基于客户分级及换乘的多车辆合乘问题算法研究

孟春华[1,3] 王洪国[2,3] 邵增珍[2,3] 于洪玲[1,3] 丁艳辉[2,3]
(山东师范大学管理科学与工程学院 济南 250014)[1] (山东师范大学信息科学与工程学院 济南 250014)[2]
(山东省分布式计算机软件新技术重点实验室 济南 250014)[3]

摘 要 多车辆合乘匹配问题(MRMP)是物流领域和交通领域的一个重要问题,现有的多车辆合乘匹配算法是以解决基本的多车辆合乘问题为主。为了提高客户的搭乘率,提出了客户分等级并且带有换乘的多车辆合乘匹配算法。该算法以蚁群优化算法为核心,分为3步:寻找起点终点集合;蚁群寻优,并在单向蚁群的基础上提出双向蚁群算法;车辆路径微调。实验仿真显示该算法获得80%以上的搭乘率,同时双向蚁群比单向蚁群具有更强的寻优能力。所得结果表明,该算法可以有效地获得带有换乘的匹配路线。

关键词 多车辆合乘匹配问题,蚁群算法,换乘,客户分级
中图法分类号 TP18 **文献标识码** A

Algorithm Research on Multi-vehicle Ride Matching Problem Based on Passengers Classification and Transfers

MENG Chun-hua[1,3] WANG Hong-guo[2,3] SHAO Zeng-zhen[2,3] YU Hong-ling[1,3] DING Yan-hui[2,3]
(Management Science and Engineering, Shandong Normal University, Jinan 250014, China)[1]
(Information Science and Engineering, Shandong Normal University, Jinan 250014, China)[2]
(Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan 250014, China)[3]

Abstract Multi-vehicle ride matching problem(MRMP) is an important problem in the field of logistics and transportation. The existing MRMP algorithms mainly solve the basic MRMP. To improve the riding rate, an algorithm for the MRMP with transfers and passengers classifing was proposed. The algorithm with ant colony optimization algorithm as the core is divided into three steps: finding the sets of start-point and end-point; ant colony optimizes the routes, and two-way ant colony is proposed based on one-way ant colony, triming the vehicle routes. Simulation experiment shows that the algorithm can achieve riding rate of 80%. And the two-way ant colony shows stronger optimization ability than one-way ant colony. The results show clealy that the algorithm can find the matching routes effectively.

Keywords Multi-vehicle ride matching problem, Ant algorithm, Transfers, Passengers classifing

1 引言

交通拥堵历来都是许多国家道路交通的大问题,随着人们生活水平的提高以及城市人口的增加,城市中机动车数量急剧增加,使得城市交通拥堵状况日趋严重。为了应对交通拥堵问题,"搭便车"的现象应运而生,在网上也可以看到很多拼车信息的发布平台供出行者选择搭车对象;与此同时,也出现了为用户出行随时随地提供搭车信息的简单的手机终端软件,但是这些都是单纯地人工进行拼车路径信息的筛选,因此对于车辆合乘问题中车辆路径选择算法的研究显得尤为重要。

针对车辆合乘问题,对支持换乘的多车辆合乘匹配模型研究得较少。国内的研究中,李玲、谷寒雨[1]研究了复杂PDPTW问题,并提出插入启发式算法用于解决车辆路径规划问题;贾永基等人[2]用快速禁忌搜索算法解决PDPTW问题。国外的Cristian E[3]提出利用分支定界法解决具有转乘特征的PDP问题;Sophie N[4]提出了基于异质用户的dial-a-ride问题解决方案;Lauir Hame[5]提出适应性插入算法解决窄时间窗口的单车dial-a-ride问题;Herbawi等人[6]提出一类多目标的动态多跳车辆合乘问题。对于蚁群算法,先前也有很多利用其来解决车辆路径问题的算法,如国内李文勇[7]的公交路径蚂蚁算法、柯梁军等[8]提出的利用改进的蚁群算法解决带时间窗的团队定向问题,国外有S. R. Balseiro[9]提出的基于插入启发的蚁群算法来解决带有时间窗的车辆路径问题,它们都证明了蚁群算法具有较高的寻优能力,但是还没有将蚁群算法应用到车辆合乘问题的先例。

到稿日期:2012-11-05 返修日期:2013-04-15 本文受山东省自然科学基金项目(ZR2011FQ029,ZR2011FL026),山东省科技发展计划项目(2011YD01099,2011YD01100),山东省高等学校科技计划项目(J11LG32)资助。
孟春华(1989—),女,硕士,主要研究方向为人工智能、物流优化,E-mail:meng1662@163.com;王洪国(1966—),男,博士,教授,主要研究方向为电子政务、最优化理论;邵增珍(1976—),男,博士,副教授,主要研究方向为物流优化、智能算法;于洪玲(1988—),女,硕士,主要研究方向为物流优化;丁艳辉(1979—),男,博士,主要研究方向为数据挖掘。

邵增珍等[10]提出了一个两阶段聚类的算法，解决了带有时间窗的静态的多车辆合乘匹配问题，本文在此基础上进行了进一步的研究，在合乘的过程中，增加了支持换乘的机制，并引入了第二等级的客户，即不考虑客户上下车时间点的时间窗口约束，针对这一等级的客户对先期车辆的匹配路线进行调整以满足需求，增加换乘之后可以以更大的概率满足不同客户的需求，并利用以蚁群为核心的算法来解决这种带有换乘的车辆合乘二次匹配的问题。

2 带有换乘的 MRMP 问题模型

2.1 问题描述

某区域有 m 辆车组成车队 F，可在满足自身出行需要的基础上提供搭乘服务，区域内有 n 个搭乘服务需求。设车辆的品牌、空余容量及行驶速度各不相同，车辆出行信息及服务需求信息确定且保持不变，车辆司机、搭乘服务需求的上下车点确定且对应固定的服务时间窗口。目标是将 n 个乘客尽可能多地合乘到 m 辆车上。这就是基本的多车辆合乘匹配问题。

