This article was downloaded by: [C N R S]
On: 05 July 2012, At: 09:38
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK

Applied Artificial Intelligence: An International Journal

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uaai20

THE COMBINATORIAL ANT SYSTEM

JOSE AGUILAR [a] , LUIS VELÁSQUEZ [b] & MARÍa E. VELÁSQUEZ [b]

[a] Departamento de Computación, Facultad de Ingenieria, Mérida, Venezuela

[b] Dpto. de Ing. en Informática e Indutrial, Universidad Nacional Exp. de Guayana, Puerto Ordaz, Venezuela

Version of record first published: 16 Aug 2010

To cite this article: JOSE AGUILAR, LUIS VELÁSQUEZ & MARÍa E. VELÁSQUEZ (2004): THE COMBINATORIAL ANT SYSTEM, Applied Artificial Intelligence: An International Journal, 18:5, 427-446

To link to this article: http://dx.doi.org/10.1080/08839510490442067

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Applied Artificial Intelligence, 18:427–446, 2004
Copyright © Taylor & Francis Inc.
ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839510490442067

Taylor & Francis
Taylor & Francis Group

THE COMBINATORIAL ANT SYSTEM

JOSE AGUILAR
Departamento de Computación, Facultad de Ingenieria, Universidad de los Andes, Mérida, Venezuela

LUIS VELÁSQUEZ and MARÍA E. VELÁSQUEZ
Dpto. de Ing. en Informática e Indutrial, Universidad Nacional Exp. de Guayana, Puerto Ordaz, Venezuela

Downloaded by [C N R S] at 09:38 05 July 2012

This paper presents a new distributed algorithm based on Ant System (AS) concepts called Combinatorial Ant System (CAS). It is oriented to solve static discrete-state combinatorial optimization problems. Our approach consists of mapping the solution space of the combinatorial optimization problem in the space where the ants will walk, and defining the transition probability and the pheromone update formula of the Ant System, according to the objective function of the Combinatorial Optimization Problem. We test our approach on the graph partitioning, graph coloring and traveling salesman problems.

Real ants are capable of finding the shortest path from a food source to their nest without using visual cues by exploiting pheromone information (Bonabeau et al. 1999). 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 Ant System (AS), an algorithm in which a set of artificial ants determine the solution of a problem by exchanging information via pheromone deposited on a graph. Dorigo (1992) proposed the first AS in his Ph.D. thesis. Currently, considerable work is being done in the direction of applying AS to combinatorial optimization problems (Bonabeau et al. 1999; Dorigo 1992; Corne et al. 1999; Costa and Hertz 1997; Dorigo et al. 1996; Dorigo and Gambardella 1997; Hidrobo and Aguilar 1998; Kuntz and Snyers 1994; Kuntz et al. 1997; Schoonderwoerd et al. 1997; Stutzle and Hoos 1997). AS has been applied to the traveling salesman problem and quadratic assignment problem, among others. On the other hand, different groups have been working on various extended versions of the

Address correspondence to Jose Aguilar, CEMISID, Departamento de Computación, Facultad de Ingenieria, Universidad de los Andes, Merida, 5201 Venezuela. E-mail: aguilar@ing.ula.ve

AS paradigm (Ant-Q, etc.), (Bonabeau et al. 1999; Dorigo and Gambardella 1997; Hidrobo and Aguilar 1998).

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 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 information gained from both 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 to use the AS on the TSP:

1. The TSP graph represents the solution space of this problem. This TSP graph is 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).
2. The AS transition function has goals similar to the TSP objective function. The AS transition function goal is a trade-off between *visibility* (which says close nodes should be chosen having 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 once.

Downloaded by [C N R S] at 09:38 05 July 2012

That is not the case for other combinatorial optimization problems. We propose a new distributed algorithm based on AS concepts, called the Combinatorial Ant System (CAS), to solve static discrete-state combinatorial optimization problems. The main novel idea introduced by our model 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), graph coloring problem (GCP), and TSP.

THEORETICAL ASPECTS

Ant Systems

In general, the behavior of ant colonies is important for their objectives of survival. It is derived from a process of *collective behavior.* This process is based on the ant communication capacities, which define the inter-relations between them. These inter-relations permit the transmission of information

each ant is processing. The communication among agents (ants) is made through a trace called *pheromone.* Thus, an ant leaves a certain quantity of pheromone trail as it moves. In addition, the probability that an ant follows a path depends on the number of ants having taken the path (a large quantity of pheromone in a path means a large probability that it will be visited).

AS is the progenitor of all research efforts with ant algorithms and it was first applied to the TSP problem (Dorigo 1992; Dorigo et al. 1996). Algorithms inspired by AS have manifested as heuristic methods that permit resolving combinatorial optimization problems. These algorithms mainly rely on their versatility, robustness, and operations based on populations. The procedure is based on the search of agents called "ants," that is, agents with very simple capabilities that try to simulate the behavior of the ants.

AS utilizes a graph representation (*AS graph*) where each edge (r, s) has a desirability measure γ_{rs}, 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 presence. 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 the tour was. Then we continue with a new iteration of the process.

