系 统 科 学 与 数 学
J. Sys. Sci. & Math. Scis.
46(2) (2026, 2), 642–656

一种基于多分治策略的大规模多中心 CARP 求解算法*

张玉州[1] 梅 一[2] 江健生[3] 张海奇[4] 赵 芬[1]

(1. 南京晓庄学院信息工程学院, 南京 211171; 2. 惠灵顿维多利亚大学计算机科学与工程学院, 惠灵顿 6012;
3. 安庆师范大学计算机与信息学院, 安庆 246133; 4. 南京理工大学计算机科学与工程学院, 南京 210094)

摘要 多中心限量弧路由问题 (multi-depot capacitated arc routing problem, MDCARP) 是基本 CARP (capacitated arc routing problem) 的重要扩展模型, 任务可归属于任一中心点, 较 CARP 更为复杂. 大规模 MDCARP (large scale multi-depot capacitated arc routing problem, LSMDCARP) 在问题结构以及解空间等方面均极具挑战性. 虽然 RoCaSH2 在较大规模的 MDCARP 测试集上, 取得了较其他算法好的结果, 当问题规模上升至一定程度后, 子 CARP 亦呈规模较大状态, 其性能会受到其子 CARP 求解过程的影响. 鉴于此, 文章在该算法的基础上, 设计了一种基于多分治策略的 LSMDCARP 求解方法 MDCSbA (multiple divide-and-conquer strategies based approach), 其中分治策略分布于 MDCARP 的分解、子问题间的协同以及子问题的分解, 寻求在多个环节上降低问题的复杂性. 为了验证 MDCSbA 对于 LSMDCARP 的有效性, 在规模至 3584 的测试集上进行了验算, 结果表明, 文章算法在相同的计算时间内, 效果明显优于包括 RoCaSH2 在内的其他算法.

关键词 组合优化, 大规模, 限量弧路由问题, 多中心, 分治策略.

MR(2020) 主题分类号 90B20

DOI 10.12341/jssms240656

A Multiple Divide-and-Conquer Strategies Based Approach to Large Scale Multi-Depot Capacitated Arc Routing Problem

ZHANG Yuzhou[1] MEI Yi[2] JIANG Jiansheng[3] ZHANG Haiqi[4] ZHAO Fen[1]

(1. *School of Information Engineering, Nanjing Xiaozhuang University, Nanjing* 211171; 2. *School of Engineering and Computer Science, Victoria University of Wellington, Wellington* 6012; 3. *School of*

* 安徽省自然科学基金 (1808085MF173), 南京晓庄学院高层次人才科研启动项目 (4172322), 安徽省教育厅科研计划项目重点项目 (2024AH051099), 并行与分布处理国防科技重点实验室开放课题 (WDZC202252501) 资助课题.
收稿日期: 2024-08-21, 收到修改稿日期: 2024-12-04.
通信作者: 张玉州, Email: yzhzhang@mail.ustc.edu.cn.
编委: 张新雨.

Computer and Information, Anqing Normal University, Anqing 246133; 4. *School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing* 210094)

Abstract Multi-depot capacitated arc routing problem (MDCARP) is an important extension version of the basic capacitated arc routing problem (CARP). In MDCARP, each task can be assigned to any depot, so MDCARP is more complex than CARP. Large scale multi-depot capacitated arc routing problem (LSMDCARP) is extremely challenging on the problem structure and the vast solution space. Though, RoCaSH2 is a state-of-the-art approach which outperformed the other algorithms on the test data with a larger scale, it will encounter the straits when the problem scale ranges up to a certain level in which the scale of the sub-CARPs appears to be larger. In RoCaSH2, the operator for the sub-CARP ignores the scalability of the problem. In view of this, a multiple divide-and-conquer strategies based approach (MDCSbA) is proposed for LSMDCARP based on RoCaSH2, where the divide-and-conquer strategies are distributed in the decomposition of MDCARP, the collaborative optimization among sub-CARPs and the decomposition of sub-CARPs. As a result, the complexity of the problem is decreased at multiply stages. In order to verify the effectiveness of MDCSbA, it is evaluated on two test sets (i.e., mdHefei and mdBeijing) in which the number of the tasks is up to 3584. The results show that MDCSbA can outperform the compared algorithms significantly within runtime budget.

Keywords Combinatorial optimization, large scale, capacitated arc routing problem, multi-depot, divide-and-conquer strategy.

