An Ant-based Technique for the Dynamic Generalized Traveling Salesman Problem

CAMELIA M. PINTEA[1] , PETRICĂ C. POP[2], D. DUMITRESCU[3]
[1]Department of Computer Science, Faculty of Mathematics and Computer Science
Babeş-Bolyai University
Cluj-Napoca 400084
ROMANIA

[2]Department of Mathematics and Computer Science, Faculty of Sciences
North University of Baia Mare
Baia-Mare 430122
ROMANIA

[3]Department Computer Science, Faculty of Mathematics and Computer Science
Babeş-Bolyai University
Cluj-Napoca 400084
ROMANIA

Abstract: - In this paper we present an effective metaheuristic algorithm based on ant colony system in the case of the dynamic generalized traveling salesman problem. The same technique can be used for other dynamic large scale *NP*-hard problems encountered in telecommunications, transportation, network design, etc. In the dynamic versions that we consider, if we look at the distance between nodes as travel times they are no longer fixed.
Computational results are reported in the case of dynamic generalized traveling salesman problem for some real data sets.

Key-Words: ant colony algorithms, generalized traveling salesman problem

1 Introduction

Ant-systems based techniques are metaheuristics able to solve large *NP*-hard problems. The *Ant Colony System* (*ACS*) is a system based on agents which simulate the natural behavior of ants, including mechanisms of cooperation and adaptation, [2].
A metaheuristic based on ant algorithms for the static Generalized Traveling Salesman Problem (GTSP) was given in [7]. This paper shows a dynamic version of the Generalized Traveling Salesman Problem based on Ant Colony System. In GTSP the nodes of a complete undirected large graph are partitioned into set of nodes, called clusters. The dynamism is there at each moment when a cluster, determined with a given probability, is missing from the tour. In the real life frequently appear blocked ways due to poor weather, accidents, maintenance, etc. The objective is to find a minimum cost tour passing through exactly one node from each available cluster.
The paper is structured as follows. The preliminary section shows the Ant based system for the Generalized Traveling Salesman Problem. The main section of the paper is The Ant Colony System for Dynamic GTSP followed by the numerical tests section. Two sets of real data are used to illustrate the numerical results in the case of static and dynamic generalized traveling salesman problem.

2 Ant-based system for the Generalized Traveling Salesman Problem

Given an undirected graph $G = (V, E)$ with the nodes partitioned into clusters and edges defined between nodes from different clusters having a non-negative cost, the *Generalized Traveling Salesman Problem* (*GTSP*), introduced by introduced by Laporte and Nobert [6], asks for finding a minimum cost tour H spanning a subset of nodes such that H contains exactly one node from each cluster.
In [7], Pintea *et al*. proposed a new algorithm based on *Ant Colony System*, in order to solve the *Generalized Traveling Salesman Problem* (*GTSP*). In Ant Colony System (ACS) for GTSP the pheromone trails are updated

immediately after the ants have chosen the corresponding edges to travel on. This local updating decrease the amounts of pheromone on the edges just selected, preventing in this way stagnation. When all ants have constructed their tours, a global updating will be performed. The global updating means pheromone evaporation on all edges, excepting those belonging to the global best tour, which will receive additional pheromone. The best tour is the solution of the given problem solved with ant-based system.

3 The Ant Colony System for Dynamic GSPT

The dynamic *GTSP*, is a variation on the *GTSP* in the sense that can appear delays due to maintenance work, accidents, etc. and therefore the total cost of a tour in a graph may vary.

We mention that several others variants for combinatorial optimization problems are considered, such as variants resulting from insertion or deletion of cities, see [4, 5].

At the beginning there is relatively much exploration, after a while all connections that are not promising will be slowly cut off from the search space because they do not get any positive reinforcement and the associated pheromones have evaporated over time. In the Dynamic *GTSP*, at each iteration of the new algorithm, one cluster is blocked.

For solving the Dynamic *GTSP*, the algorithm from [7] should be changed following the characteristics of the dynamic problem: starting from a specific city from a cluster, the salesman has to visit a given number of cities and finally he must return to the starting cluster.

In the *Ant Colony System* for *Dynamic GTSP* algorithm m ants incrementally construct solutions. The choice of the next node is based on two main components: pheromone trails and a heuristic value called visibility.

At the beginning, the ants are placed in a node from a chosen starting cluster and all the edges are initialized with a certain amount of pheromone.

At each moment $(t+1)$ a random cluster is missing. Every ant moves to a new node from an unvisited *cluster* and the parameters controlling the algorithm are updated.

For each edge (i, j), where *i* is the starting node and *j* is the arrival node from an unvisited cluster, at time *t*, each edge is labeled by a trail intensity, $\tau_{ij}(t)$.

