REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 2, número 1

Págs. 7-18

A General Ant Colony Model to solve Combinatorial Optimization Problems

José Aguilar[*]

Abstract

An Ants System is an artificial system based on the behavior of real ant colonies, which is used to solve combinatorial problems. This is a distributed algorithm composed by a set of cooperating agents called ants which cooperate among them to find good solutions to combinatorial optimization problems. The cooperation follows the behavior of real ants using an indirect form of communication mediated by a pheromone. In this work, we present a new distributed algorithm based on Ant System concepts, called the General Ant System, to solve Combinatorial Optimization Problems. Our approach consist on mapping the solution space of the Combinatorial Optimization Problem on the space where the ants will walk, and on defining the transition probability of the Ant System according to the objective function of the Combinatorial Optimization Problem. We test our approach on the Graph Partitioning and The Traveling Salesman Problems. The results show that our approach has the same performances than previous versions of Ant Systems.

Keywords: *Combinatorial Optimization Problem, Ant System, the Graph Partitioning and The Traveling Salesman Problems.*

1 Introduction

The Collective Intelligence studies how the actions and inter-relations of a set of simple agents (for example, bees, ants, etc.) carry out global objectives of the system where these agents are immersed. Each agent collaborates in the realization of the tasks to complete those objectives, without central coordination or control, through mechanisms of inter-relation and communication between them [1, 10]. In these systems, their agents are not individually intelligent, but their actions as a whole have an intelligent behavior in order to complete certain objectives of the systems (for example: the search of food sources).

Examples of these systems are Insect Systems (for example: Bee or Ant Colonies). For example, real Ants are capable of finding the shortest path from a food source to their nest without using visual cues by exploiting pheromone information [2, 5, 6, 7]. While walking, ants deposit pheromone trails on the ground and follow pheromone previously deposited by other ants. The above behavior of real ants has inspired the Ants System (AS), an algorithm in which a set of artificial ants cooperate to the solution of a problem by exchanging information via pheromone deposited on a graph. The main characteristics of the AS are: it follows a positive feedback (it allows finding good solutions quickly),

[*] Universidad de los Andes, Mérida, 5101. Venezuela, CEMISID. Departamento de Computación, Facultad de Ingeniería, fax: (58.74) 402872 fax(home): (775) 5422983, email: aguilar@ing.ula.ve

it is based on distributed computation (it avoids premature convergence), it uses a constructive greedy heuristic (it helps find acceptable solutions) and it is a population-based approach. Dorigo has proposed the first AS in his Ph.D. thesis [5]. Currently, most work has been done in the direction of applying AS to combinatorial optimization problems [4, 6, 10, 11, 12, 13, 14]. AS has been applied to the traveling salesman problem, to the quadratic assignment problem, among others. On the other hand, different groups have been working on various extended versions of the AS paradigm (Ant-Q, etc.) [7, 8, 9].

In the AS applied to the Traveling Salesman Problem (TSP), a set of cooperating agents, called ants, cooperate to find good solutions to TSP's using an indirect form of communication through of pheromone trails that they deposit on the edges of the TSP graph while building solutions. Informally, each ant constructs a TSP solution in an iterative way: it adds new cities to a partial solution by exploiting both informations gained from past experience and a greedy heuristic. Memory takes the form of pheromone trails deposited by ants on TSP edges, while heuristic information is simply given by the edge's weights. There are two reasons for an easy application of the AS on the TSP:

a. The TSP graph represents the solution space of this problem. This TSP graph can be used to describe the space where the ants walk (AS graph). That is, the TSP graph can be directly used by the AS because its structure is the same as the AS uses to build the solutions (AS graph).

b. The transition function has similar goals as the TSP objective function. The transition function goal is a trade-off between visibility (which says close nodes should be chosen with high probability) and trail intensity at a given time (the pheromone update formula is based on the edge length and the ants traffic). The TSP goal is to find a minimal length closed tour that visits each city one. That is not the case for other combinatorial optimization problems.

