Hans Journal of Data Mining 数据挖掘, 2022, 12(3), 246-258
Published Online July 2022 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2022.123025

Hans汉斯

基于离线关键站点和路径阻抗的地铁路线推荐方法

戴政君[1]，宋　莹[1,2*]

[1]北京信息科技大学，数字文化重点实验室，北京
[2]计算机体系结构国家重点实验室，中国科学院计算技术研究所，北京

收稿日期：2022年6月1日；录用日期：2022年7月5日；发布日期：2022年7月12日

摘　要

目前主流的路线推荐方法先通过寻径算法在满足各种约数条件下寻找K条可行路线，然后根据乘客的出行偏好从K条可行路线中选择一条路线推荐给乘客。然而，由于寻径算法的约束条件只能通过实时计算得到K条可行路线，当同时进行大量乘客的路线推荐时，由于无法快速获得推荐路线从而影响乘客的用户体验。因此本文提出基于离线关键站点的K最短时间算法，通过构造关键站点图和哈希表离线辅助查找K条最短路径，从而降低算法的时间复杂度。此外，通过分析发现，乘客对出行的负面感受与地铁换乘次数及拥挤度间并不是简单的线性关系，随着换乘次数和拥挤度的增加，乘客对出行的满意度会迅速衰减，所以，目前通过最大最小值方法计算出的乘客偏好路线并不一定是最佳路线。因此，本文设计了基于路径阻抗的地铁路线推荐方法，其中路径阻抗由不同权重的地铁时间成本和拥挤成本组成，通过计算K条最短时间路径的路径阻抗，从中选出符合乘客偏好的出行线路推荐给乘客。通过理论分析，随着查询次数的增多，本文提出的基于离线关键站点的K最短时间算法时间复杂度要远小于其他约数条件下的寻径算法。最后通过具体的实验示例验证本方法完全可以满足不同乘客的路线推荐需求。

关键词

地铁刷卡数据，寻径算法，路线推荐

Offline Key Station and Path Impedance Based Subway Route Recommendation Method

Zhengjun Dai[1], Ying Song[1,2*]

[1]Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing Information Science and Technology University, Beijing

[*]通讯作者。

文章引用: 戴政君, 宋莹. 基于离线关键站点和路径阻抗的地铁路线推荐方法[J]. 数据挖掘, 2022, 12(3): 246-258.
DOI: 10.12677/hjdm.2022.123025

[2]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing

Received: Jun. 1st, 2022; accepted: Jul. 5th, 2022; published: Jul. 12th, 2022

Abstract

The current mainstream route recommendation method first finds K feasible routes by path finding algorithm under various approximate conditions, and then selects a route from K feasible routes to recommend to passengers according to their travel preferences. When a large number of passengers' routes are recommended at the same time, the recommended routes are not available quickly and thus affect the passengers' user experience. Therefore, this paper proposes the K shortest time algorithm based on offline key sites, which can reduce the time complexity of the algorithm by constructing a key site graph and hash table to assist in finding K shortest routes offline. In addition, it is found through analysis that there is not a simple linear relationship between passengers' negative feelings about the trip and the number of interchanges and crowding, and passengers' satisfaction with the trip will decay rapidly as the number of interchanges and crowding increase. Therefore, the current passenger preference route calculated by the maximum-minimum method is not necessarily the best route. Therefore, this paper designs a subway route recommendation method based on path impedance, where the path impedance consists of different weights of subway time cost and congestion cost, and by calculating the path impedance of the K shortest routes, the travel routes that meet passenger preferences are selected from them and recommended to passengers. Through theoretical analysis, with the increase of the number of queries, the time complexity of the K shortest time algorithm based on offline key stations proposed in this paper is much smaller than that of the path finding algorithm under other approximate conditions. Finally, it is verified through specific experimental examples that this method can fully meet the route recommendation needs of different passengers.

Keywords

Metro Smart Card Data, Path Finding Algorithm, Route Recommendation

1. 引言

1.1. 研究背景

近年来，随着外来人口的增加以及私家车拥有量的迅猛增长，国内外各大城市的交通压力陡然剧增，交通拥挤已经成为城市面临的一大难题。大力发展城市公共交通则是解决道路拥堵的一个重要方法，目前，地铁等公共交通已经成为如纽约、上海等许多城市地区日常通勤者的主要交通方式[1]。但是，过多乘客选择地铁出行造成了地铁线路的拥挤，早晚高峰时段地铁车厢和站台拥挤已成为常态。

与此同时，人们对地铁乘坐舒适度的需求与日俱增，在行程时间允许的情况下，乘客普遍希望尽可能在乘坐地铁时有较为宽松舒适的乘车环境，但是如果增加过多的车次或者增加过多的车厢在乘客人流量较少时又会造成浪费。这就对地铁管理者们提出新的要求，即需要在地铁运输效率和乘客舒适度之间

做出权衡。对于乘客来说不仅考虑出行时间，还应该考虑乘坐地铁时车厢拥挤程度。许多现有的地图软件，如谷歌地图和百度地图等，大多只是根据地铁乘坐时间或换乘次数来推荐路线，通常忽略了车厢拥挤的问题[1]。基于以上分析，根据乘客出行时间与拥挤度权衡的线路推荐方法研究是很有必要的，可以为乘客提供以舒适度为目标的路线推荐。

1.2. 研究现状

本节主要介绍目前各种基于约束条件下的寻径算法和为了满足人们多样的出行需求衍生出的各种路径推荐方法。

1.2.1. 寻径算法研究现状

地铁系统的寻径方法主要采用最短路径算法、K 最短路径算法和基于地铁动态调度的寻径算法。经典的最短路径算法有 Dijkstra 算法[2]等，用于查找网络拓扑模型两个节点中的最短路径。由于 Dijkstra 算法其时间复杂度高，无法满足求解大型地图上最短路径问题，所以基于 Dijkstra 算法提出了 A*算法[3]。A*算法是一种启发式搜索算法，通过加入了启发信息来指引搜索的方向，从而避免遍历图中的所有节点，因此路径查找效率要明显高于 Dijkstra 算法。