The state transition rule used by AS is given by the Eq. (1), which gives the probability with which ant k in city r chooses to move to the city s while building its *tth* tour (transition probability from node r to node s for the *kth* ant) (Bonabeau et al. 1999; Dorigo 1992; Corne et al. 1999; Dorigo et al. 1996; Dorigo and Gambardella 1997):

$$P_{rs}^{k}(t) = \begin{cases} [\gamma_{rs}(t)]^{\alpha}[\eta_{rs}]^{\beta} / \sum_{u \in J_r^k} [\gamma_{rs}(t)]^{\alpha}[\eta_{rs}]^{\beta} & \text{If } s \in J_r^k \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

where $\gamma_{rs}(\mathrm{t})$ is the pheromone at iteration t, η_{rs} is the inverse of the distance between city r and city $s(\mathrm{d(r,s)}), \mathrm{J_k(r)}$ is the set of nodes that remain to be visited by ant k positioned on node r, and β and α are two adjustable parameters which determine the relative importance of trail intensity (γ_{rs}) versus visibility (η_{rs}).

In AS, the global updating rule is implemented as follows. Once all ants have built their tours, pheromone (that is, the trail intensity) is updated on all edges, according to the equation (Bonabeau et al. 1999; Dorigo 1992; Corne et al. 1999; Dorigo et al. 1996; Dorigo and Gambardella 1997):

$$\gamma_{rs}(t) = (1-\rho)\gamma_{rs}(t-1) + \sum_{k=1}^{m} \Delta\gamma_{rs}^{k}(t) \tag{2}$$

Downloaded by [C N R S] at 09:38 05 July 2012

where ρ is a coefficient such that $(1-\rho)$ represents the trail evaporation in one iteration (tour), m is the number of ants, and $\Delta\gamma_{rs}^{k}(t)$ is the quantity per unit of length of trail substance laid on edge (r, s) by the kth ant in that iteration:

$$\Delta\gamma_{rs}^{k}(t) = \begin{cases} 1/L_k(t) & \text{If edge (r, s)} \in \text{tour completed by ant k} \\ 0 & \text{Otherwise} \end{cases}$$

where $L_k(t)$ is the length of the tour performed by ant k at iteration t. 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 summarized as follows:

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

The time complexity of AS is $O(t \cdot n^2 \cdot m)$, where t is the number of iterations done (until the system convergence) and n the number of nodes to be visited.

Different versions to improve the classic AS have been proposed (Bonabeau et al. 1999; Dorigo and Gambardella 1997; Hidrobo and Aguilar 1998). 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 (Dorigo 1992; Kuntz and Snyers 1994), new extension to AS is proposed, called ACS (Ant Colony System). 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.

Downloaded by [C N R S] at 09:38 05 July 2012

- 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.

Graph Partitioning Problem

The GPP problem consists of 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 (Hidrobo and Aguilar 1998; Battiti and Bertossi 1999; Bui and Moon 1996). In order to formulate mathematically the problem, the following definition is necessary:

$$\mathrm{G} = (\mathrm{N}, \mathrm{A}),$$

where G is an undirected graph and $\mathrm{N} = \{1, \ldots, \mathrm{n}\}$ is a set of n nodes on which we can associate a weight function $\mathrm{Q} : \mathrm{N} \rightarrow \mathrm{R}$. In our paper, $\mathrm{Q(i)} = 1$ for $\mathrm{i} = 1, \ldots, \mathrm{n}$ and $\mathrm{A} = \mathrm{a_{ij}}$ are node pairs that define the arcs. It is known as the adjacency matrix.

According to certain constraints, the problem consists of dividing the graph into 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 use the following cost function (Hidrobo and Aguilar 1998; Battiti and Bertossi 1999; Bui and Moon 1996):

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

where $\mathrm{D} = \{\mathrm{i} \in \mathrm{Gm}, \mathrm{j} \in \mathrm{G1} \text{ and } 1 \neq \mathrm{m}\}$; $\mathrm{N_{Gz}} =$ number of nodes in subgraph z; and 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

Downloaded by [C N R S] at 09:38 05 July 2012

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:

$$F = MIN(C_f)$$

The Traveling Salesman Problem

The traveling salesman problem is a classical optimization problem that could be described 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 as:

$$G = (N, A).$$

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

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

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

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

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

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

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

The matrix E allows an array V to be defined with n elements. This array contains the city that was visited in each position.

$$V_j = i \,(\text{if the city } i \text{ was visited in the position } j).$$

Finally, we use a function composed of two parts. The first one calculates the distances between the cities, and the other determines the grade of validity of the solution. These functions are (Bonabeau et al. 1999; Dorigo and

Downloaded by [C N R S] at 09:38 05 July 2012

Gambardella 1997; Hidrobo and Aguilar 1998; Freisleben and Merz 1996):

$$F1 = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n-1} L'_{ik} e_{ij} e_{kj+1};$$

where L'_{ij} = distance between the cities i and j

and

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

where C = penalty factor.

Finally,

$$\mathrm{C_r} = \mathrm{F1} + \mathrm{F2} \tag{4}$$

The problem consists of finding the tour of cities that minimizes the value of the cost function $\mathrm{C_f}$.