1 引 言

限量弧路由问题 (capacitated arc routing problem, CARP) 最先由 Golden 和 Wong [1] 于 1981 年提出, 因其在实际生活与工程中的广泛应用背景, 而备受关注, 如邮件投递、道路保养、垃圾回收、冬季铲雪、校园巴士等 [2]. CARP 的目标是为驻扎在中心点处的车队规划路径, 以便对分布于道路网中的需求给予服务, 在满足特定约束条件下, 总费用最小化. 显然, CARP 属于典型的组合优化问题, 也已被证明为 NP-hard 问题 [1], 且已有大量的研究成果产生 [3–5].

然而, 随着应用的不断拓广, 基本 CARP 模型难以应对. 对基本模型进行扩展, 适应实际问题的需要, 成为 CARP 近年来研究的热点, 如周期型 CARP [6]、异车型 CARP [7]、多中心 CARP (multi-depot CARP, MDCARP) [8]、不确定 CARP [9, 10]、多层 CARP [11] 及软聚束 CARP [12] 等. 在 MDCARP 中, 存在多个中心点, 车辆可驻扎在任一中心点处, 现实中城市多回收站街道垃圾清扫问题可归属该问题模型. 对于 MDCARP 而言, 道路需求可归属至任一中心点处被服务, 显然相对于基本 CARP, MDCARP 的解空间呈 $n_d^{n_v}$ 倍增长 [13], n_d、n_v 分别为中心点数和车辆数. 所以 MDCARP 较 CARP 更为复杂, 亦为 NP-hard 问题.

尽管 MDCARP 具有很高的实际应用价值, 相比较 CARP 的充分研究, 其受关注程度较低. Zhu 等 [8] 与 Kansou 和 Yassine [14] 分别提出了混合遗传算法 (hybrid genetic algorithm, HGA) 以及多中心文化基因算法 (multi-depot memetic algorithm, MDMA), 并在小规模算例上进行了验证. 然而, 在现实世界里的问题往往呈现大规模状态, 如垃圾清扫问题中的街道数突破千条. 另一方面, MDCARP 的解空间随着问题规模的增长呈指数级增长, 所以传统算法难以应对大规

模 MDCARP (large scale MDCARP, LSMDCARP). 如何在巨大的解空间中进行有效搜索, 在有限的时间内寻找问题的优秀解, 是 LSMDCARP 面临的挑战.

面对大规模或复杂的问题, 分治策略能够将原问题分解为若干相对独立的子问题, 随后分别对子问题求解, 并将所有子解组合, 获得原问题的解. 显然, 分治策略能够迅速降低问题的规模或复杂性, 使得搜索在相对较小的空间中进行, 从而该策略被广泛采用 [15–18]. 分治策略的关键在于对原问题的分解策略的制定, Zhang 等 [19] 依据 MDCARP 的结构特点, 设计了一种三标准任务分配策略, 将问题中的任务分配至合适的中心点, 然后由中心点及分配至其的任务构成若干子 CARP, 对子 CARP 求解, 即规划各中心点对应车辆的路径. 合并所有子 CARP 对应的解, 形成路径集合, 并使用三标准策略对路径重新聚类至各中心点, 从而提出了路径聚类搜索启发式方法 (route clustering and search heuristic, RoCaSH). RoCaSH 以迭代为算法框架, 对分配至中心点处的路径重新规划, 以取得更优的问题解. RoCaSH 在三种规模测试集上进行了验证, 分别为小规模测试集 mdgdb、中等规模 mdegl 和大规模测试集 mdHefei、mdBeijing, 取得了较 HGA、MDMA 更好的效果, 尤其在大规模测试集 mdHefei、mdBeijing 上, RoCaSH 显示了良好的问题求解能力.

对于 MDCARP, 任意一对任务均可安排于同一条路径进行服务, 并且相互的位置对于总费用产生影响, 所以 MDCARP 属于完全不可分离组合优化问题. RoCaSH 中各子 CARP 独立求解, 以至于某些适合安排在同一条路径上的任务被隔离开, 影响了解的质量. 尽管 RoCaSH 采取了迭代结构, 然而后期以路径为对象的聚类策略难以顾及上述任务. 由此, Zhang 等 [20] 提出了子 CARP 之间任务移动算子 (task moving among sub-problems, TMaS), 进一步加强子问题之间的交流, 对任务重新寻找合适的中心点和路径中的位置. 对于 TMaS, 若以整个 MDCARP 为对象, 进行任务重新寻址, 则工作量巨大. 所以采取类似于 POPMUSIC 策略 [21] 设计 TMaS, 仅对距离较近且来自于不同中心点路径中的任务进行局部探索, 以更高效地为其重定位. TMaS 嵌入 RoCaSH, 方便起见, 记所形成的算法为 RoCaSH2, 实验证明了其对于 LSMDCARP, 较 RoCaSH 更有效.