We propose a new distributed algorithm based on AS concepts, called the General Ant System (GAS), to solve Combinatorial Optimization Problems. The main novel idea introduced by our approach is the definition of a general procedure to solve Combinatorial Optimization Problems using AS. In our approach, the graph that describes the solution space of the Combinatorial Optimization Problem is mapped on the AS graph, and the transition function and the pheromone update formula of the AS are built according to the objective function of the Combinatorial Optimization Problem. We test our approach on the classical Graph Partitioning Problem (GPP) and the TSP. This paper is organized as follows: section II presents the AS and the GPP. Then, we present our approach. Section IV presents the experiments. We compare our results with previous one for the GPP and the TSP. Finally we present our conclusion.

2 Theoretical Aspects

2.1 Ant Systems

The Collective Intelligence studies how appear collective cognitive capacities on Insect Systems [1, 10]. These cognitive capacities on Insect Systems allow them to complete objectives, which guarantee their survival/vitality on hostile environments with a great efficiency. Examples of these systems are ant colonies, bee colonies, wasp colonies, etc. Thus, the collective intelligence is an emergent phenomenon from activities of individuals which generate a collective and sophisticated behavior in the system (for example: the nests construction, the foods search, etc.). The main characteristics of these systems are the following:

- They have not a central control.
- They have auto-organization capacities.
- Exist a common objective for all the individuals/agents of the system.
- They are based on a tasks division.

In general, the behavior of Ant Colonies is impressing to perform their objectives of survival. It is derived of a process of Collective Intelligence. This process is based on the ant communication capacities, which define the inter-relations between them. These inter-relations permit to transmit the information that each ant goes processing. The communication among agents (ants) is made through a trace, called pheromone. Thus, an ant leaves a certain quantity of pheromone trail when it moves it. In addition, the probability that an ant follows a path depends on the number of ants that has taken the path (a large quantity of pheromone in a path means a large probability to be visited).

AS is the progenitor of all research efforts with ant algorithms and was first applied to the TSP [2, 3, 5, 6]. Algorithms inspired on AS have appeared like heuristic methods that permit resolving combinatorial optimization problems. The principal characteristics of these algorithms are the following: their versatilities, their robustness and their operations based on populations. The procedure is based on the distribution of the search on agents called «ants», that is, agents with very simple capacities that try to simulate the behavior of the ants.

AS utilizes a graph representation where (AS graph) each edge (r, s) has a desirability measure g(r,s), called -pheromone, which is updated at run time by artificial ants. Informally, the AS works as follows. Each ant generates a complete tour by choosing the nodes according to a probabilistic state transition rule; ants prefer to move to nodes that are connected by short edges, which have a high pheromone amount. Once all ants have completed their tours, a global pheromone updating rule is applied: a fraction of the pheromone evaporates on all edges, and then each ant deposits an amount of pheromone on edges which belong to its tour in proportion to how short its tour was. The process is then iterated.

The state transition rule used by ant system is given by the equation (1), which gives the probability with which ant k in city r chooses to move to the city s (transition probability from node r to node s for the kth ant):

$$P_k(r,s)= \begin{cases} [[\gamma(r,s)\ n(r,s)]^{\beta} / \Sigma u\in J_k(r)\ [\gamma(r,u)\ n(r,u)]^{\beta} & \text{if } s\in J_k(r) \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

Where γ is the pheromone, n=1/d is the inverse of the distance d(r,s), $J_k(r)$ is the set of nodes that remain to be visited by ant k positioned on node r ,and β is a parameter which determines the relative importance of pheromone versus distance.