Downloaded by [C N R S] at 09:38 05 July 2012

The Graph Coloring Problem

The graph coloring problem is one of the best known combinatorial optimization problems in the class of NP-hard and arises in numerous applications. A graph coloring of a graph G is a function which associates a color with each node such that no two nodes of the same color have common edges. This problem can be expressed of the following manner.

$$\mathrm{G} = (\mathrm{V}, \mathrm{E}, \mathrm{A})$$

where V is a set of n vertices, E is a set of edges, and $\mathrm{A} = \mathrm{a_{ij}},$, is the adjacency matrix of size $\mathrm{n} \cdot \mathrm{n}$.

Then, if we suppose that the colors are numbered from 1 to q, we are asked to find the minimum chromatic number q, such that the vertices $\mathrm{Vi} \in \mathrm{V}$ can be partitioned into q color classes, or nonempty disjoint subsets $\mathrm{SVp} \subset \mathrm{V}$ (for $\mathrm{p} = 1, \ldots, \mathrm{q}$); none of which contains both endpoints of any edge in E. We formulate the GCP as a typical combinatorial optimization problem and handle the constraint about edges by introducing penalty for constraint violation. Let $\mathrm{Ps} = \{\mathrm{SV_1}, \ldots, \mathrm{SV_q}\}$ be an arbitrary partition of the set V, and SEp (for $\mathrm{p} = 1, \ldots, \mathrm{q}$) be the subset of edges from E both of whose endpoints are included in the same SVp. For this problem, an objective function can be defined as (Costa and Hertz 1997)

$$C_f = -\sum_{p=1}^{q} |SVp|^2 + \sum_{p=1}^{q} 2|SVp||SEp| \tag{5}$$

When a partition is one of the feasible solutions of the GCP, all the subsets of edges become empty (SEp = 0). An important observation about the objective function in Eq. (5) is that all its local optimal solutions correspond to the feasible solutions of the GCP. Another objective function for the GCP, expressed through a state matrix X that indicates the color of each node, is:

$$C_f = C\sum_{i=1}^{n}\sum_{c=1}^{q}(Xic-1)^2 + \sum_{c=1}^{q}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}XicXjc \qquad (6)$$

where $X_{ic} = \begin{cases} 1 & \text{if vertice } i \text{ has a color equal to } c \\ 0 & \text{Otherwise} \end{cases}$

C = factor of penalty.

In the second case, we obtain a feasible only when $C_f = 0$. The problem consists of finding the chromatic number q with its respective partition (cost function 1) or the state matrix (cost function 2) that minimizes the value of C_f.

Downloaded by [C N R S] at 09:38 05 July 2012

OUR APPROACH: THE COMBINATORIAL ANT SYSTEM

There are two reasons for using AS on the TSP. First, the TSP graph can be directly mapped on the AS graph. Secondly, the transition function has similar goals to the TSP. This is not the case for other combinatorial optimization problems. We propose a new distributed algorithm based on AS concepts, called the Combinatorial Ant System (CAS), 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 will walk).
- The transition function and the pheromone update formula of the CAS, 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 defines 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 of CAS are:

1. Build the AS graph.
2. Define the transition function and pheromone update formula of the CAS.
3. Execute the classical AS procedure (or one of the improved versions).

Building 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 registers the relationship between the elements of the solution space (COP matrix). The second one registers the pheromone trail accumulated on each edge (pheromone matrix). This weight matrix is calculated/updated, according to the pheromone update formula. When the incoming edge weights of the pheromone matrix for a given node become higher, this node has similarly and a higher probability to be visited. If an edge between two nodes of the COP matrix is low, it 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, it means that they are incompatible.

We define a data structure to store the solution that every ant k is building. This data structure is a vector (A^k) with a length equal to the length of the solution (number of nodes that an ant must visit). For a given ant, the vector keeps a register of each node of the solution space that it visits.

Downloaded by [C N R S] at 09:38 05 July 2012

Defining the Transition Function and the Pheromone Update Formula

The state transition rule depends on the ant traffic and the trail intensity at a given time. 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 given by:

$$Tf(\gamma_{rs}(t), Cf^k_{r\to s}(z)) = \frac{\gamma_{rs}(t)^{\alpha}}{Cf^k_{r\to s}(z)^{\beta}}$$

where $Cf^k_{r\to s}(z)$ is the cost of the partial solution that is being built by the ant k when it crosses the edge (r, s). If it is in the position $r, z-1$ is the current length of the partial solution (current length of A^k), and α and β are two adjustable parameters that control the relative weight of trail intensity (γ_{rs}(t)) and the cost function. In the CAS, the transition probability is as follows: An ant positioned on node r chooses the node s to move according to a probability $P^k_{rs}(t)$, which is calculated according to the equation given by:

$$P^k_{rs}(t) = \begin{cases} \dfrac{Tf(\gamma_{rs}(t), Cf^k_{r\to s}(z))}{\sum_{u\in J^k_r} Tf(\gamma_{ru}(t), Cf^k_{r\to u}(z))} & \text{If s} \in \text{J}^{\text{k}}_{\text{r}} \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

when $\beta = 0$, we exploit previous solutions (only trail intensity is used) and, when $\alpha = 0$, we explore the solution space (a stochastic greedy algorithm is obtained). A trade-off between quality of partial solutions and trail intensity is necessary. The pheromone updating rule is defined by Eq. (2), where the quantity per unit of length of trail substance laid on edge (r, s) by the kth ant in that iteration ($\Delta\gamma_{rs}^k(t)$) is calculated, according to the following formula:

$$\Delta\gamma_{rs}^k(t) = \begin{cases} \dfrac{1}{C_f^k(t)} & \text{If edge (r,s) has been crossed by ant } k \\ 0 & \text{Otherwise} \end{cases} \quad (8)$$

where $C_f^k(t)$ is the value of the cost function (objective function) of the solution proposed by ant k at iteration t.

