A Path Planning Algorithm Based on Typical Case Reasoning

WENG Min  WEI Xiuqin  QU Rong  CAI Zhongliang

Abstract  Case-based reasoning is an AI technique in which the previous solutions are stored for future use. People are used to guiding themselves according to familiar routes that are stored in their memories and have been used by them before. It is just based on people’s preference to familiar routes, which are gained through the study of the cognitive activities. We propose to apply the intelligent method based on the case reasoning to path planning. It is impossible for a case base to store all the solutions to all the shortest paths; therefore, part of them should be stored. However, which routes should be stored and which should not be? How do we adapt the cases that have already been stored and how do we acquire the shortest route based on them? All these issues need to be explained by integrating knowledge of the network on account of case-based reasoning techniques. This paper suggests the case-based reasoning in another point. This means finding some irreplaceable links on the basis of the complete analysis of the problems space, which are called the must_be_passed link between the source and destination. Merely compute the shortest path case from those best exit/entry nodes of the grids to the irreplaceable links, and then add them into the case base storing for future use. This method is based on case-based reasoning technique and completely considers the properties of the problem space. In addition to the use of knowledge of the natural grid in the route network, this method is more efficient than existing algorithms on computing efficiency.

Keywords  path planning; case-based reasoning; typical case

CLC number  P283.7

Introduction

The critical problem of path planning is to find the optimal route between a source and a destination in a digital route map. The common methods are heuristic searching strategies directing to the destination using spatial modeling definitions in the navigation world. There are plenty of heuristic models, such as greedy algorithm exquisite exploration in operating data structure and its searching method\cite{1-3}, directive strategy reasonable restricted by searching areas\cite{4}, as well as hierarchical strategy of optimal transport network constructed by applying hierarchical reasoning\cite{5,6}. All of them are based on spatial models and excel in searching in a limited scale. Case-based reasoning offers an alternative method for path planning, by which path planners can get new solutions by means of adapting and retrieving previous solutions with the similar goals. It aims at reasoning through previous experience; the key of which is applying previous experiences in solving old problems to the new problems—using cases to reason. In such a method, the cases of typical problems have been stored in a case
base. When the user would like to resolve a new problem, he can use more cases in the case base rather than reason and search delicately from the beginning to the end every time. It is similar to human experience. For instance, if a person wants to go to a place he has been to before, he will probably use the last route he took to guide the present trip. If he wants to go to a place near which he has been to before, he will probably modify and extend that old solution to the new problem. In the process of route finding, people prefer to drive on the familiar route; naturally, case-based reasoning of path planning can meet their preferences. This method, however, may not be the best, but it is a rather efficient and immediate solution, especially in path planning algorithm, because they could avoid certain unnecessary searches and reduce searching space and time, if they used several common sense and geographical spatial knowledge.

1 Case-based reasoning algorithms in path planning

According to input source and destination, case-based reasoning technique access to the shortest path can be classified as follows:

(1) Perfectly matched. If an old case contains both the source and destination of a new problem, provide this perfectly matched part from the old case to the user directly.

(2) Partially matched. When no previous case exactly matches the source and destination, the adaptation is required. However, this adaptation is limited to nearby junctions. This simple modification can acquire a new solution with less computation, which can be stored in the case base then.

(3) Hardly matched. If neither (1) nor (2) succeeds, it is impossible to resolve the present problem merely depending upon the case-based reasoning algorithm. Thus, other algorithms such as Dijkstra algorithm need to be integrated to solve the problem.

Although the path planning methods on the basis of case-based reasoning attract more attention in the point of cognition, there are still many problems left, which make it inefficient if applied simply.

(1) In a huge road network, it is impossible to store all the shortest paths from each node to all other nodes. If none of these old cases can match the new problem, the old case has to be modified. However, complicated adaptation takes more time, cannot always guarantee an optimal solution, and will be impossible if there is no similar case in the case base.

(2) Even if people can set up a complete case base, they still have two problems: enough space for enormous storage and more time for case retrieval.

Therefore, it is indispensable for us to look for storing and retrieving case techniques in accordance with characters and knowledge of road network on the basis of case-based reasoning.