In AS, the global updating rule is implemented as follows. Once all ants have built their tours, pheromone is updated on all edges according to (that is, the trail intensity is updated):

$$\gamma(r,s) = (1-\alpha)\gamma(r,s) + \Sigma_{k=1}^{m} \Delta\gamma_k(r,s) \qquad (2)$$

Where α is a coefficient such that $(1 - \alpha)$ represents the trail evaporation in one iteration, m is the number of ants, and $\Delta\gamma_k(r,s)$ is the quantity per unit of length of trail substance laid on edge (r, s) by the kth ant in that iteration

$$\Delta\gamma_k(r,s)=\begin{cases} 1/L_k & \text{if edge } (r,s) \in \text{ tour done by ant k} \\ 0 & \text{otherwise} \end{cases}$$

Where L_k is the length of the tour performed by ant k.

Pheromone updating is intended to allocate a greater amount of pheromone to shorter tours. Pheromone placed on the edges plays the role of a distributed long-term memory; this memory is not locally within the individual ants, but is distributed on the edges of the graph. The general algorithm is the following:

```
1.        Place the m ants randomly on the nodes of the AS graph
2.        Repeat until system convergence
2.1       For i=1, n2.1.1          For j= 1, m
2.1.1.1.  Choose the node s to move to, according to the transition
          probability (equation 1)
2.1.1.2.  Move the ant m to the node s
2.2       Update the pheromone using the pheromone update formula (equation 2)
```

The time complexity of AS is $O(t_* n^2{}_* m)$, where *t* is the number of iterations done (until system convergence). Different versions to improve the classic AS have been proposed [6, 7, 8, 9]. Two of them are the ant-density and ant-quantity algorithms. They differ in the way the trail is updated. In these models each ant lays its trail at each step, without waiting for the end of the tour. In the ant-density model a quantity Q of trail is left on edge (r, s) every time an ant goes from r to s; in the ant-quantity model an ant going from r to s leaves a quantity Q/d(r,s) of trail on edge (r, s) every time it goes from r to s. In a most recent work, they propose a new extension to AS, called ACS. The ACS differs from the previous one on:

- The state transition rule provides a direct way to balance between exploration of new edges and exploitation of a priori and accumulated knowledge about the problem.
- The global updating rule is applied only to edges which belong to the best ant tour.
- A local pheromone-updating rule is applied while ants construct a solution.

2.2 Graph Partitioning Problem

The GPP consists on dividing a graph in several subgraphs, so as to minimize the connection costs between them. We can complicate the problem by weighting the arcs. In this case, we must minimize the sum of the weights between the subsets. Also, we can add a weight to the nodes and define again what we want to minimize according to the particular characteristics of the problem [1, 13]. In order to formulate mathematically the problem, the next definition is necessary:

$G = (N, A)$,

where,

- G is a undirected graph,
- N = {1, ...,n} is a set of n nodes on which we can associate a weight function Q : N -> R. In ours paper Q(i)=1 for i=1,.., n,
- A= a_{ij}, are node pairs that define the arcs. It is known as the adjacency matrix.

According to certain constraints, the problem consists on dividing the graph in *K* different subgraphs. The classic constraints are:

- Subgraphs must have a specific size or must have a weight sum of nodes less than a given value.
- Arcs with extremities in different subgraphs must be minimal, or the weight sum of arcs which join nodes in different subgraphs must be minimized.

The cost function associates a real value to every subgraph configuration. We propose the next cost function:

$$Fc= \sum_{i,j\in D} a_{ij} + b \sum_{z=1}^{K} (N_{Gz} - n/K)^2/K \qquad (3)$$

where: - $D=\{i\in Gm \text{ and } j\in Gl \text{ and } l\neq m\}$
- N_{Gz}= number of nodes in subgraph z,
- b = balance factor [0,2].

The first term minimizes the weight sum of arcs that belong to the cut. The second summation term will have a minimum value only when the number of nodes by subgraph is the same. The balance factor (b) defines the importance of the interconnection cost with respect to the imbalance cost. The GPP is reduced to find a subgraph configuration with minimum value for the cost function:

F = MIN(Fc)

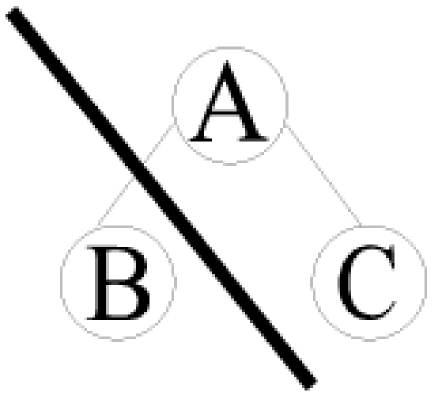

Figure 1: Partition of a graph *G* with 3 nodes in 2 subgraphs.

