第 45 卷 第 10 期 计 算 机 工 程 2019 年 10 月
Vol. 45 No. 10 Computer Engineering October 2019

·开发研究与工程应用· 文章编号：1000-3428(2019)10-0308-06 文献标志码：A 中图分类号：TP18

一种求解共享单车再平衡问题的遗传算法

刘喜梅，潘立军

（湖南工程学院 管理学院，湖南 湘潭 411104）

摘 要： 共享单车再平衡问题（BRP）是单一商品旅行商问题（1-PDTSP）的扩展，是一类 NP 难问题。针对已有算法求解速度慢，不利于实现实时调度优化的缺点，提出一种求解 BRP 的非代际遗传算法。基于个体搜索机制保留优异个体，设计线路交叉算子和 k 点破坏修复变异算子，引入破坏修复机制，当算法收敛变慢时自动生成新个体进入种群以避免陷入局部最优解。应用 BRP 标准算例测试表明：在小规模算例上该算法均能找到最优解，平均 CPU 消耗为 3.8 s；在中等规模与大规模算例上，该算法找到 9 个算例的最优解，并且其运算速度相较于分支定界算法和线路破坏与修复启发式算法提升 77% 以上。

关键词： 车辆路径问题；共享单车再平衡问题；遗传算法；线路交叉；破坏修复变异

开放科学（资源服务）标志码（OSID）：

中文引用格式： 刘喜梅，潘立军. 一种求解共享单车再平衡问题的遗传算法[J]. 计算机工程，2019，45(10)：308-313.

英文引用格式： LIU Ximei，PAN Lijun. A genetic algorithm for solving bike-sharing rebalancing problem[J]. Computer Engineering，2019，45(10)：308-313.

A Genetic Algorithm for Solving Bike-sharing Rebalancing Problem

LIU Ximei，PAN Lijun

（School of Management，Hunan Institute of Engineering，Xiangtan，Hunan 411104，China）

【Abstract】 The Bike-sharing Rebalancing Problem (BRP) is an extension of the Single Commodity Traveler Problem (1-PDTSP) and a type of NP-hard problem. To address the slow solution speed of the existing algorithms, which is not conducive to real-time scheduling optimization, a non-intergenerational genetic algorithm for solving BRP is proposed. The excellent individuals are retained based on the individual search mechanism, and the line crossover operator and k-point destroy and repair mutation operator are designed. The destroy and repair mechanism is introduced, and when the convergence speed of the algorithm slows down, new individuals are automatically generated into the crowd to avoid falling into the local optimal solution. Computational results on the benchmark problems show that the algorithm can find the optimal solution for the small-scale benchmark problems with an average CPU consumption of 3.8 s. For the medium-scale and large-scale benchmark problems, the algorithm finds the best solutions for 9 benchmark problems, and the speed is improved by over 77% than those of B&C and D&R algorithms.

【Key words】 Vehicle Route Problem (VRP); Bike-sharing Rebalancing Problem (BRP); genetic algorithm; line cross; destroy and repair mutation

DOI： 10.19678/j.issn.1000-3428.0054037

0 概述

城市共享单车在方便居民短距离出行、缓解城市交通拥堵等方面发挥着重要作用。由于单车用户用车具有随机性、单向性等特点，导致各停放点单车数量分布不均，需定期组织货车回收、再投放单车，平衡各停放点单车数量。随着我国各城市共享单车系统规模的扩大，单车再平衡工作量与日俱增，设计开发共享单车再平衡智能调度软件以实现高效的共享单车再平衡，已成为共享单车运营主体降本增效、提升服务质量的关键。

共享单车再平衡过程可描述为：利用具有相同载重的车辆从单车维护与补给中心出发，完成一定区域内各停放点单车的再分配，使各停放点达到预

基金项目： 湖南省自然科学基金（2019JJ60038）；湖南省双一流应用特色学科工商管理资助项目（湘教通[2018]469 号）。

作者简介： 刘喜梅（1978—），女，讲师、硕士，主研方向为物流系统优化；潘立军，副教授、博士。

收稿日期： 2019-02-27 **修回日期：** 2019-04-03 **E-mail：** liuxm006@163.com

先设置的容量后回到中心。在此过程中,运输车辆可以在各停放点调配运送单车,也可以在出发前从维护与补给中心载入一部分单车调配到各停放点,或将各停放点多余的单车运回维护与补给中心。而共享单车再平衡问题(Bike-sharing Rebalancing Problem,BRP)即为在共享单车再平衡过程中,对所有的车辆求解最短的行驶距离或最少的行驶时间。

