



求解带时间窗车辆路径优化问题的改进细菌觅食算法

李 珺,郝丽艳,何奕涛,段钰蓉

(兰州交通大学 电子与信息工程学院,兰州 730070)

摘 要:为求解带时间窗的车辆路径问题(VRPTW),提出一种改进的细菌觅食算法。将待配送的客户点依据地理位置进行K-means聚类,使得到的分类结果在满足时间窗的要求下,按顺序插入配送路径的最佳位置中,构造VRPTW问题的初始解,同时通过结合趋化操作与大邻域搜索中的removal算子进行距离寻优,扩大算法搜索范围并提高运行效率。实验结果表明,在规定时间内,改进算法能合理安排配送路径并最小化总配送成本。

关键词:车辆路径问题;时间窗;细菌觅食算法;大邻域搜索;贪婪插入法

开放科学(资源服务)标志码(OSID):



中文引用格式:李珺,郝丽艳,何奕涛,等.求解带时间窗车辆路径优化问题的改进细菌觅食算法[J].计算机工程,2021,47(11):44-53.

英文引用格式:LI J, HAO L Y, HE Y T, et al. Improved bacterial foraging algorithm for solving vehicle routing optimization problem with time windows[J]. Computer Engineering, 2021, 47(11): 44-53.

Improved Bacterial Foraging Algorithm for Solving Vehicle Routing Optimization Problem with Time Windows

LI Jun, HAO Liyan, HE Yitao, DUAN Yurong

(School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

[Abstract] This paper proposes an improved Bacterial Foraging Algorithm(BFA) to solve the Vehicle Routing Problem with Time Windows (VRPTW). The to-be-distributed customer points are clustered by K-means according to the geographical location, and the classification results are inserted into the optimal location of the distribution path in order within the time window, so the initial solution of the problem is constructed. Then the chemotaxis operation is combined with the removal operator in Large Neighborhood Search(LNS) for optimization, which expands the search range and improves the operation efficiency of the algorithm. Experimental results show that the algorithm can shorten the distribution path and minimize the overall distribution cost within the time window.

[Key words] Vehicle Routing Problem(VRP); time window; Bacterial Foraging Algorithm(BFA); large neighborhood search; greedy insertion method

DOI:10.19678/j.issn.1000-3428.0059260

0 概述

车辆路径问题(Vehicle Routing Problem, VRP)一直是物流配送活动中的基本问题,受到国内外学者的广泛关注。随着电子商务的发展以及客户需求的提高,企业在设计物流系统时需要在规定时间窗内完成服务,从而提高服务水平。带时间窗的车辆路径问题(Vehicle Routing Problem with Time Windows, VRPTW)是经典VRP问题的一个重要扩展,每个顾客有预先设定的时间窗,车辆不能早于顾

客允许的最早服务时间提供服务,也不能晚于顾客允许的最晚服务时间提供服务。

求解VRPTW问题的算法大致可以分为精确算法和启发式算法两类。精确算法在理论上可以找到问题的最优解,但由于在实际应用中消耗的空间和时间成本较大,因此计算机内存要求较高,仅适用于求解较小规模的路径优化问题。启发式算法不管是求解小规模的问题还是大规模的问题,都能够在一定范围和较短的时间内给出满意解或最优解。因此,目前相关领域的学者专注于设计不同的启发式算法寻找该问题的近

基金项目:甘肃省科技厅自然科学基金(1506RJZA084);甘肃省教育厅科研基金(1204-13);兰州市科技局科研基金(2015-2-74)。

作者简介:李珺(1974—),女,副教授、博士,主研方向为智能计算、图像处理;郝丽艳、何奕涛、段钰蓉,硕士研究生。

收稿日期:2020-08-14 **修回日期:**2020-11-08 **E-mail:**lijane@mail.lzjtu.cn

似最优解,特别是对现代启发式算法的研究。文献[1]利用分布式多agent系统实现分布式求解VRPTW问题。文献[2]介绍基于量子蚁群算法的VRPTW问题。文献[3]介绍一种求解VRPTW问题的离散粒子群算法。文献[4]介绍改进的蚁群算法求解带时间窗的车辆路径问题。文献[5]介绍一种新的求解带时间窗车辆路径问题的遗传算法。文献[6]介绍考虑载荷约束的求解时间窗车辆路径问题的遗传算法。

目前,还有一些学者采用深度强化学习方法求解车辆路径问题。文献[7]提出一个基于深度强化学习框架的改进启发式规划方法来求解车辆路径问题。通过设计基于自我关注的深层架构策略网络,指导下一步解决方案的选择。学习到的策略能很好地概括不同的初始解和问题规模,并在实际数据集上给出了较好的解决方案。文献[8]提出一种基于紧急程度的插入启发式算法来构造初始解,在局部搜索改进阶段使用强化学习来指导搜索。实验结果表明,基于单个解的算法中的抽样方案并没有显著提高解的质量,但是可以大幅降低搜索过程中发现的不可行解的比例,在大规模搜索空间内具有很强的搜索能力。文献[9]提出一种具有动态编译码结构的动态关注模型,该模型能够在不同的构建步骤中动态探索节点特征并有效挖掘隐藏的结构信息,但是训练阶段比较耗时,仅可针对中小规模问题实例进行训练。文献[10]使用基于蒙特卡罗预测及深度策略梯度学习的智能车辆路径规划方法,设计一种基于环境感知预测、行为决策和控制序列生成的深度强化学习框架,并将多个单一的强化学习算法应用到智能车路径规划系统的不同子任务中。实验结果验证了车辆具有的预估风险能力,并以尽量大的安全距离调整车辆状态,行驶时有更加连续的转角控制序列,实现快速、安全、平稳的路径规划。文献[11]提出一种基于强化学习的超启发式算法,设计算法的高层启发式策略,包括选择策略和解的接收准则,算法在实验求解过程中整体性能较好,但在客户规模较大的车辆路径问题上仍具有改进空间。目前,关于使用深度强化学习方法求解VRPTW问题的研究相对较少,可进行更深入的探讨。

细菌觅食算法(Bacterial Foraging Algorithm, BFA)作为现代启发式算法之一,是由PASSINO^[12]于2002年提出的一种仿生随机搜索算法,具有局部搜索能力强、并行搜索、实现简单等优点,目前已在优化计算、图像处理、模式识别、系统仿真、人脸识别等领域得到广泛应用^[13-15]。本文提出一种改进的细菌觅食算法来求解VRPTW问题,使用K-means算法根据地理位置对待配送点进行聚类,依据聚类结果采用贪婪插入法生成初始解。将细菌觅食的趋化操作与大邻域搜索相结合,设计4种removal算子进行寻优,扩大搜索范围,在不增加时间复杂度的情况下,