2.3 The Traveling Salesman Problem

The Traveling Salesman problem is a classical optimization problem that could describe according to the next statement: given n cities, the Salesman should visit each city one time and the total cost of the tour should be minimal. We can define the cost of the tour as the sum of the distances between the visited cities. This problem can be expressed of the following manner:

$G = (N, A)$

where: - $N = \{1, ..., n\}$, it is the graph with n nodes,
- $A = \{a_{ij}\}$, it is the adjacency matrix.

If we suppose that the cities are numbered from 1 to n, a solution to the problem could express through a state matrix (E) that indicates the order in that the cities are visited:

$$e_{ij} = \begin{cases} 1 & \text{if the city i was visited in the position j} \\ 0 & \text{Otherwise} \end{cases}$$

The matrix E will allow to verify the validity of a solution, that is all the cities must be visited only once:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} e_{ij} = n$$
$$\sum_{i=1}^{n} e_{ij} = 1$$
$$\sum_{j=1}^{n} e_{ij} = 1$$

The matrix E allows to define an array V with n elements, which contain the city that was visited in each position.

$V_j = i$ (If the city i was visited in the position j)

Finally, we propose a function composes for two parts, the first one calculates the distances between the cities, and the other one determines the grade of validity of the solution. These functions are:

$$F1 = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{j=1}^{n} l_{ik}\, e_{ij}\, e_{kj+1}$$

where l_{ij} = distance between the cities i and j

and

$$F2 = C(| \sum_{i=1}^{n} \sum_{k=1}^{n} e_{ik} - n| + \sum_{i=1}^{n} | \sum_{k=1}^{n} e_{ik} - 1| + \sum_{k=1}^{n} | \sum_{i=1}^{n} e_{ik} -1|)$$

Finally,

$Fc = F1 + F2$

where C = factor of penalty.

The problem consists of finding the tour for the cities that minimize the value of the cost function Fc.

3 Our approach: the General Ant System

There are two reasons for an easy application of the AS on the TSP: the first one, the TSP graph can be directly mapped on the AS graph. The other one, the transition function has similar goals than the TSP. The AS goal is a trade-off between visibility (which says close nodes should be chosen with high probability) and trail intensity at a given time (the pheromone update formula is based on the edge length and the ants traffic). The TSP goal is to find a minimal length closed tour that visits each city one. That is not the case for other combinatorial optimization problems.

We propose a new distributed algorithm based on AS concepts, called the General Ant System (GAS), to solve Combinatorial Optimization Problems. In our approach, we must define:

- The graph that describes the solution space of the Combinatorial Optimization Problem (COP graph). The solution space is defined by a graph where the nodes represent partial possible solutions to the problem, and the edges the relationship between the partial solutions. This graph will be used to define the AS graph (this is the graph where the ants willwalk).
- The transition function and the pheromone update formula of the AS, which are built according to the objective function of the Combinatorial Optimization Problem.

In this way, we can solve any Combinatorial Optimization Problem. The COP graph define the structure of the AS graph. Each ant builds a solution walking through this graph according to a transition rule and a pheromone update formula defined according to the objective function of the Combinatorial Optimization Problem. The main steps are:

a) Build the AS graph
b) Define transition function and the pheromone update formula of the AS
c) Execute the classical AS procedure (or one of the improved versions)

3.1 Build the AS graph

The first step is to build the COP graph, then we define the AS graph with the same structure of the COP graph. The AS graph has two weight matrices: the first one is defined according to the COP graph and saves the relationship between the elements of the solution space (COP matrix). The second one saves the pheromone trail accumulated on each edge (pheromone matrix). This weight matrix is calculated/updated according to the pheromone update formula. A node has more probability to be visited if edges weight of the pheromone matrix incoming it are high. If an edge between two nodes of the COP matrix is low, that means that ideally if one of these nodes belongs to the final solution, the other one must belong too. If the edge is equal to infinite means that they are incompatible.