当任务数上升至一定程度, LSMDCARP 经过分解, 子问题往往亦为较大规模. 所以, 对子 CARP 采用传统方法处理, 则难以取得满意的问题解. 故而, 对子 CARP 进一步采用分治策略处理, 是 RoCaSH2 需要更进一步考虑的问题.

Mei 等 [22] 提出了基于路径距离的聚类策略 RDG, 并结合 MAENS [23], 形成 RDG-MAENS 对大规模 CARP 问题进行求解, Shang 等 [24] 对 RDG-MAENS 进行了改进, 提出了 IRDG-MAENS 对大规模多目标 CARP 进行求解. 文献 RDG-MAENS 及 IRDG-MAENS 的最大规模测试算例中的任务数为 375, 但此规模往往远不及现实中问题的任务数. 鉴于此, Tang 等 [25] 提出了一种层次化的问题分解 (hierarchical decomposition, HD) 策略, 以虚拟任务形式对聚类的子解进行封装, 由底层向顶层逐层聚合, 从而设计了基于 HD 的方法 SAHiD. 纵观 RDG 与 HD, RDG 以线性方式对问题进行分解, 而 HD 则以层次化的方式, 显然 HD 较 RDG 更高效, 因其呈对数级强度对问题分解. 并且, Tang 等 [25] 依据合肥和北京地图, 设计了两个测试集 Hefei 与 Beijing, 各包含 10 个算例, 其中 Hefei 算例规模至 1212, 而 Beijing 则上升至 3584. 两个测试集上的计算结果表明, SAHiD 明显好于其他算法, 包括 RDG-MAENS.

本文旨在解决任务数上升至一定程度的 MDCARP, 为此, 在 RoCaSH2 的基础上, 引入层次分解策略 HD, 并对其进行改进, 重新设计子 CARP 的求解方案, 以期能够高效地解决各子 CARP, 从而为大规模 MDCARP 的快速解决赢取时间. 显然, 在所设计的算法中, 分治策略采纳

于三处: 1) 任务或路径集合的划分, 用于原问题的分解; 2) 子问题之间协同交流算子的设计, 使得协同过程轻量化; 3) 子问题的求解过程, 实现问题的进一步分解. 分治结构如图 1 所示, 记本文的算法为 MDCSbA (multiple divide-and-conquer strategies based approach).

图 1　分治结构框架

(Figure 1　The framework of divide-and-conquer structure)

2　问题模型

2.1　问题描述

MDCARP 可描述为给定无向图 $G=(V_N,V_D,E)$ 上寻求满足一定约束条件的最优路径集合, 其中 V_N、V_D 分别代表非中心点顶点集合和中心点顶点集合, E 为连接顶点的边集合. 由集合 V_N、V_D 构成顶点集合 V, 即 $V=V_N\cup V_D$. 集合 V_D 各中心点处驻扎了具有相同容量 Q 的车队 K, 以为需求边提供服务. 边集 E 中, 每条边 e 可由非负属性值: 需求量 $d(e)$, 服务费用 $sc(e)$ 和经过费用 $dc(e)$ 进行描述. 在边集 E 中, 存在一些需求边, 其需求量为正值, 即 $d(e)>0$, 称这样的边为任务, 所有任务构成任务集 T, 即 $T=\{e|e\in E\wedge d(e)>0\}$. 表 1 列出了问题描述中符号及其含义. MDCARP 的目标为合理地分解任务集 T 至 V_D 各中心点处, 由驻扎在中心点处的车辆为其服务, 同时要求行车服务费用最小化, 并满足以下约束条件:

1) 车辆由某一中心点出发, 最后返回同一中心点;

2) 车辆进行服务的总需求量不能超过容量 Q;

3) 任务集 T 中任一任务均被服务, 且只服务一次.

表 1　MDCARP 模型符号含义说明表

(Table 1　The symbols used in the definition of of MDCARP)

符号	含义说明	符号	含义说明
V_N	非中心点顶点集合	t	任务集合 T 中的一个任务
V_D	中心点顶点集合	K	车辆集合
E	边集合	Q	车的最大负载量
e	边集合 E 中的一条边	$head(t)$	任务 t 的第一端点
$d(e)$	边 e 上的需求量	$tail(t)$	任务 t 的第二端点
$sc(e)$	边 e 上的服务费用	$inv(id)$	标号 id 对应任务的反向标号
$dc(e)$	边 e 上的经过费用	$\mathfrak{R}$	一条路径
T	任务集合	$\mathbb{S}$	MDCARP 的一个解

