Solving the Shortest Path Problem in Vehicle Navigation System by Ant Colony Algorithm

Yong JIANG Wan-liang WANG Yan-wei ZHAO
(Collego of Information Engineering, Zhejiang University of Techonology, Hangzhou 310014, China)

Abstract: A shortest path search method based on ant colony algorithm is proposed. The method contracts the search space appropriately, obtaining, in a short time, a path that is as close as possible to the path obtained by the Dijkstra method (the optimum path). To improve the performance, we modify the pheromone update rule and introduce a learning strategy into the ant colony algorithm. The method obtains a solution to a problem given within a specified time, such as path search in a vehicle navigation system. The effectiveness of the method is described through use of simulations.

Key words: Vehicle navigation; Shortest path problem; Ant colony algorithm

1 Introduction

The path search problem involves finding the optimum path between the present location and the destination under given conditions. Currently, these problems arise in networks such as the highway system, railroads, and communication networks, and cover a wide range of applications. In particular, vehicle navigation systems have exhibited explosive growth in popularity due to recent advances in science and technology. The actual route found in vehicle navigation is not necessarily the optimum solution because of hardware restrictions and the calculation time required for the path search if the network changes dynamically and has broad search scope. A number of path search methods have been proposed for use in vehicle navigation systems, but while all of those methods have their respective merits and demerits, not exceptionally good method has yet been established.

The Dijkstra method[1], and the A^* algorithm are the main types of algorithms that are being used or studied for use in path search problems. The Dijkstra method is an algorithm for finding the optimum path. Because this algorithm searches for the minimum-cost path among all paths in order, beginning from the starting point, the search region expands concentrically. This method thus has the disadvantages of poor search efficiency and a long search time when the distance to the destination is large. The A^* algorithm is based on the approach of the Dijkstra method, but eliminated fruitless searches by considering the distance to the destination. Besides, genetic algorithms [2-4] are also used to solve the path search problem. These algorithms imitate a number of processes that are seen in natural evolution. In the case of path search, a path is selected and regarded as a gene, and the solution path is obtained by performing calculations that emulate genetic crossing, spontaneous mutations, and so on. HUI F. and ZHEN H. [5] proposed a basic ant colony algorithm for shortest path problem. The algorithm can only solve the problem effectively under the condition that the route network has a small number of nodes.

In this paper, we propose a method of path search that is based on the ant algorithm, which aims to obtain a path that is as close as possible to the optimum solution in a shorter time even with thousands of nodes in the route network. This paper organized as follows. In section 2, we introduce the basic ant algorithm. In section 3, the algorithm based on ant algorithm for solving the shortest path problem in vehicle navigation is proposed. Section 4 presents preliminary simulation results. Section 5 concludes the work.

2 Ant colony algorithms

Ant colony algorithms are inspired on an analogy with real life behavior of a colony of ants when looking for food. In their search they mark the trails they are using by laying a substance called pheromone. The amount of pheromone in a path lets other ants to know if it is a promising path or not. This observation inspired Colorni, Dorigo and Maniezo[6] for proposing

a metaheuistics technique: ants are procedures that build solutions to an optimization problem. According to how the solution space is explored some valued are recorded in a similar way as pheromone acts, and objective values of solutions are associated with food sources. An important issue of these algorithms is parallelism: several solutions are built at the same time and they interchange information during the procedure and use information of previous iterations.

At each iteration of the basic Ant Colony method, each ant builds a solution of the problem step by step. At each step the ant makes a move in order to complete the actual partial solution choosing between elements of a set A_k of expansion states, following a probability function. Probability $P_k(i,j)$ of moving from present state i to another state j is calculated for each ant k taking into account:

(1) Attractiveness η of the move according to the information of the problem.

(2) Level τ of pheromone of the move that indicates how good the move was in the past.

(3) A $Tabu_k$ list of forbidden moves.

In the ant algorithm original version formula for $P_k(i,j)$ is:

$$P_k(i,j)=\begin{cases}0 & if\ (i,j)\in Tabu_k\\ \dfrac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum_{(i,z)\notin Tabu_k}[\tau(i,z)]^{\alpha}[\eta(i,z)]^{\beta}} & otherwise\end{cases}$$

Where a,β are parameters that are used to establish the relative influence of η versus τ. After iteration t is complete, that is when all the ants have completed their solutions. The pheromone levels are updated to:

$$\tau(i,j)=\varphi\cdot\tau(i,j)+\Delta\tau(i,j)$$

Where φ is a coefficient representing the level of pheromone persistence and $\Delta\tau(i,j)$ represents the contributions of all ants that chose move (i,j) in their solution.

