense: the nature has had more than a million years to purify algorithms that assure an optimal use of the resources during the functions of survival of species.

The study of the various natural processes is being in the last years a source of inspiration of incalculable value in different investigation areas like DNA computing, membrane computing, synthetic biology, artificial neural networks or genetic algorithms. In those cases in which we want to find the optimal behavior of several robots in a community we can base our developments on ant colony optimization. It seems difficult and interesting to understand how almost blind animals, moving approximately randomly, can find the shortest way from its nest to the food source and return. The ant when moving leaves signals which are called pheromones so that the others can follow it.

The methodological philosophy in which ACO is included is called emergence [Clark, 1999]. Emergence is central to the theories of integrative levels and of complex systems. It makes reference to those properties of those processes of a system which are not reducible to the properties or processes of its constituent parts. The emergence concept is related closely to autoorganization concepts and it is defined in opposition to the concepts of reductionism and dualism. The emergence concept has acquired renewed force as a result of the height of complexity sciences and plays a fundamental role in the philosophy of the mind and the philosophy of Biology.

This model of algorithmic computation was first presented in the doctoral thesis of Marco Dorigo [DOR] who in 1996 published three possibilities to solve the well known Traveling Salesman Problem. The main characteristic of Ant Colony Optimization algorithms is the explicit use of elements that belongs to previous solutions. This characteristic was also introduced by genetic algorithms (GA's) which are based on the results returned by previous generations in order to improve the results of the next one. However, in ACO the probabilistic distribution of partial results is explicitly detailed in previous steps without using extra algorithms (like selections and crossovers in the case of GA's). During the next section it is explained how ACO algorithms work in the domain of problems tackled in this paper. The rest of the sections show how a global memory can be very useful for their resolution.

Ant Colony Optimization

Swarm intelligence is based on the observation of nature in order to construct computational models [Maniezzo, 2004].. Figure 1 show graphically and in a very intuitive way how the algorithms based on ant colonies work by showing the behaviour of real ants. In the first image (figure 1a) the ants travel straight to the food. There is only one path and only one possibility to place the pheromones. In figure 1b an obstacle is situated in the middle of the route used by the ants. In that moment, the ants that are behind the obstacle can smell the pheromones of the ants in front so they start looking for alternative paths to reach the lost route (figure 1c). What happens in the last step that can be seen in figure 1d is that the ants decided to travel across the shorter path which avoids the obstacle. That decision is based on the amount of pheromones present in the path. Those ants who first decided to take the shorter path reached the original route before those ants who first decided to take the longer one so the shorter path will be full of pheromones quickly.