The general procedure of our approach is summarized as follows:

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. Repeat until system convergence.
   4.1. Place the m ants on different nodes of the AS graph.
   4.2. For i=1, n
      4.2.1. For j= 1, m
         4.2.1.1. Choose the node s to move to, according to the transition probability (Eq. 7).
         4.2.1.2. Move the ant m to the node s.
   4.3. Update the pheromone using the pheromone update formula (Eqs. 2 and 8).

The time complexity of our algorithm is the same as the AS algorithm, $O(t \cdot n^2 \cdot m)$, where t is the number of iterations done until the system convergence.

EXPERIMENTS

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

1. Graph class: which defines the AS graph.
2. Ant class: which manages, for example, the vector (A^k) of each ant.
3. Ant System: which manages the AS (pheromone updating, etc.).

Downloaded by [C N R S] at 09:38 05 July 2012

Building the AS Graph

In the case of the GPP, the COP graph is defined as (see Figure 2): Nodes represent the possible assignment of each node of the graph to be divided (for example, the node (A, 1) in Figure 2 represents the assignment of the node A of the graph *G* in Figure 1 on the first subgraph). Arcs represent the relationship between the possible assignment. A weight edge equal to infinite means that these nodes cannot belong to the final solution together (they constitute an incompatible solution). For example, the node A cannot 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 do not introduce additional communication costs together (because they represent assignments of the same subgraph). A weight value equal to a_{AB} corresponds to the edge weight between nodes A and B of the graph *G* to be divided (graph in Figure 1). We use this information to build the AS graph structure and its COP matrix.

In the case of the GPP problem, the data structure registering the solution that every ant *k* is building (A_i^k) is a vector of length *n*. Each element (A_i^k) registers the assignment of one node of the graph to be divided in a

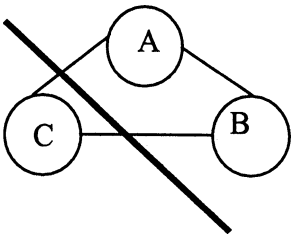

FIGURE 1. Partition of a graph *G* with three nodes in two subgraphs.

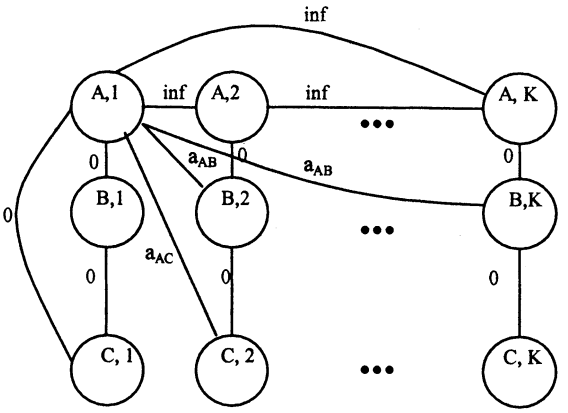

FIGURE 2. COP graph for the GPP problem of the graph *G* defined in Figure 1.

Downloaded by [C N R S] at 09:38 05 July 2012

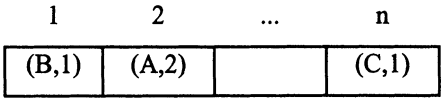

FIGURE 3. Data structure of an ant for the GPP problem.

given subgraph. For example, Figure 3 shows that node B is assigned to subgraph 1, etc.

In the case of TSP, the AS graph is the TSP graph that describes the distance between cities. Each element of $A(A_i^k)$ keeps the node (city) that has been visited in this position.

In the case of the GCP, the COP graph is defined as (see Figure 5): Each node represents the possible color of each node of the graph to color (for example, the node (A, 1) of Figure 5 represents that the node A of the graph G of Figure 1 has a color equal to 1). Arcs represent the relationship between the nodes colored. A weight edge equal to infinite means that these nodes cannot belong to the final solution together (they are an incompatible solution). For example, the node A cannot have color 1 (A, 1) and 2 (A, 2) at the same time, or the nodes (A, 1) and (B, 1) cannot have the same color because they have a common edge. A weight arc equal to 0 means that these nodes can have the same color (because they do not have common edges) or they are compatible solutions (for example, the nodes (A, 1) and (B, 2)).

In this case, the value of Q of the COP graph is equal to n (maximal number of colors). For each ant, we fix the value of $q^k(t) < n$ (the maximum chromatic number that each ant is going to use) in the following manners:

- When we place the ant on a node of the AS graph, randomly (SI).
- When we start the execution of a new iteration (the same value of q for the set of ants during that iteration), randomly (S2).
- When we add an outloop to carry out an exhaustive search over the different possible values of q (S3). In this way, we modify our general algorithm with a time complexity equal to $O(t \cdot n^3 \cdot m)$, in the following manner:

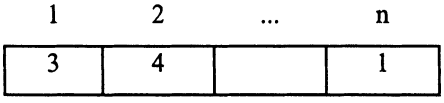

FIGURE 4. Data structure of an ant for the TSP problem.

