A New Path Search Algorithm for Providing Paths among Multiple Origins and One Single Destination

Liang Zhao¹, Wenjia Wang²

Hitachi (China) Research & Development Corporation,
301 North Wing Tower C Raycom Infotech Park,
2 Kexueyuan Nanlu, Beijing, China
¹lzhaohitachi.cn; ²wjwang@hitachi.cn

Abstract
Route planning services provide directional functionalities allowing people to make transport plan in which such services are now closely interrelated to people’s daily life. Existing route planners such as digital map only offer the path search services originated from one place which cannot fulfill the increasing needs of people. In this paper, we first described a new service for digital map that provides route plans for people from different origins towards the same destination for the purpose of group travelling. Since the performance of route planner highly depends on the shortest path search algorithms, in this paper, weal so introduced our newly proposed path search algorithm, namely, Multi-origins Common Destination algorithm. The initial evaluation results showed that it out performs the counterparts.

Keywords
Group Travel; Route Planning; Multiple Origins

Introduction
Modern lifestyle propels the intensive needs of people’s movement, i.e. transportation. Therefore, it is of vital importance for providing navigation or map services to transportation among places, in order to meet the fast-paced life. Nowadays, the popularity and proliferation of navigation technologies and services are growing rapidly and especially used for route planning in densely-populated metropolitan areas. The provisioning of intelligent navigation services to citizens and business has been a strategic objective for organizations and governments worldwide to promote the quality of life. The intelligent route planning technologies provide point-to-point (s) route planning for both public transports and private transports through digital devices such as PC, laptop, smart phone or on-board navigation system etc., which allow users to find the most efficient route with saving up gasoline prior to one’s trip or during one’s trip. A lot of people like traveling, shopping, or hanging out in a group. While travelling together, a group of friends can first gather in a place from various departure places, then move together to the same destination. This allows participant familiar with the route that can guide others to the destination. Moreover, it also saves gasoline if participants gather in the rendezvous point then drive together towards the destination. However, with regards to such service, there is no existing map service and efficient technology to provide route plans for meeting such needs. In addition, in this context, this problem is to find a rendezvous-point before destination with the total minimized distance of paths on road network. However, existing path search algorithms cannot be applied directly to solve this problem as all of existing algorithms are designed for finding paths originated from single node. Therefore, to address this problem, in this paper, we proposed a new algorithm to find the approximate optimal rendezvous-point and the corresponding routes. The reminder of this paper is organized as following: Section 2 reviews the related work. Section 3 describes the Dijkstra’s algorithm and Section 4 introduces the proposed algorithm (MOD). In Section 5, the performance of MOD is evaluated. Finally, Section 6 concludes and directs the future work.

Related Work
In recent years, route planning services and technologies have attracted a lot of research attention. In the service point of view, existing route planning services can be divided into followings, the single-origin-to-single-destination route planning service, and the single-origin-to-multiple-destinations route planning service, which are offered by the major map search engine such as Google Map, and Bing Map for providing driving, public transports, walking and
hybrid transportations. Basically, single-origin-to-single-destination route planning service provides travel route from one origin to one destination. In contrast, the single-origin-to-multiple-destinations route planning service plans the routes from one origin to a final destination through one or multiple waypoints. In the technology point of view, route planning technologies are instances designed to solve the shortest path problem. As a type of shortest path problem, the single-pair shortest path problem is to find the shortest path from one source (origin) to one destination. The algorithms designed for this problem commonly are referred to making the single-origin-to-single-destination route plans in the digital map services.