但是最短路径算法只能求解两点之间最短的那一条路径，无法满足求得最短、次短、再次短等路径。由此 K 最短路径算法被提出来寻找网络中一对节点之间的 K 条最短路径。K 最短算法的传统方法是通过计算足够多的最短路径，并删除不满足约束条件的路径。比如 Yen 提出了求解 K 最短路径问题的 Yen 算法，该算法以 Dijkstra 为基础，使用递推法中的偏离路径的算法思想来逐一求解前 K 条最短路径[4]。

然而，这些方法没有考虑到地铁列车运行的时间特征，因此，许多学者使用列车时刻表提出了基于列车动态调度的最优寻径算法。如 Xu 等人根据基于调度的模型，提供了一种两阶段标记算法来搜索 K 条最短路径[5]。Zhou 等人提出了一种利用上车和下车时间的约束从有效乘客路径集中删除不可行路径，从而获取可能路径集的算法[6]。Li 等人将时空棱镜引入到路径集生成中，创建了一种兼顾地铁系统时空特性的路径集生成算法[7]。Jarlyasunant 等人利用换乘点，预先计算出每条公交线路的起始站到其他每条公交线路的终点站之间的可行路径查找表，这种通过预先计算路径的方法可以大幅减小寻径时间[8]。Yang 等人使用时间依赖模型对 K 最短路径算法进行了改进，以解决时间表信息化交通系统的 K 最早到达问题[9]。Wang 等人采用时间扩展方法对时间表信息进行建模，然后使用 Martins-Santos (MS)算法寻找 K 最早到达路径[10]。Jeon 等提出了一种改进的基于调度的公交路线算法，该算法使用了换乘阻力进行多路径搜索[11]。

1.2.2. 路线推荐方法研究现状

常见的路线推荐算法是基于最快到达或者最少换乘，然而这些算法推荐出来的线路不一定满足乘客的用户需求。为了满足不同乘客出行偏好的需求，一些研究人员对图搜索算法进行改进，这些改进算法的基本思路是构造一些指标来定量表示乘客的出行偏好，然后将乘客的出行偏好融入到图搜索过程中。如 Delling 等人提出一个以必应地图(Bing Maps)引擎为核心的考虑现实交通状况的路线推荐引擎[12]。Kim 等人研究了拥堵对乘客路线选择的影响，结果表明，人们改变路线不仅是为了避免拥挤造成的延误也是为了避开拥挤的人群[13]。

然而，在地铁网络中的进行路线推荐时，除了考虑上述的约束条件，还应该考虑地铁列车的运行时间表。Li 等人使用刷卡数据和列车时刻表研究乘客对不同路径的偏好选择[14]。Li 等人首先通过出发站的前向搜索计算出到达站的可达换乘站，然后通过目的站的后向搜索确定出站的可达换乘站，将两个可行轨迹的换乘站交点作为乘客的可行路径集，然后考虑到乘客对出行时间、换乘惩罚和拥塞容忍度的不

同感知进而进行路线推荐[15]。

1.3. 研究内容

本文的路线推荐方法包括寻找可行路径的方法和乘客偏好的路线计算方法两部分。对可行路径的查找方法主要使用时间约束或者结合换乘站来降低寻找 K 条可行路径的时间复杂度，然而这些在线计算的方法无法满足路线推荐系统低延迟、高并发的要求。因此本文使用基于离线关键站点的 K 最短时间算法，通过构造关键站点图降低算法的时间复杂度，并且通过构建哈希表辅助查找 K 条最短路径，进一步降低了时间复杂度。

此外，通过分析发现，乘客对出行的负面感受与换乘次数及拥挤度间并不是简单的线性关系，随着换乘次数和拥挤度的增加，乘客对出行的满意度会迅速衰减。因此，目前通过最大最小值方法计算乘客出行偏好的路线并不一定是最佳路线。因此本文设计了基于路径阻抗的地铁路线推荐方法，该方法的路径阻抗是不同权重的乘客乘坐地铁时间成本和拥挤成本组成的。最后通过计算 K 条最短时间路径的路径阻抗，从中选出符合乘客偏好的出行路线推荐给乘客。

2. 研究方法

本文提出的基于离线关键站点和路径阻抗的地铁路线推荐方法包括两个部分，第一部分是使用本文提出的基于离线关键站点的 K 最短时间算法找到 K 条最短时间路径，第二部分是使用本文提出的基于路径阻抗的地铁路线推荐方法根据乘客不同的偏好从中选择合适的路线推荐给乘客。

2.1. 基于离线关键站点的 K 最短时间算法

在城市中，由于市中区人口众多、人流量大，导致地铁站点在市中区分布密集，其中包含大量的换乘站。然而在城市边缘，地铁线路非常稀疏，通常不会和其他线路相交。因此在查找两个站点之间的 K 条最短时间路径时，如果对全部地铁网络图 $G=\langle V,E\rangle$ 进行路径查找无疑会增加大量的无用运算，其中 $V=\{v_1,\cdots,v_n\}$ 表示 n 个地铁站点组成的集合，$E=\{e_1,\cdots,e_m\}$ 表示 n 个站点构成的 m 条路段的集合。

地铁站点分为起始站、终点站、普通站和换乘站，并且普通站的数量要远大于其他三种站点数量。在查找可行路径时只有在遍历到换乘站时才会产生不同的分支路径，如果只保留换乘站作为关键站点，那么就构造出一个新的地铁网络图 $G'=\langle V',E'\rangle$，本文将其命名为地铁关键站点图，其中 $V'=\{v_1',v_2',\cdots,v_{n'}'\}$ 表示 n' 个关键站点组成的集合，$E'=\{e_1',e_2',\cdots,e_{m'}'\}$ 表示 n' 个关键站点组成的 m' 条关键路段集合。因为地铁关键站点图 G' 是一个小而稠密的图，那么在这个地铁关键站点图 G' 中查找可行路径的时间复杂度将会显著降低。