本文提出的基于客户分级和换乘特点的多车辆合乘匹配问题是指在基本的 MRMP 的基础上，考虑搭乘服务需求的异构性，这种异构性导致了服务需求的优先级不同，在这里将客户分为两个等级。第一等级的客户属于基本的多车辆合乘匹配问题研究的范畴，第二等级客户的特点为可以换乘而且无严格的上下车时间窗口约束，即能够在一定的时间内到达即可。要解决的便是在前期车辆合乘阶段车辆的固定路线的基础上，寻找第二等级客户的换乘路线，其中换乘点是在先前路线中的交叉点中搜寻。该换乘点不是固定的，因此不会因为增加固定换乘点而产生不灵活性[3]。

2.2 问题形式化及符号定义

假设车辆数量为 m，乘客总数量为 n，车辆路径的交叉点数目为 k，具有第一优先级的乘客数量为 $n-d$，剩余的具有第二优先级的乘客的数量为 d。设 F 为车辆集合，$F=\{1,2,\cdots,m\}$；P 为乘客集合，$P=\{1,2,\cdots,n\}$；D 为第二优先级的客户的集合，$D=\{1,2,\cdots,d\}$。

设客户点 i 的起点集合 $S_i=\{s_{i1},s_{i2},\cdots,s_{ik1}\}$（$i$ 为客户点的编号，$k1$ 为满足约束的车辆的个数）。客户点 i 的终点集合 $E_i=\{e_{i1},e_{i2},\cdots,e_{ik2}\}$（$i$ 为客户点的编号，$k2$ 为满足约束的车辆的个数）。

F^+ 为车辆的上车点集合，F^- 为车辆的下车点集合，P^+ 为乘客上车点集合，P^- 为乘客下车点集合，V 为所有车辆路径上的点的集合。Tr 为车辆路径交叉点集合，$Tr=\{1,2,\cdots,k\}$。

Q_k 为车辆 k 的剩余容量，$k\in F$。t_{ij}^k 为第 k 辆车从点 i 行驶到点 j 所需要的时间。

F_T^i 表示客户 i 的换乘次数；T_{tr}^j 表示车辆 j 到达换乘点 tr 处的时间。

2.2.1 目标函数定义

该问题的目标是在保证搭乘成功的前提下，尽可能地降低成本。系统搭乘客户 i 的成本可分为两部分：一部分是所有车辆运行总成本中由客户 i 所分摊的那部分成本，另一部分是由于换乘导致的等待成本。

$$Cost_i^f=\sum_{j\in F}(ac_j\cdot(\sum_{x\in V}\sum_{y\in V\setminus\{x\}}X_{x,y}^{i,j}\cdot a_{x,y})) \tag{1}$$

式中，$Cost_i^f$ 是指客户 i 所分摊的行驶成本，$X_{x,y}^{i,j}$ 为二进制变量，$X_{x,y}^{i,j}=1$ 时，说明车辆 j 从点 x 直接行驶到点 y 时搭乘了顾客 i，而 $X_{x,y}^{i,j}=0$ 则说明没有搭乘。单位里程内每增加一位乘客增加成本为 ac_j，$a_{x,y}$ 是 x 点到 y 点的距离。

$$Cost_i^t=\sum_{tr\in Tr}(\overline{ac}\cdot(\bar{v}\cdot(T_{tr}^{j_2}-T_{tr}^{j_1})))\cdot IFT_i^{tr} \tag{2}$$

式中，$Cost_i^t$ 是指顾客 i 由于换乘导致的成本。$\overline{ac}$ 是所有车辆在单位里程每增加一位乘客所增加的平均成本，$\bar{v}$ 是指所有车辆的平均速度，$\bar{v}\cdot(T_{tr}^{j_2}-T_{tr}^{j_1})$ 模拟了在换乘点的车辆行驶距离。IFT_i^{tr} 是二进制变量，$IFT_i^{tr}=1$ 说明顾客 i 在换乘点 tr 处换乘，$IFT_i^{tr}=0$ 说明没有换乘。

该问题的目标就是使成本最低，即

$$\mathrm{Min}(Cost_i^f+Cost_i^t) \tag{3}$$

2.2.2 约束条件

1）车主以及第一等级客户的时间窗口约束（此约束为最初的基本约束，将排除掉肯定不符合时间窗口约束的交叉点）

车辆 j 必须在规定时间窗口内到达上、下车点，否则服务需求无法在规定时间得到满足。有

$$e_x\leqslant T_x^j\leqslant l_x,x\in P,j\in F \tag{4}$$

特殊情况下会出现车辆 j 早于时刻 e_x 到达顶点 x 的情况，处理方法见前文所述，则可修正为

$$T_x^j\leqslant l_x,x\in P,j\in F \tag{5}$$

2）车辆的剩余运量约束

行驶过程中，任意时刻不能超过车辆容限。有

$$q_x^j=initq_j,x\in F^+,j\in F \tag{6}$$

$$q_x^j=initq_j,x\in F^-,j\in F \tag{7}$$

$$q_x^j\leqslant Q_j,x\in P,j\in F \tag{8}$$

其中，式(6)表示车辆 j 从起点出发时的原有载货量；式(7)表示车辆到达终点时所有搭乘全部完毕，只剩余原有载货量，$initq_x$ 为常量；式(8)表示车辆在搭乘过程中总货品数量不能超过车辆搭载容限。

