한국 퍼지 및 지능시스템 학회 2002, Vol. 12, No. 4, pp. 361-365

Decision Support Method in Dynamic Car Navigation Systems by Q-Learning

홍수정* · 홍언주 · 오경환
Soo-Jung Hong, Eon-Joo Hong, and Kyung-Whan Oh

*한국전자통신연구원, 서강대학교
*Electronics and Telecommunications Research Institute,
Dept.of computer science and Engineering, Sogang University

Abstract

오랜 세월동안 위대한 이동수단을 만들어내고자 하는 인간의 꿈은 오늘날 눈부신 각종 운송기구를 만들어 내는 결실을 얻고 있다. 자동차 네비게이션 시스템도 그러한 결실중의 한 예라고 할 수 있을 것이다. 지능적으로 판단하고 정보를 처리할 수 있는 자동차 네비게이션 시스템을 부착함으로써 한 단계 발전한 운송수단으로 진화할 수 있을 것이다.
이러한 자동차 네비게이션 시스템의 단점이라면 한정된 리소스만으로 여러 가지 작업을 수행해야만 하는 어려움이다. 그래서 네비게이션 시스템의 주요 작업중의 하나인 경로를 추출하는 경로추출(Route Planning) 작업은 한정된 리소스에서도 최적의 경로를 찾을 수 있는 지능적인 방법이어야만 한다.
이러한 경로를 추출하는 작업을 하는 데 기존에 일반적으로 쓰였던 두 가지 방법에는 Dijkstra's algorithm과 A* algorithm이 있다. 이 두 방법은 최적의 경로를 찾아낸다는 점은 있지만 경로를 찾기 위해서 알고리즘의 특성상 각각, 넓은 영역에 대하여 탐색작업을 해야 하고 또한 수행시간이 많이 걸린다는 단점과 또한 경로를 계산하기 위해서 Heuristic function을 추가적인 정보로 계산을 해야 한다는 단점이 있다.
본 논문에서는 적은 탐색 영역을 가지면서 또한 최적의 경로를 추출하는데 드는 수행시간은 작으며 나아가 동적인 교통환경에서도 최적의 경로를 추출할 수 있는 최적 경로 추출방법을 강화학습의 일종인 Q- Learning을 이용하여 구현해 보고자 한다.

Keywords : Reinforcement Learning, Search algorithm, Intelligent navigation system

1. Introduction

Human being has been dreaming about creating a great mobility. One of the results is vehicle navigation system. With the assistance of the vehicle navigation system driver can do many intelligent works that once those never be concerned as a possible task.

But, there are still several obstacles to overcome. In particular, the route planning task that is one of goals of navigation systems need to have intelligent method to determine the optimal route.

There are two well-known methods to find the optimal route. One is Dijkstras algorithm and the other is A* algorithm. Those two methods find the optimal route but both of them have disadvantages, respectively.

According to the Dijkstras algorithm, it has to visit all the nodes in a given map, therefore it needs long execution time. In case of A*-algorithm, it has to maintain the extra knowledge to calculate the heuristic cost. These facts can be the critical matters to be applied in a navigation system, which faces not only static environment but also dynamic environment.

This paper proposes the new method to find the optimal route in navigation system. Proposed method is implemented based on the idea of Q-Learning.

2. Background

2.1 Autonomous car navigation systems

Human beings have never stopped pursuing the dream of great mobility. Thousands of years of civilization have led to the modern vehicle location and navigation systems of today. There are, and always will be, a range of systems from the very low and to the very high end. Figure 1 illustrates the building modules of a navigation system. Main modules are route planning, route guidance, digital map database, positioning and map matching.

In particular, Route planning is the process of helping vehicle drivers to plan a route prior to or during their journey, based on a given map provided by the map database module. A commonly used technique is to find a minimum-travel-cost route based on criteria such as time, distance, and complexity.