提高收敛精度。

1 考虑时间窗的车辆路径优化问题

VRPTW问题描述:有一定数量的相同型号的车辆,从中心仓库出发,对若干个客户节点进行服务,要求满足约束条件,合理安排配送路径,使总配送成本最小。基本假设如下:

1)只有一个配送中心,车辆从配送中心出发并最终返回。

2)配送中心及客户位置已知。

3)每个客户需求量及服务时间一定。

4)每个客户仅被一辆车服务一次。

5)车辆必须在客户指定的时间窗内进行配送。

6)每辆车的载重不得超过车辆最大载重能力。

符号定义如下:

S 表示客户节点集合, $S=\{1, 2, \dots, N\}$ 。

S_0 表示顶点集合, $S_0=S \cup \{0\}$, 0代表配送中心。

K 表示所有车辆的集合。

V_k 表示车辆 k 访问的客户节点集合。

D_i 表示客户节点 i 的需求量。

C_{ij} 表示客户节点 i 到 j 之间的距离,配送中心到客户节点 i 之间的距离为 C_{0i} 。

Q 表示车辆的最大负载能力。

L 表示车辆的最大行驶距离。

E_i 表示客户节点 i 的最早服务时间。

L_i 表示客户节点 i 的终止服务时间。

x_{ij}^k 表示当车辆 k 由客户节点 i 直接行驶到 j 时, $x_{ij}^k=1$, 否则 $x_{ij}^k=0$ 。

y_i^k 表示客户节点 i 是否被车辆 k 服务, $y_i^k=1$ 为服务, $y_i^k=0$ 为不服务。

t_{ij} 表示客户 i 到客户 j 所需的行驶时间。

s_i 表示客户节点 i 的开始服务时间。

对VRPTW建立如下模型:

$$\min Z = \sum_{k \in K} \sum_{i \in S_0} \sum_{j \in S_0, j \neq i} c_{ij} x_{ij}^k \quad (1)$$

满足如下约束条件:

$$\sum_{i \in S_0} \sum_{k \in K} x_{ijk} = 1, \forall j \in S \quad (2)$$

$$\sum_{i \in S_0} \sum_{j \in S_0} c_{ij} x_{ijk} \leq L, \forall k \in K \quad (3)$$

$$\sum_{k \in K} y_i^k = 1, \forall i \in S \quad (4)$$

$$\sum_{j \in S} x_{0j}^k = \sum_{i \in S} x_{i0}^k = 1, \forall k \in K \quad (5)$$

$$\sum_{i \in V_k} D_i y_i^k \leq Q, \forall k \in K \quad (6)$$

$$E_i \leq s_i \leq T_i, \forall i \in S \quad (7)$$

$$\sum_{i \in S, j \in S} x_{ij}^k = |V_k| - 1, \forall k \in K \quad (8)$$

式(1)表示最小化车辆总行程;式(2)表示每个客户仅被一辆车服务一次;式(3)表示每辆车的行驶

距离都不得超过最大行驶距离;式(4)表示每个客户只被同一辆车服务;式(5)表示车辆从配送中心出发最终必须再回到配送中心;式(6)表示每辆车配送的客户总需求量不得超过车辆最大载重;式(7)表示客户 i 的开始服务时间必须满足给定的时间窗;式(8)表示消除子回环。

2 改进的细菌觅食算法

改进的细菌觅食算法(Improved Bacterial Foraging Algorithm, IBFA)是在细菌觅食算法的基础上结合大邻域搜索(Large Neighborhood Search, LNS)的优化算法^[16-18]。该算法主要通过趋化操作、复制操作和迁徙操作的迭代计算来搜索问题的最优解。趋化操作是算法的核心操作,因此本文在趋化操作中结合了大邻域搜索算法中的removal算子,运用4种removal算子进行寻优,扩大细菌的搜索范围。大邻域搜索算法由SHAW^[19]于1998年提出,克服局部搜索的缺陷,提高BFA优化能力,能够更快地找到最优解,对于求解VRPTW问题十分有效。

2.1 编码

本文采用非负整数编码方式表示解空间,配送中心编号为0,客户节点编号为 $1, 2, \dots, N$,令 $S=[S_1, S_2, \dots, S_k]$ (k 为车辆数目)为VRPTW问题的一组可行解,假设车辆路径规划为 $S=[S_1, S_2]=[\{0, 3, 2, 8, 7, 6, 0\}, \{0, 1, 4, 5, 9, 0\}]$,第1条路径从配送中心出发,服务客户3、2、8、7、6后回到配送中心,第2条路径从配送中心出发,服务客户节点1、4、5、9后回到配送中心,如图1所示。

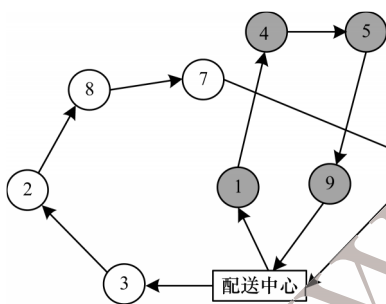


图1 配送示意图

Fig.1 Schematic diagram of distribution

2.2 初始解构造

本文采用基于K-means算法的贪婪插入法构造VRPTW问题的初始解。在运用贪婪插入法构造初始解时,由于前期客户的插入次序对解的构造有较大影响,因此使用K-means算法对客户节点进行预处理,其中 $k \in \left[\left\lceil \sum_{i=1}^N d_i / Q \right\rceil, V \right]$, V 为配送中心所拥有的最大车辆数目, k 为随机选取该范围中的一个整数,利用客户节

点的位置信息对客户节点聚类,根据聚类结果产生客户的插入序列,使用贪婪插入法生成路径。贪婪插入法首先构建一条从配送中心出发再返回配送中心的路径,即 $[0, 0]$,然后运用K-means算法排列好的客户节点依次插入路径中的最佳位置,当路径中无法再插入新客户时,重新构建一条 $[0, 0]$ 的路径,重复上述操作,直到所有客户都插入为止。

为验证采用K-means构造初始解对算法的影响,本文选取Solomon测试集中的R211算例进行验证。测试集共100个客户节点,车辆载重为1000,使用K-means算法聚类后的客户排列顺序和原始客户集中的客户排列顺序分别生成初始解,共进行10次实验,采用K-means聚类后得到的最终解平均值为782.43,使用初始客户集生成最终解的平均值为791.08。图2给出了使用K-means算法得到的最优解构造的初始路径。图3给出了使用原始序列得到的最优解构造的初始路径。实验结果表明,使用K-means算法对客户集进行预处理得到的解更优。