3）访问顺序约束

任何搭乘服务需求的上车时间加上乘车时间，应在该搭乘服务需求的下车时间之前。有

$$T_x^j+RT_x\leqslant T_{m+n+x}^{j'},x\in \mathrm{N}^+,j,j'\in F \tag{9}$$

4）任意顾客的换乘次数的约束

设换乘次数不超过 r，则

$$F_T^i\leqslant r \tag{10}$$

5）换乘点的时间窗口

一方面是第二等级客户在换乘点的等待时间约束。对于乘客来说，在换乘点等待车辆是很烦心的，因此，虽然该类客户在上下车点没有时间窗口约束，但是在换乘点，需要给客户一个等待的时间保障，即等待时间不能超过设定的最大等待时间 T_d，即

$$T_{tr}^{j_2}-T_{tr}^{j_1}\leqslant T_d^i,i\in D,j_1,j_2\in F \tag{11}$$

式中，j_1 代表换乘之前客户搭载的车辆编号，j_2 代表换乘之后客户搭载的车辆编号，利用该式进行判断的前提是车辆 j_2 在车辆 j_1 之前到达换乘点。

若车辆 j 在车辆 i 之后到达换乘点，则客户不需要等待，需要车辆进行等待，车辆进行等待的时间要满足式(4)或

式(5)中其他客户的时间窗口约束。

3 求解支持换乘的 MRMP 问题的蚁群算法

蚁群算法是一种用来解决复杂的组合优化问题的随机算法。蚂蚁通过觅食过程中留下的信息素来相互通信，蚂蚁根据找到的食物源的质量的好坏来决定释放的信息素的多少，而后面的蚂蚁也会根据信息素的多少来选择信息素多的路径。利用蚂蚁觅食的原理，人们设计了蚁群算法来对已知的起、终点的最短路径问题进行求解[7]。Bullnheimer、Hartl ant Stauss(1999)[11]证明了在车辆路径问题中，蚁群算法与其他的亚启发式算法如禁忌搜索算法、模拟退火算法等相比具有更强的竞争性。蚁群算法中最重要的两个部分——信息素更新策略和选择策略，会在后面根据本文的问题进行详细的设计。

基于蚁群解决该问题的算法的流程如图 1 所示。

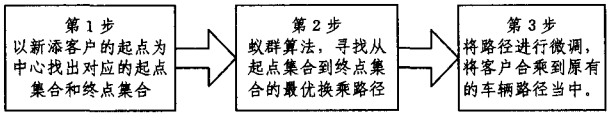

图 1 算法流程

这 3 步在寻找每一个客户点的换乘路线时是串行进行的，以防止路线之间出现混乱的情况。图 2 为一个寻找路线的示意图。

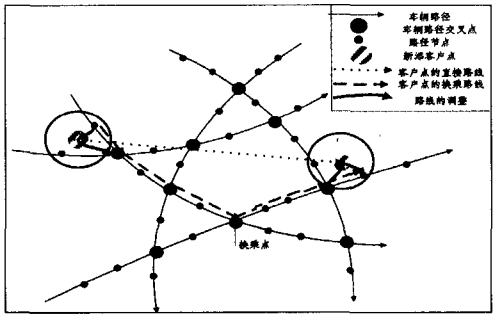

图 2 寻找路线示意图

3.1 算法第 1 步：找出起点和终点集合

客户点要顺利换乘，有时不需要调节原有的车辆路线，但是很多情况下需要对原有路线进行微调才能将该换乘点搭载上。这个问题可以从客户点的角度出发，以客户点的起点或者终点为中心，比如说以客户点的起点为中心从该起点出发，寻找在该起点周围满足时间约束和运力约束的车辆路线，然后对每一条车辆路线都找出一个从起点出发到车辆路线路程最短的那个点，有多少条满足约束的车辆路线，就会有多少个这样的点，将这些点定义为该客户点的起点集合：$S_i=\{s_{i1}, s_{i2}, \cdots, s_{ik}\}$($i$ 为客户点的编号，k 为满足约束的车辆编号)，然后以客户点的终点为中心，按照同样的方法找出一些距离最近的点，将这些点定义为该客户点的终点集合：$E_i=\{e_{i1}, e_{i2}, \cdots, e_{ik}\}$($i$ 为客户点的编号，k 为满足约束的车辆编号)。

在寻找起点集合和终点集合时，运用一个距离聚类，设置一个系统能够接受的距离参数 di，以起点或者是终点为中心，寻找周围距离该点在 di 以内的所有的在原有车辆路线上的点作为下一步蚁群寻优的蚁穴和食物源。

3.2 第 2 步：蚁群算法寻找最优换乘线路

利用蚁群算法将起点的集合作为蚁群算法中蚁群的蚁穴，终点的集合作为蚁群算法的食物源。此时，所有的车辆路径形成该子问题的路网，只要让蚂蚁从起点集合中的一点出发找到一条相对较优的路径到终点集合中的任意一点即可，其中要考虑的因素有：车辆的时间窗、车辆上客户的时间窗、车辆的搭载能力约束。这里的最优是指运行的时间最短(不能用距离来衡量，因为辆车的速度是不一样的)、换乘次数最少。本文提出两种蚁群寻优的算法并对两者进行了比较，一种是单向蚁群即所有蚂蚁都从起点集合出发向终点集合寻找路线；另一种是双向蚁群，即将蚂蚁分成两部分分别从起点集合和终点集合出发寻找最优路线。

3.2.1 单向蚁群

将等量的蚂蚁数平均放到起点集合中的各个起点上，蚂蚁在选择出行路径时，从所在的起点出发，根据从起点所能搭载的 n 条车辆路径(若起点是交叉点，则可以搭乘多条线路)的信息素强度，利用一定的随机策略选择第 i 条，继续前进到达下一个站点，并判断是否达到食物源(即终点集合中的任意一点)。若该只蚂蚁到达终点成功，则改变经过的线路的激素强度(计算所通过路径的距离、换乘的次数、对于车辆以及其他服务需求的时间窗的影响)；若还没有到达，则继续向前查找，直到搜寻到食物源为止。