접수일자 : 2001년 10월 26일
완료일자 : 2002년 5월 2일

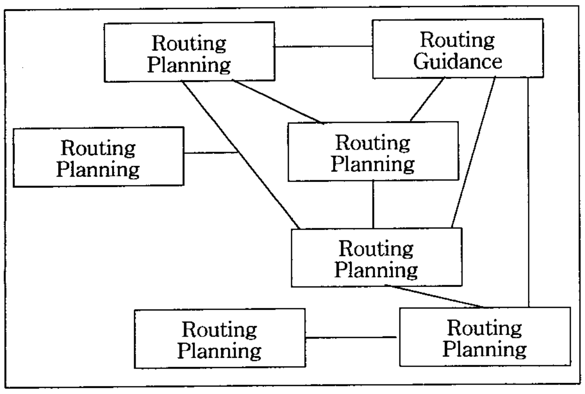

Fig. 1. Basic module for a navigation system

2.2 Optimal route planning in navigation system

Route planning is an essential part of the navigation system. In the computer literature, people often refer to finding a route from point A to point B as a 'shortest path' problem. Many algorithms have been developed to solve single-origin shortest path and all pairs shortest path problem. These algorithms treat situations that are quite analogous to single-vehicle route planning and also multi-vehicle route planning. A variety of route optimization criteria (planning criteria) may be used in route planning. We refer to all these factors as the 'travel cost'. Some drivers may prefer the shortest distance. Others may prefer the shortest travel time. Thus the evaluation function chosen to minimize this cost depends on the system design and user preference.

2.3 The Dijkstras algorithm & The A*-algorithm

The Dijkstra's algorithm is one of the main representatives of shortest path algorithms [5-6]. This algorithm is a general method to solve the single-source shortest path problem. The Dijkstras algorithm is a prime example of a greedy technique, which generally solves a problem in stages by doing what appears to be the best thing at each stage.

The A*-algorithm is one of the most popular heuristic search methods [3]. In general, there are different search strategies to realize a heuristic search method. A heuristic is a technique for improving the efficiency of a search process by possibly sacrificing claims of completeness. The merit of using heuristic information is that this information helps to determine which is the "most promising" node, which successors to generate, and also it tells which irrelevant search branches to prune [2]. In the A*-algorithm this heuristic evaluation function enables the algorithm to search the most promising nodes first. In other words: to find the most promising node, the evaluation function is used [1].

2.4 Q-Learning

Reinforcement learning is learning from positive and negative rewards [2].

Reinforcement learning algorithms find a behavior for the system that maximizes the total reward for every possible start state [10]. Such a behavior is usually specified as a stationary, deterministic policy.

One of the most important breakthroughs in reinforcement learning was the development of Q-learning [4]. As can be seen from Figure 2, Q-learning consists of a termination checking step (line 2) and action selection step (line 3), an action execution step (line 4), and a value update step (line 5). For the moment, the initial Q-values stay unspecified.

The action selection step implements the exploration rule ("which state to go to next"). It is constrained to look only at information local to the current state s of the system.

```
For each state and action,initialize the Q-value to zero.
Observe the current state.

1. Set s : = the current state.
2. If $s \in \overline{G}$ , then stop. ( $\overline{G}$ is goal state).
3. Select an action $a \in \overline{A}(s)$.
4. Execute action a./* As a consequence, the agent
   receives reward $\overline{\gamma}(s,a)$ and is in state $\overline{\text{succ}}(s,a)$.
   Increment the number of steps taken, i.e. set t : =
   t+1*/
5. Set Q(s,a) := $\overline{\gamma}(s,a) + \gamma U(\overline{\text{succ}}(s,a))$.
6. Go to 1.
where total reward U(s) : = max $a \in \overline{A}(s) Q(s,a)$ at
every point in time.
```

Fig. 2 The Q-learning algorithm

3. Decision Support method in car navigation systems by Q- learning

This part describes how a method for decision support to optimize route planning in car navigation system is constructed. The proposed method for route planning is implemented based on Q-learning. The reason for using the Q-learning algorithm for route planning is that this method can improve the performance on-line, while the system and the environment interact. A car navigation system faces a real-time traffic environment, thus Q-learning provide proper way to adapt to a dynamically changing environment. Figure 3 is the feagure of digital map for the experiment.

