# The Optimal Routing of Cars in the Car Navigation System by Taking the Combination of Divide and Conquer Method and Ant Colony Algorithm into Consideration

Pirayeh Yousefi and Roghayeh Zamani

***Abstract*—In this paper we propose an optimal routing method for cars in car navigation system. The proposed method finds the paths with a combination of Divide and Conquer method and Ant Colony algorithm. In order to do this, the road network is divided to small areas. Then the learning operation is done in these small areas. Then different learnt paths are combined together to make the complete paths. This method causes balance and reduces the traffic in lanes of the pathes, because it not only consider lengths of the paths for learning operations, it also considers the other factor which is traffic condition of the lanes of the paths. Consequently this method reduces the average triptime of the cars in comparison with existing Ant Colony, Genetic and Dijkstra algorithms by reducing and balancing of road traffic. Therefore, some improvements in the average traveling time of vehicles are achieved. Also, this results in short and precise paths and learning stage becomes faster.**

***Index Terms*—Ant Colony algorithm, controlling system, car navigation system, Divide and Conquer method, routing algorithm.**

## I. Introduction

The optimal routing issue is inevitable in lots of cases including railroad and in communication and urban road networks.

Since the road traffic changes in rush hours and with respect to the fact that the number of people using personal cars increases, there is an urgent need to have a system which can find optimal path with the short response time.

Because of the nature of the car navigation issue, as soon as a vehicle arrives to a junction, the system must determine the optimal path and represent it to the vehicle. If the response time of the system is long, it will result in heavy traffic on roads.

There are a lot of implementations for optimal routing of vehicle in the past with certain and uncertain algorithms and experiments show that the uncertain algorithms produce better results in comparison to certain algorithms. Yet, each of them has its own advantages and disadvantages which will be discussed in detail in the following paragraphs. Much of the researches and algorithms in this system have been based on Dijkstra's shortest path, Restricted Search, and A* algorithms [6].

On the other hand, the road network is a big graph with lots of edges and real-time response is necessary. So, most of the researches attempted to optimize that parameter.

The disadvantage of these methods is that they just consider physical distance as the only optimizing criteria and they ignore the most important factor which is the traffic condition of the paths.

Some other methods, which use graph partitioning with respect to find the real time solution, are proposed in literature but their solutions do not have high quality [7], [8]. The most important factor in these methods (graph partitioning) is to divide the graph and to determine intermediate edges among these sub graphs.

One of the difficulties with the above mentioned methods is that they are not able to consider turning restriction in the junctions and paths, i.e. Dijkstra, restricted search and A* algorithms cannot consider these restrictions for determining the shortest path from one node to other nodes, yet this restriction is inevitable. Also the paths with the restricted search algorithm may just be locally optimal [1].

In this paper, we propose a method of optimal path searching that is based on combination of divide and conquer method and Ant colony algorithm. This method gives responses for driver's requests in a short time and we can consider restrictions such as turning left and right in the junctions. The pathes which are found with our method are globally optimal. So our proposed method doesn't have most of the restrictions and disadvantages which the previous methods faced. To avoid confusing between two terms, "path" and the "path which our method registers in the nodes", we call a learned path of proposed method as "Antpath". This paper is organized as follows.

In Section 2, we introduce the basic Ant Colony algorithm. In Section 3, we explain our proposed method which is based on divide and conquer method and Ant colony algorithm. Section 4 deals with explanation of experiments. Section 5, represents a pseudo code of implemented method and the last section represents the results of the experiments.

## II. Ant Colony Algorithm

Real ants are capable of finding the shortest path from a food source to the nest, without using visual ques. Also, they are capable of adapting to changes in the environment, for example finding a new shortest path once the old one is no longer feasible due to a new obstacle. Ants are moving on a straight line that connects a food source to their nest. The Ant Colony algorithm is inspired of the behaviour of real ants and with respect to how to use pheromone, tries to find near optimal path or every other parameter that should be

Manuscript received September 19, 2012, revised November23, 2012.

The authors are with the Marand Branch, Islamic Azad University, Marand, Iran (e-mail: p_yousefi@marandiau.ac.ir; r_zamani84@yahoo.com).

optimized [3].

