Годишник на секция "Информатика"
Съюз на учените в България
Том 4, 2011, 62-67

Annual of "Informatics" Section
Union of Scientists in Bulgaria
Volume 4, 2011, 62-67

ON A REPRESENTATION OF ACO-ALGORITHM BY GAME METHOD FOR MODELLING

Krassimir Atanassov[1] and Stefka Fidanova[2]

[1] Institute of Biophysics and Biomedical Engineering
Bulgarian Academy of Sciences
105 Acad. G. Bonchev Str., 1113 Sofia, Bulgaria,
e-mail: krat@bas.bg
[2] Institute of Information and Communiction Technologies - Bulgarian
Academy of Sciences,
e-mail: stefka@parallel.bas.bg

Abstract: The combinatorial game-method has mostly theoretical applications in astronomy for example. It must be noted that there exist various methods which use different approaches and have applications in other mathematical areas. In this paper is shown that the game method for modelling can be used for a representation of the ACO-algorithms.
Keywords: ACO-algorithm, Game method for modelling, Modelling

1. Introduction

Real ants foraging for food lay down quantities of pheromone (chemical cues) marking the path that they follow. An isolated ant moves essentially at random but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce it with a further quantity of pheromone. The repetition of the above mechanism represents the autocatalytic behaviour of a real ant colony where the more the ants follow a trail, the more attractive that trail becomes.

The Ant Colony Optimization (ACO) procedure is one of the modern mathematical tools for optimization, based on metaheuristic rules. The solution of a given optimization problem is represented by a path or a tree in a graph. The shortest path or the tree with smallest (greatest) coefficients weight (if the problems is related to minimum or maximum, respectively) is searched [3, 4, 5].

The ACO algorithm uses a colony of artificial ants that behave as cooperative agents in a mathematical space where they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. The problem is represented by graph and the ants walk on the graph to construct solutions. The solutions are represented by paths in the graph. After the initialization of the pheromone trails, the ants construct feasible solutions, starting from random nodes, and then the pheromone trails are updated. At each step the ants compute a set of feasible moves and select the best one (according to some probabilistic rules) to continue the rest of the tour. The structure of the ACO algorithm is shown by the pseudocode below (Figure 1). The transition probability $p_{i,j}$, to choose the node j when the current node is i, is based on the heuristic information $\eta_{i,j}$ and the pheromone trail level $\tau_{i,j}$ of the move, where $i, j = 1, \ldots, n$.

$$p_{i,j} = \frac{\tau_{i,j}^a \eta_{i,j}^b}{\sum_{k \in Unused} \tau_{i,k}^a \eta_{i,k}^b}, \qquad (*)$$

where $Unused$ is the set of unused nodes of the graph.

The greater the value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search. In the beginning, the initial pheromone level is set to a small positive constant value τ_0; later, the ants update this value after completing the construction stage. ACO algorithms adopt different criteria to update the pheromone level [6, 7].

```
Hybrid Ant Colony Optimization
Initialize number of ants;
Initialize the ACO parameters;
while not end-condition do
    for k=0 to number of ants
        ant k starts from random node;
        while solution is not constructed do
            ant k selects higher probability node;
        end while
    end for
    Update-pheromone-trails;
end while
```

Figure 1: Pseudocode for ACO

The pheromone trail update rule is given by:

$$\tau_{i,j} \leftarrow \rho\tau_{i,j} + \Delta\tau_{i,j},$$

where $\rho \in (0, 1)$ models evaporation in nature and $\Delta\tau_{i,j}$ is newly added pheromone which is proportional to the quality of the solution.

The (artificial) ant starts from an arbitrary node of the graph and with its path in the graph, it constructs its individual solution of the problem. It includes a new node in the list of it solution, using the probability of the transition to it. The conditions, that the solution must satisfy, are included in the probability of ant's transition. The ant adds new nodes in the graph until the moment when it satisfies the final (stop) condition, that shows when it had already constructed a correct solution. When all ants construct their solutions, these solutions are comprised, and a new pheromone is added in the graph components (nodes and/or arcs). This pheromone is inversely proportional to the value of the objective function, if the problem is related to searching a minimum. So, the graph components (nodes and/or arcs) corresponding to the better solutions obtain more pheromone than the other components, and on the next iteration they will be more desirable by the ants.

2. Main results

In this paper, we show representation of ACO algorithm by Game Method for Modelling (GMM, see [1]).

Let us have a graph on which the ants will transfer, this graph represents the problem to be solved. If the graph is not planar, then we can construct a 3-dimensional GMM-grid similarly to the form of the 3-dimensional grid of the GMM representing a neural network from [2]. For brevity and better visualization, we will use a planar graph. For it, we can construct a GMM-grid. Cells of the grid correspond to graph, and corridors connecting two neighboring cells, nodes correspond to graph arcs. The ant-transfer will be realized only in these corridors.

Now, GMM-objects will stay in cells, marked by "*", and these objects have as a characteristic:

A) the value $\tau(x)$ of the pheromone from previous ant movements through the same cell "x";

B) a transition function that is calculated on the basis of the first cell characteristic (the value of the pheromone) and of a heuristic information, that is a characteristic of the ant, entering the cell "x".