The construction of a decision support method for route planning is essentially based on Q-learning. In particular, route planning in the static environment completely depends on the Q-values that are generated by Q-learning. In the dynamic environment, route planning does search the path based on the Q-values, too. However, in the dynamic case the total costs are determined by adding a penalty delay value to the

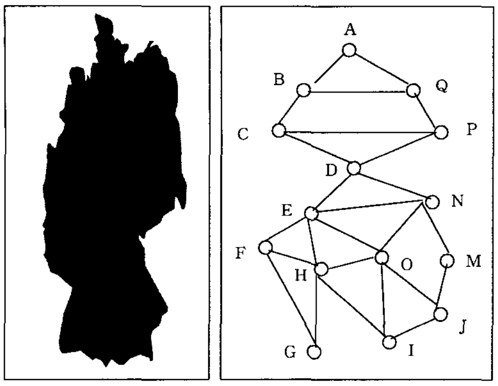

Fig. 3 Partial map of digital map database and corresponding graph

corresponding Q-value.

The following parts are the description of how Q-learning works in navigation systems. Q-learning estimates the values of each state and its action pairs. In the case of navigation system, this state is the vertex the system considers at the moment. When the system's current state is 'i', then the system considers the vertex 'i'. The state is not in the time domain, thus there is no time related order between states. An action of the system is movement to the next vertex.

Every action has an immediate reward $\bar{r}(s,a)\in\Re$ that is obtained when the system executes an action. The reward function is described as goal-reward function.

In this paper another reward function $\bar{r}(s, a)$ is proposed. When the system makes an action, such as move from vertex 'A' to vertex 'B', then the reward corresponding with the distance value or time value between two vertices is received.

The distance means the distance between two vertices and time is time to reach one vertex from another one. These two values (distance, time) are also evaluation criteria for route quality.

When the system reaches the goal state, then the total reward $U(S)$ is the accumulated sum of all distance values of a particular path. The discounting factor is 1 in this case, therefore the total reward $U(S)$ is

$$\sum_{s=0}^{n}\bar{r}_s .$$

Setting the discounting factor to 1 means that there is no process to decrease the sum of distance value or time value until the state reaches the goal.

After the Q-learning has determined, each vertex on the graph knows the best Q-value to reach the destination vertex.

As described before, one part of the Q-learning algorithm has to be adjusted, to be able to apply Q-learning for the use in navigation systems.

The original idea of Q-value (Quality value) is that once these values have been learned, the optimal action from any state is the one with the highest Q-value. But when it is applied to a navigation system, the optimal action is the one with the minimum Q-value, since costs in contrast to rewards are considered.

Mostly the navigation system wants to find the route with the minimum travel cost. Thus when we use reward with distance cost or travel time cost, then the total reward value should have minimum value. It means that the function $U(s)$ should be represented by one of the followings.

$$U(s) = \min_{a\in\bar{A}(s)} Q(s,a)$$

$$U(s) = \max_{a\in\bar{A}(s)} \frac{1}{Q(s,a)}$$

Consequently both functions do the same work. In this paper, the first total reward function is applied. Once Q-learning obtains the Q-value for each route, the Q-value is saved in a 'Q-value table'. Each vertex has its own Q-value table. When the system plans the route these tables are looked up.

When the system tries to find the optimal route from a starting vertex to a destination vertex, it looks up the Q-value table and compares all the Q-values of every path and then chooses the best Q-value. When the Q-value table is based on distance values, then the best Q-value is the minimum distance value to the destination. In the other case, the Q-value table contains values in terms of time, hence, the best Q-value gives the minimum time to reach the destination.