共享单车在世界各地的普及,使 BRP 问题受到广泛关注。在国外的研究主要将 BRP 模型分为静态和动态 2 类进行求解。

静态 BRP 模型基于假设:在车辆开始服务各停放点后,各停放点单车的平衡数量不发生变化。文献[1]提出该类模型并对其求解;文献[2]设计分枝定界法求解该模型;文献[3]研究目标函数为用户不满意度、车辆行驶时间最少的 BRP 模型,并利用分阶段启发式算法及 CPLEX 求解包含 200 个停靠点的 BRP;文献[4-5]对 BRP 进行深入研究,不仅构建了 4 种 BRP 数学模型、标准测试算例,还运用破坏修复启发式算法求解包含 500 个停靠点的 BRP;文献[6]研究只有一辆运输车辆且允许对停靠点进行多次访问的 BRP,利用可变邻域搜索技术设计启发式算法,并对包含 500 个停靠点的 BRP 求解。

由于在共享单车再平衡过程中,用户仍然在使用单车,各停靠点单车平衡数量具有动态性,因此需要建立动态 BRP 模型。文献[7]运用时间序列化方法,以时长 T 为周期将动态 BRP 模型划分为若干个静态 BRP 模型来求解,根据用户服务的分布函数确定每一个周期各停靠点的单车平衡数量,并将优化目标设置为最小化各点单车平衡数量的不匹配度;文献[8-9]应用在线处理方法求解动态 BRP 模型,在运输车辆服务完某一停靠点后,实时更新其他停靠点的取送量信息进行下一步决策。

目前,国内对 BRP 尚缺少系统研究,已报道的多为案例研究,如文献[10]构建基于用户满意度的多目标共享单车调度模型并采用遗传算法对其求解,同时以南京市江宁区的实际算例进行验证;文献[11]通过预测各停靠点的取送车需求量,运用动态调度优化策略建立系统的动态调度模型,设计模拟退火遗传算法对其求解,并以西安市雁塔区的数据进行验证;文献[12]引入停靠点取送车的时间窗约束、多调度中心约束条件,建立 BRP 模型并运用混合粒子群算法对其求解。此外,文献[13-15]也对 BRP 问题进行了研究。

国外学者较系统地研究了 BRP 问题的特点及经典启发式算法,但求解速度与质量有待进一步提升,而国内研究多为案例研究,研究成果难以推广应用。考虑到遗传算法对车辆路径问题(Vehicle Route Problem,VRP)具有较好的求解效果,例如:文献[16-18]分别使用遗传算法求解了周期性、开放式和多目标 VRP,取得了较好的效果,本文采用遗传算法并结合破坏修复启发式算法的启发规则来求解 BRP。

1　BRP 数学模型

根据图论的定义,将 BRP 表示为一个完备图 $G=(V,E)$,其中,$V=\{1,2,\cdots,n\}$ 表示顶点集,且 1 表示单车维护与补给中心,$V'=\{2,3,\cdots,n\}$ 表示单车停放点集,$E=\{(i,j)\mid i,j\in V,i\neq j\}$ 表示弧集或边集。本文符号定义如下:M 表示实施再平衡运输的车辆数,Q 表示运输车辆的最大载重量,d_i 表示车辆在第 i 个单车停放点取送量(取值范围为 $[-Q,Q]$,$d_i<0$ 表示该停放点需补充单车,$d_i>0$ 表示该停放点要回收单车),c_{ij} 表示车辆访问弧 (i,j) 产生的运输成本或时间,θ_j 表示车辆访问第 j 个停放点后的载重量(即车辆线路载重量)。定义决策变量 $x_{ij}=1$ 表示车辆访问了弧 (i,j),否则,$x_{ij}=0$。BRP 的数学模型如下:

$$\min F(x)=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij} \tag{1}$$

s. t.