There is quite a number of existing shortest path search algorithms designed for the single-pair shortest path problem. Among them, Dijkstra’s shortest path algorithm is the most commonly known to find the optimal shortest path from one origin to all vertices, while it is also can be used to find the route from one origin to one destination. Bidirectional Dijkstra’s algorithm is a variant of Dijkstra’s algorithm with improved calculation speed. A Star algorithm, Bidirectional A Star algorithm and the variants etc., are the most popular path search algorithms designed for the single-pair shortest path problem, implemented by major web-based map services to find the approximate shortest path between two nodes due to its low computational time and low memory consumption. These algorithms are heuristics while they estimate the minimum possible distance to the destination end during the calculating. In A Star algorithm, it estimates the distance from the current node (including the origin and all other nodes along the search) to the destination node in order to narrow the search direction and reduce the search complexity. Its bidirectional derivative, i.e. Bidirectional A Star algorithm runs two searches simultaneously, and sets the opposite end-node as the target to estimate the distance, i.e. a search state originated from origin node sets the destination node as the distance estimation target, while the origin node is the destination estimation target for the search state originated from destination node. Existing algorithms for the single-pair shortest path problem cannot provide effective solution to realize the group route planning, i.e. ‘the rendezvous-point shortest path problem’—finding of paths from multiple sources to single destination via a common rendezvous-point. Unlike the fixed positions of origin and destination, the location of rendezvous-point is vague until the route search is completed. For this reason, modifications of the heuristic algorithms are required to determine the rough position of rendezvous-point in the starting of the algorithms to estimate the distance to each search end, in order to be applied to solve the rendezvous-point shortest path problem. In contrast, Dijkstra’s algorithm and A Star algorithm can be implemented to find the rendezvous-point with some minor modifications. The modified algorithms should calculate the shortest paths from each starting point (including the origins and destination) to rest of vertices on the map, then these paths are compared to get one node with the minimum cost to all starting points where this node is the rendezvous-point. Although the modifications simple and straightforward, the modified algorithms are inefficient due to its nature of full graph search, which are infeasible for route planning system to handle thousands of multiple queries at the same time.

Deng et al., presents a multi-source skyline query processing scheme for road networks, where as this work focuses on providing the shortest paths from multiple origins to a common destination. Compared to the objective of proposal in this paper, the work is only designed to find the paths from multiple origins to the same destination where the paths are not through a common rendezvous-point for allowing participants to meet before moving towards the destination.

In order to provide efficient route search algorithm for the group travel planning system, we proposed new algorithm where the detailed algorithm is describe in Section 4. To better understand MOD, the basic computing process of Dijkstra’s algorithm has been described which can in Section 3.

**Dijkstra’s Algorithm**

A road network can be defined as a graph $G = (E, V)$, where $V$ is a set of nodes corresponding to road junctions and $E$ is a set of non-directional edges (an edge can be either a straight line or a polyline) between two nodes from $V$ corresponding to road segments. If two nodes in $V$ are linked as an edge in $E$, then they are adjacent to each other. All the adjacent nodes of $v \in V$ are stored in set $U$. Let $d(v, v')$ be the total length of the edges along a path (i.e. distance) between the two nodes, $v$ and $v'$. If $v$ and $v'$ are not connected on any path, then $d(v, v') = \infty$. The shortest distance of path length of the shortest path between $v$ and $v'$ is denoted as $d_t(v, v')$.

Dijkstra’s algorithm finds the shortest route from a
given origin node, \( v_o \) to a destination node \( v_d \) in the graph. During the search, if a shortest path from \( v_o \) to a node \( v' \) is found, \( v' \) will be moved from set \( U \) to set \( S \) in which \( S \) is a set to store all the shortest-path-founded nodes. In the first step of this algorithm, every node \( v \in U \) with the distance \( d(v_o, v) \) is put into \( S \). Then, the algorithm iterates to replace a node \( v \in S \) (if \( d(v_o, v) \) is the minimum for all nodes in \( S \)), by all the nodes in set \( U \). For example, for each node \( v' \in U, d(v_o, v') \) is set to \( d(v_o, v) + d(v, v') \) if \( v' \) has not yet been moved to \( S \) or (in case \( v' \) in \( S \) the existing estimate of \( d(v_o, v') \) is larger than \( d(v_o, v) + d(v, v') \). This process terminates when \( v_d \) is selected from \( S \) while \( d(v_o, v_d) = d(v_o, v) \). If the termination condition is not set as finding the shortest path to \( v_d \), Dijkstra’s algorithm then continues to compute the shortest paths from \( v_o \) to all other nodes in the graph.