The method which ants use for marking these shortest paths is called "pheromone". In the other words, ants deposit a certain amount of pheromone while walking and each ant probably prefers a direction that has a high level of pheromone. Due to this positive feedback process, all of the ants will rapidly choose the shortest path.

There are agents in this algorithm that are called "ants", and they memorize all of nodes which they pass over them and when arrive at a junction, they can determine which of the front lanes are allowed. We do this by using an array called Tabu list. A Tabu list contains the nodes which are crossed over by an ant. So, an ant is not allowed to cross a lane which led to a junction that is already inside its Tabu list. This list is individual for each ant.

Initially, all of the paths of a node which go to all of the edge nodes have an initial amount of pheromone. Suppose ant $k$ is in the node $r$ and chooses node $j$ selected from allowed nodes (nodes which are not in the Tabu list of this ant and also are one of adjacent nodes of $r$) with (1):

$$j = \begin{cases} \underset{u}{argmax}\left\{[\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}\right\}_u & if\ q \le q_0\ and\ u \notin M_k \\ S & otherwise \end{cases} \quad (1)$$

where $S$ is a random selected node according to $P_k(r,j)$ (2), which favors edges which are shorter and have a higher level of pheromone trail, $\tau(r,j)$ is the amount of pheromone trail on edge $(r,j)$, $\eta(r,j)$ is a heuristic function, which was chosen to be the inverse of the distance between node $r$ and $u$. $\alpha$ and $\beta$ are parameters which weigh the relative importance of pheromone trail and of closeness, $M_k$ is the Tabu list of vehicle $k$, $q$ is a value chosen randomly with uniform probability in [0,1], $q_0$ $(0 \le q_0 \le 1)$is a parameter [2].

$$P_k(r,j) = \begin{cases} \frac{[\tau(r,j)]^{\alpha}.[\eta(r,j)]^{\beta}}{\sum_{u \notin M_k}[\tau(r,u)]^{\alpha}.[\eta(r,u)]^{\beta}} & if\ j \notin M_k \\ 0 & otherwise \end{cases} \quad (2)$$

$u$ is the every node that isn't in the Tabu list of ant $k$ and is one of adjacent nodes of $r$ too.

To find the shortest path in the car navigation system, the length of the edge is not important; however the total length of path is important.

The best ant, among those that have the same sources and destinations are the same, is the one which finds the shortest path. There were found several paths between node a and b, and now if the path which is found by an ant from node a to b is shorter than previously found ones, it should be updated by global updating (3) and (4), otherwise it should be updated with local updating (5).

Global trail updating is as the following:

$$\tau'^{(r,j)} = (1-\rho).\tau(r,j) + \rho.\Delta\tau(r,j) \quad (3)$$

where $\tau(r,j)$ is the amount of pheromone trail of edge $(r,j)$ before being updated called "visibility", $\tau'(r,j)$ is the amount of pheromone trail of edge $(r,j)$ after being update, $\rho$ is coefficient of evaporation of pheromone inspired from natural evaporation of pheromone, $\rho \in (0,1)$, $\Delta\tau(r,j)$ is the amount of pheromone the best ant deposits on edges of its Antpath that is inverse proportion of the length of the path.

$$\Delta\tau(r,j) = \frac{1}{l_{sp}} \quad (4)$$

where $l_{sp}$ is the length of the shortest Antpath.

The Antpaths from node a to node b whose lengths are equal with previously founded Antpaths length should be updated locally. The local trail updating equation is as bellow:

$$\tau'(r,j) = (1-\rho).\tau(r,j) + \rho.\tau_0 \quad (5)$$

where $\tau(r,j)$ is the amount of pheromone trail of edge $(r,j)$ before doing local update. $\tau_0$ must be smaller than $\Delta\tau(r,j)$ which is used in (3).

The proposed method for solving the problem of optimal path in vehicle navigation system is implemented in a java base simulator which is called Green Light District (GLD). Road network is generated manually in this simulator and we can consider restrictions such as turning left or right in junctions and lanes and we can have several types of vehicles with different speeds. There are several traffic lights controlling policies that can be chosen optionally, as well.