$$\sum_{i=1}^{n}x_{ij}=1,j\in V' \tag{2}$$

$$\sum_{i=1}^{n}x_{ji}=1,j\in V' \tag{3}$$

$$\sum_{j=1}^{n}x_{1j}\leqslant M \tag{4}$$

$$\sum_{j=2}^{n}x_{1j}=\sum_{j=2}^{n}x_{j1} \tag{5}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}x_{ij}\leqslant|S|-1,S\subseteq V',S\neq\varnothing \tag{6}$$

$$\theta_j\geqslant(\theta_i+d_j)x_{ij},i\in V,j\in V' \tag{7}$$

$$\theta_i\geqslant(\theta_j-d_j)x_{ij},i\in V',j\in V \tag{8}$$

$$\max(0,d_j)\leqslant\theta_j\leqslant\min(Q,Q+d_j),j\in V \tag{9}$$

式(1)为目标函数,最小化运输总成本或时间;式(2)和式(3)确保除维护与补给中心点外其余单车停放点均被访问且只访问一次;式(4)和式(5)确保所有的运输车辆在完成任务后均回到维护与补给中心;式(6)为消除子回路约束;式(7)~式(9)为 BRP 车辆载重约束,确保各运输车辆在实施单车回收与投放过程中,车辆线路载重量不超过额定载重量,其假设在某一停放点回收的单车可被投放到任意其他需要的停放点,且在车辆出发或者回到补给中心时,载重可不为 0(因为可以事先装入部分单车,或者带回部分单车)。

2　求解 BRP 的遗传算法

2.1　算法基本框架

基本的遗传算法是基于种群的并行搜索机制,本文采用基于个体搜索机制的非代际遗传算法[19],该机制能在进化的过程中更好保留优异个体的同时兼顾种群的多样性,算法描述如下:

1)初始化种群 Pop。

2)若终止条件不符合:

3)按染色体的适应度选择 2 个染色体 x、y。

4)依交叉概率 P_{cross} 对 x、y 实施交叉算子,生成 2 个新个体 x'、y'。

5)依变异概率 P_{mut} 对 x'、y' 实施变异算子,生成 2 个新个体 x''、y''。

6)更新种群,将 x''、y'' 替换 *Pop* 中 2 个适当度最差的染色体。

7)算法结束,返回种群中适应度值最大的染色体。

2.2 算法的编码与初始解生成

本文算法采用多维整数编码方法,1,2,…, n 代表单车停放点位置编号,其中编号 1 代表单车维护与补给中心。每一基因位除包含单车停放点位置编号外,还包含以下信息:对应的线路编号信息,该基因位在线路中的排列顺序,当前线路中该停放点之后位置再插入点时对应单车容量的上限(U_i)与下限(D_i)。染色体编码示意图如图 1 所示。

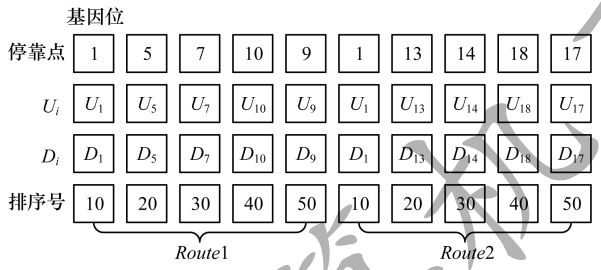

图 1 染色体编码示意图

BRP 模型具有以下性质[20]:

性质 1 设 *Route* 为一条可行线路,*Position*($P_1,P_2,\cdots,P_{i-1},P_i,P_{i+1},\cdots P_n$)为在构造线路时可插入待访问取送点的位置(如图 2 所示),设 P_i 位置可插入车数范围为[D_i,U_i],若要保持插入后 *Route* 仍可行,则有:

$$U_i = Q + \min\{\theta_1,\theta_2,\cdots,\theta_i\} - \max\{\theta_i,\theta_{i+1},\cdots,\theta_n\}$$

$$D_i = -Q - \min\{\theta_i,\theta_{i+1},\cdots,\theta_n\} + \max\{\theta_1,\theta_2,\cdots,\theta_i\}$$

图 2 线路可插入位置示意图

根据性质 1,本文算法可实时计算对应插入位置单车容量的上限与下限,该容量区间表示确保插入停靠点后的新路径仍然可行的回收与投放单车数量区间。

本文算法初始解生成采用随机选择停靠点作为插入位置,通过容量区间实时检测并判断每次插入操作是否可行。如果插入操作可行,则允许实施,否则,选择其他停靠点检测并判断。如果对所有停靠点插入操作均不可行,则重新生成一条新路径,直到所有的停靠点均插入到当前解。