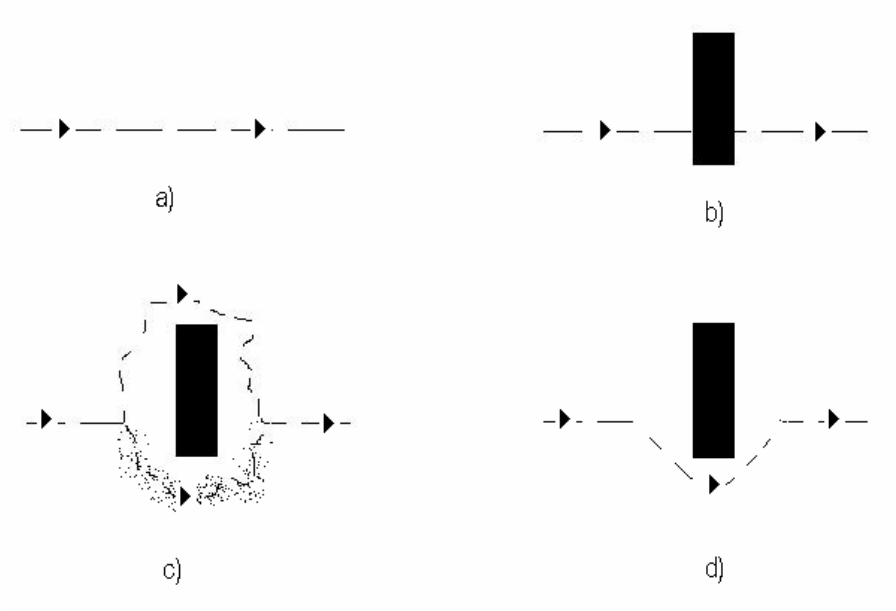


Figure 1. Ant's behaviour

It is necessary to translate the behaviour of ants explained above to computer software. In order to construct a model based on an analogy of nature it is created a distributed computer model. This model creates a process for each ant so that they can work in parallel following each ant of the colony its own synchronism. The ants can also be activated at the same time. This last option is better when trying to find improvements to the algorithm. The pseudo-code of the entire procedure would be similar to:

```
procedure ACO algorithm()
        while (final_objective_not_satisfied)
                ant_action()
                pheromone_evaporation()
                memory_evaluation()
        end while
end procedure
```

The evaporation of pheromones is done after the action of every ant. This function is always executed in a synchronous way. That means that in every iteration of the algorithm we evaporate the same amount of pheromones in all the spaces of the map which pheromones are placed. This function is the target of lots of studies and developments because the way the pheromones are evaporated can change the resolution of the problem.

The function ant_action() is subdivided in several steps. This function handles all the input ants of the problem in a synchronous or asynchronous way like it was explained before. The pseudo-code that leads the action of the ants would be similar to:

```
procedure ant_action()
        decide_movement()
        put_pheromones()
        if (target_found)
                eat_target()
        end
end procedure
```

The function put_pheromones() is also very important to the correct resolution. The way each ant put the pheromones affects very much the time of execution of the method. It is not trivial the fact that an ant could put one pheromone unit or two. If the model evaporates one pheromone unit per iteration it could easily been changed the time that pheromones are present in the map.

In this work we focus our attention on the function decide_movement() in which the global memory structure presented in this paper is used. If several paths part from the point where an ant is, that ant must decide which way should it go. At the beginning of the problem all the paths are pheromone free so the probability of choosing one path ore another is exactly the same. As the problem is being resolved the paths can contain different amounts of pheromones and this function must choose probabilistically the path to follow. The global memory structure provides the ant the ability of avoiding the useless paths.

Memory Requirements

Over the last years there are a lot of papers published in which the researchers try to apply memory structures to ant colonies [Manfrin, 2006]. That is due to the fact that ACO have become a strong alternative to classic algorithms. However, the biggest disadvantage of ACO is the lack of structures which can provide every individual of the population with simple memory mechanisms.

The most intuitive problem in which we are bound to apply ACO is any problem that belongs to the “Check Problem Class” (CPC). We would say that a problem is included in CPC when the input of such a problem consists of a set of three elements:

- A number n of input ants. The problem starts with n ants which are seen like the processes we can manipulate in order to solve the problem. This number can be variable during the resolution.
- A graph G = (N, E) where N is the set of nodes of the problem and E is the set of edges. Each edge is in the form E = (N1, N2) where N1 and N2 are the nodes that this edge connects.
- A number m of input targets. The targets are spread along the input graph. The characteristic of those objectives depends on each problem. Usually the objectives are in the nodes of the graph because the graph represents a set of rooms in a building and the edges the doors existing between the rooms. In the case that two rooms communicate through a long corridor, the objectives could be on the edge. However, the case we face in this paper is the first.

The main objective of those problems is to check that a concrete goal is achieved or to check the state of all the targets. In the next section we will see how to implement a memory structure over a problem of the first class, in which the n initial ants will try to complete a concrete mission. The other class would contain those problems related to the typical problem of the security in an art gallery, where it is very important to develop a trusty security device which could cover the entire gallery during the night and check if the different objects are all in their correct place.

The problem solved in this paper and improved by a global memory is a problem in which the ants (cleaner robots in our problem) should walk around the rooms of a building floor checking for food (dirty marks in our problem) and eat it (clean them would be the action for our robots) if they found it. After carrying out its action with a target, an ant must go on walking around the map. The problem will be finished when all the targets disappear. After finishing the problem (all the dirty marks have been cleaned) the ants should return to their starting point. The global memory structure (GMS) developed in this work will help the ants to know when the goal is achieved so that they do not have to continue wandering the map.