## III. Brief Explanation of the Proposed Method

Vehicles are considered as "ants" in this simulator and they enter and leave road network by nodes that are called "edge nodes". We can set their entry frequency of vehicles manually. Each Antpath per node *i* contains outgoing lanes of it with different amount of pheromone. We do this method in two phases:

### A. Initially Learning Phase

First phase is initially learning phase. In this phase we just consider the length of the path, and not path traffic condition. In the other words, in this stage the learned Antpaths are optimal according to the total length of the Antpath. One of the goals of this stage is assigning some pheromone to the lanes of Antpaths with Ant Colony algorithm. The other goal is determining the Antpaths parameters such as the shortest trip time, total length of the Antpaths. One Antpath may have several lanes with different pheromones. This amount of the pheromone of each lane is not unique. But the amount of pheromone of each lane is different in the different ones. This amount of the pheromone determines whether it is an optimal one to achive the destinated node or not.

In this learning stage, with determining the number of sub graphs, we divide road network into selected number of sub graphs. Then Ant colony algorithm is running on all of these sub graphs respectively. So, learning of total graph is done in several short cycles.

We use Fig.1 to explain how learning of new Antpaths from adjacent nodes is done. Each node only learns Antpaths to edge nodes of its own sub graph. For example, in sub graph1, nodes 7, 8, 9 and 10 learns Antpaths to edge node 2 and 3, the edge node 2 learns the Antpath to the edge node 3 and the edge node 3 learns the Antpath to edge node 2. In sub graph 2, node 4, 5 and 6 learns the Antpaths to edge nodes 0

and 1, edge node 0 learns the Antpath to edge node 1 and edge node 1 learns the Antpath to edge node 0. The learning action in whole graph is starting just after the learning of sub graphs is finished. It means that, adjacent nodes of different sub graphs, tell its own knowledge of Antpaths to each other. Then boundary nodes, tell new Antpaths to adjacent nodes that are member of their own sub graphs. At this graph, node 8, 10 are the boundary nodes of sub graph 1. Node 4, 5 are the boundary nodes of sub graph 2. So, node 8 learns Antpaths to edge nodes 0, 1 from node 4, node 4 learns Antpaths to edge nodes 2, 3 from node 8, node 7 learns new Antpaths from node 8, nodes 1, 5 learns new Antpaths from node 4, and this will be done for every node of the graph. (Each of nodes selects the Antpath to destination x, among the Antpaths of adjacent nodes which their destination is edge node x and selects the shortest of them and registers the outgoing lane that connect it to the shortest adjacent node.) In Fig.1 the black arrows show the comparison and learning of Antpaths among boundary nodes and the green arrows show comparison and learning of Antpaths among the adjacent nodes of each of sub graphs. All of the green arrows haven't been shown.

Fig. 2 shows a small road network which is generated in the GLD simulator. In this map, edge nodes or destinations are 0, 1, 2, 3 and the junctions are 4, 5, 6, 7, 8, 9 and 10. For each of nodes, the node themself is the source of the Antpath and the destination is one of the edge nodes. For example $L_2$-$L_6$-$L_8$ is a shortest path from node 5 to node 2.

In this method, $L_2$ will be registered as an Antpath from node 5 to node 2 and $L_6$ will be registered as an Antpath from

This phase (first stage) of learning is done to assigning amount of pheromone trail to the shortest Antpaths (Dijkstra paths) of the graph and not to optimal paths.

In this method, as soon as a vehicle receives at a junction, if there is a learned Antpath to the destination of it, the system selects one of the lanes of that Antpath as a solution (else selects one of the outgoing lanes of the current junction by random) which the equation of selecting optimal lane of that Antpath is:

$$j = \begin{cases} argmax\ \tau(r,u)_u & if\ \ q \le q_0\ and\ u \notin M_k \\ S & if \qquad q > q_0 \end{cases} \tag{6}$$

The (6) shows that vehicle $k$ in node $r$, selects node s with probability of $q$, if it didn't passed it over. In the other words, $j$ is not in the Tabu list of it and is the one of adjacent nodes of $r$ too. If $q > q_0$ , $S$ is selected by random according to $P_k(r,j)$ that its equation is represented as (7).

$$P_k(r,j) = \begin{cases} \frac{\tau(r,j)}{\sum_{u \notin M_k} \tau(r,u)} & if\ j \notin M_k \\ 0 & otherwise \end{cases} \tag{7}$$

As mentioned above, the amount of the pheromone of the Antpath's lanes, are updated based on the total length of it, so there is no parameter as a total length of the path in (7).