2.3 交叉算子

为了不在交叉过程中过度破坏染色体片断,交叉算子采用线路交叉方法,即在参与交叉的 2 个染色体中随机各选取一条线路作为交换片段进行交换。在交叉过程中涉及大量的停靠点插入操作,可通过采用容量区间实时检测,确保当前插入操作可行,具体过程如下:

1)对参与交叉的 2 个父染色体 x 与 y 随机各选取一条参与交叉的线路,将从 x(y)染色体上选取的线路中的点形成待插入 y(x)染色体的位置集合 *Insert_to_y*(*Insert_to_x*)。

2)在染色体 x(y)中将与集合 *Insert_to_x*(*Insert_to_y*)重复的点删除。在此过程中可能会造成被删除点的线路变为不可行,本文采用 2 种策略进行处理。

策略 1 一旦线路变为不可行,则将该线路所有的点放入 *Insert_to_x* 集或 *Insert_to_y* 集。

策略 2 采用分裂策略,即删除重合点,将重合点前后两部分点按原有排列顺序各形成一条子线路放在原染色体中。

策略 1 有助于提升解的多样性,而策略 2 有助于保留原染色体的优质基因片段。本文在运用交叉算子时混合使用二种策略。

3)利用节约算法将 *Insert_to_x*(*Insert_to_y*)插入 x(y)。

2.4 变异算子

本文算法共使用了 3 种变异算子:

1)Swap 变异算子,随机从参与变异的染色体 x' 或 y' 中选取 2 个点进行交换,形成新的染色体 x''、y''。

2)Merge 变异算子,在参与变异的染色体 x' 或 y' 中随机选择 2 条线路,将第 1 条线路的尾点与第 2 条线路的首点相连。

3)Destroy& Repair 变异算子,从参与变异的染色体 x' 或 y' 中随机选取 3 个点,将其从当前解中移出,并保证移出 3 个点后 x' 或 y' 仍可行,再插入到 x' 或 y' 中形成新的染色体 x'' 或 y''。

在运算过程中等概率随机采用上述 3 种变异算子,而再插入过程均采取最优插入方法,即选择最优节约值最大的插入位置插入,且采用保优策略,保证变异后的新染色体 x''、y'' 的适应度优于原染色体 x'、y'。

2.5 算法的终止条件与多样性保持策略

当算法运行到指定搜索的最大循环次数 *Search_Max* 时,算法停止搜索,输出结果。当算法运行 *Not_Improve* 次循环,种群中的当前最优解未出现改进时,引入破坏修复机制来扰动原有种群,增强种群多样性,提升算法寻优能力。具体方法为从种群中选出 30% 的染色体,对每染色体采用类似于 Destroy& Repair 变异算子的方法进行破坏修复,按 10% ~30% 比例随机选择染色体中的点进行删除,并保持原染色体可行,再根据最优插入方法进行插入,形成新的染色体。为了保证算法收敛,在种群引入破坏修复机制时,采取种群保优策略,即原种群中的最优染色体始终被保留。

3　算法测试

3.1　测试环境与测试方法

本文运用 BRP 问题标准测试算例进行算法测试,该算例可在网站 http://www.or.unimore.it/resources/BRP/home.html 下载。将测试算例按单车停靠点数量分为小规模算例($n \leqslant 50$)、中等规模算例($50 < n \leqslant 100$)以及大规模算例($n > 100$),采用 Matlab2014 编程实现算法,硬件条件为 CPU(core i3,3.1 GHz)、Rom(4 GB)。参数设置为:变异概率取 0.6,交叉概率取 0.3,种群规模为 100,最大循环次数 *Search_Max* 分别为 5 000(小规模)、10 000(中规模)和 20 000(大规模),*Not_Improve* 分别为 500(小规模)、1 000(中规模)和 2 000(大规模),每个算例运行 10 次,取最好的解进行比较。