Downloaded by [C N R S] at 09:38 05 July 2012

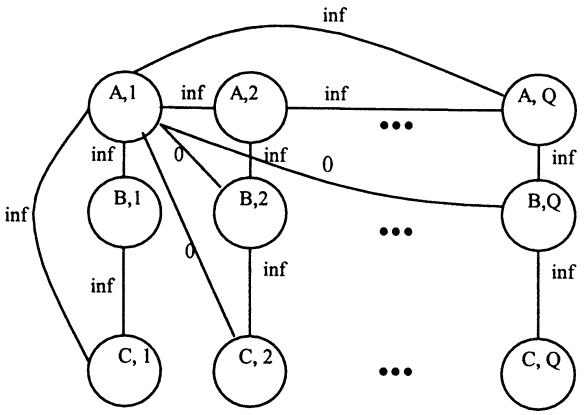

FIGURE 5. COP graph for the GCP problem of the graph G defined in Figure 1.

Downloaded by [C N R S] at 09:38 05 July 2012

For q = 1, n
  Repeat until system convergence.
    Place the *m* ants on different nodes of the AS graph.
    For i = 1, n
      For j = 1, m
        Choose the node *s* to move to, according to the transition probability (Eq. 7).
        Move the ant *m* to the node *s*.
    Update the pheromone using the pheromone update formula (Eqs. 2 and 8).

In the case of the GCP problem, each element (A_i^k) registers the color of each node of the graph to color (for example, Figure 6 shows that node C has a color 1, etc.).

Defining the Transition Function and the Pheromone Update Formula

For the GPP problem, $Cf_{r\to s}^k(z)$ is defined according to Eq. 3:

$$Cf_{r\to s}^k(z) = \mathrm{Cc} + \mathrm{Cb}$$

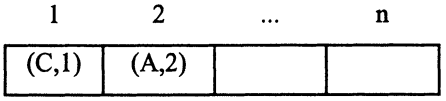

FIGURE 6. Data structure of an ant for the GCP.

where Cc is the communication cost, $\mathrm{Cc} = \sum_{((y,p),(x,m)\in A^{\mathrm{k}} \text{ and } p,m\in D1)} a_{yx}$ $\mathrm{D1} = \{\mathrm{p} \neq \mathrm{m}\}$, and Cb is the balance cost.

$$Cb = \frac{\sum_{j=1}^{K}\left(z/K - N_{Gj}^{k}\right)^2}{K}$$

where $\mathrm{N}_{\mathrm{Gj}}^{\mathrm{k}}$ is the number of nodes that ant k has assigned to subgraph Gj.

In the case of TSP, the cost function $Cf_{r\to s}^{k}(z)$ is:

$$Cf_{r\to s}^{k}(z) = \sum_{m=1}^{z-1}\sum_{j\in D2} L'_{ij} \qquad D2 = \{A_m^k = i\delta A_{m+1}^k = j\}$$

In the case of GCP, we try to minimize the objective function Eq. (5). Then, the cost function $Cf_{r\to s}^{k}(z)$ is:

$$Cf_{r\to s}^{k}(z) = -\sum_{p=1}^{q^k}\left|SV_p^k(z)\right|^2 + \sum_{p=1}^{q^k} 2\left|SV_p^k(z)\right|\left|SE_p^k(z\right|$$

where $\{\mathrm{SV}_1^k(z),\ldots,\mathrm{SV}_q^k(z)\}$ is the partial set of colors of the graph G proposed by ant k, and $\mathrm{SEp}^k(z)$ is the subset of edges from E both of whose endpoints are included in the same $\mathrm{SVp}^k(z)$.

$$SVp^k(z) = \sum_{i=1}^{z}\sum_{A_i^k=p} 1$$

In each case, to calculate $Cf_{r\to s}^{k}(z)$ the node s is included on A_z^k.

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, the Web page http://mat.gsia.cmu.edu/COLOR/instances.html for the GCP, and the Web page http://rtm.science.unitn.it/intertools/graph-partitioning/benchmark.html and the graph generated on (Hidrobo and Aguilar 1998) for the GPP. Our approach has been tested on problems of various sizes and compared with other general purpose heuristics and with one specialized graph partitioning heuristic, the Reactive Randomized Tabu Search (RRTS) heuristic (Battiti and Bertossi 1999). In the case of TSP, we compare our approach with the last version of the Ant system (ACS) proposed by Dorigo et al. (Bonabeau et al. 1999; Dorigo and Gambardella 1997), the best results reported for Genetic Algorithms (GA) (Freisleben and Merz 1996), our previous work (Hidrobo and Aguilar 1998), and the optimal length

Downloaded by [C N R S] at 09:38 05 July 2012

TABLE 1 Comparison of the Best Results on TSP Problems
Results for ACS are from Bonabeau et al. (1999) and Dorigo and Gambardella (1997). Results for GA have been obtained using the approach proposed on Hidrobo and Aguilar (1998).