本文提出的基于关键站点的 K 最短时间算法主要分为以下三步：

第一步：确定乘客起始站 O 和目的站 D 在地铁关键站点图 G' 中的位置。设起始站 O 和目的站 D 分别位于关键站点图 G' 中的关键路径 e_O' 和 e_D'，如果通过遍历全部地铁网络图 G 确定关键路径无疑会产生额外的计算。本文使用以空间换时间的代价提高查找效率，即使用直接地址法 H 构建哈希表，提高查找效率，以起始站 O 为例，如公式(1)所示。

$$H(O)=e_O' \tag{1}$$

第二步：在地铁关键站点图中寻找 K 条最短时间关键路径集 R_{OD}'。本文使用 Yen 算法[4]寻找 K 条最短关键路径，其中 Yen 算法的时间复杂度取决于在支路计算中使用的最短路径算法，而 Dijkstra 算法[2]的时间复杂度为 $O(n'^2)$，Fibonacci 堆[16]的时间复杂度为 $O(m'+n'\log n')$。因此使用基于 Fibonacci 堆的 Yen 算法寻找 K 条最短时间关键路径集 R_{OD}' 的平均时间复杂度为 $O\left(K\log_{n'}(m'+n'\log n')\right)$。其中在使用

Yen 算法寻找 K 条最短时间关键路径时，考虑到乘坐地铁的总出行时间 T' (公式(2))除了包括在列车上的乘坐时间 T'_{travel}，还包括候车时的等待时间 T'_{wait} 和换乘时间 T'_{trans}，其中换乘时间表示从一个线路站台到另外一条线路站台的时间。本文定义平均等待时间 t'_{wait}，平均换乘时间 t'_{trans}，对于 t 次换乘的出行线路则有换乘时间 T'_{travel} (公式(3))，等待时间 T'_{wait} (公式(4))。

$$T' = T'_{wait} + T'_{travel} + T'_{trans} \tag{2}$$

$$T'_{trans} = t \cdot t'_{trans} \tag{3}$$

$$T'_{wait} = (t+1)t'_{wait} \tag{4}$$

第三步：通过还原 K 条最短时间关键路径 R'_{OD} 得到完整的 K 条最短时间路径集 R_{OD}。对于 R'_{OD} 中的一条关键路径 $r' = \langle e'_O, \cdots, e'_i, \cdots, e'_D \rangle$，通过地铁关键站点图 G' 与地铁完整站点图 G 的对应关系还原出一条完整路径 $r = \langle e_O, \cdots, e_j, \cdots, e_D \rangle$，最终得到起始站 O 和目的站 D 完整的 K 条最短时间路径集 R_{OD}。

由于给定的两个站点之间的 K 条最短时间路径是确定的，如果每次查找都使用上述方法不仅增加额外查询时间也会造成计算资源的浪费。因此，本文在基于关键站点的 K 最短时间算法基础上通过构建新的哈希表 Htable 辅助查找 K 条最短路径，并将其命名为基于离线关键站点的 K 最短时间算法，具体过程如下：

对于给定的起始站 O 和目的站 D，首先通过哈希函数 $Hash(O,D)$ 查找 Htable。若返回值为空，即未命中，则说明 Htable 还没有保存 O 到 D 之间的 K 条最短时间路径集 R_{OD}，因此需要使用基于关键站点的 K 最短时间算法查找 R_{OD}，并将 R_{OD} 保存到 Htable 中；否则表示命中，直接返回 R_{OD}。

2.2. 基于路径阻抗的路线推荐方法

在上一节获得始发站到目的站的 K 条最短路线集后，通过计算每条路线的路径阻抗对路线进行排序，根据乘客不同的偏好从中选择合适的路线推荐给乘客。其中，路径阻抗是乘客乘坐地铁时间成本和拥挤程度这两个乘客偏好的权衡。本文的路径阻抗 I 的定义如公式(5)，路径阻抗 I 由出行时间成本 T 和拥挤成本 C 构成，w_1 和 w_2 分别表示时间成本和拥挤成本的权重，其中 $w_1 + w_2 = 1$。

$$I = w_1 T + w_2 C \tag{5}$$

为了便于描述又不失一般性，本文做了以下基本假设：

1) 所有列车都准确地按照列车时刻表运行。
2) 列车从进入站台到离开站台的时间为 1 分钟。
3) 所有乘客在换乘站的换乘时间 t^{trans} 相同。
4) 乘客在到达目的站或换乘站之前不得下车。

2.2.1. 出行时间成本

在 2.1 节基于离线关键站点的 K 最短时间算法中，计算乘坐地铁的总出行时间 T' (公式 2)是基于理论上的平均等待时间 T'_{wait}。然而在现实中，乘客等待列车到站的时间有长有短，特别是经过多次换乘后乘客出行的理论时间与实际时间差别很大，所以在计算出行成本的时候使用实际出行时间成本 T (公式(6))。并且随着换乘次数的增加，换乘时间和实际等待的时间也随之线性增加，但是给乘客带了的负面感受却是迅速增加的。因此定义实际出行总时间成本 T，如公式(6)所示。

$$T = t_0^{wait} + T^{travel} + T^{trans} \tag{6}$$

其中 t_0^{wait} 表示乘客刷卡进站后等待第一趟列车到站的实际时间，T^{travel} 表示乘客在列车上的总乘坐时间成本，是由第 i 个列车的乘坐时间 t_i^{travel} 组成的，T^{trans} (公式(7))表示总换乘时间成本。

乘客在换乘时不仅需要步行从一个站台走到另一个站台，并且还需要在站台等待地铁列车的到来，乘客对出行的满意度会迅速衰减，因此本文使用指数形式来表示乘客负面情绪的衰减程度。公式(7)表示 k 个换乘站的总换乘时间成本，其中 $\alpha(\alpha \geq 1)$ 为换乘时间放大因子。

$$T^{trans} = \left(\sum_{i=1}^{k}\left(t^{trans} + t_i^{wait}\right)\right)^{\alpha} \tag{7}$$