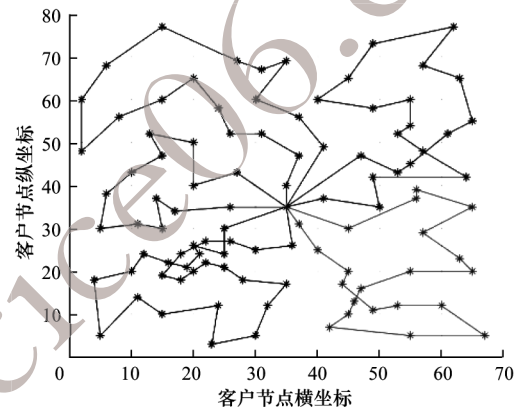


图2 基于K-means算法最优解的初始路径

Fig.2 Initial path based on the optimal solution obtained by the K-means algorithm

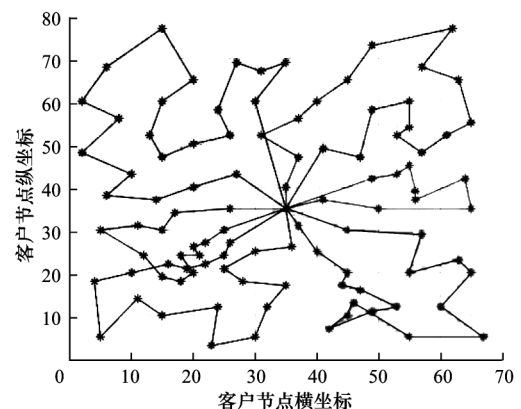


图3 基于原始序列最优解的初始路径

Fig.3 Initial path based on the optimal solution obtained by the original sequence

2.3 趋化操作改进

在细菌觅食优化算法中,对问题的解空间进行寻优的核心操作是趋化操作,包括旋转和游动。细菌向任意方向移动单位步长定义为旋转。旋转后计算适应度值,若得到的路径适应度值更小则继续沿着该方向游动,直到达到最大步长或者适应度值不发生改变,此过程称为游动。由于细菌的步长和方向会影响到寻优效果,因此本文结合 VRPTW 问题的特性,将大邻域搜索中的 removal 算子结合到趋化操作中。为达到更好的搜索效果,共设计 4 种 removal 算子,分别为 Random removal 算子、Least profit removal 算子、Route removal 算子和 Relatedness removal 算子。这 4 种算子作为细菌游动的方向,游动步长设置为 m 。用轮盘赌方法选择一种 removal 算子,用来模拟细菌在旋转过程中任意选择的方向,确定方向后进行移除插入操作。如果该算子取得的适应度值变好,则继续采用,直到达到细菌的最大游动次数,此过程模拟了细菌在该方向上的游动。若旋转后适应度值没有改善,则选择另一个 removal 算子,即另一个方向,继续进行游动。每个细菌都从旋转操作开始,在达到趋化算子次数前不断反复交替执行旋转和游动操作,更新路径。将 removal 算子应用到趋化操作中,扩大搜索范围,有利于局部搜索。在移除了 m 个客户节点后,运用之前的贪婪插入法重新将移除的客户进行插入。

下面以一个 Solomom 数据集中的实例说明利用 removal 算子寻优。假设 VRPTW 的配送中心和客户节点信息如表 1 所示,编号 0 表示配送中心,编号 1~9 表示客户节点,每辆车的载重为 100。采用贪心策略插入法生成的细菌个体为 $S=[\{0,8,3,9,5,0\},\{0,1,7,2,4,0\},\{0,6,0\}]$,即由三辆车配送,适应度值为 328.883 0。removal 算子的移除个数 $m=3$,最大游动次数为 4。

表 1 配送中心及客户节点信息

Table 1 Information of distribution center and customer nodes

编号	客户节点横坐标	客户节点纵坐标	D_i	E_i	L_i	S_i
0	40	50	0	0	3 390	0
1	52	75	10	311	471	90
2	25	85	20	2 119	2 279	90
3	23	52	10	209	369	90
4	8	40	40	685	845	90
5	26	32	10	2 292	2 452	90
6	33	32	20	2 666	2 826	90
7	85	25	10	2 513	2 673	90
8	70	58	20	1 641	1 801	90
9	28	70	10	2 601	2 761	90

假设选定 Random removal 算子第一次游动移除的客户节点为 7、2、8,在之前生成的解 S 中找到 7、2、8 并将其移除,移除后的路径 $S_{\text{new}}=[\{0,3,9,5,0\},\{0,1,4,0\},\{0,6,0\}]$ 。首先,插入客户节点 7,在满足约束条件后,在每条路径中插入的具有最小配送成本的最佳位置如图 4 所示。客户节点 7 插入 3 条路径最佳位置的适应度值分别为 354.033 4、301.987 5、273.882 6,通过比较得出最小适应度值为 273.882 6,因此将客户节点插入路径 3,此时 $S_{\text{new}}=[\{0,3,9,5,0\},\{0,1,4,0\},\{0,7,6,0\}]$ 。然后,插入客户节点 2,在满足约束条件后,在每条路径中插入的具有最小配送成本的最佳位置如图 5 所示。客户节点 2 插入 3 条路径最佳位置的适应度值分别 355.298 4、293.456 7、374.964 9,通过比较得出最小适应度值为 293.456 7,因此将客户节点 2 插入路径 2,此时 $S_{\text{new}}=[\{0,3,9,5,0\},\{0,1,2,4,0\},\{0,7,6,0\}]$ 。最后,插入客户节点 8,在满足约束条件后,在每条路径中插入的具有最小配送成本的最佳位置如图 6 所示。客户节点 8 插入 3 条路径最佳位置的适应度值分别 296.257 3、319.856 7、326.907 1,通过比较得出最小适应度值为 296.257 3,因此将客户节点 8 插入路径 1,此时 $S_{\text{new}}=[\{0,8,3,9,5,0\},\{0,1,2,4,0\},\{0,7,6,0\}]$ 。 S_{new} 的适应度值为 296.257 3, S 的适应度值为 328.883 0,经过比较, S_{new} 的适应度值较小,则用 S_{new} 的位置更新 S ,继续采用 removal 算子进行局部寻优,直到达到最大游动次数。如果该 removal 算子没有改善适应度值,则维持 S 的位置不变,结束此次游动,细菌进行翻转,用另一个 removal 算子进行第二次游动,重复以上操作,直到适应度值不再发生改变。