If there is any learned Antpath to destination of the vehicle at current node, the system calls a function which selects a random lane among outgoing lanes of it. Then the vehicle enters to this random selected lane.

When a vehicle reaches to its destination, the amount of pheromone of its path's lanes must be updated. Each lane is outgoing lane of one of nodes. Then the amount of pheromone of lanes of the Antpath in each of the nodes which include in this Antpath must be updated.

In each of nodes, If the total length of the path is smaller than registered Antpath of that, the amount of pheromone of this Antpath's lane must be updated by global updating (3), else if the length of the path is equal to registered Antpath of that node, the amount of pheromone of this path's lane must be updated by local updating (5). This will be done for each lane of the.

The value of $\tau_0$ which was used in the (5) is:

$$\tau_0 = \frac{1}{n * l_{sp}} \tag{8}$$

where $n$ is the number of nodes of the map.

### B. Second Phase or Execution Phase

Second phase is execution phase, but we can consider this stage as a learning phase too. Two phases have some different points which we will detail in bellow. At this stage, the shortest Antpaths have been learned, so selecting random lanes is not allowed.

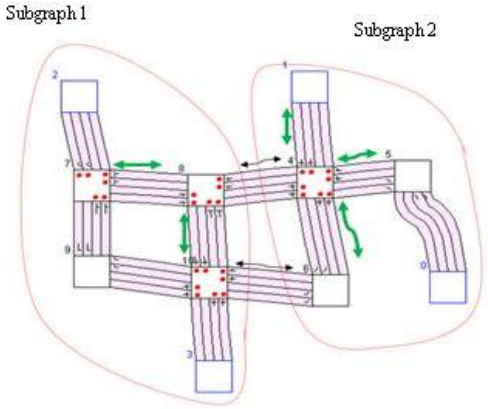


Fig. 1. A road network which is divided in two sub graphs

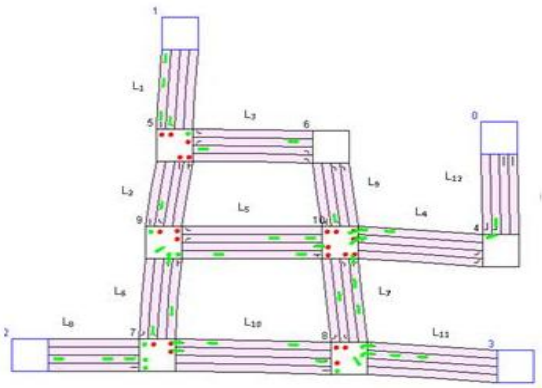

Fig. 2. A road network which is generated in GLD simulator

When a vehicle receives at a junction, the system finds the Antpath which its destination is the same with the destination of it and then selects one of the lanes of that Antpath with the (6),(7).

When a vehicle reaches to its destination, the amount of pheromone of its path's lanes must be updated with (9).

$$\tau'(r,j) = (1-\rho).\tau(r,j) + \rho.\tau_0 \tag{9}$$

We assign the pheromone to lanes of an Antpath with

relative shorter distance and balanced state of traffic on it. It means that at this stage, $\tau_0$ is variable of physical distance and delay of trip.

$$\tau_0 = \begin{cases} \frac{Q}{l_{path}} & \text{if } \frac{trip\ time_{path}}{l_{path}} \le \frac{trip\ time_{sp}}{l_{sp}} \\ \frac{-Q}{l_{path}} & \text{if } \frac{trip\ time_{sp}}{l_{sp}} < \frac{trip\ time_{path}}{l_{path}} \le (Q' * \frac{trip\ time_{sp}}{l_{sp}}) \\ \frac{-Q''Q}{l_{path}} & otherwise \end{cases} \quad (10)$$

where $Q$, $Q'$ and $Q''$ are coefficients, $l_{path}$ is the total length of the path which the vehicle passed it over, $trip\ time_{sp}$ is the time which this vehicle spent from start node to destination, $trip\ time_{sp}$is the shortest time which registered on that Antpath. As shown in (10), we have negative reward or negative$\tau_0$, this is because of the delay of the vehicle in its path.