利用蚁群算法解决的步骤如下：

单向蚁群算法步骤

Initialization

构造信息素矩阵，并将同等数量 m 只的蚂蚁分别放在该客户点的起点集合 S 中的各个点上；

Repeat

  Repeat

    从 S 中的一个起点开始，蚂蚁按照转移规则移动到下一点，并将该点加入到相应蚂蚁的禁忌表中；

    若换乘次数达到 r，则该只蚂蚁自然死亡；

    若有蚂蚁到达终点集合 E 的任意一点，则记录该路线并更新最优解，进入下一个起点的寻优；

  Until 所有起点的蚂蚁完成环游；

  根据信息素修正规则，更新路径的信息素；

Until 满足迭代次数 Nc 的约束；

3.2.2 双向蚁群

将蚂蚁分成两部分，一部分放到起点集合，另一部分放到终点集合，即从起点站和终点站同时进行搜索，直到有两只蚂蚁在某一点相遇为止。

步骤如下：

双向蚁群算法步骤

Initialization

构造信息素矩阵，并将同等数量 m/2 只的蚂蚁分别放在该客户点 i 的起点集合 Si 和终点集合 Ei 的各个点上；

Repeat

  Repeat

    从 Si 和 Ei 出发的蚂蚁按照转移规则同时移动到下一点，并将该点加入到相应蚂蚁禁忌表中；

    若换乘次数达到 r，则该只蚂蚁自然死亡；

    若从 Si 出发的蚂蚁与从 Ei 出发的蚂蚁所在的站点有交集，或者是从 Si(Ei)出发的蚂蚁到达 Ei(Si)，则更新最优解，并根据信息素修正规则，更新路径的信息素；

  Until 所有蚂蚁最终能完成搜索或者没有路径可走；

Until 满足迭代次数 Nc 的约束；

3.2.3 蚁群信息素更新和转移概率

在这两种蚁群算法中，都涉及到蚂蚁信息素的更新以及蚂蚁路径选择的转移概率，由于蚂蚁寻找路径时参照的不是路径之间的距离而是时间，因此对信息素更新策略以及转移概率做出了改进。由以下方法给出：

(1)信息素的更新

蚂蚁进行路径选择时，应该考虑两个主要因素：换乘次数和出行时间，由这两个因素给出路径信息素的更新规则：

当蚂蚁到达终点集合中的任意一点时，蚂蚁行走过的路线的信息素强度的增加值为：

$$\Delta R=R_{old}\lambda\left(\frac{n-f}{n}\right)^{\alpha 1}\left(1+\frac{1}{\sqrt{t^{\beta 1}}}\right)$$

- λ 为选择车辆路线的重要程度，主要是根据车辆路线的剩余运力情况以及时间窗的约束情况等来综合确定，一般取值为 0.8～1.2；
- n 为最大换乘次数；
- f 为蚂蚁本次出行所需要的换乘次数；
- $\alpha 1$ 为换乘次数对路径选择的影响系数；
- t 为蚂蚁本次出行所需要的时间；
- $\beta 1$ 为出行时间对路径选择的影响系数；
- R_{old} 为路线之前的信息素的强度。

当蚂蚁完成一次觅食时，沿途中未经过的线路的信息素的强度会随着时间的推移而逐渐减弱。可选取信息素的消散率为 $\rho=10\%\sim30\%$。经过一段时间，所有路线的信息素都会有一定程度的消散。

所以信息素的更新公式为：

$$R_{new}=\begin{cases}(1-\rho)R_{old}+\Delta R, & \text{蚂蚁经过的路径}\\(1-\rho)R_{old}, & \text{蚂蚁未经过的路径}\end{cases}$$

(2)转移概率

蚂蚁在运行的过程中不是盲目进行转移的，对于蚂蚁已经走过的路径来说，是不可行的，可以对蚂蚁建立行走的禁忌表 T_k，而对于可行的路径集合，蚂蚁需要根据一定的概率进行转移。转移概率由下式给出：

$$p_{ij}(k)=\begin{cases}\dfrac{R_{ij}^{\alpha}\times\eta_{ij}^{\beta}}{\sum\limits_{h\notin T_k}R_{ih}^{\alpha}\times\eta_{ih}^{\beta}}, & j\notin T_k\\0, & \text{其他}\end{cases}$$

- $p_{ij}(k)$ 为第 k 只蚂蚁选择边(i,j)的概率；
- R_{ij} 为边(i,j)的信息素的大小；
- η_{ij} 为边(i,j)的能见度，它的值由一个贪婪算法得到，即鼓励蚂蚁走局部的最优路径，这里 $\eta_{ij}=1/t_{ij}$；
- α 为信息素大小对转移概率的影响系数；
- β 为能见度的大小对转移概率的影响系数。

3.3 第 3 步：路径微调

3.3.1 微调的基本方法

将蚁群算法找出的最优路径进行微调，即将客户的起点和终点合乘到相应的车辆上，得到最终的合乘解。

例如起点是 s、终点是 e 的客户 u 通过蚁群算法得到的换乘路径是〈r_1-r_2-r_3-r_4-r_5-r_6-r_7(换乘点)-r_8-r_9〉，该路径中起点和终点分别属于起点集合 S 和终点集合 E，即起点 r_1 和终点 r_9 并不一定是起点 s 和终点 e。若〈r_1-r_2-r_3-r_4-r_5-r_6-r_7(换乘点)〉属于第 i 辆车上的路径，〈r_7(换乘点)-r_8-r_9〉属于第 j 辆车的路径，则客户需要先乘坐第 i 辆车到换乘站 r_7 下车，再乘坐第 j 辆车到达终点。若设定客户的起点和终点是不能移动的，即需要车主稍微改变原来路线，在不影响第一类客户需求的情况下到达客户的起点或者终点，所以就需要将搭载的第一辆车和最后一辆车的路径进行微调(若有多个换乘点，中途的车辆不需要改变原有路径)。