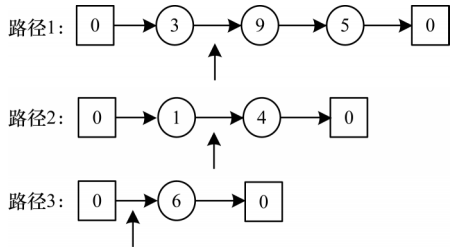


图 4 客户节点 7 插入每条路径的最佳位置

Fig.4 The best position for inserting client node 7 into each path

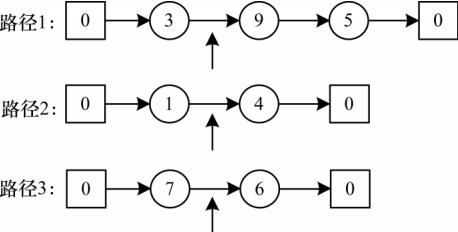


图 5 客户节点 2 插入每条路径的最佳位置

Fig.5 The best position for inserting client node 2 into each path

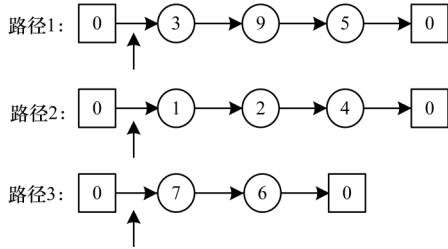


图6 客户节点8插入每条路径的最佳位置

Fig.6 The best position for inserting client node 8 into each path

图7和图8分别给出了初始路径和经过趋化操作的改进路径。

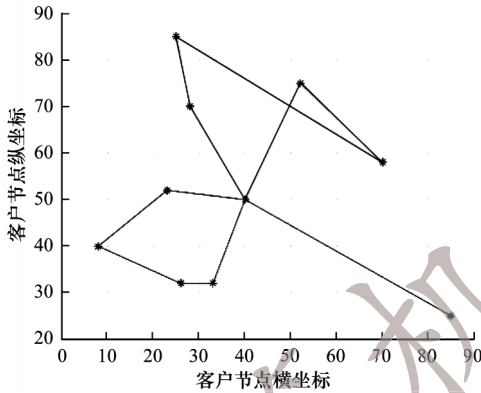


图7 初始路径

Fig.7 Initial path

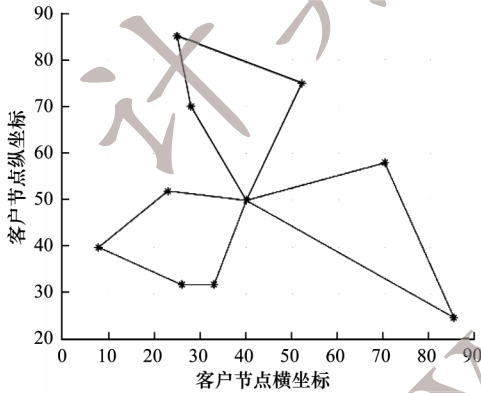


图8 经过趋化操作的改进路径

Fig.8 Improved path after chemotaxis

4种 removal算子的具体描述如下:

1) Random removal算子。该算子是随机选择 q 个客户节点进行移除,再将其进行插入。该算子的随机性增加了搜索的多样性。

2) Least profit removal算子。该算子主要移除成本较高的客户,给定一个客户 i 和一个解 s ,定义客户节点 i 的成本为 $C_i(s) = f(s) - f_{-i}(s)$,其中 $f_{-i}(s)$ 表示移除客户节点 i 之后的适应度值,对 $C(s)$ 值进行降序排列,选择成本较高的 q 个客户节点进行移除。

3) Route removal算子。该算子主要是减少寻优过程中车辆的使用数目,选取配送客户节点较少的

车辆中的客户节点进行移除。在采用 Route removal算子进行局部搜索过程中,可以发现某些车辆所服务的客户节点数量非常少,这样对车辆资源造成较大的浪费,也不利于总配送路径的减少。因此,在 Route removal算子中应该优先移除这些车辆配送数目较少的客户节点,再将移除的客户节点用贪婪插入策略重新插入路径中。

4) Relatedness removal算子。该算子移除相关性较高的客户节点,根据两个客户节点 i 和 j 的距离 C_{ij} 、送货需求的差异 $|D_i - D_j|$ 以及两个客户的最早开始时间 $|E_i - E_j|$ 的差异来衡量两个客户节点 i 和 j 的相关性。每个局部相关性度量分别使用 α 、 β 、 γ 进行加权,并对问题实例给出的所有客户 S 集合的各个极值进行归一化。因此,两个客户节点 i 和 j 的相关性度量 R_{ij} 表示如下:

$$R_{ij} = \alpha \frac{C_{ij}}{\max_{i,j \in S} (C_{ij})} + \beta \frac{|D_i - D_j|}{\max_{i \in S} (D_i) - \min_{i \in S} (D_i)} + \gamma \frac{|E_i - E_j|}{\max_{i \in S} (E_i) - \min_{i \in S} (E_i)} \quad (9)$$

其中: $\max(C_{ij})$ 是任意两个客户之间的最大距离; $\max(D_i)$ 是交付的最大需求值; $\min(D_i)$ 是交付的最小需求值; $\max(E_i)$ 是客户开始服务时间的最大值, $\min(E_i)$ 是客户开始服务时间的最小值。对任意两个客户节点间的相关性进行排序,移除相关性较大的 q 个客户节点,重新将它们插入到路径中,从而得到新的解决方案。

2.4 算法流程

改进的细菌觅食算法步骤具体如下:

1) 初始化参数。初始化细菌种群数目 N 、趋化操作次数 N_c 、最大游动次数 N_s 、复制次数 N_{re} 、迁徙次数 N_{ed} 、迁徙概率 P_{ed} 、removal算子中的移除个数 q , 根据基于 K-means 的贪婪插入法构造初始解。

2) 对种群中的所有细菌个体进行趋化操作。

3) 对细菌的适应度值进行评价并按照升序排列。将前面 $N/2$ 个细菌个体的位置复制给排列在后面的 $N/2$ 个细菌个体。

4) 为每个细菌个体随机生成一个概率 rand 。如果 rand 小于 P_{ed} , 则随机生成 N_{num} 到 2 的全排列数据,将生成的数据按照贪心策略插入法生成细菌个体;如果 rand 大于等于 P_{ed} , 则维持原细菌个体位置不变。