Successful ant algorithms have been developed for several combinatorial optimization problems.

3 An ant colony algorithm for shortest path problem in vehicle navigation

We resume here the main characteristics of our ACO algorithms for shortest path problem.

(1) Solution construction: at each iteration of ACO each ant builds a solution for the shortest path problem, moving to next node in the traffic network according to transition rules based on a combination of the amount of pheromone at each arc and the current node information. The role of Tabu list mentioned at previous section is taken here by a set of already visited neighbors, forbidden for the ant at current iteration.

(2) Transition rule (Pseudo-Random-Proportional rule): this rule for choosing next node to visit combines random selection with best option. Let q_0 such that $0\le q_0\le 1$, we generate q a random number in $[0,1]$, then next node j is chosen according to:

$$j=\begin{cases}\max_{u\in\Gamma(i)}[\tau(i,u)]^{\alpha}[\eta(i,u)]^{\beta} & if\ q\le q_0\\ J & \text{if } q>q_0\end{cases} \quad (1)$$

Where J is randomly selected according to $P_k(i,j)$. We tried following three alternatives for $\eta(i,j)$:

① $\eta_1(i,j)=\dfrac{1}{d_{ij}}$, where d_{ij} is the length of arc (i,j)

② $\eta_2(i,j)=\dfrac{1}{d_{ij}+dis_{jd}}$, where dis_{jd} is the Euclidean distance between the candidate node j and the destination node d.

③ $\eta_3(i,j)=\dfrac{\cos(Ang(\vec{ij},\vec{id}))+1}{2d_{ij}}$, where $Ang(\vec{ij},\vec{id})$ is the angle between the vector $\vec{ij}$ and $\vec{id}$.

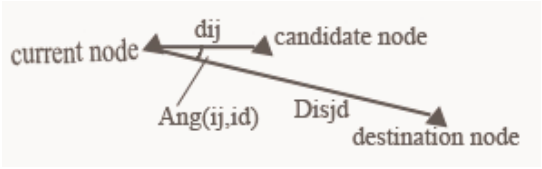

Fig.1. Graph description of $\eta(i,j)$

(3) Pheromone update

When all ants complete their tours, the objective function (the length of route) is calculated for each run. The pheromone levels are updated to:

$$\tau(i,j)=(1-\rho)\cdot\tau(i,j)+\rho\cdot\Delta\tau_{ij}^{rb},\ \rho\in(0,1) \quad (2)$$

$$\Delta\tau_{ij}^{rb}=\begin{cases}Q/L^{gb}, & if\ (i,j)\ \text{in the best route}\\ 0, & otherwise\end{cases} \quad (3)$$

Where ρ, Q are coefficients and L^{gb} is length of the best ant tour.

Because of the restrictions of the traffic network (turning penalty), an ant may be trapped during the solution construction (no available candidate node can be selected). In such a situation, the ant can't construct a feasible solution and will be regarded as a "dead" ant, and the relative route will be regarded as a "death"

route. To avoid too much ants being trapped, the pheromone levels in the "death" route are updated as follows:

$$\tau(i,j)=\begin{cases} Drho\cdot\tau(i,j), & if\ (i,j)\ \text{in the "death" route} \\ \tau(i,j), & otherwise \end{cases} \quad (4)$$

Where $Drho=\exp(-\rho_d * step)$, $\rho_d>0$ is a coefficient, and $step=1,2,3,4,5...$ representing the sequence order of arc (i,j) in the "death" route, which expresses the idea that the closer the arc from the "death" node (the node where the ant is trapped), the more pheromone in the arc should be vaporized.