In a dynamic environment the navigation system notices several changes of traffic situation. One of theses influences on the traffic situation is a traffic jam. In this thesis, the difference between the static environment and the dynamic environment is indicated by the fact that the traffic situation might include a sudden traffic jam or might not. A static environment continues to keep the same environment until it finds the optimal route.

The Q-value table is determined by off-line learning. Therefore, when the system does route planning in a static environment, these Q-value can be considered to be 100% reliable. But in a dynamic environment these Q-values cannot be completely reliable anymore. To compensate, an update process is needed. This update process will be done on-line, i.e. when the navigation system receives information about a traffic jam through, e.g. a wireless communication module, a penalty value will be used in combination with the Q-values for finding the optimal route. This means that when the system considers to proceed by moving through a congested street segment during the route planning, then a penalty value is added. If the system doesnt take the congested segment, then no penalty value will be added. Following function describes the summarized explanation.

$$\bar{r}'(s,a) = \begin{cases} \bar{r}(s,a) & \textit{, when action } a \neq \textit{selection of traffic jam segment} \\ \bar{r}(s,a) + \textit{penalty value} & \textit{, when action } a = \textit{selection of traffic jam segment} \end{cases}$$

When a segment a has a penalty value, then the cost along segment increase. The new costs are given by the sum of the original Q-value and the penalty value. The penalty value can be described as delay time or delay cost.

4. Experimental results

when a navigation system is trying to find the route requiring the minimum travel time, then time is the criterion being applied. And when the system wants to find the shortest path, then distance is the criterion being applied. These two criteria are commonly used for finding an optimal route.

In a static environment, each of the all three routing methods Dijkstra, A*- algorithm, Q-values finds the optimal route regarding travel distance or travel time.

Table 1. Average execution times in a static environment

pair	Dijkstra's	A*	Q-learning
A - B	0.990	0.060	< 0.001
A - C	0.770	0.050	< 0.001
A - P	0.610	0.060	< 0.001
A - D	0.660	0.050	< 0.001
A - E	0.550	0.050	< 0.001
A - O	1.100	0.060	< 0.001
A - F	0.710	0.050	< 0.001
A - H	0.660	0.060	< 0.001
A - M	0.770	0.050	< 0.001
A - J	0.710	0.050	< 0.001
A - I	1.200	0.060	< 0.001
A - G	1.100	0.060	< 0.001

Table 2. Average execution time of each method in static environment

	Average execution Time	Normalized Time	Improvement Row by row
Dijkstra	0.8192	100 %	0 %
A*	0.0550	6.71 %	93.29 %
Q-learning	<0.001	0.23 %	98.2 %

The accuracy of Dijkstra's algorithm, A*-algorithm and Q-learning method in dynamic environment is the same as before. Table 3 shows the execution time to get the optimal route in dynamic environment. The execution time in Table 3 is obtained from the experimental result when the starting vertex is 'A' on the map.

Table 3. Average execution time in the dynamic environment

Dijkstra's		A*	Q-learning
A - B	0.940	0.220	0.050
A - C	1.100	0.220	0.100
A - P	0.820	0.170	0.110
A - D	0.990	0.220	0.060
A - E	0.710	0.220	0.110
A - O	1.100	0.170	0.060
A - F	1.150	0.160	0.110
A - H	0.770	0.220	0.060
A - M	0.880	0.280	0.110
A - J	0.830	0.160	0.060
A - I	1.150	0.160	0.110
A - G	1.250	0.330	0.120

Table 4. Average execution time of each method in the dynamic environment

	Average execution Time	Normalized Time	Improvement Row by row
Dijkstra	0.9742	100 %	0 %
A*	0.2108	21.64 %	78.36 %
Q-learning	0.0750	9.06 %	64.42 %

5. Conclusion and Prospects

One of the objects in car navigation systems is to provide the optimal route information for the user. This task for route planning is done in the car navigation system's module called 'Route Planning' (Figure 1).