2 Definition and storage of typical case

It is impossible for a case base to store all the solutions to all the shortest paths; anyway, part of them should be stored. However, which routes should be stored and which should not be? How to adapt cases that have already been stored and how to acquire the shortest route based on them? All these issues need to be explained by integrating knowledge of network on account of case-based reasoning techniques. Anwar & Takaichi provided must_be_passed links based on the characteristics of the Japanese national road network in order to reduce the case base [6]. Thus, this paper utilizes the typical case base of network knowledge to reason so as to improve it.

A must_be_passed link is an important part that lies in a link that connects two route sub-networks and connects a source and destination in such two sub-networks. That is to say, if the source and the destination lie on two sides of the route network, the shortest path between the source and the destination must pass this link. Urban road networks develop into natural grids according to advanced grading division of the route. If each grid has a shortest path stored as a case from an entry/exit node of a grid to this must_be_passed link, naturally, finding a shortest path between this pair of nodes across the must_be_passed links is not a time-consuming task.
Typical cases can be defined as follows.

1. Store the shortest route from a grid to the must_be_passed link as a case.
2. Store the shortest path between two grids as a case, if both of them lie on the same side of the must_be_passed link.

The first case consists of three kinds of information: the source grid, the entry/exit node of the source grid and the shortest path which connects the grid and the must_be_passed link. The other one includes separated grids that belong to the source and the destination, the entry/exit node of the two grids, the shortest path which connects the grid and the must_be_passed link. The cases can be indexed by the ID of grids that belong to the source and the destination of the stored route.

Owing to the expression of the case, the issue of how to organize the cases becomes the key to improving the efficiency of the case-based reasoning system. According to the hierarchy structure mechanism of the route network, this hierarchy structure of the network and sub-network (as Fig.1) serves as an index mechanism for cases storing organizing, which means the organization of case-storing is the hierarchy structure of the network and sub-network and the neighborhoods are applied as case index.

3 Path planning algorithm of typical case-based reasoning

3.1 Steps of algorithm

Suppose that a traveler intends to go from the source O to the destination D (as Fig.2). According to the case base, the procedure of the path planning algorithm of the typical case-based reasoning approach is described as follows.

Step 1 Find the grids containing source O and destination D.

Step 2 If the source and the destination grids lie on each side of a must_be_passed link (HI), then:

1. Tell whether there is a shortest path (GH) between the source grid and a must_be_passed link, if so, retrieve the case and go to (3), or
2. Run the shortest path algorithm in the sub-network M containing the source grid to find the shortest path GH from the best exit/entry node G of the source grid to the node H on the same side of the must_be_passed link, and add the new solution into the case base.
3. Tell whether there is a shortest path IJ between the destination grid and a must_be_passed link; if so, retrieve that case and go to (5), or
4. Run the shortest path algorithm in the sub-network N containing the destination grid to find the shortest route IJ from the best exit/entry point J of the destination grid to the point I on the same side of the must_be_passed link, and add the new solution into the case base.
5. Run the shortest path algorithm in the source grid to find the shortest path OG between the source O and the best exit/entry node of (1) and (2).
6. Run the shortest path algorithm in the destination grid to find the shortest path DJ between the destination D and the best exit/entry node of (3) and (4).
7. Connect OG, GH, HI, IJ and DJ, which forms the shortest path between source O and destination D.

Or

Step 3 If the source and the destination grids lie on the same side of a must_be_passed link and both grids are different or nonadjacent, then:

1. Tell whether there is a case stored in the case base perfectly matching the path between the source grid and destination grid; if so, retrieve this case and then go to (3) in step 2; or
(2) Run the shortest path algorithm in the sub-network with the same side of two grids to find the shortest path from the best exit/entry node of the source grid to the best exit/entry node of the destination grid, and add the new solution into the case base.

(3) Run the shortest path algorithm separately in both the source and the destination grids to find the shortest path from the source and destination to the best exit/entry node of cases in (1) or (2) in step 2. Connect case (1) or case (2) and two shortest paths in (3) in step 2, which form the shortest path from the source \( O \) to the destination \( D \).