3.2　测试结果

在已有的 BRP 求解算法中,文献[2]运用了分支定界法(B&C)、文献[4]运用了线路破坏与修复启发式算法(D&R)分别求解了该问题的标准算例,且其测试硬件环境与本文基本相同。因此,本文选择上述 2 种算法作为对比算法。表 1 ~ 表 3 为3 种算法计算结果与计算时间比较。其中,N/C 分别代表算例的停靠点数量与运送单车的卡车额定载重量,S_i 代表各算法的求解结果,CPU_i 表示各算法的求解 CPU 消耗时间,$g\_B\&C = (S_2 - S_1)/S_2$、$g\_D\&R = (S_3 - S_1)/S_3$,分别表示本文算法相较于B&C、D&R 在求解质量上的提升比例,$g\_t\_B\&C = (CPU_2 - CPU_1)/CPU_2$、$g\_t\_D\&R = (CPU_2 - CPU_1)/CPU_2$,分别为本文算法相较于 B&C、D&R 算法在求解时间上的节约比例。在表 1 中,对于 B&C、D&R 2 种算法计算时间小于 2 s 的情况未进行比较。

表 1　小规模算例测试结果对比

算例	N/C	本文算法		B&C 算法		D&R 算法		提升比例/%			
		S_1	CPU_1/s	S_2	CPU_2/s	S_3	CPU_3/s	$g\_B\&CCC$	$g\_D\&R$	$g\_t\_B\&C$	$g\_t\_D\&R$
Bari	13/30	14 600	2.4	14 600	0.0	14 600	0.0	0.0	0.0		
	13/20	15 700	2.5	15 700	0.0	15 700	0.0	0.0	0.0		
	13/10	20 600	2.9	20 600	0.0	20 600	0.0	0.0	0.0		
Regg	14/30	16 900	2.6	16 900	0.0	16 900	0.0	0.0	0.0		
	14/20	23 200	3.0	23 200	0.0	23 200	0.0	0.0	0.0		
	14/10	32 500	2.9	32 500	0.1	32 500	0.0	0.0	0.0		
Berg	15/30	12 600	2.5	12 600	0.0	12 600	0.0	0.0	0.0		
	15/20	12 700	2.7	12 700	0.0	12 700	0.0	0.0	0.0		
	15/12	13 500	3.1	13 500	0.1	13 500	0.0	0.0	0.0		
Parm	15/30	29 000	2.6	29 000	0.0	29 000	0.0	0.0	0.0		
	15/20	29 000	2.6	29 000	0.0	29 000	0.0	0.0	0.0		
	15/10	32 500	3.6	32 500	0.1	32 500	0.0	0.0	0.0		
Trev	18/30	29 259	3.4	29 259	0.1	29 259	0.0	0.0	0.0		
	18/20	29 259	3.4	29 259	0.0	29 259	0.0	0.0	0.0		
	18/10	31 443	3.9	31 443	0.1	31 443	0.1	0.0	0.0		
Lasp	20/30	20 746	3.4	20 746	0.0	20 746	0.0	0.0	0.0		
	20/20	20 746	3.7	20 746	0.0	20 746	0.0	0.0	0.0		
	20/10	22 811	4.0	22 811	0.1	22 811	0.1	0.0	0.0		
Otta	21/30	76 999	3.5	76 999	0.4	76 999	0.0	0.0	0.0		
	21/20	91 619	3.6	91 619	3.6	91 619	4.2	0.0	0.0	-0.9	14.0
	21/10	16 202	3.9	16 202	0.0	16 202	0.0	0.0	0.0		
Buen	21/30	16 202	4.4	16 202	0.0	16 202	0.0	0.0	0.0		
	21/20	17 576	4.6	17 576	0.1	17 576	0.0	0.0	0.0		
Sana	23/30	22 982	4.4	22 982	0.1	22 982	0.0	0.0	0.0		
	23/20	24 007	4.0	24 007	0.1	24 007	0.1	0.0	0.0		
	23/10	40 149	3.8	40 149	1.1	40 149	0.4	0.0	0.0		
Bres	27/30	30 300	4.2	30 300	0.1	30 300	0.1	0.0	0.0		
	27/20	31 100	4.2	31 100	0.2	31 100	0.1	0.0	0.0		
	27/11	35 200	3.8	35 200	1.4	35 200	0.4	0.0	0.0		
Madi	28/30	61 900	3.9	61 900	0.8	61 900	0.4	0.0	0.0		
	28/20	66 600	4.1	66 600	1.7	66 600	3.5	0.0	0.0		-17.9
	28/10	68 300	4.4	68 300	0.6	68 300	0.0	0.0	0.0		
Roma	28/30	29 246	4.1	29 246	0.0	29 246	0.0	0.0	0.0		
	28/20	29 839	4.2	29 839	0.1	29 839	0.0	0.0	0.0		
	28/18	33 848	4.4	33 848	0.5	33 848	0.2	0.0	0.0		
Guad	41/30	57 476	6.4	57 476	0.2	57 476	2.1	0.0	0.0		
	41/20	59 493	5.3	59 493	0.3	59 493	1.5	0.0	0.0		
	41/11	64 981	5.2	64 981	1.8	64 981	2.4	0.0	0.0		-117.0
Dubl	45/30	33 548	5.7	33 548	6.1	33 548	2.3	0.0	0.0	5.9	-148.7
	45/20	39 786	5.4	39 786	76.7	39 786	3.7	0.0	0.0	93.0	-43.8
	45/11	54 392	5.1	54 392	610.8	54 392	5.3	0.0	0.0	99.2	3.7
平均		34 361	3.8	34 361	17.3	34 361	0.7	0.0	0.0	77.7	-483.6