Graph	CAS	ACS	GA	Optimal
Ei150	425 (431)	425 (1830)	426 (5590)	425
Ei175	535 (931)	535 (3480)	540 (9445)	535
KroA100	21282 (1025)	21282 (4820)	21291 (12031)	21282

reported on the Web page for the TSP or Bonabeau et al. (1999); Dorigo and Gambardella (1997); and Freisleben and Merz (1996). In the case of GPP, we compare our approach with the results reported for Hidrobo and Aguilar (1998), Battiti and Bertossi (1999); and Bui and Moon (1996). In the case of GCP, we compare our approach with the optimal results reported for Costa and Hertz (1997). Results are shown in the next tables. For Tables 1 and 4, the first number is the solution cost (the value of the objective function of the best solution) and the second number represents the amount of iterations of the algorithm before the best solution was discovered. For Tables 2, 5, and 6, the first number is the value of the objective function of the best solution and the second number is the average CPU time in seconds for a simple run. For the rest of the tables, we present only the solution cost. The default value of the parameters are: $\alpha = 1, \beta = 1, \rho = 0.5$, and $m = 20$. In general, the values tested were $\alpha \in \{0, 0.5, 1\}, \beta \in \{0, 1, 2\}, \rho \in \{0.1, 0.5, 0.9\}$, and $m \in \{5, 10, 15, 20\}$. We obtain very good solutions for different parameter combinations ($\alpha = 1, \beta = 1, \rho = 0.5, m = 20; \alpha = 1, \beta = 2, \rho = 0.5, m = 20$, etc.). They have the same performance level. For more details about the parameter values for each test, see Aguilar (2000).

In the case of the TSP, our approach obtains good results with a low number of iterations (see Table 1) or execution times (see Table 2). In some cases, we have obtained similar or better results than the ACS and GA approaches (see Tables 1, 2, 3). Our approach needs the least number of iterations. In the case of the GPP, our approach can obtain the optimal result (see Table 5) or equivalent results to our GA approach (see Table 4) with a low

TABLE 2 Comparison of the Best Results on Symmetric TSP Problems
Results for ACS are from Bonabeau et al. (1999) and Dorigo and Gambardella (1997); results for GA are from Freisleben and Merz (1996).

Graph	CAS	ACS	GA	Optimal
d198	15780 (224)	15780 (238)	15780 (253)	15780
lin318	42029 (423)	42029 (537)	42029 (2054)	42029
att532	27690 (687)	27693 (810)	27686 (11780)	27686
Rat783	8809 (1054)	8818 (1280)	8806 (21210)	8806

Downloaded by [C N R S] at 09:38 05 July 2012

TABLE 3 Comparison of the Best Results on Asymmetric TSP Problems
Results for ACS are from Bonabeau et al. (1999) and Dorigo and Gambardella (1997); results for GA are from Freisleben and Merz (1996).

Graph	CAS	ACS	GA	Optimal
ry48p	14430	14422	14422	14422
kro124p	36230	36230	36230	36230
ftv170	2755	2755	2755	2755

TABLE 4 Comparison of CAS with our GA (Hidrobo and Aguilar 1998) on Randomly Generated GPP

Graph	CAS	GA
20 nodes and 4 subgraphs	46 (215)	46 (1103)
50 nodes and 5 subgraphs	244 (334)	244 (2903)
50 nodes and 7 subgraphs	260 (355)	262 (3276)
100 nodes and 5 subgraphs	1517 (823)	1517 (9819)
200 nodes and 5 subgraphs	1775 (1903)	1772 (19101)

Downloaded by [C N R S] at 09:38 05 July 2012

number of iterations, but with a moderate execution time (see Table 5). Its performance level is lower than that of specialized algorithm for the GPP (RRTS). In the case of the GCP, an exhaustive search of the different values of q (S3) is very expensive at the level of the execution time (see Table 6). We can obtain good results with the other approaches (S1, S2) with one short execution time. We obtain the best result with S1 because this approach carries on a better search over the different possible value of q. In the case of S1, the ants carry on a parallel search over the space of possible value of q in each iteration. For the GCP, if we compare CAS with a specialized ant system

TABLE 5 Comparison of the Best Results on GPP Problems
Results for RRTS are from Battiti and Bertossi (1999). Results for the "Best" column are from http://rtm.science.unitn.it/intertools/graph−partitioning−benchmark.html, these values are either the globally optimal ones, or the best values obtained by all algorithms considered in http://rtm.science.unitn.it/intertools/graph−partitioning−benchmark.html.

Graph	Best	RRTS	CAS
G500.2.5	49	49 (2)	49 (74)
G500.05	218	218 (2.5)	218 (64)
G1000.05	445	445 (6.5)	445 (73)
G1000.20	3382	3382 (14.7)	3384 (91)
U500.05	2	2 (1.7)	2 (65)
U500.40	412	412 (10.2)	412 (65)
U1000.40	737	737 (24.3)	737 (103)
bcsstk13	2355	2355 (6.7)	2355 (181)
Nasa4704	1292	1292 (11.1)	1294 (323)

TABLE 6 Comparison of the Best Results over 100 Runs
Obtained by the different versions of the CAS approach on GCP problems with those reported by http://mat.gsia.cmu.edu/COLOR/instances.html.