为了计算乘客第 i 个换乘站实际等待时间 t_i^{wait}，需要使用列车到站时刻表加以确认。地铁等公共交通系统都按照时间表调度运行的，也就是说，地铁列车会在预定的时间到达和离开站台，因此根据乘客所处的站点以及当前时间从列车时刻表中找到可以乘坐的地铁列车。

定义 N 条线路的线路集合 $lines = \{1,2,\cdots,l,\cdots,N\}$，有 M 个站点的第 l 线的站点集合 $stations = \{1,2,\cdots,i,\cdots,M\}$，$t_{in}$ 表示乘客刷卡进站时间，$A_{l,i}^{j}$ 表示 l 地铁线 i 站第 j 趟列车到达站台的时间，$D_{l,i}^{j}$ 表示 l 线 i 站第 j 趟列车离开站台的时间。对于 l 线的每一趟列车，如果乘客进入 l 线的 i 站的时间 t_{in} 满足公式(8)，那么表示该乘客能够乘坐列车 j。

$$D_{l,i}^{j-1} < t_{in} \leq D_{l,i}^{j} \tag{8}$$

乘客换乘一次的出行线路如图 1 所示，乘客在 t_{in} 时间刷卡进入 l 线的 i 站，等待 t_0^{wait} 时间乘坐最近一趟列车 j，经过 t_1^{travel} 时间列车到换车站 i'，经过 t^{trans} 时间换乘到 i' 线路，等待时间 t_1^{wait} 后，经过 t_2^{travel} 时间到达目的地，行程结束。因此通过这种基于列车到站时刻表的路径搜索可以确定所有实际等待时间 t_i^{wait}。

此外，当在时间 t_{in} 向前无法查找到满足公式 8 的列车 j 时，则说明此时该路段当天已经没有可以乘坐的列车了，即末班车已经过去了，因此在路线推荐时直接放弃该路线。

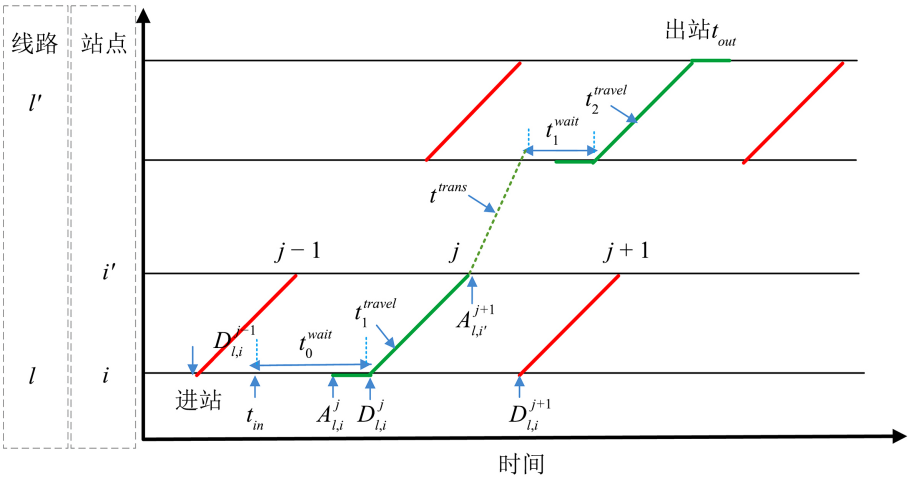

Figure 1. The schematic diagram of passengers changing once to the subway
图 1. 乘客换乘一次地铁示意图

2.2.2. 拥挤成本

不同的乘客对出行时间、拥挤程度有不同的偏好，比如部分乘客对出行时间不敏感，他们反而更关心乘车时的舒适度。某两站之间线路 i 上的地铁车厢内乘客的负面感受与车厢内拥挤程度 c_i 和列车行驶时间呈正 t_i 相关，且随着拥挤度和列车行驶时间的增加，乘客对出行的满意度会迅速衰减，因此一共经过 m 条边路的拥挤总成本 C 的定义如公式(9)所示，其中 $\beta(\beta > 1)$ 是地铁拥挤度放大因子。

$$C_i = \left(\sum_{i=1}^{m} c_i t_i\right)^{\beta} \tag{9}$$

当乘客经过 k 个换乘站、途经 m 条路段的出行线路的最终总路径阻抗 I 如下：

$$I = w_1\left(t_0^{wait} + T^{travel} + \left(\sum_{i=1}^{k}\left(t^{trans} + t_i^{wait}\right)\right)^{\alpha}\right) + w_2\left(\sum_{i=1}^{m} c_i t_i\right)^{\beta} \tag{10}$$

其中 w_1 和 w_2 分别表示时间成本和拥挤成本的权重，并且 $w_1 + w_2 = 1$，α 和 β 分别是换乘时间和地铁拥挤度放大因子，并且 $\alpha \geq 1$，$\beta > 1$。

当 w_1 为 1、w_2 为 0、α 为 1 时，I 表示只考虑出行的总时间，并不考虑换乘次数和拥挤度；当 w_1 为 1、w_2 为 0、α 大于 1 时，此时 I 等于 T，表示只考虑总出行时间成本，考虑到换乘次数，I 弱化为计算总出行时间成本；当 w_1 为 0、w_2 为 1 时，此时 I 等于 C，表明只考虑乘车的舒适程度，I 弱化为计算总拥挤成本；当 w_1 和 w_2 均不为 0 且 $\alpha > 1$ 时，表明同时考虑了时间成本和拥挤成本。

3. 实验与结果分析

本章首先介绍实验使用到的数据，其次理论分析对比两个约数条件下的 K 最短时间算法[6] [15]的时间复杂度，验证本文所提出的基于离线关键站点的 K 最短时间算法的优势，最后通过示例对比分析本文所提出的基于路径阻抗的路线推荐算法的效果。