5) 重复步骤 2~步骤 4, 直至达到最大迁徙次数 N_{ed} 。

3 实验与结果分析

3.1 实验环境

实验中使用的计算机配置为 2 GHz CPU、8 GB RAM、64 位操作系统。编程语言使用 Matlab 语言, 版本为 Matlab r2017b。

3.2 算例测试与比较

为评价改进算法的性能,使用 Solomon 数据集进行实验,将所得最短配送距离与对比算法进行比较。Solomon 数据集的 VRPTW 共分为 C1、C2、R1、R2、RC1、RC2 等 6 类。C1 和 C2 类的客户呈现分块聚集分布;R1 和 R2 类的客户是随机分布的;RC1 和 RC2 类的客户既有随机分布,又有分块聚集分布。R1、C1、RC1 数据集的车辆容量小、时间窗口窄,每条路线只允许少数客户;R2、C2 和 RC2 数据集的车辆容量大,且在较长的计划周期内允许许多客户使用同一辆车进行服务。本文算法在每个算例上独立运行 10 次。将实验结果与文献[20]中的具有交叉操作的人工蜂群(Artificial Bee Colony with Crossover operation, ABC-C)算法、具有扫描策略的人工蜂群(Artificial Bee Colony with Scanning strategy, ABC-S)算法和改进人工蜂群(Improved Artificial Bee Colony, IABC)算法所给出的算例进行比较,将全部算例分别与文献[21]中的混合遗传算法(Hybrid Genetic Algorithm, HGA)和文献[22]中的自适应记忆算法(Adaptive Memetic Algorithm, AMA)进行比较。具体比较情况如表 2~表 4 所示,其中“—”部分表示对比算法未计算相应数据。改进的细菌觅食算法中的参数设置为:细菌种群数目 $N=30$,趋化操作次数 $N_c=50$,最大游动次数 $N_s=3$,复制次数 $N_{re}=5$,迁徙次数 $N_{ed}=2$,邻域搜索长度 $W=10$ 。

表 2 C 类算例测试结果比较

Table 2 Comparison of test results in C class example						
算例	ABC-C	ABC-S	IABC	HGA	AMA	IBFA
C101	827.30	827.30	827.30	846.137 8	828.94	828.94
C102	831.50	831.50	831.50	847.456 3	828.94	828.94
C103	834.40	834.40	834.40	845.162 3	828.06	828.06
C104	842.10	845.60	842.10	840.125 6	824.78	824.78
C105	828.94	828.94	828.94	841.299 7	828.94	828.94
C106	833.70	833.70	833.70	839.156 3	828.94	828.94
C107	837.20	837.20	837.20	843.164 2	828.94	828.94
C108	833.60	835.30	830.60	845.372 6	828.94	828.94
C109	828.94	828.94	828.94	844.544 6	828.94	828.94
C201	—	—	—	596.914 5	591.56	591.56
C202	—	—	—	594.286 9	591.56	591.56
C203	—	—	—	598.178 9	591.17	591.17
C204	—	—	—	595.164 2	590.60	590.60
C205	—	—	—	593.654 1	588.88	588.88
C206	—	—	—	593.162 4	588.49	588.49
C207	—	—	—	595.156 2	588.29	588.29
C208	—	—	—	593.125 3	588.32	588.32

表 3 R 类算例测试结果比较

Table 3 Comparison of test results in R class example						
算例	ABC-C	ABC-S	IABC	HGA	AMA	IBFA
R101	1 618.30	1 618.30	1 618.30	1 687.246 5	1 650.80	1 642.88
R102	1 465.00	1 465.00	1 465.00	1 521.264 1	1 486.12	1 473.84
R103	1 207.00	1 207.00	1 207.00	1 326.854 3	1 292.68	1 222.05
R104	996.240	996.24	996.24	1 035.326 4	1 007.31	992.73
R105	1 394.30	1 398.80	1 390.50	1 413.648 5	1 377.11	1 370.80
R106	1 263.12	1 263.12	1 263.12	1 287.145 2	1 252.03	1 252.17
R107	1 126.30	1 126.30	1 126.30	1 134.834 6	1 104.66	1 095.50
R108	972.80	972.80	972.80	988.126 5	960.88	957.12
R109	1 028.50	1 028.50	1 028.50	1 226.478 9	1 194.73	1 156.70
R110	1 091.50	1 095.90	1 088.20	1 147.456 7	1 118.84	1 098.29
R111	1 099.46	1 099.46	1 099.46	1 128.526 4	1 096.73	1 060.93
R112	960.50	960.50	960.50	1 007.155 4	982.14	971.25
R201	—	—	—	1 257.462 8	1252.37	1 173.70
R202	—	—	—	1 201.452 1	1191.70	1 055.60
R203	—	—	—	945.124 5	939.50	882.10
R204	—	—	—	831.124 7	828.78	748.96
R205	—	—	—	1 002.369 1	994.43	982.32
R206	—	—	—	911.154 8	906.14	898.24
R207	—	—	—	902.651 4	890.61	827.05
R208	—	—	—	733.845 2	726.82	722.52
R209	—	—	—	915.145 2	909.16	890.96
R210	—	—	—	943.489 6	939.37	930.36
R211	—	—	—	896.741 2	891.11	776.07

表 4 RC 类算例测试结果比较

Table 4 Comparison of test results in RC class example						
算例	ABC-C	ABC-S	IABC	HGA	AMA	IBFA
RC101	—	—	—	1 730.196 5	1 696.95	1 654.10
RC102	—	—	—	1 587.154 2	1 554.75	1 481.06
RC103	—	—	—	1 290.874 2	1 261.67	1 296.90
RC104	—	—	—	1 163.187 4	1 135.52	1 167.90
RC105	—	—	—	1 655.621 3	1 629.44	1 539.59
RC106	—	—	—	1 483.162 8	1 424.73	1 404.80
RC107	—	—	—	1 253.353 2	1 230.48	1 256.90
RC108	—	—	—	1 167.593 5	1 139.82	1 160.90
RC201	1 263.10	1 263.10	1 258.60	1 413.465 9	1 406.94	1 288.80
RC202	1 178.90	1 178.90	1 178.90	1 375.793 6	1 365.64	1 143.10
RC203	1 083.60	1 083.60	1 083.60	1 057.156 4	1 049.62	960.74
RC204	799.12	799.12	799.12	804.854 2	798.46	795.01
RC205	1 330.70	1 330.70	1 321.30	1 307.694 8	1 297.65	1 190.00
RC206	1 171.20	1 171.20	1 171.20	1 154.234 6	1 146.32	1 097.10
RC207	1 096.50	1 096.50	1 096.50	1 068.731 6	1 061.14	992.18
RC208	833.97	833.97	833.97	834.144 2	828.14	786.13

在与ABC-C、ABC-S和IABC算法的比较中:由表2可以看出,对于C1类算例,IBFA算法除了C101求解性能较弱,其他算例均优于对比算法;由表3可以看出,对于R1类算例,IBFA算法除了R101、R102、R103、R109、R110和R112的求解性能较弱以外,其他算例都优于对比算法;由表4可以看出,对于RC类算例,IBFA算法除了RC201算例求解性能较弱以外,其他算例都优于对比算法。以上结论说明IBFA算法是有效的。

在与HGA算法的比较中:由表2可以看出,IBFA算法对于C类算例的求解性能均优于对比算法;由表3可以看出,IBFA算法对于R类算例的求解效果也优于对比算法;由表4可以看出,在RC类算例中,IBFA算法除了RC103、RC104和RC107求解性能弱于对比算法以外,其他算例均优于对比算法。整体而言,IBFA算法的改进效果较好。