GMS consists of a knowledge base of the superfluous nodes of the graph so that the probabilistic movement decision of the ants takes it into account in order to avoid non useful movements. A superfluous node is that one that has no objectives in it (a room without dirty marks) and does not make shorter the path of an ant who passes through it. In order to incorporate GMS to the robots they must store the information about the paths they followed in the past so that they could know how costly one path or another is. The cost of the paths is calculated by all the rooms or graph nodes that the ant has passed through during its path. All the nodes have the same cost C so that $C(node_i) = 1, \forall \; i \in [1...N'last]$.

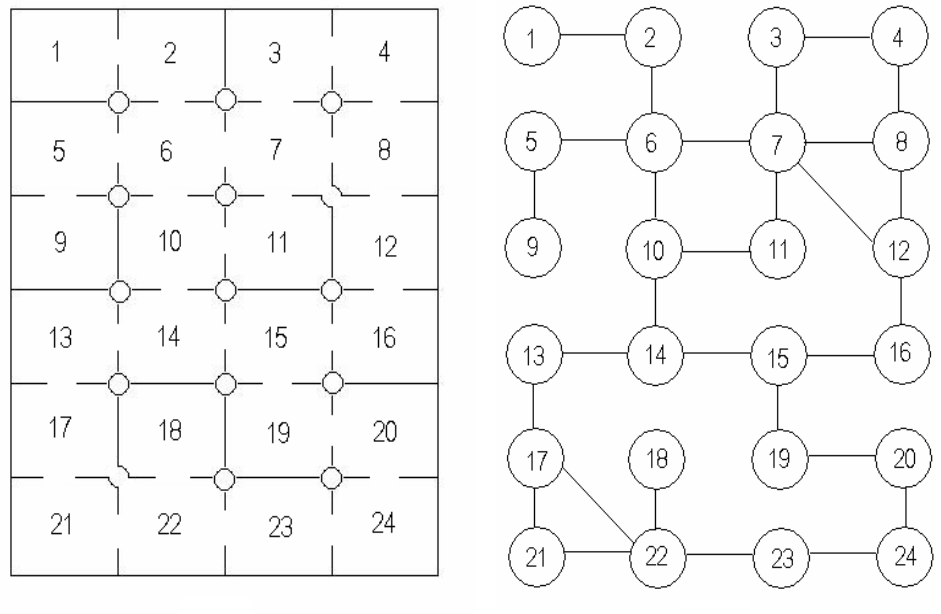
The memory structure will work as a parallel process so that all the ants can have access to the information about the superfluous nodes of the graph in order to avoid travelling along them. In this way, all the ants will collaborate to create the GMS and all of them will also be benefited from it. This structure will need more computability charge in each ant because of the knowledge base update necessary in every movement. However, the time saved thanks to GMS in solving the problem exceeds the computability charge used.

In the next section there is an example explained about how to add GMS to a single problem solved with ant colony optimization.

Superfluous Nodes Elimination

In this section it is explained how the global memory structure works in an ant colony. In the present problem we have several input robots which are all in a concrete room of an art gallery. As they are cleaner robots, we suppose that this room is the room of the building where the robots are kept. That room would be the nest in an analogy with real ants. All the robots must work in a complete concurrent and asynchronous way in order to simulate a real ant-algorithm.

The floor plan of the gallery can be seen in figure 2a. Each room $r_i \; \forall \; i \in [1..24]$ must have at least 1 connection and a maximum of 8 connections with the surrounding rooms. A room has as an attribute an array of rooms connected to it. For example, room r_7 has the attribute $connected_7 = [r_3, r_8, r_{12}, r_{11}, r_6]$. An abstraction of the gallery is shown in figure 2b in the form of a graph. That graph is the real input of the problem. Another attribute of each node is the number of marks or targets there are in that room. As the aim of this paper is to show how to create a global memory structure, the attribute *pheromone* is omitted in this example. A novel reader would understand how that parameter acts following the brief explanation of previous sections.



a) b)

Figure 2. Map of the gallery and abstraction graph

All robots are initially kept on room 1 and they do not know where the targets are. When the problem starts, all the robots must start wandering around the map following the actions we have programmed in advance as it was been seen previously in this paper. In order to decide a movement each robot takes into account all the rooms which surround the actual room in which the robot is in. What the GMS is going to do is to eliminate from the initial graph those nodes which are considered superfluous so that the size of the map decreases. All nodes can become superfluous at any time so it is necessary to check them every time a robot makes an iteration. It is important to notice that the GMS is common for all the ants so it is very important to synchronize the access to it by every robot.