**TABLE 1 COMPUTATION PROCESS OF DIJKSTRA’S ALGORITHM**

<table>
<thead>
<tr>
<th>Step</th>
<th>Set ( S )</th>
<th>Set ( U )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Move ( A ) to set ( S ), then ( S = {A} ), Shortest paths: ( A \rightarrow A = 0 ) Start search by applying ( A ) as mid-point</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( U = {B, C, D, E} ) Tentative distances: ( A \rightarrow B = 6 ) ( A \rightarrow ) (other nodes in ( U )) ( \rightarrow ) Find ( A \rightarrow C = 3 ) is the shortest path to ( C ). Tentative distances: ( A \rightarrow B = 6 )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Move ( C ) to set ( S ), then ( S = {A, C} ) Shortest paths: ( A \rightarrow A = 0 ), ( A \rightarrow C = 3 ) Applying ( C ) as mid-point, and start search from the shortest path of ( A \rightarrow C = 3 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( U = {B, D, E} ) Find new path ( A \rightarrow C \rightarrow B = 5 ) as the shortest path to ( B ) (shorter than the path ( A \rightarrow B = 6 ) from last step). Tentative distances: ( A \rightarrow C \rightarrow D = 6 ), ( A \rightarrow C \rightarrow E = 7 )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Move ( B ) to set ( S ), then ( S = {A, C, B} ) Shortest paths: ( A \rightarrow A = 0 ), ( A \rightarrow C = 3 ), ( A \rightarrow C \rightarrow B = 5 ) Applying ( A ) as mid-point, and start search from the shortest path of ( A \rightarrow C \rightarrow B = 5 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( U = {D, E} ) Find new path ( A \rightarrow C \rightarrow B \rightarrow D = 10 ) (longer than path ( A \rightarrow C \rightarrow D = 6 ) from step 2). Therefore, ( A \rightarrow C \rightarrow D = 6 ) is the shortest path to ( D ). Tentative distance: ( A \rightarrow C \rightarrow E = 7 )</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Move ( D ) to set ( S ), then ( S = {A, C, B, D} ) Shortest paths: ( A \rightarrow A = 0 ), ( A \rightarrow C = 3 ), ( A \rightarrow C \rightarrow B = 5 ) Applying ( D ) as mid-point, and start from the shortest path ( A \rightarrow C \rightarrow D = 6 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( U = {E} ) Find new path ( A \rightarrow C \rightarrow D \rightarrow E = 8 ) (longer than path ( A \rightarrow C \rightarrow E = 7 ) from step 2). Therefore, ( A \rightarrow C \rightarrow E = 7 ) is the shortest path to ( E )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Move ( E ) to set ( S ), then ( S = {A, C, B, D, E} ) Shortest paths: ( A \rightarrow A = 0 ), ( A \rightarrow C = 3 ), ( A \rightarrow C \rightarrow B = 5 ), ( A \rightarrow C \rightarrow E = 7 )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Now set ( U ) is empty, search ends.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1 shows an example of a five nodes topology with weight value beside each link. In this example, Dijkstra’s algorithm is applied to find paths from node \( A \) to rest of nodes in this weighted graph, while Table 1 lists the detailed computing process of using this algorithm.

**Group Travel Planning for People from Multiple Origins towards the Same Destination**

This system is designed for group travelling to find the travel routes, while especially driving routes can be found for people originated from multiple places towards the same destination and intend to meet in a rendezvous-point before moving to the destination. The system requires user to put in search requirements, including the name of departure place of each participator in this group travel event, and the name of common destination place and other information such as e-calendar, email address, and mobile phone numbers etc. In particular, a new algorithm has been proposed which finds paths for group travel, to realize this system. Our proposed algorithm finds the rendezvous-point on the shortest paths to participators’ location and the destination. In addition, by implementing the proposed algorithm, system then can generate estimated driving time, gasoline cost and distance for paths from each participator to rendezvous-point and from rendezvous-point to destination respectively. All search results are shown on the GUI. In this paper, anew algorithm was proposed, namely, Multi-Origins Common Destination algorithm (MOD). For multiple given source vertices (origin nodes) and one common destination vertex (destination node) in the graph, this algorithm finds a suitable rendezvous-point, in which the total cost of paths from each given vertex via the rendezvous-point to the destination vertex is the approximate lowest.

Figure 2 shows the operation process of MOD algorithm. The MOD is an algorithm trying to solve the following problem: given a number of origins \( S \) and a single destination node \( v_o \), find a rendezvous-point \( r \) that minimizes \( d(v_o, l) + \sum d(s, v_o) \), where \( \forall s \in S, d(s, r) \) is the shortest distance between node \( s \) and node \( r \). Each starting node (including origins and destination)
starts the searches. MOD algorithm operates one search each time. For each search state, one step of Dijkstra’s algorithm is implemented. Here, one step of Dijkstra is one single step or an iteration of Dijkstra’s algorithm shown in Table 1 in which all search states run one-by-one simultaneously in round-robin fashion. The MOD algorithm terminates when a node is visited by all search states. Hence, this node is the rendezvous-point in which the corresponding paths are also generated. As the searching ends at a premature termination, the rendezvous-point is possible to be non-optimal which means continuing the search could still lead better solution. However, our current solution of finding the optimal rendezvous-point is not quite efficient. Therefore, in this paper, we only showed the MOD with the approximate optimal rendezvous-point and related paths.