微调时，由于该类客户不考虑时间窗口约束，先期只需根据距离对路径进行插入排序即可。但是在该插入排序的过程中，要注意其中的第一类客户点的需求，路线微调的过程中，不能将客户点调走，同时微调出来的路线要满足各个客户的时间窗口约束。

首先，将第一个换乘点之前的路径所在的车辆找出来，然后将客户的上车点添加到该辆车的路线当中。方法：以第一个换乘点为界，对换乘点之前的车辆上的路径，先把路径中的客户点找出来，将这些客户点和换乘点组成新的序列，然后将所要搭乘的上车点插入到这个序列当中，并找到代价最小的排列方式，排列完成之后，将这些节点插入到原路线当中，便可得到关于该辆车的一个新的路线。例如第 i 辆车原先的路径为：〈i_1-i_2-r_1-r_2-r_3-r_4-r_5-r_6-r_7-i_3-i_4〉，经过微调后得到的路线为：〈i_1-i_2-s-r_1-r_2-r_3-r_4-r_5-r_6-r_7-i_3-i_4〉(下划线的部分是该客户走的路径)，这样就将客户起点顺利匹配到了第 9 辆车中。

最后的换乘点，用同样的方法将所要搭乘的客户的下车点微调到最后换乘点后的路径所在的车辆路线当中。例如第 j 辆车原先的路线为：〈j_1-r_7-r_8-r_9-j_2-j_3-j_4-j_5-j_6〉，经过调整后的路线为：〈j_1-r_7-r_8-r_9-j_2-e-j_2-j_3-j_4-j_5-j_6〉。

而中间的换乘点则无需考虑线路微调，只考虑换乘造成的时间代价是否满足客户的时间窗口的约束。

这样就可以得到该客户的最终的路线：〈s-r_1-r_2-r_3-r_4-r_5-r_6-r_7(换乘)-r_8-r_9-j_2-e〉。

3.3.2 可行性检测

在生成最终的解的路线之后需要对解的可行性进行检测，可行性检测主要包括车主以及第一等级客户的时间窗口的检测、第二等级客户在换乘点的等待时间约束，还有车辆在行驶过程中的剩余容量是否满足客户的搭载容量需求。

(1)对于车主和第一等级客户，将两类都并为第一等级的客户，检测时应对换乘路线中所涉及的所有车辆都进行约束检测，对第 j 辆的车辆路线 $r=\langle r_0\text{-}r_1\text{-}r_2\text{-}r_3\text{-}\cdots\text{-}r_m\rangle$，从起点 r_0 的时间窗口$[er_0,lr_0]$中随机选择一个时间段进行对于第一等级客户的时间窗口的检测。由于车的车速 v_j 是固定的，因此可以轻松地计算出从一个客户点到下一个客户点的时间是 $D_{p,p+1}$，设从一个客户点 p 出发的时间是 d_{tp}，那么到达下一个客户点的时间是：

$$at_{p+1}=d_{tp}+D_{p,p+1}/v_j$$

对于客户点 $p+1$ 来说，应该满足时间窗口约束：

$$at_{p+1}\leqslant lr_{p+1}$$

即到达时间应该早于 $p+1$ 等待的最晚时间。若到达时间比时间窗的最早时间还要早，那么在 $p+1$ 点的客户应该等到时间窗的上限才能出发。

(2)对于在换乘点处的时间约束，因为搭乘的前后两辆车

到达换乘点有一定的时差，若是前一辆早到，那么客户有一个等待时间，若是后一辆车早到，那么后一辆车需要等待。对于客户的等待，需要设置一个客户心理能够接受的时间限制 TL，若需要等待的时间超过客户所设置的限制，该路线不能被采用。对于后一辆车的等待，要考虑到后面所有客户点的时间窗口要求，若等待的时间过长达不到路径中后面的客户的要求，则该路线也不能被采用。

(3)对于剩余容量的检测，由于客户在中途有上车有下车，因此需要在每个客户点对剩余容量进行判断，以保证下一个客户点能够被成功搭载。同时换乘点也是客户的上车点和下车点，所以在判断车辆剩余容量时，需要考虑换乘点的容量的变化。

3.4 算法复杂度分析

在上述算法的3个步骤当中，第1步找起点集合和终点集合的复杂度为 $2*O(n)$。蚁群算法中，初始化参数的复杂度为 $O(n^2+m)$，设置蚂蚁的禁忌表的复杂变为 $O(m)$，每只蚂蚁单独构造解复杂度为 $O(n^2m)$，解的评价和轨迹更新量的计算的复杂度为 $O(n^2m)$，信息素轨迹浓度的更新的复杂度为 $O(n^2)$，判读是否到达算法的终止条件(达到设定的最大循环次数 Nc)的复杂度为 $O(nm)$，输出计算结果的复杂度为 $O(1)$。当 n 足够大时，低次幂的影响可以忽略不计，整个蚁群算法的复杂度为 $O(Nc*n^2*m)$，最后一步的路线调整的复杂度为 $2*O(n\log n)$。所以整个算法的复杂度为 $2*O(n)+O(Nc*n^2*m)+2*O(n\log n)\leqslant O(n^4)$，说明算法为多项式算法，具有可接受的计算复杂度。