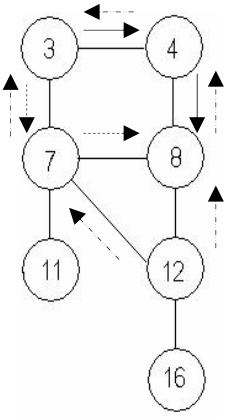


Figure 3. Detail of R_4

Next it is shown how to eliminate a node. In the next example it is eliminated node number 4 because is consider as superfluous. In order to simplify the problem we will focus our attention in the movement across the neighbours of r_4 when the situation is: $r_3 = ([7], [r_4, r_7])$; $r_4 = ([2], [r_3, r_8])$; $r_7 = ([8], [r_3, r_8, r_{12}, r_6])$; $r_8 = ([11], [r_4, r_7, r_{12}])$; $r_{12} = ([9], [r_8, r_7, r_{16}])$ where the first

attribute of a room is the number of objectives and the second attribute is the connections. As the algorithm runs, the number of objectives decreases because when a robot gets into a node he cleans a mark and then decide if moving to another room or staying in the actual room in order to look for more targets.

The first room which will be totally cleaned is room number 4 because the number of targets it has is much slower than the rest of the rooms. After several iterations of the robots, $r_4 = ([0], [r_3, r_8])$. The first condition that a node must have to be ambiguous is then validated in r_4. Now, the process in charge of GMS must study the cost of the paths which go through r_4 and the cost of the alternative paths. In figure 3 are described all the paths that a possible robot can choose travelling through r_4. Between r_3 and r_8 there are two paths whose cost is 3 in both cases: $<r_3, r_8> = (C [r_3, r_4, r_8] = 3, C [r_3, r_7, r_8] = 3)$.

Other routes are:

$$<r_3, r_{12}> = \langle \; C [r_3, r_4, r_8, r_{12}] = 4; C [r_3, r_7, r_8, r_{12}] = 4; C [r_3, r_7, r_{12}] = 3; C [r_3, r_4, r_8, r_7, r_{12}] = 5 \rangle$$

$$<r_7, r_{16}> = \langle \; C [r_7, r_3, r_4, r_8, r_{12}, r_{16}] = 6; C [r_7, r_8, r_{12}, r_{16}] = 4; C [r_7, r_{12}, r_{16}] = 3 \rangle$$

$$<r_8, r_3> = <r_3, r_8>; <r_{12}, r_3> = <r_3, r_{12}>; <r_{16}, r_3> = |<r_{12}, r_3>| + 1$$

$$<r_8, r_{12}> = \langle \; C [r_8, r_{12}] = 1; C [r_8, r_4, r_3, r_7, r_{12}] = 5; C [r_8, r_7, r_{12}] = 3 \rangle$$

As it can be seen in the routes shown above, the cost of every possible path ρ_i that a robot can take between the two nodes of the route which is $C [\rho_i] = \sum C (r_j), \forall \; r_j \in \rho_i$, is always higher when that path cross r_4. The fact that turns r_4 a superfluous node is that there is always a possible path for each route θ with lower or equal cost as the path across r_4. If we divide each set of paths that forms a route into two new subsets S_1 and S_2 in such a form that S_1 contains all the paths that travel along r_4 and S_2 contains all the paths that avoid r_4, we can affirm that $\forall \; \theta \; / r_4 \in \theta \; [\rho \text{'first}..\rho \text{'last}] \Rightarrow \exists \; \rho_i \in S_2 \wedge \; C (\rho_i) \leq C (\rho_j)$ where ρ_j represents any path of S_1.

That is why the node number 4 can be considered as superfluous and it is eliminated from the abstracted graph which is stored in the GMS. Remember that GMS is common for all the robots so that as far as the problem is running each robot will take into account a smaller graph to make the decision of where to move next. The time saved by the lack of non useless movements of the ants increases as the problem grows.