We define a data structure to save the solution that every ant is building. This data structure is a vector (Vk) with a length equal to the length of the solution (number of nodes that an ant must visit). For a given ant, the vector saves each node of the solution space that it visits.

3.2 Define the transition function and the pheromone update formula

The state transition rule depends of the trail intensity at a given time and of the ant traffic. The trail intensity at a given time is defined by the pheromone update formula. The state transition rule and

the pheromone update formula are built using the objective function of the combinatorial optimization problem. The transition function between nodes is the following:
$Ft(\gamma(r,s),Fc^{z})= \gamma(r,s)/Fc_{k}(r\text{->}s)^{z+1}$

Where $Fc_{k}(r\text{->}s)^{z+1}$ is the new cost when an ant k cross the edge (r, s) at the time z if it is in r.

The transition probability is calculated according to the next equation:

$$P_k(r,s)=\begin{cases} Ft(\gamma(r,s), Fc_k^{\,z})/ \Sigma u\in J_k(r)\ Ft(\gamma(r,u),Fc_k^{\,z}) & \text{if } s\in J_k(r) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

And the pheromone updating rule is calculated according to the next formula:

$$\Delta\gamma_k(r,s) = \begin{cases} 1/Fc_k & \text{if edge (r,s) has been crossed by ant k} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The general procedure of our approach is the following:

```
1.      Generation of the AS graph
2.      Initialization of the pheromone matrix
3.      Definition of the state transition rule and the pheromone update formula according to the
        Combinatorial Optimization Problem
4.      Place the m ants on different nodes of the AS graph
5.      Repeat until system convergence
5.1   For i=1, n
5.1.1       For j= 1, m
5.1.1.1.  Choose the node s to move to, according to the transition probability (equation 4)
5.1.1.2.   Move the ant m to the node s
5.2    Update the pheromone using the pheromone update formula (equations 2 and 5)
```

The time complexity of our algorithm is the same as the AS algorithm, $O(t_* n^2{}_* m)$, where *t* is the number of iterations done.

4 Experiments

We have developed a version of our approach on C++, and we have executed our program on a PC. We have tested our approach for a set of benchmark for the GPP and TSP. We have run our algorithm 30 times to calculate every point. Our computational implementation is composed by three classes:

a) Graph class: which defines the AS graph.
b) Ant class: which manages the data structure of each ant (Vk), etc.
c) AntSystem: which manages the AS (pheromone updating, etc.)

4.1 Build the AS graph

For the case of the GPP, the COP graph is defined by nodes that represent the possible assignment of each node of the graph to divide (for example, the node (A, 1) of the figure 2 represent the assignment of the node A of the graph G of the figure 1 on the first subgraph), and arcs that represent the relationship between the possible assignment. A weight edge equals to infinite means that these nodes can not belong to the final solution together (they are incompatible solution). For example, the node A can not be assigned to subgraphs 1 (A, 1) and 2 (A, 2) at the same time. A weight arc equal to 0 means that these nodes together do not introduce additional communication costs (because they represent assignments to the same subgraph). A weight value equals to a_{AB} corresponds to the edge weight between nodes A and B of the graph G to divide (graph of the figure 1). We use this information to build the AS graph structure and its COP matrix.