3.1. 实验数据与预处理

本实验数据主要包括上海一卡通乘客刷卡数据和地铁首末班车时刻表数据。

3.1.1. 地铁网络交通线路数据

本文使用到地铁乘客刷卡数据来自 2015 年 4 月 1 日到 2015 年 4 月 30 日一共 30 天的上海一卡通乘客刷卡数据[17]。每条原始一卡通刷卡数据如表 1 所示，包括卡号、刷卡日期、刷卡时间、公共交通信息、交通方式、交易金额和交易性质。因为本文是针对地铁数据的研究，所以根据交通方式可以筛选出全部的地铁刷卡数据，本文后面使用的乘客刷卡数据指交通方式为地铁的上海一卡通乘客刷卡数据。

Table 1. Smart-card passenger swipe card data
表 1. 一卡通乘客刷卡数据

属性	定义	示例
卡号	乘客标识符	2201252167
刷卡日期	一卡通刷卡日期	2015-04-01
刷卡时间	一卡通刷卡时间	08:55:44
公共交通信息	公交线路、轨道交通线路、出租及轮渡等信息	1 号线陕西南路
交通方式	公交、地铁、出租、轮渡、P + R 停车场	地铁
交易金额	公共交通刷卡价格	4.0
交易性质	是否优惠	非优惠

3.1.2. 地铁首末班车时刻表数据

因为上海地铁线路非常多，由于篇幅和时间的原因，本文选择具有代表性的环路 4 号线、南北方向并且有大小交路的 8 号线和东西方向的 9 号线作为研究对象，并且这三条线路也都有交叉站点，即换乘车站。

上海地铁官网提供了每条线路的首末班车时刻表和列车运行间隔数据，表 2 是上海地铁 8 号线首末班车时刻表部分数据[18]，根据首末班车时刻表可以获得经过相邻两个站点列车运行的时长，并且对于同一条线路，具体分为上行和下行两个方向。

Table 2. Partial schedule of the first and last train of Shanghai Metro Line 8
表 2. 上海地铁 8 号线首末班车部分时刻表

站名	首班车发车时刻		末班车发车时刻	
	往沈杜公路↑	往市光路↓	往沈杜公路↑	往市光路↓
沈杜公路	---	05:30	---	22:30
联航路	06:33	05:32	23:33	22:32
江月路	06:31	05:34	23:31	22:34
…	…	…	…	…
翔殷路	05:33	06:32	22:33	23:32
嫩江路	05:31	06:34	22:31	23:34
市光路	05:30	---	22:30	---

3.1.3. 数据预处理

首先使用原始地铁刷卡数据确定刷卡数据的进出站，然后对乘客刷卡数据无换乘和有换乘两种情况分别提取地铁乘客出行链路数据，再根据提取的列车到站时间表将每条出行链路数据划分到具体时间具体线路的列车上，以此获得地铁线路上列车的人数，最后经过数据处理得到地铁车厢的拥挤度。

3.2. 基于离线关键站点的 K 最短时间算法时间复杂度的分析

图 2 是由上海地铁 4 号线、8 号线和 9 号线构成的关键站点图，其中矩形表示换乘站。值得注意的是 4 号线与 8 号线共有的换乘站只有陆家浜站和西藏南路站，4 号线上环路与 8 号线没有相交。

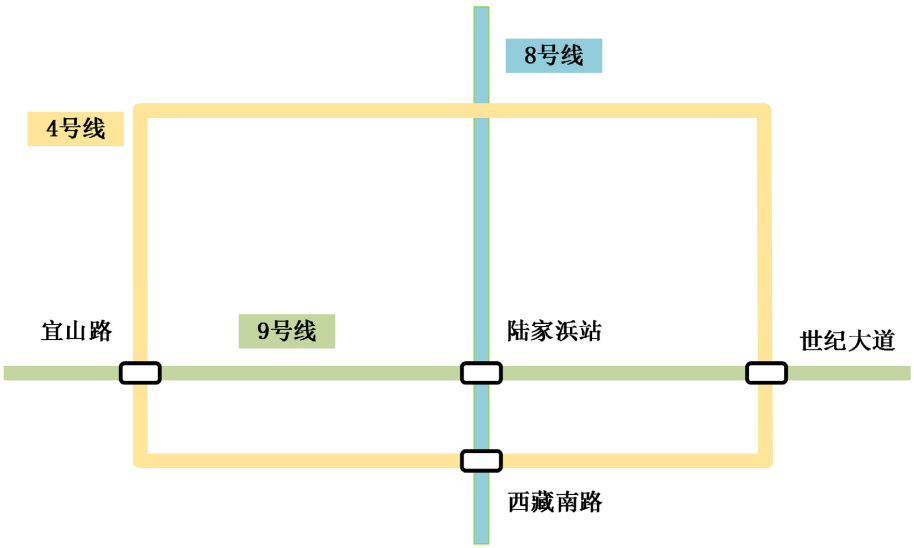

Figure 2. Map of key subway stations
图 2. 地铁关键站点图

设 K 为需求路径数，n 和 n' 分别为地铁网络总站点数和换乘站数，m 为一条路径的平均路段数，m' 为换乘站点组成的关键路段数，P 表示需要查询的列车数量。

本文提出的基于离线关键站点的 K 最短时间算法记为算法 A，因为算法 A 通过构建哈希表 Htable 辅助查找 K 条最短路径，因此分为命中和未命中两种情况进行讨论。对于给定的起始站 O 和目的站 D，当第一次执行算法 A 查找 O 到 D 的 K 条最短路径 R_{OD} 时，由于哈希表里面没有 R_{OD}，此时为查找未命中，需要使用基于关键站点的 K 最短时间算法得到 R_{OD}，并将 R_{OD} 保存到哈希表中，则平均时间复杂度为 $O\left(K\log_{n'}\left(m'+n'\log n'\right)\right)$；当再次查找 R_{OD} 时，因为可以在哈希表直接获取，此时为查找命中，则算法的时间复杂度为 $O(1)$。因此，算法 A 的时间复杂度可以表示为 $\left\{O\left(K\log_{n'}\left(m'+n'\log n'\right)\right),O(1)\right\}$。