An ant decides in which arrival node is the next move with a probability that is based on the time and the amount of trail intensity on the connecting edge. The *visibility* η_{ij} is the inverse of distance from a starting node to the next arrival node.

At each time unit evaporation takes place. This is to stop the intensity trails increasing unbounded. In order to stop ants visiting the same *cluster* in the same tour a tabu list is maintained. This prevents ants visiting *clusters* they have previously visited. This tabu list is cleared after each completed tour.

In order to favor the selection of an edge with high pheromone intensity τ, and high visibility η, a probability function p_{iu}^k is considered. Let J_i^k be the unvisited neighbors of node i by ant k and if $u \in J_i^k$ $u = J_k(y)$, where y is a node from the unvisited cluster J_k. The probability function is defined as follows:

$$p_{iu}^k(t) = \frac{[\tau_{iu}(t)]^{\alpha}[\eta_{iu}(t)]^{\phi}}{\Sigma_{o \in J_i^k}[\tau_{io}(t)]^{\alpha}[\eta_{io}(t)]^{\beta}}, \quad (1)$$

where α and β are parameters used for tuning the relative importance of edge time in selecting the next node, p_{iu}^k is the probability of choosing $j = u$, where $u = J_k(y)$ is the next node, if $q > q_0$ (the current node is i). If $q \le q_0$ the next node j is chosen as follows:

$$j = \arg\max_{u \in J_i^k}\left\{[\tau_{iu}(t)]^{\alpha}[\eta_{iu}(t)]^{\beta}\right\}, \quad (2)$$

where q is a random variable uniformly distributed over [0,1] and q_0 is a parameter similar to the temperature in simulated annealing, $0 \le q_0 \le 1$.

In the dynamic case, solutions that are bad before a change in the environment might be good afterwards. Now, if the ant system has converged to a state where those solutions are ignored, very promising connections will be lost and the result will be a suboptimal solution. That is way we use a new local update rule.

A technique called *shaking* is used in order to smooth all the pheromone levels in a certain way. If the amount of pheromones on an edge becomes much higher than all the other edges, this edge will be always be chosen. That is a way for the static case to ensure that a good connection will always be followed, but it prevents ants from taking another connection when the good connection is blocked. The formula used in shaking is the local update rule:

$$\tau_{ij}(t+1) = \tau_0\left(1 + \log\left(\frac{\tau_{ij}(t)}{\tau_0}\right)\right). \quad (3)$$

In *Ant Colony System* only the ant that generates the best tour is allowed to *globally* update the pheromone. The global update rule is applied to the edges belonging to the *best tour*. The correction rule is

$$\tau_{ij}(t+1) = (1-\rho)^2\tau_{ij}(t) + \rho^2\Delta\tau_{ij}(t), \quad (4)$$

where $\Delta\tau_{ij}(t)$ is the inverse time of the best tour.

At the beginning, the ants are in their nest and will start to search food in a specific area. An ant goes exactly once through a given number of clusters, nc and returns to the nest. The more pheromone there is on a certain edge the bigger the chance that edge will be taken. Therefore, the pheromone trails will guide the ants to the shorter path, a solution of *ACS* for *Dynamic GTSP*.

After each completed tour the amount of pheromone is updated. On every edge some fraction of pheromone evaporates. All edges on the best tour get an extra amount of pheromone. The entire process goes on until a certain condition, such as a certain number of iterations N_{iter} have been achieved.

The *ACS* for *Dynamic GTSP* algorithm can be stated as follows:

Algorithm 1. Ant Colony System for Dynamic GTSP
begin
for every edge (i, j) do $\tau_{ij}(0) = \tau_0$
for $k = 1$ to m do
  place ant k on a *specified* chosen node from a *specified* cluster
let T^+ be the shortest tour found and L^+ its length
for $i = 1$ to N_{iter} **do**
  for $k = 1$ to m **do**
    random choose a cluster and set the cluster visited
    build tour $T^k(t)$ by applying $nc - 1$ times
      choose the next node *j* from an unvisited cluster (1)(2)
      update pheromone trails by applying the local rule (3)
end for
for $k = 1$ to m **do**
    compute the length $L^k(t)$ of the tour $T^k(t)$
if an improved tour is found **then** update $T^k(t)$ and $L^k(t)$
for every edge $(i, j) \in T^+$ **do**
    update pheromone trails by applying the global rule (4)
end for
print the shortest tour T^+ and its length L^+
end

The next figure describes the ACS for Dynamic GTSP. After several iterations per- formed by each ant the partial solutions are visible. The largest pheromone trail gives the partial solutions of the problem. On all the other trails the pheromone is evaporating. The shortest path found after a given number of iterations is the optimal solution.