表 2 中等规模算例测试结果对比

算例	N/C	本文算法		B&C 算法		D&R 算法		提升比例/%			
		S_1	CPU_1/s	S_2	CPU_2/s	S_3	CPU_3/s	g_B&CCC	g_D&R	g_t_B&C	g_t_D&R
Denv	51/30	51 583	31.7	51 583	0.7	51 583	0.5	0.0	0.0	-4 631.8	-6 504.8
	51/20	53 465	32.0	53 465	25.3	53 465	24.7	0.0	0.0	-26.5	-29.8
	51/10	67 459	22.7	67 459	231.5	67 459	103.0	0.0	0.0	90.2	77.9
Riod	55/30	122 547	29.5	122 547	65.6	122 547	198.7	0.0	0.0	54.9	85.1
	55/20	155 446	25.3	155 446	3 600.0	155 517	304.1	0.0	0.0	99.3	91.7
	55/10	253 690	23.9	253 690	3 600.0	257 147	241.1	0.0	1.3	99.3	90.1
Bost	59/30	65 669	34.5	65 669	28.1	65 669	16.2	0.0	0.0	-22.6	-113.5
	59/20	71 879	36.4	71 879	473.8	71 916	178.2	0.0	0.1	92.3	79.5
	59/16	74 790	37.5	74 790	3 600.0	75 085	280.5	0.0	0.4	99.0	86.6
	75/30	56 011	45.2	47 634	13.8	47 634	246.4	-17.6	-17.6	-227.5	81.7
	75/20	64 106	37.3	50 204	859.7	50 438	251.8	-27.7	-27.1	95.7	85.2
	75/10	94 583	34.7	58 814	3 600.0	61 717	303.9	-60.8	-53.3	99.0	88.6
	80/30	55 609	48.3	40 794	3 600.0	41 390	201.5	-36.3	-34.4	98.7	76.0
	80/20	54 758	46.1	42 621	3 600.0	46 631	269.4	-28.5	-17.4	98.7	82.9
	80/12	61 908	47.3	54 238	3 600.0	58 539	321.4	-14.1	-5.8	98.7	85.3
	82/30	15 4861	31.5	152 229	3 600.0	154 038	335.1	-1.7	-0.5	99.1	90.6
	82/20	225 593	31.5	209 379	3 600.0	214 250	265.1	-7.7	-5.3	99.1	88.1
	82/10	402 220	44.4	390 536	3 600.0	397 921	300.3	-3.0	-1.1	98.8	85.2
	90/30	94 417	45.2	67 894	3 600.0	72 279	213.9	-39.1	-30.6	98.7	78.9
	90/20	102 336	42.9	88 952	3 600.0	94 319	254.2	-15.0	-8.5	98.8	83.1
	90/17	125 896	39.4	99 714	3 600.0	103 658	359.6	-26.3	-21.5	98.9	89.0
平均		114 706	36.5	105 692	2 309.5	107 772	222.4	-8.5	-6.4	98.4	83.6