According to trip time and the length of the path which a vehicle passed it, we classify the path in (10). If the path of the vehicle is a longer one, according to its length or its trip time, we assign amount of negative pheromone to its lanes.

To avoid search stagnation (the situation where all of the vehicles follow the same path, that is they construct the same solution), the allowed range of the pheromone level is limited to below [2].

$$\tau(i,j) = \begin{cases} \tau_{max} & if\ \ \tau(i,j) \ge \tau_{max} \\ \tau_{min} & if\ \ \tau(i,j) \le \tau_{min} \end{cases} \quad (11)$$

## IV. Experiment Conditions

We set initial amount of parameters in the initial learning stage such as bellows:

$\rho = 0.1, = 1$ , $\beta = 0$, $q_0 = 0.7$,$\tau_{min} = 1$ and $\tau_{max} = 5$

And we set them in the execution phase such as bellows and get the best results of our experiments:

$\rho = 0.001$, $Q' = 1.3$, $'' = 3$ , $\alpha = 1$ , $\beta = 0$ , $q_0 = 0.6$, $\tau_{min} = 1$ and $\tau_{max} = 5$

The implemented program is executed in pc with Intel(R) Core(TM) 2 T700 2GH CPU and 256MB of RAM, and the traffic lights controlling policy was relative longest queue. This policy makes the traffic lights green for the lanes which have relatively longer queues. This policy was selected because of improving the results of Dijkstra algorithm to show that despite of selecting best controlling traffic light policy, the results of proposed method is better.

## V. Experiments

We implemented this problem with several algorithms such as Dijkstra, Genetic, Ant Colony and proposed method (combination of Divide and Conquer method and Ant Colony algorithm) with GLD simulator. The results of our executions are shown in Fig. 3. and Fig. 4. The explanation of implementation of this problem with Genetic Algorithm was detailed in [4], [5].

## VI. The Pseudo Code of Implemented Method

*The initial learning phase:*

While Dijkstra paths are not found do

Begin

*Step1.* The Cars are generated at each cycle with determined entry frequencies. A random destination will be selected for each car.

These should be done for each entered car:

*Step2.* The car which reaches one of the junctions, if there is a found Antpath to destination of the car on that junction, then it will be selected with (6), (7). Otherwise, one of the outgoing lanes of this junction will be selected randomly.

*Step3.* When a vehicle reaches to one of the edge nodes, if this edge node is the destination of vehicle, then the amount of pheromone of its path will be updated according to (3),(4) or (5), (8) (depend on local or global path conditions) and with consideration of (11). (The value of $\tau_0$ which was used in (5) depends on length of the path of the vehicle.)

End while

*Execution Phase:*

*Step4.* As soon as *t*he car which reaches to the one of the junctions, the Antpath whose distination is the same with the distination of the vehicle will be selected. Then one of the lanes of the selected Antpath (one of outgoing lanes of the junction) will be selected according to (6), (7).

*Step5.* When a vehicle reaches to the edge node, the amount of pheromone of the path of the vehicle will be updated by (9), (10) by taking (11) into consideration. (The value of $\tau_0$ in (9) depends on the length of the path of the vehicle and its trip time.)

The execution phase will be finished if we close the simulator.

## VII. Experiments

We implemented this problem with several algorithms such as Dijkstra, Genetic, Ant Colony and proposed method (combination of Divide and Conquer method and Ant Colony algorithm) with GLD simulator. The results of our executions are shown in Fig. 3 and Fig. 4. The explanation of implementation of this problem with Genetic Algorithm was detailed in [4], [5].

Fig. 3 shows the results of our experiments on a road network with 120 nodes with entry frequency of the edge nodes equal with 0.07 and 0.08. Entry frequency equal with 0.08 means that at each cycle of execution of the program, 0.08 numbers of vehicles enter at each edge node.

At Fig. 3, the horizontal axis shows the time steps of 50 cycles and vertical axis shows trip time average of vehicles.