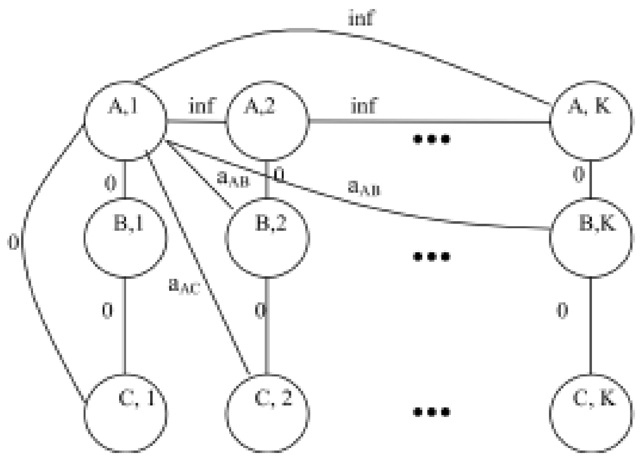

Figure 2. COP graph for the GPP of the graph *G* defined on figure 1.

For the case of the GPP problem, the data structure (Vk) to save the solution that every ant is building is a vector of length n. Each element (V_k^i) saves the assignment of one of the nodes of the graph to divide in a given subgraph (for example, figure 3 shows that node B is assigned to subgraph 1, etc.).

1	2	...	n
(B,1)	(A,2)		(C,1)

Figure 3. Data Structure of an Ant for the GPP.

In the case of TSP, the AS graph is the TSP graph that describes the distance between cities. Each element of Vk (V_k^i) saves the node that has been visited in this position.

1	2	...	n
3	4		1

Figure 4. Data Structure of an Ant for the TSP.

4.2 Define the Transition Function and the Pheromone update Formula

For the GPP, Fc_k is defined according to the equation 3:

$Fc_k = Cc + Cb$

Where Cc is the communication cost

$Cc = \Sigma_{((r,p),(x,m) \in Vk \text{ and } p,m \in D1)} a_{rx}$ $\qquad D1=\{p \neq m\}$

and Cb is the balance cost

$Cb = \Sigma_{j=1}^{K} (c/K - N_{Gj}^{k})^2/K$

Where c is the number of node assigned by the ant k (current length of Vk) and N_{Gj}^{k} is the number of nodes that ant k has assigned to subgraph Gj.

For the case of TSP, the cost function (Fc_k) is:

$Fc_k = \sum_{(m=1\ \delta\ ij \in D2)}^{c-1} l_{ij}$ $\qquad D2=\{i=V_k^{m}\ \delta\ j=V_k^{m+1}\}$

Where c is the number of node traversed by the ant k (current length of Vk) and l_{ij} is the distance between the cities i and j.

This cost is equivalent to the Lk parameter of the $\Delta\gamma_k(r,s)$ equation. To calculate $Fc_k\ (r\text{->}s)^{z+1}$, the node *s* is included on Vk. In this way, we can use the same cost functions for each problem.

4.3 Result Analysis

To test our algorithm, we use the graphs of the web page http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html for the TSP and the graph generated on [10] for the GPP. For the case of TSP, we compare our approach with the last version of the ant system proposed by Dorigo (ACS) and the best results reported for Genetic Algorithms (GA) [1]. For the case of GPP, we compare our approach with our previous work [10]. The results are shown in the next tables. The first number is the solution cost and the second number is the number of iterations of the algorithm.

Table 1. Results for the TSP

Graph	GAS	ACS	GA
Eil50	428 (431)	425 (1830)	428 (25000)
Eil75	549 (931)	535 (3480)	545 (80000)
KroA100	21766 (1025)	21282 (4820)	21761 (103000)

Table 2. Results for the symmetric TSP

Graph	GAS	ACS	GA
d198	15810	15780	15780
lin318	42057	42029	42029
att532	27701	27693	27686

Table 3. Results for the asymmetric TSP

Graph	GAS	ACS	GA
ry48p	14440	14422	14440
kro124p	36244	36230	36230
ftv170	2761	2755	2755

Table 4. Results for the GPP

Graph	GAS	GA
20 nodes and 4 subgraphs	46 (15)	46 (23)
50 nodes and 5 subgraphs	244 (82)	244 (203)
50 nodes and 7 subgraphs	267 (111)	262 (276)
100 nodes and 5 subgraphs	1521 (523)	1517 (809)
200 nodes and 5 subgraphs	1778 (1423)	1772 (2121)