在与AMA算法的比较中:由表2可以看出,在C类算例中,IBFA算法与对比算法取得了相同的效果;由表3可以看出,在R类算例中,IBFA算法仅有R106的求解性能劣于对比算法,其他算例均优于对比算法;由表4可以看出,在RC类算例中,IBFA算法仅有RC103、RC104、RC107和RC108的求解性能劣于对比算法,其他算例均优于对比算法。整体而言,IBFA算法的寻优效果较好。

为进一步验证IBFA算法的有效性,同样使用Solomon数据集进行测试,将其与已知最优的启发式结果进行比较,已知最优启发式结果中的数据从<http://w.cba.neu.edu/~msolomon/heuristi.htm>中选取。在表5~表7中,TD为已知启发式最优解,IBFA为本文算法得到的最优解,GAP表示本文算法最优解与已知启发式最优解之间的百分比偏差,用于评估算法的质量。

表5 C类算例中IBFA与已知最优解的比较

Table 5 Comparison of IBFA and known optimal solutions in C class example

算例	TD	IBFA	GAP/%
C101	828.94	828.94	0.00
C102	828.94	828.94	0.00
C103	828.06	828.06	0.00
C104	824.78	824.78	0.00
C105	828.94	828.94	0.00
C106	828.94	828.94	0.00
C107	828.94	828.94	0.00
C108	828.94	828.94	0.00
C109	828.94	828.94	0.00
C201	591.56	591.56	0.00
C202	591.56	591.56	0.00
C203	591.17	591.17	0.00
C204	590.60	590.60	0.00
C205	588.88	588.88	0.00
C206	588.49	588.49	0.00
C207	588.29	588.29	0.00
C208	588.32	588.32	0.00

表6 R类算例中IBFA与已知最优解的比较

Table 6 Comparison of IBFA and known optimal solutions in R class example

算例	TD	IBFA	GAP/%
R101	1 645.79	1 642.88	-0.18
R102	1 486.12	1 473.84	-0.83
R103	1 292.68	1 222.05	-5.78
R104	1 007.24	992.73	-1.46
R105	1 377.11	1 370.80	-0.46
R106	1 251.98	1 252.17	0.02
R107	1 104.66	1 095.50	-0.84
R108	960.88	957.12	-0.39
R109	1 194.73	1 156.70	-3.29
R110	1 118.59	1 098.29	-1.85
R111	1 096.72	1 060.93	-3.37
R112	982.14	971.25	-1.12
R201	1 252.37	1 173.70	-6.70
R202	1 191.70	1 055.60	-12.89
R203	939.54	882.10	-6.51
R204	825.52	748.96	-10.22
R205	994.42	982.32	-1.23
R206	906.14	898.24	-0.88
R207	893.33	827.05	-8.01
R208	726.75	722.52	-0.59
R209	909.16	890.96	-2.04
R210	939.34	930.36	-0.97
R211	892.71	776.07	-15.03

表7 RC类算例中IBFA与已知最优解的比较

Table 7 Comparison of IBFA and known optimal solutions in RC class example

算例	TD	IBFA	GAP/%
RC101	1 696.94	1 654.10	-2.59
RC102	1 554.75	1 481.06	-4.98
RC103	1 261.67	1 296.90	2.72
RC104	1 135.48	1 167.90	2.78
RC105	1 629.44	1 539.59	-5.84
RC106	1 424.73	1 404.80	-1.42
RC107	1 230.48	1 256.90	2.10
RC108	1 139.82	1 160.90	1.82
RC201	1 406.91	1 288.80	-9.16
RC202	1 367.09	1 143.10	-19.59
RC203	1 049.62	960.74	-9.25
RC204	798.41	795.01	-0.43
RC205	1 297.19	1 190.00	-9.01
RC206	1 146.32	1 097.10	-4.49
RC207	1 061.14	992.18	-6.95
RC208	828.14	786.13	-5.34

由表5可以看出,对于C1和C2类算例,IBFA算法与已知启发式最优解之间没有百分比偏差。

由表6可以看出,对于R类算例,IBFA算法除了R106的百分比偏差是正值以外,其余均是负值,范围为 -15.03% 到 -0.18% ,证明求解质量较好。由表7可以看出,对于RC类算例,IBFA算法中RC103、RC104、RC107和RC108的百分比偏差为正值,其余均是负值,范围为 -19.59% 到 -0.43% ,证明求解效果较好。

图9给出了IBFA算法与其他算法在不同算例上

的最短配送距离比较结果。由图9(a)和图9(b)可以看出,在C类算例上,IBFA算法基本取得了最佳效果。由图9(c)和图9(d)可以看出,改进细菌觅食算法整体上优于其他算法。由图9(e)和图9(f)可以看出,RC1类算例RC103、RC104、RC107和RC108效果比较差,其他算例效果整体较好。以上结论证明了IBFA算法改进效果较好。图10给出了IBFA算法的部分算例的路径规划结果。