To avoid search stagnation (the situation where all the ants follow the same path, that is, they construct the same solution), the allowed range of the pheromone level is limited to

$$\tau(i,j)=\begin{cases} \tau_{\min} & if\ \tau(i,j)\le\tau_{\min} \\ \tau_{\max} & if\ \tau(i,j)\ge\tau_{\max} \end{cases} \quad (5)$$

(4) Learning strategy between ants: we introduce the component of the dynamic programming algorithm into our algorithm, and here we call this component the learning strategy (LS). When an ant reaches a node, it exchanges its route information with the node. The one with worse route solution gets the route information from the other. Specifically, if the route length of the ant is shorter than the node's, the node gets the route information from the ant. Otherwise, the ant gets the route information from the node. Such a strategy can be regarded as the learning strategy between the ants by the node.

(5) Stopping rules: ACO procedure stops when there is not improvement on the solution after several iterations or when $n\max$ number of iterations is reached.

4 Computational results

Two kind of traffic network are used to test the effectiveness of the algorithm. One is the Hangzhou city traffic network, and the other is the simulated traffic network Px by creating a node grid pattern of paths with different node number. Px represents the network has x nodes, e.g. $P100$ is the traffic network with 100(10×10) nodes. The length of every arc in the simulated traffic network is a random number between 1 and 10. All experiments are performed on personal computer with CPU Celeron 2.6GHz and main memory 256MB. The basic parameters of the ACO are set as: 30 ants, $\alpha=1,\beta=2$, $\tau_{\max}=3.5$, $\tau_{\min}=0.5,\rho=0.05,q_0=0.8,\ \rho_d=0.001.$

First we test the performance of the algorithms with different $\eta(i,j)$ by applying them into the vector road network of Hangzhou to do searching comparison about the shortest path when given a specific origin node and destination node. Fig.2 shows the comparison of solution evolution. We find that the algorithm with $\eta_3(i,j)$ performs better than the others. From Fig.3, as the nodes number increases in the traffic network, the number of dead ants during the iteration also increases using $\eta_1(i,j)$ or $\eta_2(i,j)$, which makes the algorithm get less feasible solutions during the iteration, and thus affects the effectiveness of the algorithm.

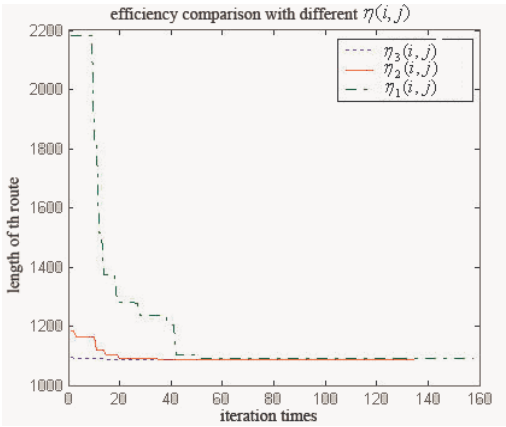

Fig.2. Comparison of performance of the algorithm for searching in the HZ traffic network with different $\eta(i,j)$

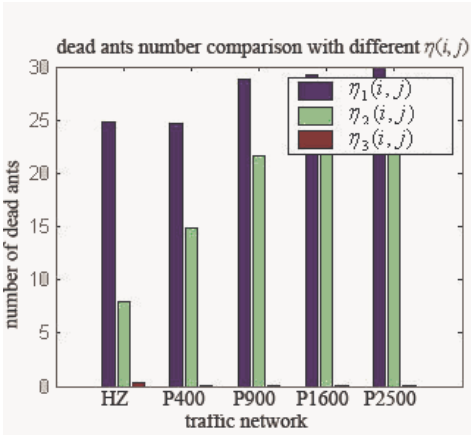

Fig. 3. Comparison of the No. of dead ants in one iteration with different $\eta(i,j)$

To avoid the ants being trapped in the network and improve the performance of the algorithm, we modify the pheromone update rule and introduce the learning strategy into the algorithm described in section 2. Fig.3 shows the comparison result between the algorithms with the "death route" pheromone update rule (DRPU) and without by applying them to do searching shortest path in Hangzhou traffic network with restrictions such as turning penalty. We find the algorithm with DRPU can restrain the "death" of ants effectively. Fig.4 shows the comparison result between

the algorithms with the learning strategy and without.