由于边集 E 的无向性, 即任意边 e 均可按照两个方向经过. 为了便于描述, 为任务 $t=(x,y)$ 分配两个整数标号 $id1$, $id2$, 分别代表正反两个方向. 对任务 t 的标号 $id1$ 而言, 它的反向标号

$id2$ 可记作 $inv(id1)$, 即 $id2 = inv(id1)$, 同时 $id1 = inv(id2)$. 值得注意的是, 任务正反向的所有属性值均相同, 即具有等值的服务需求量、服务费用以及路径费用. 由于中心点是车辆的必经顶点, 所以其被认为特殊的任务, 所有属性值均为 0, 且为每个中心点分配一标号. 任务是 MDCARP 中的基本单位, 为了表示其对应的两个端点, 定义 $head(\cdot)$、$tail(\cdot)$ 分别表示任务的第一和第二端点. 如 $t = (x, y)$ 的标号 $id1$ 和 $id2$ 分别代表正向边 (x, y) 和反向边 (y, x), 则有 $head(id1) = tail(id2) = x, tail(id1) = head(id2) = y$. 图 2 为一包含 3 个中心点、7 个任务的 MDCARP 示例.

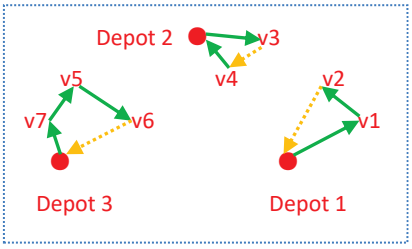


图 2 MDCARP 示例

(Figure 2 An example of MDCARP)

2.2 数学模型

MDCARP 解可表示为若干条路径的集合, 而每条路径则是由相应车辆所服务的任务组成, 即

$$\mathfrak{R}_{d,k} = (t_{dk1}, \cdots, t_{dk|\mathfrak{R}_{d,k}|}), \tag{2.1}$$

其中, $\mathfrak{R}_{d,k}$ 表示由中心点 d 出发的第 k 条路径, $|\mathfrak{R}_{d,k}|$ 表示该路径的任务数, t_{dk1} 与 $t_{dk|\mathfrak{R}_{d,k}|}$ 为中心点 d 对应任务标号. 由中心点 d 出发的路径集合:

$$\mathfrak{R}_d = \{\mathfrak{R}_{d,1}, \cdots, \mathfrak{R}_{d,|\mathfrak{R}_d|}\}, \tag{2.2}$$

其中, $|\mathfrak{R}_d|$ 表示由中心点 d 出发的路径数, 从而问题解 $\mathbb{S}$ 可表示为

$$\mathbb{S} = \{\mathfrak{R}_d | d \in V_D\}. \tag{2.3}$$

路径 $\mathfrak{R}_{d,k}$ 的负载量 $load(\mathfrak{R}_{d,k})$ 及费用消耗 $cost(\mathfrak{R}_{d,k})$ 分别为

$$load(\mathfrak{R}_{d,k}) = \sum_{i=2}^{|\mathfrak{R}_{d,k}|-1} d(t_{dki}), \tag{2.4}$$

$$cost(\mathfrak{R}_{d,k}) = \sum_{i=1}^{|\mathfrak{R}_{d,k}|-1} (\Delta(tail(t_{dki}), head(t_{dk(i+1)})) + sc(t_{dki})), \tag{2.5}$$

其中, $\Delta(v_1, v_2)$ 代表顶点 v_1 与 v_2 的最短距离, 可由 Dijkstra 算法进行计算 [26]. 式 (2.5) 中 t_{dk1} 与 $t_{dk|\mathfrak{R}_{d,k}|}$ 为中心点 d 对应任务标号, 其对应的 $head(\cdot)$、$tail(\cdot)$ 均为自身, 且无服务费用. 问题解 $\mathbb{S}$ 的总耗费计算如下:

$$tc(\mathbb{S}) = \sum_{d \in V_D} \sum_{k=1}^{|\mathfrak{R}_d|} cost(\mathfrak{R}_{d,k}). \tag{2.6}$$

综上所述, MDCARP 可描述如下:

$$\min \quad tc(\mathbb{S}) \tag{2.7}$$

$$\text{s.t.} \quad t_{dk1} = t_{dk|\mathfrak{R}_{d,k}|} = t_d, t_{dk1}, t_{dk|\mathfrak{R}_{d,k}|} \in \mathfrak{R}_{d,k}, \quad \forall d \in V_D, \quad \forall k \in 1, \cdots, |\mathfrak{R}_d|, \tag{2.8}$$

$$\bigcup_{d\in V_D} \bigcup_{k=1}^{|\mathfrak{R}_d|} \bigcup_{i=2}^{|\mathfrak{R}_{d,k}|-1} \mathfrak{R}_{d,k}[i] = T, \tag{2.9}$$

$$\sum_{d\in V_D} \sum_{k=1}^{|\mathfrak{R}_d|} (|\mathfrak{R}_{d,k}| - 2) = |T|, \tag{2.10}$$

$$load(\mathfrak{R}_{d,k}) \le Q, \quad \forall d \in V_D, \quad \forall k \in 1, \cdots, |\mathfrak{R}_d|, \tag{2.11}$$