This paper proposes a new decision support method for route planning in car navigation systems. The proposed method is implemented by using the Q-learning algorithm.

The proposed method shows an outstanding performance in the experiments in static and in dynamic environments In the future the method proposed in this thesis will be applied to further experiments on larger area of the digital map database.

Moreover, based on this method using Q-value, a personalized routing algorithm will be constructed. The personalized routing is finding the route with personal preference.

References

[1] Yilin Zhao, Vehicle Location and Navigation Systems, Boston: ARTECH HOUSE, Inc., 1997) pp. 1~6, p. 105~112.

[2] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning, Cambridge Massachusetts : The MIT Press, 1998, pp. 3~6, pp.148~150.

[3] A.V. Aho, J.E.Hopcroft and J.D. Ullman, Data Structures and Algorithms, MA : Addition-Wesley, 1983.

[4] Sven Koenig and Reid G. Simmons, "Complexity

Analysis of Real-Time Reinforcement Learning Applied to Finding Shortest Paths in Deterministic domains", Carnegie Mellon Univeristy, December 1992.
[5] R. Agrawal, S.Dar and H.V. Jagadish, "Direct Transitive Closure Algorithms: Design and Performance," ACM Trans. Database Systems, vol.15, no.3, pp 427-458, Sept.1990.
[6] T. Cormen, C. Leiserson, and R.L. Rivest, Introduction to Algorithms, MIT Press, 1993.
[7] Ning Jing, Yun-Wu Huang, Elke A. Rundensteiner, "Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation" IEEE transactions on knowledge and data engineering, vol.10, 1998
[8] Hinrich Claussen, "Status and Directions of Digital Map Databases in Europe" IEEE-IEE Vehicle Navigation and Information Systems Conference, Ottawa, 1993
[9] S. Shekhar, A. Kohl and M. Coyle, "Path Computation Algorithms for Advanced Traveler Information Systems," Proc. Ninth Int'l Conf. Data Eng., pp.31-39, 1993.
[10] Y. W. Huang, N. Jing, and E.A. Rundensteiner, "A Semi-Materialized View Approach for Route Maintenance in Intelligent Vehicle Highway Systems," Proc. Second ACM Workshop Geographic Information Systems, pp 144-151, Nov.1994
[11] Elaine Rich, Kevin Knight, Artificial Intelligence, Second Edition, New York, McGraw-Hill, Inc., 1991, pp. 76.
[12] Mark Allen Weiss, Data Structures and Algorithm Analysis in Java, Massschusetts, Addison-Welsley Longman, Inc., 1999, pp.304~309.

저 자 소 개

홍수정(Soo Jung Hong)
1999년 2월 : 서강대학교 공과대학 컴퓨터학과 (학사)
2001년 8월 : 서강대학교 대학원 컴퓨터학과 (공학석사)
2002년 3월~현재 한국전자통신연구원 연구원
E-mail : hsj63501@etri.re.kr

홍언주(Eon-Joo Hong)
2000년 : 서강대학교 대학원 컴퓨터학과(공학석사)
2001년~현재 : 서강대학교 대학원 컴퓨터학과 박사과정
1993년~2001년 : 한국전산원 주임연구원
2001년3월~현재 : 한국관광대학 디지털콘텐츠과 전임강사
관심분야 : 에이전트, 디지털콘텐츠 관리

오경환(Kyung-Whan Oh)
1974년~1978년 : 서강대학교 수학 학사
1989년 3월~1993년 2월 : 서강대학교 전자계산학과 조교수
1993년 3월~1997년 : 서강대학교 전자계산학과 부교수
1998년~현재 : 서강대학교 전자계산학과 교수
2002년 3월~현재 : 서강대학교 기획처장