Or

**Step 4** Run the shortest algorithm directly within a certain grid or every two adjacent grids.

### 3.2 Definition of exit/entry nodes

When computing the case between grid and \textit{must_be_passed link}, or grid and the shortest path between the grids, the key of the algorithm is defining the best exit/entry nodes. Fig.3 shows the defining of best exit/entry nodes between grids and \textit{must_be_passed link}. Tell the source \( O \) lying in the grid, then get the geometrical center of the grid and follow the heuristic nodes updating technique in Reference [6]. Use Pseudo link to connect the geometrical center \( S \) of the grid with each exit/entry node, then update \( S \) into a higher hierarchy and compute the shortest path between \( SA \) (\( A \) is a node of \textit{must_be_passed link}, lying in the same side of the grid) in the higher hierarchy of the road network. If node \( B \) is considered as the best exit node on the shortest path of the grid, then store the shortest path between \( BA \) as a case. In the planning of new routes, the method for getting the path between nodes that lie in this grid and \( B \) is computing the shortest path between this node and \( B \), retrieving the case \( BA \) and then acquiring the path between this node and \( A \). Fig.4 shows the definition of best exit/entry node between two grids. Separately, grid 1 and grid 2 are the grids of source \( O \) and destination \( D \), and connecting the exit/entry nodes between \( O \) and \( D \) through pseudo link. Also, the cost of pseudo link can be substituted for the sum of the distance, between the source \( O \) (destination \( D \)) and every exit/entry node, and distance, between every exit/entry node and the destination \( D \) (source \( O \)). Then find the best exit/entry node through computing the sum of minimum distance. If the result is that \( A \) is the best exit node of grid 1 and \( B \) is the best entry node of grid 2, then, the endpoints of the case for the shortest path between grid 1 and grid 2 separately record \( A \) and \( B \), and then \( AB \) is stored as the case. For planning the new route, if the source \( O' \) lies in grid 1 and the destination \( D' \) lies in grid 2, while the distance of pseudo link between \( O' \) and \( A \) and the distance of pseudo link between \( D' \) and \( B \) are shorter than other distances between exit/entry nodes, then the shortest path algorithm can be used to compute the shortest path between \( O' \) and \( A \) or between \( D' \) and \( B \). Connecting \( O'A—AB—BD' \) will form the shortest path between \( O' \) and \( D' \). If by computing the cost of pseudo link, the best exit/entry node of grid 1 is \( A \), while the same node of grid 2 is \( C \), compute the new shortest path route \( AC \) and store it.

**Fig.3** The enter/exit node between grid and link

**Fig.4** The enter/exit node between two grids

### 3.3 Analysis of algorithm

When the source and the destination lie in two sides of the \textit{must_be_passed link}, the shortest path from the source grid to the \textit{must_be_passed link} and from the destination to the \textit{must_be_passed link} can be acquired by case-retrieving. In addition, one endpoint of these cases are the best exit/entry node of the source and destination grids, and then the shortest path between the source and the destination and these endpoints can be obtained by running the shortest path algorithm. Since such route findings are, respectively, limited in each grid, the searching space is small and the speed is fast. If the grid of the source and the destination lie in the same side of the \textit{must_be_passed link}, even though there is no shortest path case left, the speed is still fast because the searching space is limited to only one side.
and obviously reducing the number of the searching arcs and nodes. For the storage of case base, there are two improvements: (1) A must_be_passed link divides the network into two parts (supposing one part with \( m \) grids and the other with \( n \) grids), so the number of the storing cases is \( m + n \). Supposing all cases between the grids are stored, then the sum is \( \nu C_m^2 +(m+n)+C_n^2 \). Without a must_be_passed link, however, the number of cases is \( m \cdot n \), while the sum of cases is \( C_{m+n}^2 \). Especially when \( m \) and \( n \) are large, the latter is obviously larger than the former. (2) The shortest path cases between several nodes in two grids, in extreme cases, can be substituted for \( C_e^1 \times C_f^1 \) cases between two grids at most, where \( e \) and \( f \) are the numbers of exit/entry nodes in the source grid and destination grid, respectively. With the verification of the experiment, in most situations, the number of cases is far less than the extreme value, so the storage amount will obviously reduce.

4 The experiment and its result analysis

The digital route map applied in the experiment is the road map of Wuhan. In this experiment, the link that the Yangtze River Bridge lies in is defined as a must_be_passed link. Then we set up a grid index (in such a case, there are 166 grid-polygons) in accordance with the high hierarchy route network data. If the source and the destination respectively lie to the two sides of the must_be_passed link, then set up the case between the source grid or destination grid and the must_be_passed link, and find the shortest path between the grid center and one endpoint of the must_be_passed link by computing to get the exit/entry node of the grid. Thus, there is only one case, by which all the nodes lying in this grid can be retrieved. That means that the final route is consisted of five parts: two shortest paths between the source or the destination and exit/entry nodes (by computing algorithms), the shortest path from the exit node of the source grid to the must_be_passed link (through case retrieving), the must_be_passed link, the shortest path from the entry node of the destination grid to the must_be_passed link (through case retrieving), as shown in Fig.5.