Superfluous nodes will always delay the resolution of the problem so the objective of GMS is to eliminate them.

Conclusion

Ant colonies are able to recollect the food in a very special way. Every individual of the population follow the same path as the rest of the colony which turns to be the shorter path possible. That is due to the capacity they have to communicate between each other by the deposit and evaporation of pheromones. The global memory structure (GMS) presented in this paper guarantees important advantages during the process of decision that every ant of the colony must do every iteration before moving to another node of the graph.

The GMS method is based on processing the map of the problem in a different way than traditional swarm intelligence algorithms do. This method focuses its attention on how to avoid useless spaces in the map so that the ants of the colony do not travel across them. It is extremely necessary to be totally sure about the spaces to eliminate because very often a space with no food can lead the ants to another space where food is abundant. That is why GMS support a matrix of weights that calculates every moment the superfluous nodes that exists on the graph.

For the correct execution of this global memory structure, two different instances of the graph are needed. One of them is the initial input of the problem and the other one must be filled as the ants wander through the nodes. The memory of the ants is in some way downloaded to this second map which is the aim of the structure. By using this technique we can observe that the time saved by ant algorithms increases as the problem gets bigger.

Bibliography

[CNCP] Centro de Computación Nacional. Paraguay. URL: http://www.cnc.una.py/cms/cnc/index.php.

[DOR] Dorigo M., Ant Colony System: URL: http://iridia.ulb.ac.be/dorigo/ACO/ACO.html

[Holland, 1975] J.H. Holland: Adaptation in natural and artificial systems, University of Michigan Press, 1975

[Clark, 1999] Andy Clark: Estar ah__. Cerebro, cuerpo y mundo en la nueva ciencia cognitiva, Editorial Paidos, 1999.

[Lee, 2008] ZJ Lee, SF Su, CC Chuang, KH Liu. Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Applied Soft Computing Journal, Elsevier 2008.

[Ramos, 2000] Vitorino Ramos and Filipe Almeida: Artificial Ant Colonies in Digital Image Habitats - A Mass Behaviour Effect Study on Pattern Recognition

[Gagné, 2006] G Gagné, M Gravel, WL Price. Solving real car sequencing problems with ant colony optimization. European Journal of Operational Research. Elsevier 2006.

[Dorigo, 2001] Marco Dorigo: The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, Methaeuristics Handbook 2001

[Maniezzo, 2004] Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi Ant Colony Optimization. (Book Chapter)

[Socha, 2008] K Socha, M Dorigo. Ant colony optimization for continuous domains. European Journal of Operational Research. Elsevier, 2008

[Dorigo, 1997] Marco Dorigo, Luca Maria Gambardella: Ant Colonies For The Traveling Salesman Problem, BioSystems press, 1997

[Dorigo, 1999] Marco Dorigo and Gianni Di Caro: Ant Algorithms for Discrete Optimization, Artificial Life Vol5 No3 pp137-172, 1999

[Parpinelli, 2002] Rafael S.Parpinelli, Heitor S.Lopes and Alex A.Freitas: Data Mining with an Ant Colony Optimization Algorithm.

[Karpenko, 2005] Oleksiy Karpenko, Jianming Shi, Yang Dai: Prediction of MHC class II binders using the ant colony search strategy, Elsevier Artificial Intelligence in Medicine, 2005.

[Boyd, 2004] Jeffrey E. Boyd, Gerald Hushlak and Chistian J. Jacob: Swarm Art: Interactive Art from Swarm Intelligence, MM'04 New York.

[Manfrin, 2006] M Manfrin, M Birattari, T Stutzle, M Dorigo. Parallel ant colony optimization for the travelling salesman problem. Lecture Notes in Computer Science. Springer 2006.

[Birattari, 2007] M Birattari, P Pellegrini, M Dorigo. On the invariance of ant colony optimization. IEEE transactions on evolutionary computation 2007

Authors' Information

Ángel Goñi Moreno – Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: ago@alumnos.upm.es

Paula Cordero Moreno – Natural Computing Group. Universidad Politécnica de Madrid, Boadilla del Monte, 28660 Madrid, Spain: e-mail: paula.cormo@gmail.com