文献[15]提出的 K 条可行路径算法记为算法 B，则算法 B 的前两步分别使用进站时间和出站时间向前和向后搜索地铁网络和列车时刻表，从而获得两个不同的中间路径集，每步时间复杂度都是 $n'\log_P$。第三步是将两个不同的中间路径集相交形成可行路径，从中选出 K 条时间最短时间路径集，复杂度是 $O(Kn')$。因此，算法 B 进行一次查找的平均时间复杂度为 $O\left(Kn'+2n'\log_P\right)$。

文献[6]提出的基于进出站时间约束的 K 最短时间算法记为算法 C，则算法 C 包括两个阶段，第一阶段是进行 K 最短时间算法，K 最短时间算法的时间复杂度为 $O\left(Km'n'\log_{n'}\right)$。第二阶段是根据列车时刻表检查路径有效性。因此，算法 C 一次查找的时间复杂度为 $O\left(Km'n'\log_{n'}+Kn'\log_P\right)$。

算法 B 和算法 C 都使用起始站点、进站时间、目的站点、出站时间和列车到站时间数据作为约束条件，因此通过上述方法降低各自算法的时间复杂度。然而在现实的路线推荐系统中，乘客通常只需要输入起始站点和目的站点即可，进站时间默认为乘客点击查询操作的时间，因此本文提出的算法约数条件只有起始站点、进站时间和目的站点，比算法 B、C 更加人性化。

Table 3. Comparison of the time complexity of the three algorithms
表 3. 三种算法时间复杂的对比

算法	时间复杂度
算法 A	$\left\{O\left(K\log_{n'}\left(m'+n'\log n'\right)\right),O(1)\right\}$
算法 B	$O\left(Kn'+2n'\log_P\right)$
算法 C	$O\left(Km'n'\log_{n'}+Kn'\log_P\right)$

同时由于算法 B 和算法 C 在查找过程中引入了列车到站时间数据，因此每次的查找结果不一定一致，因此无法将查找的路径保存。从表 3 可以看出，本文提出的基于离线关键站点的 K 最短时间算法在未命中情况下的时间复杂度要低于算法 C，但是高于算法 B，但这是由约束条件不一样导致的。并且随着查询次数的增加，算法 B 和 C 的平均时间复杂度不变，而本文提出的算法 A 的命中率则越来越高，因此算法 A 的平均时间复杂度越来越低。当地铁任意两个站点的 K 条最短路径集都保存到 Htable 后，则本文提出的基于离线关键站点的 K 最短时间算法的时间复杂度为 $O(1)$，此时，该算法的时间复杂度要远远低于算法 B 和 C。

3.3. 基于路径阻抗的路线推荐算法的示例分析

本小节选择上海地铁 8 号线的成山路和 4 号线的大连路作为起始站和目的站，使用本章预处理得到的拥挤度数据和提出的基于路径阻抗的路线推荐算法进行路线推荐。

实验设 K 为 3，w_1 和 w_2 分为 0.4 和 0.6，α 和 β 都为 2。先使用基于离线关键站点的 K 最短时间算

法找到 3 条最短时间路径，分别命名为路线 1、路线 2 和路线 3，结果如图 3~5 所示，其中实线代表出行路径，为了方便描述，使用地铁关键站点图进行展示。然后计算三条最短时间路径的路径阻抗，结果如表 4 所示，其中途经线路表示该路线顺序经过的线路，比如 8-4 表示路线 1 途经 8 号线和 4 号线。其中路线 1、路线 2 和路线 3 与算法 B 和算法 C 找到的 3 条最短时间路径相同。

图 3~5 中红色和绿色圆形分别表示起始站和目的站，其中黑色矩形表示换乘站，需要注意的是 4 号线上环路与 8 号线没有相交。图中线路的颜色表示该路段平均拥挤程度的大小，其中绿色代表不拥挤，其取值范围[0, 0.33)，其中黄色表示拥挤程度适中，其取值范围[0.33, 0.66)，其中红色代表拥挤，其取值范围[0.66, 1]，并且颜色越深表示拥挤度越大。

Table 4. The relevant attribute values calculated for the three routes
表 4. 三条路线计算的相关属性值

路线	途经线路	途经站点	出行时间(分钟)	出行时间成本	拥挤成本	路径阻抗
路线 1	8-4	12	36.84	81.23	47.32	60.88
路线 2	8-9-4	11	42.12	99.55	26.22	55.55
路线 3	8-4	21	62.84	107.23	132.52	122.40

路线 1 的出行时间和出行时间成本都是最小值，但是西藏南路到世纪大道非常拥挤，并且拥挤路段要大于成山路到陆家浜站，因此拥挤成本比路线 2 要大。路线 2 换乘次数最多，因此出行时间成本比路线 1 大，但是由于路线 2 所途经的线路平均拥挤度相对较小，因此其拥挤成本最小，所以路径阻抗也最小。因为路线 3 途经站点最多，其出行时间和出行时间成本都是最大值，虽然路线 3 经过的线路拥挤度不是最大的，但是其累加的拥挤成本也是最大的。

因此，当乘客偏好最短的出行时间时(偏好 1)，以出行时间作为推荐基准，则推荐出行路线 1；当乘客选择最不拥挤的出行方式时(偏好 2)，以拥挤成本为基准，推荐出行路线 2；当乘客希望尽可能少换乘的时候(偏好 3)，以途经线路个数(即换乘次数)为基准，推荐出行路线 1；当乘客希望在拥挤与出行时间之间折中的时候(偏好 4)，以路径阻抗为基准，推荐出行路线 2。