The ants will be represented by objects, initially staying in some cells of the GMM-grid. These objects have as initial characteristic some heuristic information – function "η", that will be used for their transfer in the graph. The heuristic information η is a combination of parameters from objective function and constraints. Well done heuristic function direct ants to search for better solutions. For example, the only available heuristic information for Traveling Salesman Problem is the visibility of the next point to be visited, while in the Knapsack Problem there are constraints and we can construct various heuristics including in numerator the parameters from the objective function (price of the object) and in the denominator combinations of the constraint parameters (the volume of the objects and the capacity of the knapsacks).

The rules of the GMM-model are the following:

1. Each ant starts its transfer from a random cell of the GMM-grid with an initial characteristic being some heuristic function for the future path or for the future aim, and a vector $\mathcal{L}$ with one element being the identifier of the cell, where the ant is located at the initial moment.

2. With respect to GMM-object place in the corridor, it can have one, two, three or four possible directions for movement (see Figs. 1 – 4). At the following time-steps the object will have one, two or three possible directions for movement, because it cannot go in its opposite direction. i.e., through cells that it had already visited. The direction of movement is determined as follows.

2.1. if the cell "a" has only one neighboring cell ("b"), where the object can enter: $a \to b$ (see Fig. 2).

2.2. if the cell "a" has two neighboring cells ("b" and "c"), where the object can enter:

$$a \to \begin{cases} b & \text{if } \frac{\tau(b)\eta(b)}{\tau(b)\eta(b)+\tau(c)\eta(c)} = \max\limits_{x\in\{b,c\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)} \\ c & \text{if } \frac{\tau(c)\eta(c)}{\tau(b)\eta(b)+\tau(c)\eta(c)} = \max\limits_{x\in\{b,c\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)} \end{cases}.$$

(see Fig. 3)

2.3. if the cell "a" has three neighboring cells ("b", "c" and "d"), where the object can enter:

$$a \to \begin{cases} b & \text{if } \frac{\tau(b)\eta(b)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)} = \max\limits_{x\in\{b,c,d\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)} \\ c & \text{if } \frac{\tau(c)\eta(c)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)} = \max\limits_{x\in\{b,c,d\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)} \\ d & \text{if } \frac{\tau(d)\eta(d)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)} = \max\limits_{x\in\{b,c,d\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)} \end{cases}.$$

(see Fig. 4)

2.4. (only for the first time-moment) if the cell "a" has four neighboring cells ("b", "c", "d" and "e"), where the object can enter:

$$a \to \begin{cases} b & \text{if } \frac{\tau(b)\eta(b)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ & \quad = \max\limits_{x\in\{b,c,d,e\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ c & \text{if } \frac{\tau(c)\eta(c)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ & \quad = \max\limits_{x\in\{b,c,d,e\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ d & \text{if } \frac{\tau(d)\eta(d)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ & \quad = \max\limits_{x\in\{b,c,d,e\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ e & \text{if } \frac{\tau(e)\eta(e)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \\ & \quad = \max\limits_{x\in\{b,c,d,e\}} \frac{\tau(x)\eta(x)}{\tau(b)\eta(b)+\tau(c)\eta(c)+\tau(d)\eta(d)+\tau(e)\eta(e)} \end{cases}.$$

(see Fig. 5)

3. When an object enters a new cell, its characteristic is changed as follows:

$$\mathcal{L} := \langle \mathcal{L}, b \rangle.$$

4. The object stops its movement when:

   4.1. there are no more cells that have not been visited,

   4.2. some of the problem constraints are violated.

5. When all objects stop, we update the pheromone, and the cells receive as new characteristics the new value of the pheromone.

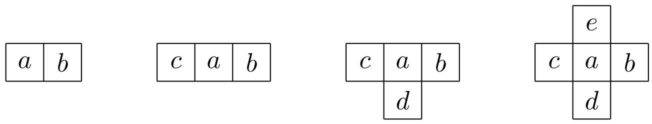

Fig. 2. Fig. 3. Fig. 4. Fig. 5.

We must note that more than one separate objects can enter one cell and they will not interact there. On the next time-step, they will continue their movements independently. Their heuristic information will be different, because they have different history (different partial solution).

3. Conclusion

In this paper, a first step to constructing of a game method for modelling representation the ACO-algorithms is done. In a next research a 3-dimensional model will be constructed. Our aim is detail representation and improvement of ACO performance.

4. Acknowledgment

The authors are grateful for the support provided by the projects DID-02-29 "Modelling processes with fixed development rules" funded by the National Science Fund, Bulgarian Ministry of Education, Youth and Science.

References

[1] Atanassov K., On the game-method for modelling, Advanced Studies in Contemporary Mathematics (in press).

[2] Atanassov, K., S. Sotirov, On a Representation of Neural networks by Game Method for Modelling, Advanced Studies in Contemporary Mathematics (in press).

[3] Dorigo M., Gambardella L.M., Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation 1, 53–66, 1997.

[4] Dorigo M., Di Caro G. Gambardella M. (1999) Ant Algorithms for Distributed Discrete Optimization. J. of Artificial Life. 5, 137 – 172.

[5] Dorigo, M., Maniezzo V., Colorni A. (1996). The Ant System: Optimization by a Colony of Cooperative Agents. IEEE Transactions on Systems, Man and Cybernetics – part B. 26(1), 29 - 41.

[6] Dorigo M., Stutzle T., Ant Colony Optimization, MIT Press, 2004.

[7] Fidanova S. (2002) ACO Algorithm with Additional Reinforcement. In Dorigo M., Di Caro G. Sampels M. eds. From Ant Colonies to Artificial Ants, Lecture Notes in Computer Science. 2463, 292 – 293.