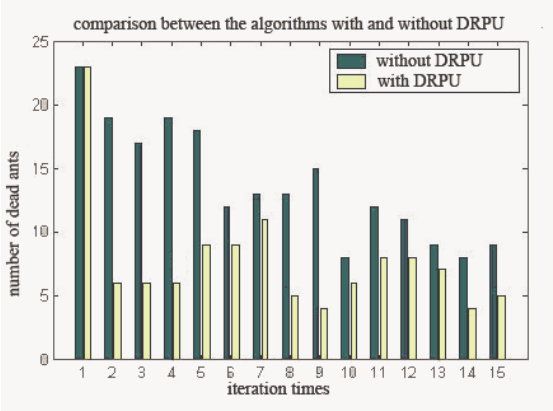

Fig.4. Effect of "death" route pheromone update rule on the restraining the "death" of ants

The data in Table 1 is presented in a form that facilitates comparison of the conventional Dijkstra method and the ant colony algorithm we proposed by applying them to different kind of traffic network to search the path between the specific origin node and the destination node. First, we consider the search time (the "Time" is the table). With the proposed ant colony algorithm, the search space is contracted, so the search time was shortened especially for the traffic network with thousands of nodes. Next, we consider the length of the path. The conventional Dijkstra method necessarily gives both the optimum solution and the minimum length. With the proposed ant colony algorithm, in all of the simulations, a near-optimum solution is obtained.

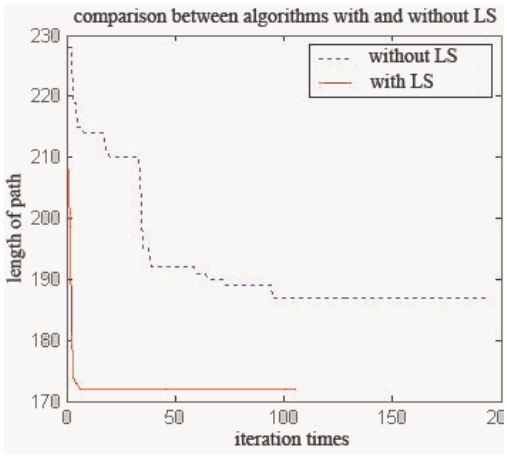

Fig.5. Effect of learning strategy on performance of the algorithm for searching shortest path between a given origin and destination node in P6400

Table 1. Comparison with the Dijkstra's

Problems		ACA proposed				Dijkstra's	
Dimension	Nodes	Best	Average	Worst	Time(s)	Best	Time(s)
P400	0-399	118	118.1	119	0.117	118	0.056
	120-180	21	21	21	0.014	21	0.016
P900	0-899	183	185.6	187	0.225	182	0.453
	100-300	63	63.6	65	0.082	63	0.297
P2500	0-2499	290	290.6	293	0.452	290	11.984
	520-785	93	93	93	0.097	93	2.735
P6400	0-6399	478	478	478	0.931	464	216.360
	2535-3598	131	131.8	132	0.197	131	69.560
P10000	0-9999	596	602.8	610	2.062	591	811.063
	4254-6785	165	165	165	0.393	165	408.297

5 Conclusions

We have proposed a shortest path search method for vehicle navigation based on ant colony algorithm and confirmed its effectiveness by means of simulation. The results show that the search time can be greatly reduced without such a large effect on the path length (movement cost). The algorithm can thus be applied to car navigation systems and other such path search problems that require a real-time solution. As can be seen from Table 1, a 100×100 search takes only about 2 seconds, so the practical use isn't limited to problems that have a broad search scope as other methods such as the Dijkstra's. As an issue to be addressed in future work, it is necessary to conduct simulations in which the results of this work are applied to actual use for vehicle routing problem.

Acknowledgments

This project is supported by the National Natural Science Foundation of China (NSFC No.60374056, 60573123).

References

[1] J. A. Bondy and U. S. R. Murty. Graph Theory

with Applications, Elsevier North-Holland, 1976.
[2] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, AddisonWesley, 1989.
[3] Y. Leung, G. Li, and Z. Gang. A Genetic algorithm for multiple destination routing problems, IEEE Trans. Evol. Comput., vol. 2, pp. 150-161, Nov. 1998.
[4] Chang Wook 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] HUI F., ZHEN H., etc. Solving a Shortest Path Problem by Ant Algorithm, in Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 3174-3177, 2004.
[6] A. Colorni, M. Dorigo, V. Maniezzo. Distributed Optimization by Ant Colonies, First European Conference on Artificial Life, 134-142, 1991.