4 实验结果展示

4.1 实验数据

表1 车辆初始信息表

车辆编号	起点/终点 时间窗	平均速度	固定成本 & 可变成本	容量 & 初始载货量	已搭客户集合 & 初始路径
1	1,[1,10] 63,[30,40]	50	50 5	500 100	{1,2,3} 1,2,10,13,12, 37,40,63
2	4,[5,14] 74,[30,40]	50	60 5	600 150	{4,5} 4,5,19,28,72,74
3	9,[2,11] 84,[35,45]	50	60 6	600 150	{6,7,8}/ 9,11,37,39,80, 64,82,84
4	6,[15,25] 93,[40,50]	60	70 7	700 200	{9,10,11} 6,17,26,31,74, 75,92,93
5	7,[10,20] 77,[35,45]	60	70 7	700 200	{12,13,14} 7,14,23,32,47, 52,70,77
6	3,[8,18] 86,[40,50]	60	80 8	800 250	{15,16,17} 3,10,8,13,43,105, 68,81,86
7	14,[7,17] 95,[40,50]	60	80 8	800 250	{18,19,20} 14,15,34,45,54, 67,97,95
8	12,[2,12] 64,[15,25]	70	90 9	900 300	{21,22} 12,36,42,59,61,64
9	5,[10,20] 39,[20,30]	70	90 9	900 300	{23,24,25} 5,18,16,33,36,37,39
10	18,[14,24] 82,[35,40]	70	100 10	1000 400	{26,27,28} 18,19,30,51,91, 90,81,82

本实验的基础数据是基于文献[9]中的数据和实验结果进行研究的，以济南交通路网为例，假设该地域中有 $m=10$ 辆可提供搭乘服务的车辆，并假设有 $n=30$ 位第一等级的客户，在一次匹配中，已经有28位客户成功被搭载，在这个基础上，每次增加一位第二等级的客户，本次实验共连续添加5位第二等级的客户。实验的内容包括：通过算法找到最优的换乘路径；两种蚁群算法性能的比较。实验基于如下环境：Intel(R)Celeron(R)CPU E3300，2.50GHz，4G 内存，操作系统为 Windows XP Professional。车辆信息以及初始路径信息、客户的信息表见表1和表2。除车辆和客户的上下车点之外，该实验将各个客户点之间的连接点也添加在内，该路网由105个点组成。

表2 客户信息表

客户编号	上车点/下车点 & 时间窗	客户等级	需求量	客户编号	上车点/下车点 & 时间窗	客户等级	需求量
1	2,[5,10]; 12,[20,25]	1	150	19	34,[15,20]; 67,[25,30]	1	90
2	10,[10,15]; 37,[25,30]	1	120	20	45,[20,25]; 97,[30,35]	1	80
3	13,[15,29]; 40,[30,35]	1	80	21	36,[1,5]; 59,[10,15]	1	110
4	5,[10,15]; 28,[18,25]	1	90	22	42,[5,10]; 61,[10,25]	1	150
5	19,[15,20]; 72,[25,30]	1	100	23	18,[10,15]; 36,[20,25]	1	110
6	11,[5,10]; 82,[30,35]	1	160	24	16,[15,20]; 37,[25,30]	1	140
7	37,[5,10]; 80,[25,30]	1	130	25	33,[15,20]; 39,[25,30]	1	120
8	39,[10,15]; 64,[25,30]	1	70	26	19,[15,20]; 51,[25,30]	1	110
9	17,[15.20]; 92,[35,40]	1	90	27	30,[20,25]; 90,[30,35]	1	90
10	26,[20,25]; 74,[30,35]	1	80	28	91,[30,35]; 81,[35,40]	1	100
11	31,[25,30]; 75,[35,40]	1	110	29	34,[45.50]; 53,[60,66]	1	150
12	14,[10,15]; 47,[25,30]	1	130	30	6,[20,26]; 61,[39,55]	1	160
13	23,[15,20]; 70,[30,35]	1	90	31	20[无]; 58[无]	2	50
14	32,[20,25]; 52,[30,35]	1	100	32	14[无]; 41[无]	2	40
15	10,[10,15]; 105,[2,25]	1	80	33	24[无]; 102[无]	2	50
16	13,[15,20]; 68,[30,35]	1	120	34	13[无]; 82[无]	2	40
17	43,[20,25]; 81,[35,40]	1	140	35	27[无]; 79[无]	2	30
18	15,[10,15]; 54,[20,25]	1	100				

4.2 参数设置

距离参数 di 取的是数据库中点与点之间的距离(该距离不是直线距离，而是实际站点相连的距离)的第一个四分位数，这样能够找到距离比较合适的起点和终点的集合。

蚁群算法的参数没有固定的最优值，而且蚁群参数的最优值随着问题的不同会产生变化[12]，所以不能将前人所用的最佳的参数组合用到本问题中。本试验对蚁群参数中的 $Nc(5\sim20)$、$m(5\sim20)$、$\rho(0.1\sim0.3)$、$\alpha(0.5\sim2)$、$\beta(0.8\sim3)$ 的选取是通过多次实验确定的相对较优的组合。

实验中参数的设置如表 3 所列。

表 3　参数设置

参数名称	参数值	参数名称	参数值
di	150	α1	0.8
m	15	β1	2
ρ	0.15	λ	(0.8～1.2)
α	0.8	f	1
β	0.5	Nc	20
TL	30		

4.3　实验结果分析

实验结果如表 4－表 6 所列，表 4 展示了单向蚁群和双向蚁群寻优的性能的比较，表 5 展示了新添客户路径微调后解的结果，表 6 从车辆的角度出发展示了车辆路线的变化。

表 4　单向蚁群和双向蚁群寻优的性能的比较

客户编号	单向/双向最优解	单向/双向换乘次数	单向/双向运行时间(s)
31	17,16,15,33,23,35,36,42,56/17,16,15,33,23,35,36,42,56	1/1	528/495
32	8,13,35,36,42/8,13,35,36,42	2/2	1860/1320
33	26,31,51/26,31,51	0	2690/2360
34	12,36,42,56,57,59,60,61,64,80/12,36,42,56,57,59,60,61,64,80	1/1	2960/2255
35	26,31,51,91,90,79/26,31,51,91,90,79	1/1	2771/1990