式 (2.8) 中, t_d 表示由中心点 d 构成的虚拟任务, 式 (2.9) 中, $\mathfrak{R}_{d,k}[i]$ 表示路径 $\mathfrak{R}_{d,k}$ 中的第 i 个任务. 式 (2.7) 为问题目标函数, 旨在最小化由式 (2.6) 计算的总耗费. 式 (2.8) 表示车辆起止于同一中心点; 式 (2.9)、(2.10) 确保每个任务被服务且仅被服务一次; 式 (2.11) 则对应车容量的约束条件.

3　算法设计

鉴于 MDCARP 的复杂性, RoCaSH2[20] 依据问题的多中心结构特点, 首先对任务集 T 进行分解, 并生成初始解. 继而对子问题解进行交流, 为任务寻找更理想的位置. 归并所有子解的路径, 并作重新分配, 形成以路径为组成元素的子问题. 采用局部搜索策略对子问题优化, 获取更优的子解. 重复上述步骤, 直至算法终止条件满足. 由于该算法中子 CARP 的求解借助于局部搜索, 当 MDCARP 中任务数达到一定规模, 显然会使得子解陷入局部最优陷阱, 而无法获取良好的问题解. 鉴于此, MDCSbA 在文献 [20] 研究基础上, 聚焦子问题的有效解决, 采用分治法进一步分解基本 CARP, 既保证解的质量, 且能满足大规模 MDCARP 求解的高效性.

3.1　子 CARP 的求解

当子 CARP 规模较大时, 子解空间依然巨大, 采用分治策略, 进行规模降解, 使得搜索能够在有效区域内进行. MDCSbA 采用文献 [25] 所提层次分解策略 HD 进行问题分解, 并结合有效的局部搜索对子 CARP 求解.

3.1.1　HD 策略

层次分解策略 HD[25] 的基本思想是由底层向顶层逐层进行问题聚合, 且采用虚拟任务概念对分解的任务子集进行封装. 对于第一层, 即最底层中的任务集合 $\{t_1^1, \cdots, t_{l_1}^1\}$, 采用某种聚类方法进行问题分解, 得到 l_2 个任务子集 $\{t_1^1, \cdots, t_{n_1}^1\}$, $\{t_{n_1+1}^1, \cdots, t_{n_2}^1\}$, $\cdots$, $\{t_{n_{l_2-1}+1}^1, \cdots, t_{l_1}^1\}$. 采用最优插入法 (best insertion heuristic, BIH) 对每个子集中的任务进行排列, 并将该结构固定, 形成虚拟任务集 $\{t_1^2, \cdots, t_{l_2}^2\}$. 进入第二层时, $\{t_1^2, \cdots, t_{l_2}^2\}$ 中的每个虚拟任务视为整体参与问题分解, 形成 l_3 个任务子集. 如此, 逐层重复相同过程的聚类, 直至最顶层 h, 形成一个包含所有任务的虚拟任务, 即 l_h=1.

3.1.2　参数改进

在 HD 中, 包含一项重要参数, 每层中的虚拟任务的划分子集数 l_i, 文献 [25] 中采用随机产

生的方式:

$$l_i = random[1, \beta \cdot l_{i-1}], \tag{3.1}$$

其中, β 取恒定值. 由式 (3.1) 知, 当前层的分解子集数在 $[1, \beta \cdot l_{i-1}]$ 之间随机产生, 其中 l_{i-1} 是其底层所产生的虚拟任务数. 然而, 随着算法迭代的进行, 当解的质量不断得到提高时, 说明当前 β 值对子集数的产生具有积极意义, 可维持该值不变化. 反之, 若当解的质量得不到改善时, 可减少划分的子集数, 如此可以使得子集对应问题可以在更大的空间中进行搜索, 从而能够获取更优的局部解. 为此, β 值需要适当较少. 本文 β 按如下公式取值:

$$\beta = \begin{cases} UpValue, & \beta < LowValue, \\ \beta - \theta \cdot noimp, & \text{其他}, \end{cases} \tag{3.2}$$

式中, $UpValue$ 和 $LowValue$ 为 β 的上下界值, $noimp$ 为解连续未改善的代数, θ 为 β 下降比例系数. 初始时, 赋予 β 上界值 $UpValue$, 当 β 值低于下界值 $LowValue$ 时, 重新恢复至上界值 $UpValue$.