Graph	Optimal	CAS(S1)	CAS(S2)	CAS(S3)
le450_15a.col	15	15 (53.9)	15 (60)	15 (183.4)
le450_15c.col	15	15 (65.1)	15 (61.1)	15 (142.1)
le450_25a.col	25	26 (64.2)	29 (64.7)	25 (211.2)
le450_25c.col	25	28 (63.1)	30 (61.2)	25 (173.3)
flat_1000_50_0.col.b	50	52 (101.2)	55 (91.3)	50 (453.2)
flat_1000_60_0.col.b	60	63 (121.2)	66 (111.3)	60 (541.1)
flat_300_20_0.col.b	20	20 (43.1)	20 (38.5)	20 (170.1)
flat_300_28_0.col.b	28	29 (34.3)	31 (40.9)	28 (212.2)
inithx.i.1.col	54	56 (65.3)	56 (76)	54 (411.1)
inithx.i.3.col	31	34 (71)	37 (69.9)	32 (432)
mulsol.i.1.col	49	49 (17.8)	49 (21.1)	49 (87)
mulsol.i.3.col	31	33 (18.9)	33 (23.5)	33 (89)
mulsol.i.4.col	31	33 (16.9)	35 (20.1)	31 (82)
miles250.col	8	8 (13.1)	8 (11.8)	8 (34.7)
miles500.col	20	22 (14.2)	20 (12.2)	20 (41.2)
miles1000.col	42	42 (12.8)	44 (13.4)	42 (38.8)
mycie13.col	4	4 (2.1)	4 (2.2)	4 (7.1)
mycie14.col	5	5 (2.1)	5 (4.5)	5 (13.2)
mycie15.col	6	6 (5.1)	6 (6.5)	6 (21.1)

Downloaded by [C N R S] at 09:38 05 July 2012

(ANTCOL) (Costa and Hertz 1997), then our approach reaches the same results, but its number of iterations is larger (see Figure 7). The advantage of our approach is at the level of implementation, which is very easy in our case.

Like ACS, our system doesn't converge to a single common solution. Figure 8 represent the evolution of the best solution and the population standard deviation in the case of the Ei150 problem. Our approach maintains a high diversity, although the best solution is obtained at the iteration 431. That is, our approach has the nonconvergence property, which is common to many swarm-based systems. It avoids getting trapped in local optima. In our system, some ants can find incorrect solutions of the problems, mainly

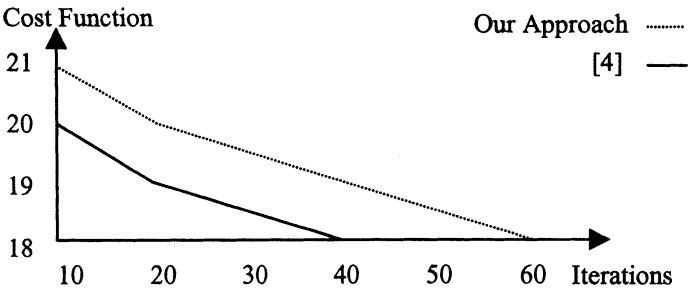

FIGURE 7. Comparison of CAS with ANTCOL for a graph with 100 vertices (Costa and Hertz 1997).

at the beginning, but the objective functions penalize that. In this way, other ants will avoid the solutions found for these ants in order to obtain correct solutions (see Figures 8, 9, 10).

Our approach never reaches the best solution within a small number of iterations. At the end, the set of ants follows similar solutions (small standard deviation). In addition, the execution time of one iteration in our approach is not small because we must calculate the partial solution costs each time an ant needs to make a decision about what is its next transition. An improved version of our approach augmented with a simple local search can reduce the number of iterations. That is, in our approach, we need to improve the combination between the global search (it expects that every ant searches a 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 the trail form when the pheromone is updated. The global search is carried out by assigning the ants to different places at the beginning. Another alternative is to avoid the same route for the ants in the first iterations. We need to implement a mechanism to undertake an improved local search.

Downloaded by [C N R S] at 09:38 05 July 2012

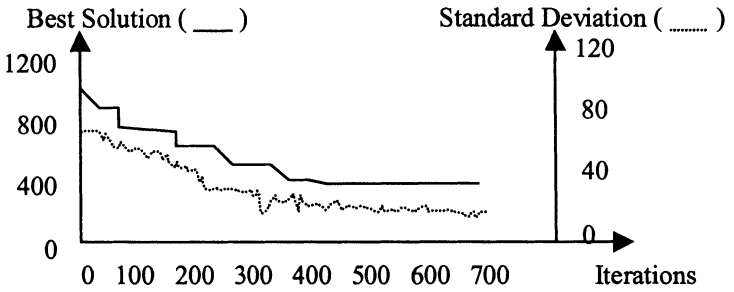

FIGURE 8. Evolution of the best solution and the population standard deviation for the case of the Eil50 problem (TSP).

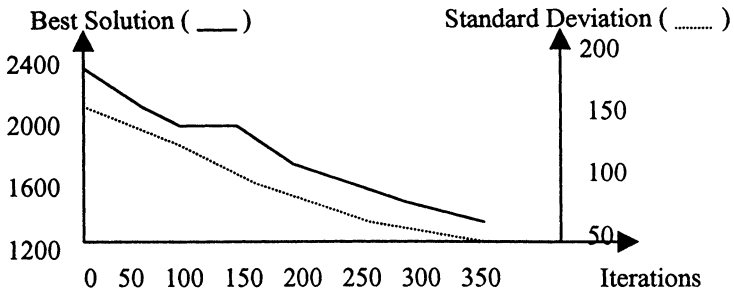

FIGURE 9. Evolution of the best solution and the population standard deviation for the case of the Nasa4704 problem(GPP).