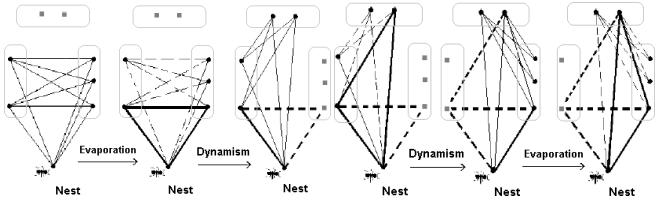

Figure 1. Starting from the nest to find food, an ant is going once through each valuable cluster and is returning to the nest; all the ways are initialized with the same pheromone quantity; after several iterations performed by each ant, the partial solutions are visible by the largest pheromone trail (thick lines); the pheromone is evaporating on all the other trails (dashed lines). The optimal solution it is the shortest path found.

4 Numerical tests

Comparative tests are performed to illustrate the robustness of the proposed metaheuristic. The tests are performed on instances from TSPLIB, [1].
In the following we choose to solve two real data sets: an Euclidean data set and a geographical one. To divide the set of nodes into subsets we used the procedure proposed in [3]. The procedure sets the number of clusters $nc = [n/5]$, identifying the nc farthest nodes from each other, called centers and assigning each remaining node to its nearest center. Some real world problems may have different cluster structures, but the solution procedure presented in this paper is able to handle any cluster structure.

The first data set is related to the *Generalized Traveling Salesman Problem* referring to a salesman which starting from a given city has to make a tour on all the available cities. In the real life can appear delays due to maintenance work, accidents, etc. and therefore the cost of a travel may vary.

Table 1 with the tests results on Euclidean instances is using the notations:

Problem: The name of the test problem. The digits at the beginning of the name give the number of clusters; those at the end give the number of nodes.

Opt. Static: The optimal objective value for the static problem, [3].

Dyn.Value: The average, minimum, and maximum objective values returned in the five trials for the dynamic problem.

Dyn.Time: The Average, Minimum and Maximum CPU time, in seconds, for each of the five trials of the dynamic algorithm.

Parameters used are $q_0 = 0.9$, $\alpha = 1$, $\beta = 7$, $\tau_0 = 0.01$, $m = 3$, $\rho = 0.01$ and maximal 30000 iterations.

Table 1. Euclidean data benchmarks for ACS for dynamic GTSP

Problem	Opt. Static	Dyn. Values			Dyn. Time		
		Min.	Avg.	Max.	Min.	Avg.	Max
11EIL51	174	111	124.8	136	82.2	82.2	208
14ST70	316	227	229	232	283.6	283.6	510
16EIL76	209	172	174	176	983.4	983.4	1712
20KROA100	9711	9316	9357.2	9397	323	323	745
21EIL101	249	226	227.6	229	9.6	9.6	29

It is possible to obtain better solutions than the minimal one. Since today there are not known the optimal values for the dynamic *GTSP* on TSPLIB instances.

The *Generalized Traveling Salesman Problem* for the large airport instance considered, refers to a salesman which starting from a given airport, from a country, have to make a tour on all the available airports around the globe. The clusters are considered the countries and the nodes of the general graph are the airports. Many delays may appear in real life and also the cost of a travel may vary.

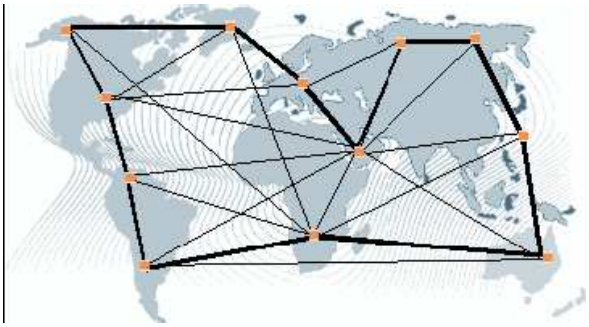

Fig. 2. Graph representation of the geographical problem

This problem is geographical. The nodes of the graph correspond to points on the earth. The distance between two points is their distance on the idealized sphere with radius 6378.388 kilometers, [10]. The node coordinates give the geographical latitude and longitude of the corresponding point on the earth. A positive latitude is "North", negative latitude is "South". Positive longitude is "East" and negative latitude is "West". Let assume that $x[i]$ and $y[i]$ are the coordinates for city i in the specified format. The input it is converted to geographical latitude and longitude given in radians, as in [10], followed by the computation of distance d_{ij} between two different nodes i and j in kilometers.