3.1.3 邻域搜索

对于复杂问题, 进行邻域进行搜索, 有助于发现更优的解, 这是改善复杂问题解的一种重要策略. 本文的邻域搜索建立在三种邻域结构上:

$N1$: 将一任务移至同一路径中另一任务之后;

$N2$: 将连续两个任务移至同一路径中另一任务之后;

$N3$: 交换同一路径中两个任务.

三种结构均基于单条路径的, 即只针对某一路径上任务的移动或交换, 且在重新为任务安排新的位置时, 考虑方向性. 基于上述三种结构的邻域为 $\boldsymbol{N}$=$\{N1, N2, N3\}$. 然而, 在整个邻域 $\boldsymbol{N}$ 上进行探索, 无疑会增加搜索量, 降低算法的效率. 去除 $\boldsymbol{N}$ 中对搜索进展意义不大的邻域, 缩小范围, 对搜索具有积极的意义. MDCSbA 在 Zhang 等 [27] 所提用于刻画任务间距概念 (任务等级) 的基础上, 进一步限制邻域 $\boldsymbol{N}$ 搜索. 为了定义任务等级概念, 首先给出任务 t_i 与 t_j 间的距离计算公式:

$$\begin{aligned} \delta_{task}(t_i, t_j) =& \frac{1}{4}(\Delta(head(t_i), head(t_j)) + \Delta(head(t_i), tail(t_j)) + \Delta(tail(t_i), head(t_j)) \\ &+ \Delta(tail(t_i), tail(t_j))), \end{aligned} \tag{3.3}$$

其中, $\Delta(v_1, v_2)$ 为顶点 v_1, v_2 的最短距离.

定义 1 [27] 令任务集 $T = t_d, t_1, \cdots, t_n$, t_d 为中心点 d 对应任务标号, n 为任务数. 对于任务 t_i, 其余任务依据距离升序排列: $(t_1^i, \cdots, t_n^i)$, 即 $\delta_{task}(t_i, t_j^i) \le \delta_{task}(t_i, t_k^i)$, $j \le k$, 则 t_1^i 的等级定义为 1, 即 $\mathrm{rank}(t_i, t_1^i) = 1$, 后续任务的等级依次为 2, 3, $\cdots$. 若 $\delta_{task}(t_i, t_j^i) = \delta_{task}(t_i, t_{j+1}^i)$, 则 t_j^i 与 t_{j+1}^i 分配相同的等级, 亦即 $\mathrm{rank}(t_i, t_j^i) = \mathrm{rank}(t_i, t_{j+1}^i)$.

由等级定义知, 等级值 $\mathrm{rank}(t_i, t_p)$ 越小, 则 t_p 越接近 t_i, 反之则有长距离的间隔. 所以, 可以通过等级进行邻域 $\boldsymbol{N}$ 的缩小, 如 t_i 在进行 $N1$ 邻域结构搜索时, 对于另一任务 t_p, 若 $\mathrm{rank}(t_i, t_p) > UpRank$, 则直接过滤 t_p 后位置的搜索, $UpRank$ 为搜索等级阈值, 如此形成邻域结构 $N1^*$. 同法, 可以分别为 $N2$ 以及 $N3$ 生成压缩邻域结构 $N2^*$ 和 $N3^*$, 从而新的邻域范围为 $\boldsymbol{N}^* = \{N1^*, N2^*, N3^*\}$.

3.1.4 子 CARP 求解算法

令当前中心点为 d, 子 CARP 为 P_d, 基于上述描述以及文献 [25] 大规模 CARP 的求解过程, MDCSbA 中子问题 P_d 的求解算法步骤如下:

步骤 1 依据 $Noimp$ 及公式 (3.2) 计算 β;

步骤 2 使用 HD 策略对 P_d 进行分解, 形成虚拟任务 VT_d, 其包含 P_d 中所有任务;

步骤 3 使用 2-OPT 对 VT_d 进行逆序搜索;

步骤 4 使用 Ulusoy Splitting 算法 [28] 对 VT_d 进行路径分割, 得到 P_d 的解 S_d;

步骤 5 采用第 3.1.3 小节所定义的结构 $\boldsymbol{N}^*$ 对 S_d 进行邻域搜索;

步骤 6 若 S_d 未得到改善, 则对其采用 MS [23] 进行大尺度的邻域搜索;

步骤 7 返回 S_d.