从表 4 中可以看出，单向和双向蚁群寻优的结果是一样的，这是由于蚁群参数中蚂蚁数目以及迭代次数设置得充分大，足够蚁群每次都能找到最优的换乘路径。但是从运行时间上可以看出，双向蚁群的运行效率要比单向蚁群高，运行的时间有一定程度的缩减。

表 5　第二等级客户路径微调后的解

客户编号	客户解的路线	换乘点	搭载车辆	总成本
31	20,16,33,36,42,58	36	9－8	5430
32	无解			
33	6,17,24,26,31,74,75,102	无	4	5534
34	13,36,42,41,58,59,61,64,80,82,	64	8－3	5892
35	27,24,26,31,51,91,90,79,	51	4－10	5690

表 5 中显示第 32 号客户无解，是因为换乘次数限制最大换乘一次，但是蚁群寻优的结果中是两次，所以该客户不能被搭乘。而且即使换乘次数不超过两次，该客户点被换乘到第 8 辆车的时候，并不能满足第 8 辆车的时间窗口约束。

下面通过客户 31 的换乘路径来描绘一下换乘之前与换乘之后的路线的调整，如图 3 所示，客户点 31 的上车点为 20，下车点为 58。由蚁群算法得出客户 31 的最优的换乘路径为：〈17,16,15,33,23,35,36,42,56〉，并得知客户可以先搭乘第 9 辆车，并在站点 36 处进行换乘，然后搭载第 8 辆车。由于问题假设客户的上下车点固定不变，因此需要对第 8 辆车和第 9 辆车的路线进行微调，以便于搭载到客户点。

经过微调之后得到关于客户的完整的路线：〈20,17,16,15,33,23,35,36,42,56,57,58〉。对于第 9 辆车来说，搭载之前的路线为：〈5,18,17,16,15,33,23,35,36,37,38,39〉，微调后的路线为〈5,18,19,20,17,16,15,33,23,35,36,37,38,39〉。第 8 辆车微调前的路线为〈12,36,42,56,57,59,60,61,64〉，搭载微调后的路线为：〈12,36,42,56,57,58,57,59,60,61,64〉。

表 6　车辆路径的变化

车辆编号	初始路径(搭乘客户集合)	加入客户 31 的路径	加入客户 32 的路径	加入客户 33 的路径	加入客户 34 的路径	加入客户 35 的路径
1	1,2,10,13,12,37,40,63	无变化	无变化	无变化	无变化	无变化
2	4,5,19,28,72,74,	无变化	无变化	无变化	无变化	无变化
3	9,11,37,39,80,64,82,84	无变化	无变化	无变化	无变化(但是搭乘)	无变化
4	6,17,26,31,74,75,92,93	无变化	无变化	6,17,24,26,31,74,75,102,92,93	无变化	6,17,27,24,26,31,51,74,75,102,92,93
5	7,14,23,32,47,52,70,77	无变化	无变化	无变化	无变化	无变化
6	3,10,8,13,43,105,68,81,86	无变化	无变化	无变化	无变化	无变化
7	14,15,34,45,54,67,97,95	无变化	无变化	无变化	无变化	无变化
8	12,36,42,59,61,64	12,36,42,58,59,61,64	无变化	无变化	12,13,36,42,58,59,61,64	无变化
9	5,18,16,33,36,37,39	5,18,20,16,33,36,37,39	无变化	无变化	无变化	无变化
10	18,19,30,51,91,90,81,82	无变化	无变化	无变化	无变化	18,19,30,51,91,90,79,81,82

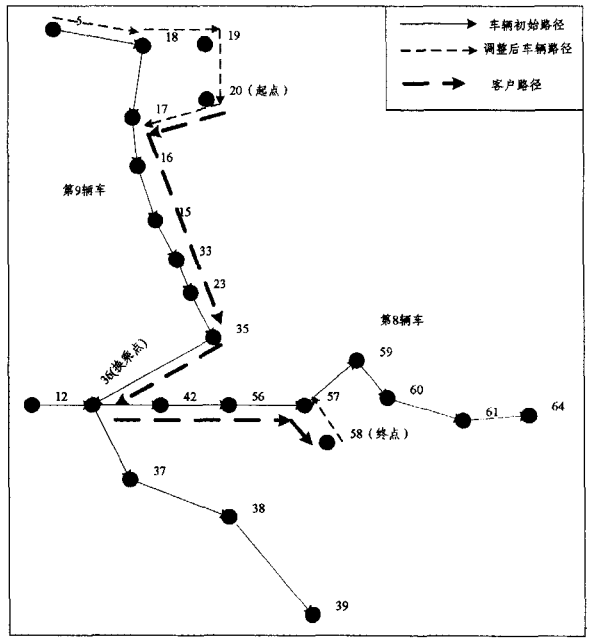

图 3　31 号车换乘路径

结束语　本文研究了带有换乘的并且客户分等级的多车辆合乘匹配问题，并提出了基于蚁群算法的 3 步寻优的解决方案：寻找起点和终点集合；蚁群寻优；路线调整。最后通过实验证明该算法能够解决二次匹配的问题，并有较好的搭乘效果，导致搭乘失败的主要原因是车辆原先路线的时间和容量的约束。对于未来的研究可以从以下两个方面入手：(1)本

(下转第 261 页)

600×600 时，加速比就已经超过了 10 倍，因此当原始矢量场分辨率较高时，本文实现的算法比 FastLIC 更快。

相对于 M. Zöckler 等人的并行 LIC 对硬件的较高要求，本文实现的算法具有相对较好的通用性。

然而，从图 8 中也可以看到，当矢量场分辨率为 100×100 时，算法的加速比小于 1，GPU 的应用没能提高程序的效率。这是由于在执行核心程序之前要将核心程序所需要的数据从 CPU 的内存中复制到显示芯片的内存中，而在最后计算完成后又要将计算结果从显示芯片的内存复制到 CPU 的内存。在数据规模较小、本来的计算时间很短的情况下，数据来回复制的过程便占据了很大部分时间，使得应用 GPU 之后的效率反而有所下降。