表 3 大规模算例测试结果对比

算例	N/C	本文算法		B&C 算法		D&R 算法		提升比例/%			
		S_1	CPU_1/s	S_2	CPU_2/s	S_3	CPU_3/s	g_B&CCC	g_D&R	g_t_B&C	g_t_D&R
Minn	116/30	15 0031	55.9	136 148	3 600	138 467	745.3	-10.2	-8.4	98.4	92.5
	116/20	17 6783	51.7	157 736	3 600	166 150	1 003.7	-12.1	-6.4	98.6	94.9
	116/17	24 6285	53.9	246 133	3 600	262 936	946.4	-0.1	6.3	98.5	94.3
Bris	150/30	139 349	56.6	108 275	3 600	115 120	742.8	-28.7	-21.0	98.4	92.4
	150/20	150 884	62.6	132 419	3 600	146 313	957.8	-13.9	-3.1	98.3	93.5
	150/17	179 985	73.5	147 236	3 600	160 015	966.4	-22.2	-12.5	98.0	92.4
Mila	184/30	187 595	96.5	145 245	3 600	167 493	871.2	-29.2	-12.0	97.3	88.9
	184/20	254 864	98.3	187 175	3 600	218 249	823.8	-36.2	-16.8	97.3	88.1
	184/18	265 474	114.8	203 716	3 600	234 423	1 048.3	-30.3	-13.2	96.8	89.0
Lill	200/30	192 744	102.5	164 149	3 600	176 976	666.7	-17.4	-8.9	97.2	84.6
	200/20	240 757	85.5	191 630	3 600	213 090	280.1	-25.6	-13.0	97.6	69.5
Toul	240/30	202 251	109.4	166 653	3 600	188 995	1 147.9	-21.4	-7.0	97.0	90.5
	240/20	304 234	127.7	190 739	3 600	228 674	1 398.5	-59.5	-33.0	96.5	90.9
	240/13	417 115	167.5	256 036	3 600	307 874	869.0	-62.9	-35.5	95.3	80.7
Sevi	258/30	302 581	96.8	194 805	3 600	225 076	898.4	-55.3	-34.4	97.3	89.2
	258/20	340 398	114.1	240 210	3 600	279 990	1 054.3	-41.7	-21.6	96.8	89.2
Vale	276/30	329 798	124.4	245 979	3 600	287 854	789.5	-34.1	-14.6	96.5	84.2
	276/20	430 878	147.1	302 368	3 600	367 201	1 064.4	-42.5	-17.3	95.9	86.2
Brux	304/30	418 569	168.3	255 259	3 600	311 097	852.0	-64.0	-34.5	95.3	80.2
	304/20	424 741	215.0	301 100	3 600	376 387	835.7	-41.1	-12.8	94.0	74.3
	304/16	689 256	243.4	348 303	3 600	424 432	1 038.3	-97.9	-62.4	93.2	76.6
Lyon	336/30	523 088	215.0	300 950	3 600	360 009	879.6	-73.8	-45.3	94.0	75.6
	336/20	693 106	275.8	356 787	3 600	433 959	1 136.0	-94.3	-59.7	92.3	75.7
Barc	410/30	774 701	340.5	311 774	3 600	543 929	458.3	-148.5	-42.4	90.5	25.7
	410/20	905 674	543.5	449 060	3 600	771 507	1 538.6	-101.7	-17.4	84.9	64.7
	410/19	955 741	517.7	429 327	3 600	800 622	1 411.3	-122.6	-19.4	85.6	63.3
Lond	564/30	897 890	903.2	385 748	3 600	699 571	1 573.0	-132.8	-28.3	74.9	42.6
	564/29	944 685	919.0	363 299	3 600	718 026	1 447.3	-160.0	-31.6	74.5	36.5
平均		419 266	217.1	247 081	3 600	333 016	980.2	-69.7	-25.9	94.0	77.8

由小规模算例测试结果可以看出,本文算法与B&C、D&R 一样,均能找到算例的最优解。本文算法的求解的平均 CPU 消耗为 3.8 s,比 B&C 的 17.3 s 快,但仍慢于 D&R 的 0.7 s。

由中等规模算例测试结果可以看出,本文算法找到9 个算例的最优解,并且其平均求解质量虽然比 B&C、D&R 低 8.5% 与 6.4% ,但求解的平均 CPU 消耗为 36.5 s,相较于 B&C 的 2 309.5 s、D&R 的 222.4 s 分别提升 98.4% 与 83.6% 。