步骤 3 中对 VT_d 进行 2-OPT 搜索, 相当于一次全局搜索, 因为无论步骤 5 中的传统邻域搜索, 还是步骤 6 中大尺度搜索 MS, 均是基于局部的, 所以 2-OPT 预先给予了全局搜索. 步骤 6 中的 MS 搜索是对步骤 5 小尺度邻域搜索的补充, 其通过不断合并、切割路径, 从而获取更优的邻域解.

3.2 算法其他部分

上述基于 HD 分解策略的子 CARP 求解方法是 MDCSbA 重要组成部分. 除此之外, RoCaSH2 提出的种群初始化以及子问题交流过程亦基于分治策略, 在 MDCSbA 中被沿用.

3.2.1 初始化过程

RoCaSH2 是基于单个体而非群体搜索方法, 通过两种方式产生一定量的解, 择优作为初始解. 两种方式分别为: 1) 将所有任务依据距离分配至最近的中心点处, 对于每个中心点所形成的子问题, 采用 BIH 构造子解; 2) 每次随机选择一任务, 分配至最近的中心点, 并采用 BIH 方法将其插入到已构建的子解中. 显然, 第二种随机方法可以增加群体的多样性.

3.2.2 子问题交流算子

令当前 MDCARP 的解为 $S = S_1 \cup \cdots \cup S_{|V_D|}$, 其中子解 $S_d\,(d \in V_D)$ 由中心点 d 出发的路径组成, 即 $S_d = \mathfrak{R}_d$. 对任务进行跨子问题的邻域搜索, 如此, 会使得位于不同子问题中联系紧密的任务能够聚集在同一条路径上, 经过后续的路径再分配, 从而进入同一子问题中, 进一步提高解的质量. 为此, 在 RoCaSH2 中, 设计了子问题间进行任务合适位置移动的算子 TMaS. 如何进行寻觅任务的合适位置, 如何移动, 这是关键问题. 对于某一任务, 在整个问题范围内搜索理想的位置, 会增加问题解决过程的复杂性.

TMaS 采用了类似于 POPMUSIC 机制, 对于某子解中路径, 在其他子解中寻找一定量距其较近的路径, 形成路径集合. 对该路径集合采用类似于邻域结构 $N1$ 和 $N3$ 进行探索, 但仅在来自于不同子解的路径之间进行, 如此可以提升搜索效率.

3.3 MDCSbA 算法

基于上述描述, 基于多分治策略 MDCSbA 具体步骤描述如下:

步骤 1 使用最近距离法生成初始解决方案 $S = S_1 \cup \cdots \cup S_{|V_D|}$;

步骤 2 $noimp \leftarrow 0$; //用于公式 (3.2) 中 β 计算;

步骤 3 $improve \leftarrow false$;

步骤 4 $S' \leftarrow S$;

步骤 5 采用 TMaS 算子对 S' 各子解中任务进行理想位置搜寻, 并移动位置. 令改变后的所有路径构成集合 TRS 中, 并置 RS 为空集, 即 $RS \leftarrow \emptyset$;

步骤 6 对于 TRS 所有路径采用 RCO [27] 路径切割技术进行切割, 形成较短的路径, 添加至 RS 中;

步骤 7 使用文献 [19] 中三标准分解策略将 RS 中的路径分配至 V_D 的相应中心点处, 得到子问题 $P_1, \cdots, P_{|V_D|}$;

步骤 8 $i \leftarrow 1$;

步骤 9 对子问题 P_{d_i} 采用第 3.1 小节所设计的算法求解, 得到子解 $S^*_{d_i}$;

步骤 10 $i \leftarrow i+1$;

步骤 11 若 $i \leq |V_D|$, 则转步骤 9, 继续求解子问题;

步骤 12 连接所有子解, 得到完整解 S^*, 即 $S^* = S^*_1 \cup \cdots \cup S^*_{|V_D|}$;

步骤 13 若 S^* 优于 S, 则以 S^* 更新 S, 即 $S \leftarrow S^*$, 并置 $improve \leftarrow true$;

步骤 14 若 $improve = true$, 则置 $noimp \leftarrow 0$. 否则置 $noimp \leftarrow noimp + 1$;

步骤 15 若停止条件不满足, 则转步骤 3 继续执行, 否则输出解 S.

MDCSbA 算法中, 步骤 1、5、7 以及 9 均基于分治策略而设计, 分别为 MDCARP 的初始解生成、子问题之间的交流、路径重新划分以及子 CARP 的求解, 能够有效地降低问题的规模. 同时, 通过迭代的方式, 弥补由于任务间的紧密联系而导致的步骤 1 和步骤 7 分解策略不够精准的缺陷, 步骤 9 中含有子解的邻域搜索, 亦是对 HD 分解中以虚拟任务间缺少搜索的填补.