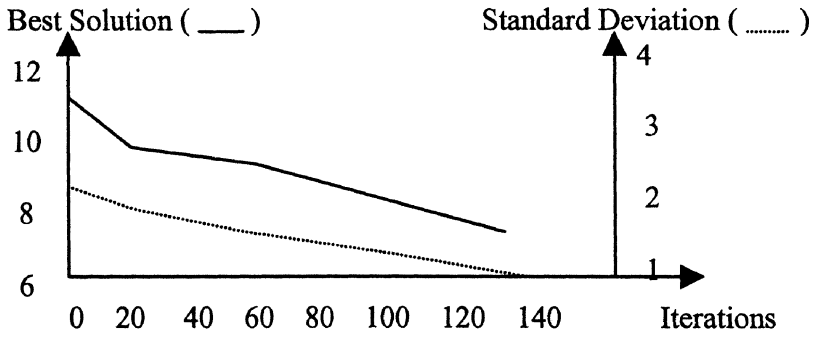

FIGURE 10. Evolution of the best solution and the population standard deviation for the case of the miles 250.col problem (GCP).

Downloaded by [C N R S] at 09:38 05 July 2012

CONCLUSIONS

In this work, we have presented a versatile approach for AS to solve combinatorial optimization problems. Our system is suited for static discrete-state combinatorial optimization problems. This versatility has been exemplified by the possibility of using the same model to solve different combinatorial optimization problems of various sizes. Our approach can be applied to any combinatorial optimization problems by defining an appropriate graph representation of the solution space of the problem considered and an objective function that guides our heuristic to build feasible solutions. In our approach, we define the solution space of the combinatorial optimization problem (COP graph) as 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 considered. In this way, we can solve any combinatorial optimization problem. We have tested our approach on the GPP, GCP, and TSP. The results show that our approach obtains good performances with a moderate number of iterations or execution times. We must improve the execution time of a given iteration and reduce the number of iterations. The execution time of a given iteration is high because our approach has an evaluation phase of the partial costs each time an ant is going to move. In general, CAS allows unfeasible solutions to make an exhaustive search the partial solution of an incorrect solution could be part of an optimal solution). Penalty functions in Eqs. (4) and (6) could be unnecessary if we set $\Delta\gamma_{rs}^{k}(t)$ to zero $\forall$rs crossed by ant k, when the solution found by ant k violates the constraints (that is an incorrect solution). In general, CAS is a versatile and general approach for static discrete-state combinatorial optimization problems, which has better performances than the classical ACS. Furthermore, we will develop a parallel

version of our approach and we will test our approach over other combinatorial optimization problems, particularly, dynamic combinatorial optimization problems.

REFERENCES

Aguilar, J. 2000. A general ant colony model to solve combinatorial optimization problems. *Technical Report, CEMISID-10-2000*, Universidad de Los Andes, Mérida, Venezuela.

Battiti, R., and A. Bertossi. 1999. Greedy, prohibition, and reactive heuristics for graph-partitioning. *IEEE Transactions on Computers*, 48:361–385.

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

Bui, T., and B. Moon. 1996. Genetic algorithms and graph partitioning. *IEEE Transaction on Computers*, 45:841–855.

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

Costa, D., and A. Hertz. 1997. Ants can colour graphs. *Journal of the Operational Research Society* 48:295–305.

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

Dorigo, M., and L. Gambardella 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, 1:53–66.

Dorigo, M., V. Maniezzo, and A. Coloni. 1996. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man, Cybern.* 26:29–41.

Freisleben, B., and P. Merz. 1996. Genetic local search algorithm for solving symmetric and assymetric traveling salesman problems. In *Proceedings IEEE International Conference on Evolutionary Computation*, pages 616–621, Chicago.

Hidrobo, F., and J. Aguilar. 1998. Toward a parallel genetic algorithm approach based on collective intelligence for combinatorial optimization problems. In *Proceedings IEEE International Conference on Evolutionary Computation*. pages 715–720, Washington.

Different results for the graph coloring problem. http://mat.gsia.cmu.edu/COLOR/instances.html

Benchmarks for the graph partitioning problem. http://rtm.science.unitn.it/intertools/graph-partitioning/benchmark.html

Kuntz, P., P. Layzell, and D. Snyers. 1997. A colony of ant-like agents for partitioning in VLSI technology. In *Proceedings Fourth European Conference on Artificial Life*. pages 417–424, Barcelona.

Kuntz, P., and D. Snyers. 1994. Emergent colonization and graph partitioning. In *Proceedings Third International Conference on Simulation of Adaptive Behavior: From Animals to Animals*, page 3.

Schoonderwoerd, R., O. Holland, J. Bruten, and L. Rothkrantz. 1997. Ant-based load balancing in telecommunications networks. *Adaptive Behavior.* 5:169–207.

Stutzle, Y., and H. Hoos. 1997. The max-min ant system and local search for the traveling salesman problem. In *Proceedings 4th Int. Conf. on Evolutionary Computation*, pages 309–314, Orlando.

Downloaded by [C N R S] at 09:38 05 July 2012