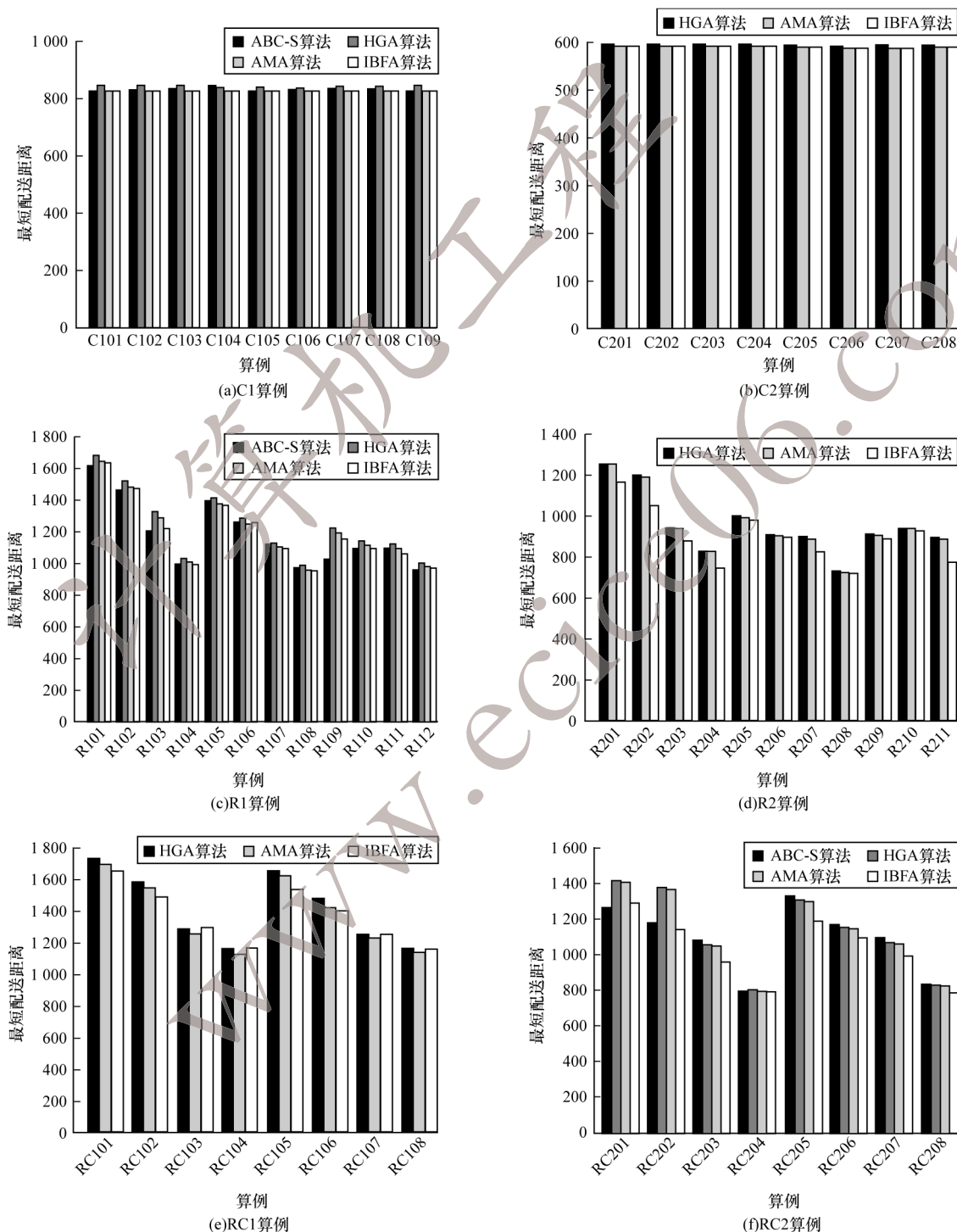


图9 6类算例上的算法最短配送距离比较

Fig.9 Comparison of the shortest distribution distances of algorithms on six class examples

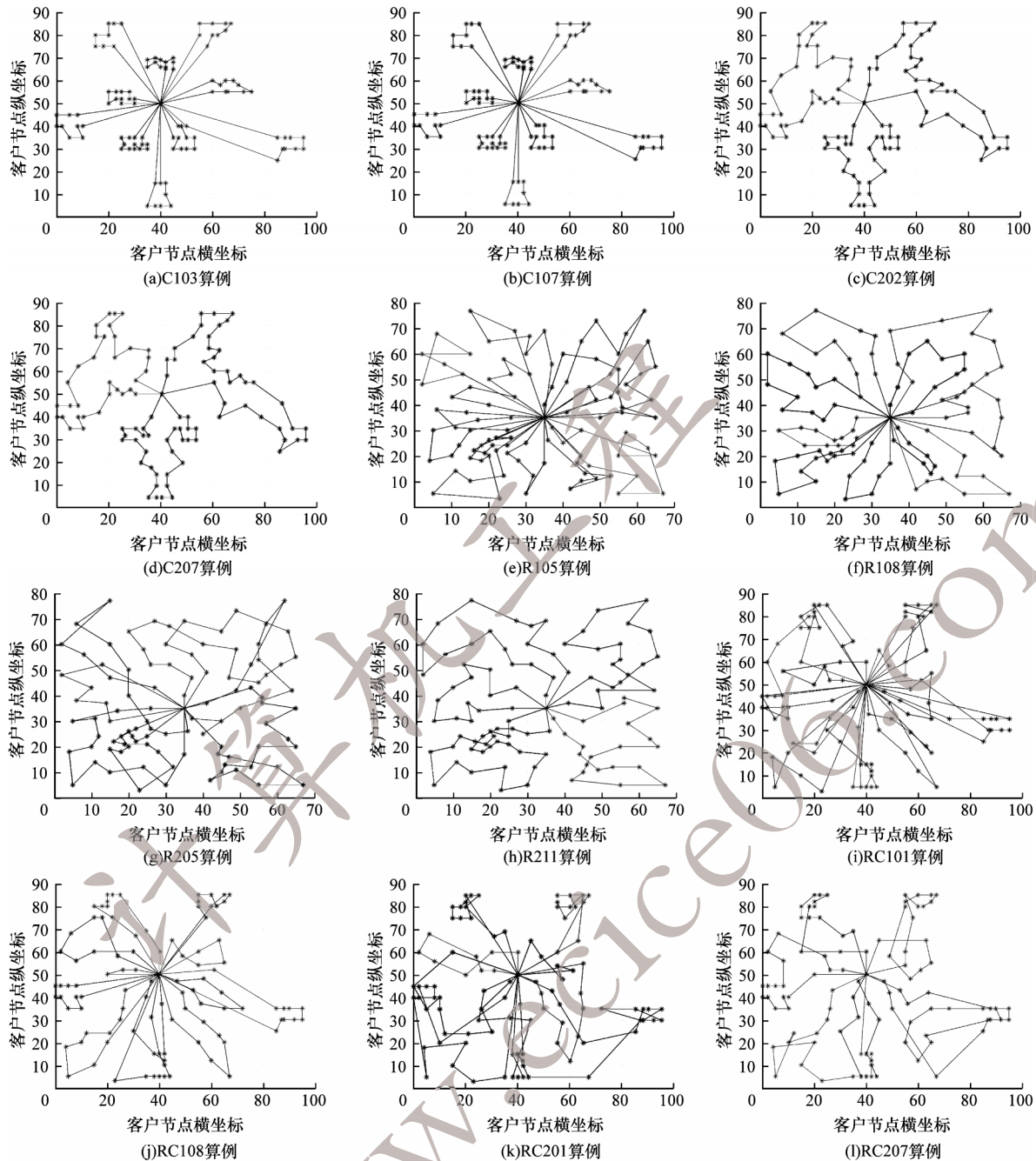


图10 部分算例的IBFA路径规划结果

Fig.10 IBFA path planning results of some examples

3.3 算法运行时间比较

为验证算法性能,将本文IBFA算法与ABC-C、ABC-S和IABC算法进行运行时间对比,依据文献[14]中的数据经过计算得到:IBFA算法在C类、R类以及RC类算例的平均运行时间分别为61.33 s、119.42 s和114.63 s;ABC-C算法在C类、R类以及RC类算例的平均运行时间分别为77.67 s、218.33 s和185.75 s;ABC-S算法在C类、R类以及RC类算例的平均运行时间分别为74.11 s、213.5 s和178.13 s;IABC算法在C类、R类以及RC类算例的平均运行时间分别为68.44 s、199.92 s和168.75 s。从平均运