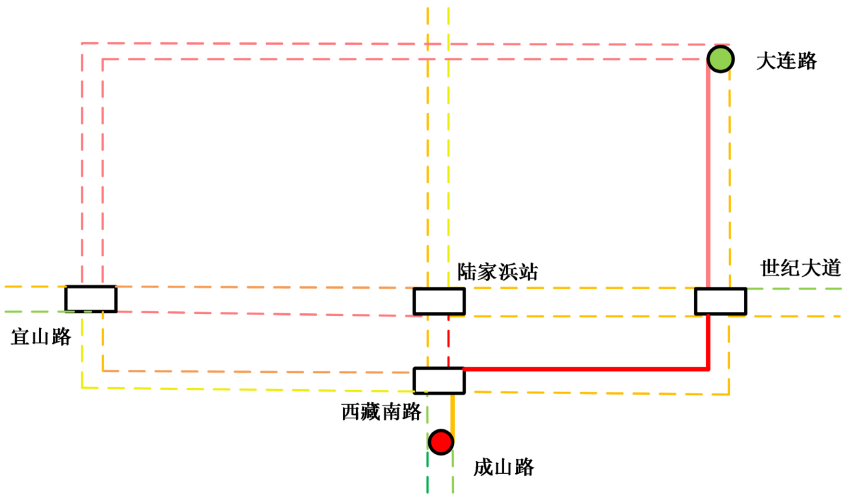

Figure 3. Travel route 1
图 3. 路线 1

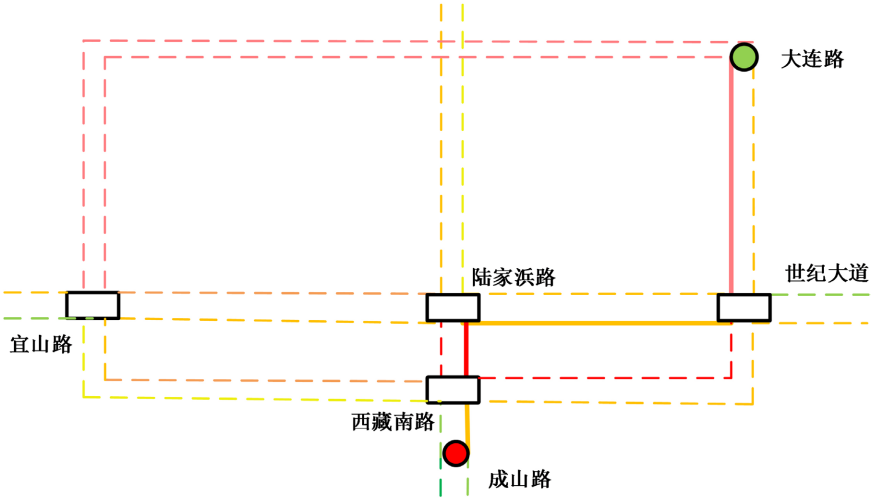

Figure 4. Travel route 2
图 4. 路线 2

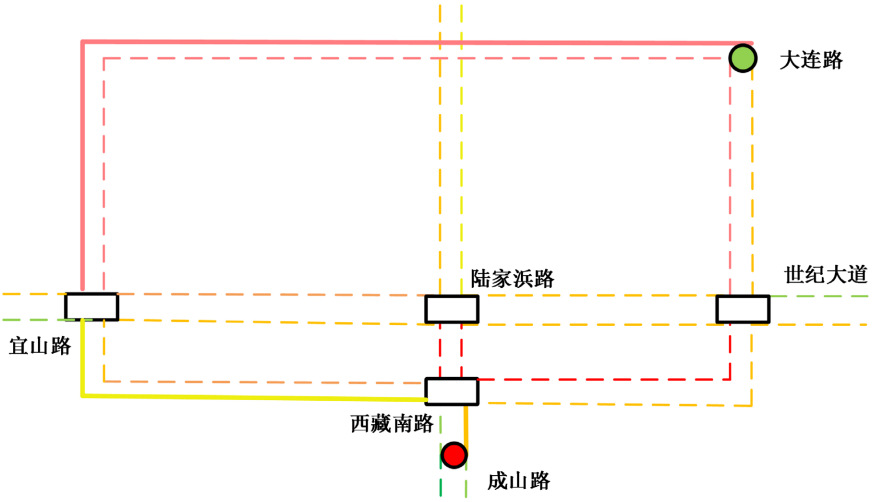

Figure 5. Travel route 3
图 5. 路线 3

文献[15]提出的路线推荐方法只能给出偏好 1、偏好 2 和偏好 3 的推荐路线，因为偏好 4 的路线推荐依赖于本文提出的路径阻抗，是推荐给那些在地铁拥挤程度与出行时间之间折中的乘客。同时，该路线推荐算法与本文提出的路线推荐算法根据偏好 1 和偏好 3 给出的推荐路线一致。

但对于偏好 2 的推荐路线，前者使用每条路线最大路段的拥挤度的作为该路线的拥挤成本，并从中选择拥挤成本最小的路线推荐给乘客，因为路线 1 的西藏南路到陆家浜路和路线 2 的西藏南路到世纪大道的拥挤度均大于路线 3 每个路段的拥挤度，因此推荐路线 3。然而实际中，这种方法欠缺合理性，虽然路线 3 途经路段不是最拥挤的，但相对还是较为拥挤，并且路线 3 途经站点最多、出行时间最长，这种长时间比较拥挤路段的后期乘客的负面感受会迅速增加，这种负面感受要强于只有一小段非常拥挤、其他路段(如路线 2)不拥挤带来的负面感受。而本文提出的路线推荐发方法考虑到乘客对出行的负面感受与换乘次数及拥挤度间并不是简单的线性关系，随着换乘次数和拥挤度的增加，乘客对出行的满意度会迅速衰减，因此在实际生活中要更加合理。