PI=3.141592;

$latitude[i] = PI \cdot (nint(x[i] + 5.0 \cdot (x[i] - nint(x[i]))/3.0)/180.0;$

$longitude[i] = PI \cdot (nint(y[i] + 5.0 \cdot (y[i] - nint(y[i]))/3.0)/180.0;$

$q1 = cos(longitude[i] - longitude[j]);$

$q2 = cos(latitude[i] - latitude[j]);$

$q3 = cos(latitude[i] + latitude[j]);$

$d_{ij} = (int)(6378.388 \cdot ArcCos(0.5 \cdot ((1.0+q1) \cdot q2) - (1.0 - q1) \cdot q3)) + 1.0);$

The problem is the instance *ali535* from TSPLIB [10], with 535 Airports around the globe (Padberg/Rinaldi). For this problem, since today there are not known the optimal values for the static and dynamic *GTSP* value.

In Table 2 we present the results of five successively runs of each of the algorithms. There are used the same specifications of the parameters as for the already tested euclidean data set.

Table 2. Ant Colony-based Systems for the geographical data set

Static Values			*Dynamic Values*		
Min.	Avg.	Max.	Min.	Avg.	Max.
58007	60137.6	62124	**51251**	52766.8	54184

The computer used for computation is a AMD 2600, 1.9Ghz, 512GB RAM. It is possible to obtain better results for the minimal value, using a computer with higher performances or/and using better value for the parameters. The average time for the static problem is 429.6 seconds and for the dynamic GTSP is 796 seconds.

4 Conclusion

We describe an effective meta-heuristic algorithm in the case of a dynamic Generalized Traveling Salesman Problem (GTSP). Within GTSP, a recently proposed NP-hard problem, the nodes of a complete undirected graph are partitioned into clusters. At each moment an unvisited cluster is unavailable (is blocked based on a given probability). The objective is to find a minimum cost tour passing through exactly one node from each available cluster. Numerical tests were performed for the first time, for some real data from TSPLIB, including Euclidean and geographical problems.

In a large graph or network, the cluster to be blocked could be chosen in a deterministic way, not only probabilistic, depending on the constraints of the problem. Other dynamic large scale NP-hard problems encountered in telecommunications, transportation, network design, etc. could be solved using the Dynamic ACS for GTSP or an improved version of the algorithm.
In the future should be done tests using other algorithms to compare the actual ant-based algorithm. For the static case [7], the results are in general better for ant-based mechanisms than other algorithms: nearest neighbor, an composite heuristic for the Symmetric GTSP [8] and a random key genetic algorithm [9].

Acknowledgements: This work was supported by the CEEX grant ET34/2006, contract no. 5867/18.09.2006 from the Romanian Ministry of Education and Research.

References:

[1] B. Bixby and G. Reinelt, http://nhse.cs.rice.edu/ softlib/catalog/tsplib.html 1995
[2] M. Dorigo and L.M. Gambardella, Ant Colony System, A cooperative learning approach to the Traveling Salesman Problem, *IEEE Trans. Evol. Comp*., Vol. 1, 1997, pp. 53-66
[3] M. Fischetti, J.J. Salazar and P. Toth, A branch-and-cut algorithm for the symmetric generalized traveling Salesman problem, *Operations Research*, Vol. 45, 1997, pp. 378-394
[4] M. Guntsch and M. Middendorf, Pheromone modification strategies for ant algorithms applied to dynamic TSP, in *Applications of Evolutionary Computing*: Proceedings of EvoWorkshops 2001, Springer Verlag 2001
[5] M. Guntsch, M. Middendorf and H. Schmeck, An Ant Colony Optimization approach to dynamic TSP, *Proceedings of GECCO'2001*, 2001, pp. 860-867
[6] G. Laporte and Y. Nobert, Generalized traveling salesman problem through n sets of nodes, An integer programming approach, *INFOR* 21, 1, 1983, pp. 61-75
[7] C. Pintea, P.C. Pop and C. Chira, The Generalized Traveling Salesman Problem solved with Ant Algorithms, *Journal of Universal Computer Science*, Graz, 2007, in press
[8] J. Renaud, and F.F. Boctor, An efficient composite heuristic for the Symmetric Generalized Traveling Salesman Problem, *European Journal of Operational Research*, 108, 3, 1998
[9] L.V. Snyder and M.S. Daskin, A Random-Key Genetic Algorithm for the Generalized Traveling Salesman Problem, *INFORMS*, San Antonio, TX 2000
[10] http://www.informatik.uni-heidelberg.de/groups/ co-mopt / software/ TSPLIB95/doc.ps.