In our approach, we obtain good results with a low number of iterations. In some case, we obtain the same result as ACS and GA approaches. Our approach is very slow (it has a large execution time), but need the least number of iterations. In our approach, we need to improve the combination between the global search (we like that every ant searches for different zone of the solution space) and the local search (the good information found for an ant must be transmitted to the others). Currently, the local search is transmitted in form of trail when the pheromone is updated, and the global search is carried out because we assign the ants in different places at the beginning. We need to add another mechanism to assure one efficient global search. One of the ways is to avoid the same route for the ants in the first iterations.

5 Conclusions

In this work we have presented a general approach for AS to solve different combinatorial optimization problems. In our approach, we define the solution space of the combinatorial optimization problem (COP graph) how the space where the ants will walk (AS graph). Ants walk through this space according to a set of probabilities updated by a state transition and a pheromone update rule defined according to the objective function of the Combinatorial Optimization Problem. In this way, we can solve any combinatorial optimization problem. We have tested our approach on the GPP and the TSP. We need to do more experiments to define the ideal combination of the parameters of our approach (number of ants, local pheromone update, etc.). The results show that our approach has the same performances as the previous versions. Currently, we are testing our approach on the graph coloring problem. At the further, we will test our approach on other combinatorial optimization problems. Of course, the solution space of the combinatorial optimization problem to test will have to be able to be described by a graph (COP graph). Also, we will improve the current computational version of our approach. We will develop a new version for workstation.

References

[1] E. Bonabeau, M. Dorigo and G. Theraulaz, From Natural to Artificial Swarm Intelligence, *Oxford University Press*, 1999.

[2] A. Coloni, M. Dorigo and V. Maniezzo, Distributed Optimization by Ant Colonies. In: *Proc. European Conference on Artificial Life*, pp. 134-142, 1991.

[3] D. Corne, M. Dorigo and F. Glove, New ideas in Optimization, *McGraw Hill*, 1999.

[4] Costa D. and A. Hertz, Ants Can Colour Graphs. *Journal of the Operational Research Society*, 48: 295-305, 1997.

[5] M. Dorigo, Optimization, Learning and Natural Algorithms, *Ph.D Thesis, Politecnico de Milano*, Italy, 1992.

[6] M. Dorigo, V. Maniezzo, A. Coloni, The ant system: Optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man, Cybern.*, 26 (2): 29-41, 1996.

[7] M. Dorigo and L. Gambardella, A study of some properties of Ant-Q. In: *Proc. 4th Int. Conf. Parallel Problem Solving from Nature*, pp. 656-665, 1996.

[8] M. Dorigo and L. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Trans. on Evolutionary Computation*, 1(1): 53-66, 1997.

[9] L. Gambardella and M. Dorigo, Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In: *Proc. of 12th Int. Conf. on Machine Learning*, pp. 252-260, 1995.

[10] F. Hidrobo and J. Aguilar "Toward a Parallel Genetic Algorithm Approach Based on Collective Intelligence for Combinatorial Optimization Problems", In: *Proc. of the IEEE International Conference on Evolutionary Computation*, pp. 715-720, 1998.

[11] P. Kuntz and D. Snyers. Emergent Colonization and Graph Partitioning. In: *Proc of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3*, 1994.

[12] P. Kuntz, P. Layzell and D. Snyers. A Colony of Ant-like Agents for Partitioning in VLSI Technology. In: *Proc. of the Fourth European Conference on Artificial Life*, pp. 417-424, 1997.

[13] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkrantz, Ant-based Load Balancing in Telecommunications Networks, *Adaptive Behavior*, 5(2): 169-207, 1997.

[14] Y. Stutzle and H. Hoos, The Max-Min Ant System and Local Search for the Traveling Salesman Problem, In: *Proc. 4th Int. Conf. on Evolutionary Computation*, pp. 309-314, 1997.