In this experiment, the author picks up eight pairs of nodes (the sources and the destinations of these 8 pairs respectively lie to the two sides of the must_be_passed link) and tests the time cost of retrieving in case bases with different scales. From Table 1, the

| Node pair ID | Path length / m | The hierarchical algorithm / ms | The case-based algorithm / ms |
|--------------|-----------------|-------------------------------|-------------------------------|
| (5 052, 5 337) | 4 065.86 | 1 663 | 510 |
| (5 069, 5 798) | 6 766.15 | 3 275 | 400 |
| (4 237, 5 271) | 7 169.46 | 1 653 | 391 |
| (3 928, 4 377) | 12 505.83 | 3 435 | 470 |
| (3 995, 5 871) | 14 704.13 | 2 153 | 410 |
| (2 527, 6 438) | 15 779.73 | 5 638 | 921 |
| (2 787, 6 046) | 17 760.86 | 1 613 | 300 |
| (1 385, 6 459) | 24 457.53 | 19 047 | 561 |

Fig.5 Retrieving the route by must_be_passed links case
time cost in case-based reasoning approaches is unrelated to the length of the route but related to the scale of the case base. When the case base increases from 166 (166 is the number of grids in hierarchy road network) to 166×3 (if there are three must_be_passed links), the time-consumption rises about 20 seconds. While the number of cases increases to 166×50, the time-consumption rises about 370 seconds, and when it increases to 166×100 (date amount is 1.76MB), the time consumption of the algorithm rises about 700 seconds, which is still less than hierarchical route finding algorithms.

5 Conclusion

It is hard for general case-based approaches to assure which cases should be stored and which should not be. This paper applies the case-based reasoning in another point, i.e., finding some irreplaceable links on the basis of the complete analysis of problems space, which are called the must_be_passed link between the source and destination. Merely compute the shortest path case from those best exit/entry nodes of the grids to the irreplaceable links, and add them into the case base stored for future use. This method is based on case-based reasoning technique and completely considers the properties of the problem space. In addition to the use of knowledge of natural grids in road networks, the method is more efficient than existing algorithms in computing efficiency. For some urban road networks, in the situation where the irreplaceable link cannot be found, it is still able to divide the route network using natural grids. By transforming the shortest case between node and node to cases between grid and grid that contain these nodes, the number of cases has been extremely reduced and the efficiency of the case index has been improved by applying hierarchical index mechanism to case storing.

References

[1] Deo N, Pang C Y (1984) Shortest-path algorithms: taxonomy and annotation [J]. Networks, 14: 275-323
[2] Zhan F B, Noon C E (1998) Shortest path algorithms: an evaluation using real road networks [J] Transport. Sci., 32 (1): 65-73
[3] Karimi H A (1996) Real-time optimal route computation: a heuristic approach [J] JSJ, 3(2):111–127
[4] Lu Feng, Lu Dongmei, Cui Weihong (1999) Time shortest path algorithm for restricted searching area in transportation networks [J]. Journal of Image and Graphics, 4A(10): 849-853
[5] Car A, Frank AU (1994) General principles of hierarchical spatial reasoning - the case of wayfinding [C] Proceedings of Sixth Int. Symposium on Spatial Data Handling, Edinburgh, Scotland
[6] Weng Min, Wu Hehai, Du Qingyun, et al. (2006) A heuristic and hierarchical wayfinding algorithm based on the knowledge of road network [J]. Geomatics and Information Science of Wuhan University, 31(4):360-363
[7] Zhu Fuxi, Tang Yiqun, Fu Jianming (2002) Artificial intelligence principle [M]. Wuhan: Wuhan University Press
[8] Weng Min (2006) Research on criteria of route selection and methods of wayfinding in the cognitive space of transportation networks [D]. Wuhan: Wuhan University
[9] Anwar M A, Yoshida (2001) Integrating OO road network database, cases and knowledge for route finding [C]. SAC2001, Toronto