由大规模算例测试结果可以看出,本文算法的求解速度优势更明显,虽然求解质量分别比 B&C、D&R 低 69.7% 与 25.9% ,但求解的平均 CPU 消耗为217.1 s,分别比 B&C、D&R 提升 94.0% 与 77.8% 。

4　结束语

BRP 是共享设施再平衡调度中重要的实际应用问题,在共享经济快速发展的背景下具有较高的研究价值。本文提出一种求解 BRP 的遗传算法。通过应用非代际遗传算法框架,采用多维整数编码方法,设计路线交叉算子和 k 点破坏修复变异算子,并引入破坏修复机制来保持种群多样性。实验结果表明,该算法对小规模算例具有良好的求解质量,所有算例均能在 4 s 内找到最优解,对于中等规模和大规模算例,本文算法求解质量有待进一步提升,但其运算速度相较于 B&C 和D&R 算法具有显著优势。下一步将考虑停靠点规划布置、用户取用车习惯以及管理机构再平衡管理目标(成本或用户满意度最优)等实际需求,设计相应的高效算法求解 BRP 问题。

参考文献

[1] RAVIV T,TZUR M,FORMA I A. Static repositioning in a bike-sharing system:models and solution approaches[J]. EURO Journal on Transportation and Logistics, 2013, 2(3):187-229.

[2] ERDOGAN G,BATTARRA M,CALVO R W. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems[J]. European Journal of Operational Research,2015,245(3):667-679.

[3] FORMA I A, RAVIV T, TZUR M. A 3-step math heuristic for the static repositioning problem in bike-sharing systems[J]. Transportation Research Part B: Methodological,2015,71:230-247.

[4] DELL M,IORI M, NOVELLANI S, et al. A destroy and repair algorithm for the bike sharing rebalancing problem[J]. Computers and Operations Research,2016,71:149-162.

[5] DELL M,HADJICOSTANTINOU E,IORI M,et al. The bike sharing rebalancing problem:mathematical formulations and benchmark instances[J]. OMEGA,2014,45:7-19.

[6] CHEMLA D,MEUNIER F,CALVO R W. Bike sharing systems: solving the static rebalancing problem[J]. Discrete Optimization,2013,10(2):120-146.

[7] CONTARDO C, MORENCY C, ROUSSEAU L M. Balancing a dynamic public bike-sharing system[M]. Montreal,Canada:Cirrelt,2012.

[8] CAGGIANI L, OTTOMANELLI M. A modular soft computing based method for vehicles repositioning in bike-sharing systems[J]. Procedia-Social and Behavioral Sciences,2012,54:675-684.

[9] PFROMMER J, WARRINGTON J, SCHILDBACH G, et al. Dynamic vehicle redistribution and online price incentives in shared mobility systems[J]. IEEE Transactions on Intelligent Transportation Systems,2014, 15(4):1567-1578.

[10] 叶丽霞. 城市公共自行车调度系统研究[D]. 南京:南京理工大学,2013.

[11] 乔晓. 城市公共自行车多车场车辆调度问题研究[D]. 西安:长安大学,2017.

[12] 蒋琳. 城市公共自行车系统调度模型及算法研究[D]. 兰州:兰州交通大学,2017.

[13] 卢琰. 共享单车车辆调度问题研究[D]. 成都:西南交通大学,2018.

[14] 靳迎新. 城市公共自行车系统调度优化研究[D]. 兰州:兰州交通大学,2018.

[15] 杨桥东,李朋州,李琮,等. 多车场协同运输的公共自行车调度方法研究[J]. 交通科技与经济,2016, 18(4):8-12.

[16] 赵宇橙. 基于 PSO 算法的秩序配送车辆路径问题的研究[J]. 现代经济信息,2017(1):30-32.

[17] 潘立军,符卓,刘喜梅. 带工作时间与时间窗的开放式车辆路径问题[J]. 计算机工程,2012,38(4):17-19.

[18] 金仙力,李金刚. 基于遗传算法的多目标路径优化算法的研究[J]. 计算机技术与发展,2018,28(2):55-58.

[19] 潘立军,符卓. 求解带时间窗取送货问题的遗传算法[J]. 系统工程理论与实践,2012,32(1):120-126.

[20] 潘立军,符卓,刘喜梅. 共享单车再平衡问题及其容差插入启发式算法[J]. 运筹与管理,2019,21(8):57-63.

编辑　刘盛龄