In Fig. 4, we show results of execution of the algorithms on the two graphs with 250 and 300 nodes for 2000 cycles of time and entry frequency of 0.04.

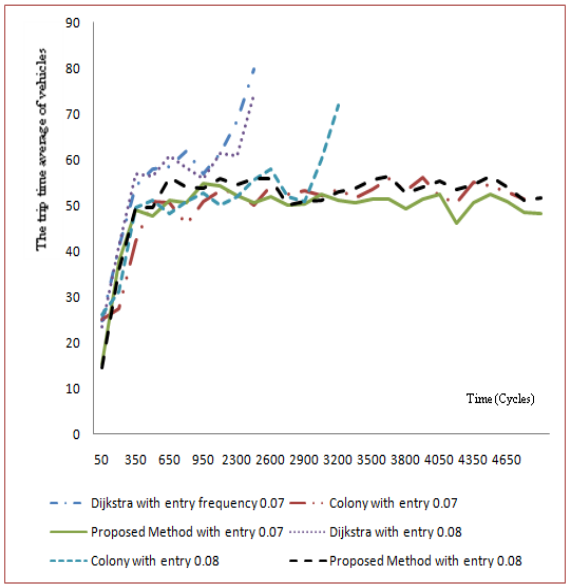


Fig. 3. Comparison of trip time average of vehicles in the road network with 120 nodes and entry frequency of 0.08

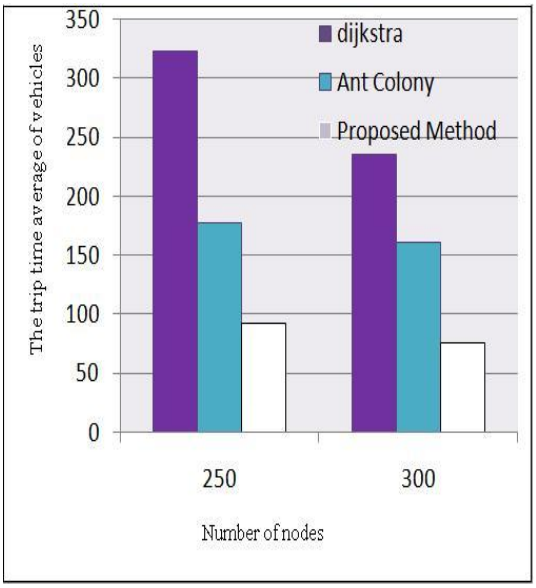


Fig. 4. The results of execution of the algorithms on two graphs with 250 and 300 nodes for 2000 cycles of time and entry frequency of 0.04

## REFERENCES

[1] M. Hashemzadeh and M. Mohammadreza, "A fast and efficient route finding method for car navigation systems with neural networks," *IEEE*, 2007.

[2] Y. Jiang, W. Wang, and Y. Zhao, "Solving the shortest path problemin vehicle navigation system by ant colony algorithm," *in Proc. 7th WSEAS Int. Conf. on Signal Processing, Computational Geometry and Artificial Vision,* Athens, Greece, August 24-26, 2007.

[3] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the travelling salesman problem," *IEEE Transactions on Evolutionary Computation*153–66, 1997.

[4] C. Ahn and R. S. Ramakrishna. "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Trans. Evol. Comput.*, 6(6), 567-569, 2002.

[5] S. Abeysundara, B. Giritharan, and S. Kodithuwakku, "A Genetic algorithm approach to solve the shortest path problem for road maps," *in Proc. International Conference on Information and Automation,* Colombo, Sri Lanka, December 15-18, 2005.

[6] I. Chabini and S .Lan, "Adaptation of the A* algorithm for the computation of fastest paths in deterministic discretetime dynamic networks," *IEEE Trans. Intelligent Transportation Systems,* vol. 3, pp. 60-74, Mar. 2002.

[7] E. P. F. Chan and N. Zhang, "Finding shortest paths in large network systems," in *Proc. of the 9th Int. Conference on Advances in Geographic Information Systems,* ACM Press, New York, NY, USA, 2001.

[8] I. Flinsenberg, "Graph partitioning for route planning in car navigation systems," in *Proc. of the 11th IAIN World Congress, Smart Navigation Systems and Services,* Berlin, Germany, Oct. 2003.