从上文的图示和分析可知，基于 CUDA 架构的 GPU 加速 LIC 对于分辨率较小的矢量场（如采样点在 40000 个以下的矢量场）加速效果不明显甚至没有，但随着数据分辨率的增大，加速比逐渐增加，当数据分辨率达到一定时（如采样点在 360000 个以上），加速比超过 10，比 FastLIC 更快。因此，本文实现的 LIC 加速算法更适合于高分辨二维矢量场的可视化，且没有特别高的硬件要求，通用性较好。

结束语 本文首先改进了基本颜色映射 LIC 法，针对用户感兴趣区域矢量场强度比较接近的问题，利用非线性颜色增强 LIC 法进行可视化，使得在矢量场大小比较集中的区域也能区分出矢量的大小。接着利用 NVIDIA 的 CUDA 架构实现了 LIC 算法的 GPU 加速，加速后的算法尤其适用于高分辨率二维矢量场的交互式可视化，且对硬件的要求不高，通用性较好。本文所实现的改进算法还只适用于二维平面空间上的矢量场，对于三维空间场的情况尚没有进行研究。后续的研究工作可以针对三维空间上的 LIC 算法展开，以满足更多的应用需求。

参 考 文 献

[1] Laramee R S, Hauser H, Doleisch H, et al. The state of the art in flow visualization: dense and texture-based techniques[J]. Computer Graphics Forum, 2004, 23(2): 203-221

[2] Stalling D, Hege H. Fast and resolution independent line integral convolution[C]// Proceedings of the ACM SigGraph'95. New York ACM SIGGRAPH, 1995: 249-256

[3] Van Wijk J J. Spot noise: texture synthesis for data visualization [J]. Computer Graphics, 1991, 25(4): 309-318

[4] Cabral B, Leedom C. Imaging vector fields using line integral convolution[J]. Computer Graphics, 1993, 27(4): 263-272

[5] Leeuw W D, Liere R V. Comparing LIC and spot noise[C]// Proceedings of IEEE Visualization'98, IEEE Computer Society. 1998: 359-365

[6] 陆剑锋，潘志庚，张明敏. 基于图像对比度数量映射的矢量场可视化算法[J]. 系统仿真学报，2004，16(7)：1502-1505

[7] 张文耀，蒋凌霜. 基于 HSV 颜色模型的二维流场可视化[J]. 北京理工大学学报，2010，30(3)：302-306

[8] 陈丽娜. 基于线积分卷积可视化矢量场大小的研究[J]. 陕西理工学院学报：自然科学版，2009，25(3)：50-52，64

[9] 朱宏玮，姜国华，王宝智. 矢量场可视化线积分卷积方法研究与系统设计[J]. 计算机应用与软件，2010，27(4)

[10] Zöckler M, Stalling D, Hege H. Parallel line integral convolution [J]. Parallel Computing, 1997, 23(7): 975-989

[11] Hlawatsch M, Sadlo F, Weiskopf D. Hierarchical line integration [J]. IEEE Transactions on Visualization and Computer Graphics, 2011, 17(8): 1148-1163

[12] NVIDIA Corporation. NVIDIA CUDA C programming guide [EB/OL]. http://developer.nvidia.com/nvidia-gpu-computing-documentation, 2012-07-06

（上接第 242 页）

文中提出的支持换乘的多车辆合乘匹配问题还处在相对静态的阶段，即没有加入道路的信息，未来可以在该多车辆合乘匹配模型中加入交通流的影响，实现动态的多车辆合乘匹配；(2)对于算法的改进，该算法中的参数组合有待进一步的提高，需要大量的实验对算法进行优化。

参 考 文 献

[1] 李玲，谷寒雨，陈坚. 复杂 PDPTW 问题的插入启发式算法[J]. 计算机工程，2003，16(2)：65-69

[2] 贾永基，谷寒雨，席裕庚. 求解 PDPTW 问题的一种快速禁忌搜索算法[J]. 控制与决策，2004(01)

[3] Cristián E, Cortés. The pickup and delivery problem with transfers Formulation and a branch-and-cut solution method[J]. European Journal of Operational Research, 2010, 200(3): 711-724

[4] Parragh S N. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem [J]. Transportation Research Part C-emerging Technologies, 2011, 19(5): 912-930

[5] Häme L. An adaptive insertion algorithm for the single-vehicle dial-a-ride problem with narrow time windows [J]. European Journal of Operational Research, 2011, 209(1): 11-22

[6] Herbawi W, Weber M. Comparison of multiobjective evolutionary algorithms for solving the multiobjective route planning in dynamic multi-hop ridesharing[A]// IEEE CEC[C]. New Orleans, June 2011

[7] 李文勇，王炜，陈学武. 公交出行路径蚂蚁算法[J]. 交通运输工程学报，2004(04)

[8] 柯良军，章鹤，尚可，等. 一类求解带时间窗的团队定向问题的改进蚁群算法[J]. 计算机科学，2012，39(4)：214-216

[9] Balseiro S R. An ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows[J]. Computers & Operations Research, 2011, 38(6): 954-965

[10] 邵增珍，王洪国，刘弘，等. 多车辆合乘问题的两阶段聚类启发式优化算法[J]. 计算机研究与发展，2013

[11] Bullnheimer B, Hartl R F, Strauss C. An Improved Ant system Algorithm for the Vehicle Routing Problem[J]. Annals of Operations Research, 1999, 89(10): 319-328

[11] 黄永青，梁昌勇，张祥德. 基于均匀设计的蚁群算法参数设定[J]. 控制与决策，2006，21(1)