行时间上看,IBFA算法在各类算例上的平均运行时间均优于ABC-C、ABC-S和IABC算法,证明IBFA算法运行效率较高。

4 结束语

本文提出一种改进的细菌觅食算法,用于求解带时间窗的车辆路径优化问题。将大邻域搜索方法与趋化操作相结合,设计4种removal算子,扩大搜索范围,提高求解质量。实验结果证明,改进算法具有较强的寻优能力,在不增加时间复杂度的基础上,能够得到更优的配送路径。下一步将从环境保护角

度出发,设计减少碳排放、提升客户满意度、降低配送成本的多目标函数,研究低碳环保的带时间窗车辆路径规划问题。

参考文献

- [1] 金淳,张雨,王聪. 带时间窗车辆路径问题的分布式多agent蚁群算法[J]. 计算机应用研究, 2018, 35(3): 666-670.
JIN C, ZHANG Y, WANG C. Distributed multi agent-based ant colony algorithm for vehicle routing problem with time windows[J]. Application Research of Computers, 2018, 35(3): 666-670. (in Chinese)
- [2] 徐廷学,张海军,付霖宇,等. 基于量子蚁群算法的VRPTW研究[J]. 火力与指挥控制, 2019, 44(8): 34-40.
XU T X, ZHANG H J, FU L Y, et al. Research on VRPTW based on quantum ant colony algorithm[J]. Fire Control & Command Control, 2019, 44(8): 34-40. (in Chinese)
- [3] GONG Y J, ZHANG J, LIU O, et al. Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(2): 254-267.
- [4] 李琳,刘士新,唐加福. 改进的蚁群算法求解带时间窗的车辆路径问题[J]. 控制与决策, 2010, 25(9): 1379-1383.
LI L, LIU S X, TANG J F. Improved ant colony algorithm for solving vehicle routing problem with time windows[J]. Control and Decision, 2010, 25(9): 1379-1383. (in Chinese)
- [5] 张丽萍,柴跃廷,曹瑞. 有时间窗车辆路径问题的改进遗传算法[J]. 计算机集成制造系统, 2002, 8(6): 451-454.
ZHANG L P, CHAI Y T, CAO R. Improved genetic algorithm for vehicle routing problem with time windows[J]. Computer Integrated Manufacturing Systems, 2002, 8(6): 451-454. (in Chinese)
- [6] LIU J S, LUO Z W, DUAN D Z, et al. A GA approach to vehicle routing problem with time windows considering loading constraints[J]. High Technology Letters, 2017, 23(1): 54-62.
- [7] WU Y X, SONG W, CAO Z G, et al. Learning improvement heuristics for solving routing problems[EB/OL]. [2020-07-11]. <https://arxiv.org/abs/1912.05784>.
- [8] CHEN B H, QU R, BAI R B, et al. A variable neighborhood search algorithm with reinforcement learning for a real-life periodic vehicle routing problem with time windows and open routes[J]. RAIRO-Operations Research, 2020, 54(5): 1467-1494.
- [9] PENG B, WANG J H, ZHANG Z Z. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems[EB/OL]. [2020-07-11]. <https://arxiv.org/abs/2002.03282>.
- [10] 余伶俐,魏亚东,霍淑欣. 基于MCPDDPG的智能车辆路径规划方法及应用[J]. 控制与决策, 2021, 36(4): 835-846.
YU L L, WEI Y D, HUO S X. Intelligent vehicle path planning method and application based on MCPDDPG[J]. Control and Decision, 2021, 36(4): 835-846. (in Chinese)
- [11] 张景玲,冯勤炳,赵燕伟,等. 基于强化学习的超启发算法求解有容量车辆路径问题[J]. 计算机集成制造系统, 2020, 26(4): 1118-1129.
- [12] PASSINO K M. Biomimicry of bacterial foraging for distributed optimization and control[J]. IEEE Control Systems Magazine, 2002, 22(3): 52-67.
- [13] 李奕颖,秦刚. 基于Spark的改进蚁群算法对带时间窗车辆路径问题的求解[J]. 计算机系统应用, 2019, 28(7): 9-16.
LI Y Y, QIN G. Solving vehicle routing problem with time window based on Spark's improved ant colony algorithm[J]. Computer Systems & Applications, 2019, 28(7): 9-16. (in Chinese)
- [14] 唐德权,黄金贵,史伟奇. 基于大数据平台的动态车辆路径调度算法[J]. 计算机工程, 2018, 44(1): 74-78.
TANG D Q, HUANG J G, SHI W Q. Dynamic vehicle routing schedule algorithm based on big-data platform[J]. Computer Engineering, 2018, 44(1): 74-78. (in Chinese)
- [15] YASSEN E T, AYOB M, NAZRI M Z A, et al. An adaptive hybrid algorithm for vehicle routing problems with time windows[J]. Computers & Industrial Engineering, 2017, 113: 382-391.
- [16] 孙沪增,李章维,秦子豪,等. 带时间窗车辆路径规划算法研究与实现[J]. 小型微型计算机系统, 2020, 41(5): 972-978.
SUN H Z, LI Z W, QIN Z H, et al. Research and implementation on time-window vehicle routing problem[J]. Journal of Chinese Computer Systems, 2020, 41(5): 972-978. (in Chinese)
- [17] QI Y T, HOU Z T, LI H, et al. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows[J]. Computers & Operations Research, 2015, 62: 61-77.
- [18] DENG Y, ZHU W H, LI H W, et al. Multi-type ant system algorithm for the time dependent vehicle routing problem with time windows[J]. Journal of Systems Engineering and Electronics, 2018, 29(3): 625-638.
- [19] SHAW P. Using constraint programming and local search methods to solve vehicle routing problems[C]//Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming. New York, USA: ACM Press, 1998: 417-431.
- [20] YAO B Z, YAN Q Q, ZHANG M J, et al. Improved artificial bee colony algorithm for vehicle routing problem with time windows[J]. PLoS One, 2017, 12(9): 18-27.
- [21] 仪孝展. 基于改进遗传算法的物流车辆路径规划方法研究与应用[D]. 西安: 西安理工大学, 2018.
YI X Z. Research and application of logistics vehicle route planning based on improved genetic algorithm[D]. Xi'an: Xi'an University of Technology, 2018. (in Chinese)
- [22] NALEPA J, BLOCHO M. Adaptive memetic algorithm for minimizing distance in the vehicle routing problem with time windows[J]. Soft Computing, 2016, 20(6): 2309-2327.

编辑 陆燕菲