**Performance Evaluation**

The performance of MOD has been primarily evaluated with comparison to Dijkstra’s Algorithm with modification (MAD) in terms of two evaluation metrics, *computation time* and *the number of nodes visited*. In Section 4.1, we first describe our comparison target, MAD. Then we show the evaluation results in Section 4.2.

To set the comparison target, the Dijkstra’s algorithm has been modified to have MAD. First of all, MAD applies the Dijkstra’s algorithm to compute and record the shortest paths from each origin node to rest of nodes, and the shortest paths from destination node to rest of nodes on the map. Then these paths are compared to find the rendezvous-point, where a node on the shortest paths to the pre-set origins and destination is chosen as the rendezvous-point.

We implemented the simulation model of the MOD algorithm and MAD algorithm in Java to examine the efficiency, where the experimental environment for analyzing the performance of algorithms in this article is as following:

Processor: Intel Core i7-2640M @ 2.80GHz

Memory: 4 GB (DDR3 1333MHz)

GPU: Nvidia NVS 4200M 1GB

Map: New York City and Florida City of US

The primary evaluation program models the simulation of up-to six origins, i.e. participants of the group travelling are from six different places towards one same destination. The evaluation program randomly selects the position of origins and destination. In this paper, MOD was compared with MAD with evaluation metrics, *the number of nodes visited* and *computation time*. The *number of nodes visited* expresses the efficiency of a path search algorithm, where the less nodes visited by a path search algorithm leads to better search speed. The *computation time* simply counts the time duration between the beginning and end of running an algorithm, which presents the overall effectiveness of a path search algorithm.

In Figure 3, the shown example of MOD algorithm intends to find the rendezvous-point and paths from two origins A and B to a common destination node D. Figure 3 (a) illustrates that the search states start from three ends, A, B and D. As the first node visited by all three searches, node M is found as the rendezvous-point in Figure 3 (b). As shown in Figure 3 (c), the rendezvous-point node M is found. Then the paths are generated by the algorithm: (1) MOD reverses the path from D to M as D is the destination node and has \(M>D\), (2) the path from A to D via M is \(A>C>M>F>D\) and the path from B to D via M is \(B>E>M>F>D\).
nodes are visited to find a rendezvous-point by MOD when there are only two origins and one destination. This number rises to around 4.65 when there are six origins and one destination. As MOD sets the first node visited by all searches to be the Rendezvous-point, it creates search overlaps on the map, causing more search overlaps when there are more search states, i.e. more origins. Consequently, it costs more if the number of searches increases. In contrast, the number of nodes visited by MAD is directly proportional to the number of origin nodes. Figure 5 shows the comparison of the computation time between MAD and MOD. As the number of origin nodes goes up, the running time rapidly increases. This clearly presents the advantage of MOD in saving time cost, and proves that MOD is suitable for the group travel system which can instantly generate the search results based on each query. There may be perplexing about the downward trend line of Figure 4, and upward trend line of Figure 5. The reason is that the evaluation metric the number of nodes visited only considers the total cost of finding the shortest paths from origin to all nodes in the graph, where it does not take the cost of comparison of these paths to find the best rendezvous-point. However, for evaluation metric computation time, it counts the total running time of each algorithm considering the node-visiting and path-finding.

To sum up the performance evaluation, with simulated two evaluation metrics, MOD outperforms its counterparts MAD, especially in processing speed.

**Conclusion and Future Work**

This paper has presented a new path search algorithm, MOD to realize the route planning service providing group travel plans for people. To best of our knowledge, MOD is first algorithm that was special designed for group travel planning. The primary performance results also show that MOD outperforms existing shortest algorithms in terms of computational speed. For our future research direction, first of all, in that MOD only provides the approximate optimal solution, MOD was expected to refine in order to find an effective way to achieve the optimal rendezvous-point and shortest paths. In addition, the feasibility of applying the heuristics search strategy in MOD has also been investigated to strengthen the performance.

**REFERENCES**