4. 总结

本文首先使用基于离线关键站点的 K 最短时间算法，通过构造关键站点图降低算法的时间复杂度，此外通过构建哈希表辅助查找 K 条最短路径，进一步降低了时间复杂度，并且随着查询次数的增多，本文提出的基于离线关键站点的 K 最短时间算法时间复杂度要远小于其他约数条件下的寻径算法。其次设计了基于路径阻抗的地铁路线推荐方法，由乘客乘坐地铁时间成本和拥挤成本共同组成路径阻抗，最后通过计算 K 条最短时间路径的路径阻抗，从中选出符合乘客偏好的乘车路线推荐给乘客，通过实验示例可以看出该方法可以满足不同乘客的路线推荐要求。

上述工作虽然取得了一定成果，但仍存在不完善之处，后续相关研究，拟从以下两个方面展开：

1) 使用上海地铁的所有线路进行研究。由于时间和篇幅的原因，本文只使用了具有代表性的上海地铁 4 号线、8 号线和 9 号线作为研究对象。但只考虑了部分线路无法代表完整地铁网络的时空特性，因此在后续的研究中需要使用完整的地铁线路。

2) 个性化路线推荐。本文提出的基于离线关键站点和路径阻抗的地铁路线推荐方法只是一个通用型的方法，其参数值需要根据乘客的出行偏好预先设定。然而，这并不符合乘客的使用习惯，因此在后续工作中可以根据每个乘客的历史出行数据从中自动学习路线推荐方法的参数，从而实现个性化路线推荐。

基金项目

国家自然科学基金面上项目(61872043)资助，河南省高等学校重点科研项目(19A520043)资助，计算机体系结构国家重点实验室开放课题(CARCHA202103)资助。

参考文献

[1] Du, B., Cui, Y., Fu, Y., Zhong, R. and Xiong, H. (2018) Smart Transfer: Modeling the Spatiotemporal Dynamics of Passenger Transfers for Crowdedness-Aware Route Recommendations. *ACM Transactions on Intelligent Systems and Technology*, **8**, Article No. 70. https://doi.org/10.1145/3232229

[2] Dijkstra, E.W. (1959) A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, **1**, 269-271. https://doi.org/10.1007/BF01386390

[3] Hart, P.E., Nilsson, N.J. and Raphael, B. (1968) A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, **4**, 100-107. https://doi.org/10.1109/TSSC.1968.300136

[4] Yen, J.Y. (1971) Finding the *K* Shortest Loopless Paths in a Network. *Management Science*, **17**, 712-716. https://doi.org/10.1287/mnsc.17.11.712

[5] Xu, W., He, S., Song, R. and Chaudhry, S.S. (2012) Finding the K Shortest Paths in a Schedule-Based Transit Network. *Computers & Operations Research*, **39**, 812-1826. https://doi.org/10.1016/j.cor.2010.02.005

[6] Zhou, F. and Xu, R. (2012) Model of Passenger Flow Assignment for Urban Rail Transit Based on Entry and Exit Time Constraints. *Transportation Research Record*, **2284**, 57-61. https://doi.org/10.3141/2284-07

[7] Guo, J. and Jia, L. (2017) A New Algorithm for Finding the K Shortest Paths in a Time-Schedule Network with Constraints on Arcs. *Journal of Algorithms & Computational Technology*, **11**, 170-177. https://doi.org/10.1177/1748301816680470

[8] Jariyasunant, J., Mai, E. and Sengupta, R. (2011) Algorithm for Finding Optimal Paths in a Public Transit Network with Real-Time Data. *Transportation Research Record*, **2256**, 34-42. https://doi.org/10.3141/2256-05

[9] Yang, Y., Wang, S., Hu, X., Li, J. and Xu, B. (2012) A Modified K-Shortest Paths Algorithm for Solving the Earliest Arrival Problem on the Time-Dependent Model of Transportation Systems. *Lecture Notes in Engineering & Computer Science*, **2196**, 1562-1567.

[10] Wang, S., Yang, Y., Hu, X., Li, J. and Xu, B. (2016) Solving the *K*-Shortest Paths Problem in Timetable-Based Public Transportation Systems. *Journal of Intelligent Transportation Systems*, **20**, 413-427. https://doi.org/10.1080/15472450.2015.1082911

[11] Jeon, I., Nam, H. and Jun, C. (2018) A Schedule-Based Public Transit Routing Algorithm for Finding K-Shortest Paths Considering Transfer Penalties. *The Journal of the Korea Institute of Intelligent Transport Systems*, **17**, 72-86. https://doi.org/10.12815/kits.2018.17.3.72

[12] Delling, D., Goldberg, A.V., Pajor, T. and Werneck, R.F. (2017) Customizable Route Planning in Road Networks. *Transportation Science*, **51**, 566-591. https://doi.org/10.1287/trsc.2014.0579

[13] Kim, K.M., Hong, S.P., Ko, S.J. and Kim, D. (2015) Does Crowding Affect the Path Choice of Metro Passengers? *Transportation Research Part A*: *Policy and Practice*, **77**, 292-304. https://doi.org/10.1016/j.tra.2015.04.023

[14] Li, W., Luo, Q., Ca, I.Q. and Zhang, X. (2018) Using Smart Card Data Trimmed by Train Schedule to Analyze Metro Passenger Route Choice with Synchronous Clustering. *Journal of Advanced Transportation*, **2018**, Article ID: 2710608. https://doi.org/10.1155/2018/2710608

[15] Li, W., Luo, Q. and Cai, Q. (2020) A Smart Path Recommendation Method for Metro Systems with Passenger Preferences. *IEEE Access*, **8**, 20646-20657. https://doi.org/10.1109/ACCESS.2020.2969075

[16] Fredman, M.L. and Tarjan, R.E. (1987) Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM*, **34**, 596-615. https://doi.org/10.1145/28869.28874

[17] SODA 大赛一卡通乘客刷卡数据[EB/OL]. https://shanghai.sodachallenges.com/, 2021-03-24.

[18] 上海地铁 8 号线首末班车时刻表[EB/OL]. http://service.shmetro.com/hcskb/250.htm, 2015-03-14.