4 实验及分析

本文采用文献 [19] 生成的两个 MDCARP 测试集 mdHefei 与 mdBeijing 对所提 MDCSbA 进行实验测试, 以检测其在大规模问题上的性能, 实验结果与有代表性的优秀算法 RoCaSH2、HGA 以及 MDMA 结果进行对比. 由于 ROCASH2 的性能优于 ROCASH [19], 故而 MDCSbA 未与 ROCASH 进行比较. 另外, Vidal [29] 设计了一种混合遗传搜索 (unified hybrid genetic search, UHGS) 用于 CARP 及其扩展模型的求解. 本文亦将 UHGS 作为对比算法, 其结果文献取自文献 [20].

4.1 测试集及运行环境

测试集 mdHefei 与 mdBeijing 来源于 Tang 等 [25] 设计的基本 CARP 测试集 Hefei 和 Beijing, 它们来自于合肥和北京的实际地图. 对于 Hefei 测试集而言, 涉及 850 顶点, 1212 条边数, 共有 10 个算例, 算例中的任务数按照 10% 的增长比例形成, 如测试算例 1 (Hefei-1) 中的任务数为 1212×10%=121, 而测试算例 10 (Hefei-10) 则有 1212 个任务. 对于测试集 Beijing, 顶点数为 2820, 边数为 3584, 共有 10 个算例, 算例中的任务数亦按照 10% 的增长比例形成, 如测试算例 1 (Beijing-1) 中的任务数为 3584×10%=358, 而测试算例 10 (Beijing-10) 则有 3584 个任务. 由于车容量较大, Hefei 和 Beijing 中的最少需求车辆数仅有 7 辆, 如 Hefei-1 和 Beijing-1. 所以 Hefei 和 Beijing 中所有测试算例的中心点数设置为 5 个, 中心点集合 V_D 的设置方法如下:

1) $V_D \leftarrow \emptyset$;

2) 将开头顶点加入到 V_D;

3) 从顶点集 V 中挑选序号为 $m \cdot \left\lfloor \frac{|V|}{n_d - 1} \right\rfloor$ 的顶点加入到 V_D 中, n_d 为预设的中心点数, $m = 1, \cdots, n_d - 1$.

MDCSbA 采用 C++ 实现, 运行环境为 Intel (R) Xeon (R) Gold 5218, 主频为 2.3 GHz, RAM 大小为 256 GB. 对于每个算例, 独立运行 30 次. 为公平起见, 各算法在同一算例上使用相同的计算时间 $TimeBudget$, MDCSbA 相关参数如表 2 所示. 算法运行时间 $TimeBudget$ 设置是至关重要, 其可反映出算法的性能. 综合考虑文献 [30] 中运行时间的设置方案, 本文 $TimeBudget$ 设置为任务数 $|T|$ 乘以 0.2 秒.

表 2 MDCSbA 参数设置

(Table 2 Parameter settings of MDCSbA)

参数	意义	值
$UpValue$	公式 (3.2) 中分解策略参数 β 的上界阈值	0.5
$LowValue$	公式 (3.2) 中分解策略参数 β 的下界阈值	0.1
θ	公式 (3.2) 中分解策略参数 β 计算系数	0.01
$UpRank$	邻域搜索等级阈值	$0.4 \cdot \|T\|$
$TimeBudget$	算例的运行时间	$0.2 \cdot \|T\|$

4.2 测试结果及分析

表 3 和表 4 为各算法在测试集 mdHefei 上的平均结果和最好结果, 表中 $|V|$, $|E|$, $|T|$ 和 Q 分别表示顶点数、边数、任务数和车容量, Avg 代表算法 30 次运行的平均结果, Std 为标准方差, Mean 为算法在测试集上所有结果的平均值, BSN 为各算法在测试集上最优解个数. 由于各算法采用相同时间对同一算例进行计算, 所以表中未附算法的运行耗时. 同一算例的最优结果以粗体显示. 由于实验中各比较算法均为基于概率算法, 故对实验结果进行秩和检验, 若算法以 95% 的水平明显好于 (劣于) MDCSbA, 则对应数据标以 “+” (“−”) . 表中 B-C-W 为 MDCSbA 在测试集上与相应算法秩和检验的统计结果, 分别为明显好、可比较以及明显劣于的算例数.

表 3 测试集 mdHefei 上的平均结果

(Table 3 The average results on the mdHefei dataset)

Name	$\|V\|$	$\|E\|$	$\|T\|$	Q	HGA		MDMA		UHGS		RoCaSH2		MDCSbA	
					Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
mdHefei-1	850	1212	121	9000	203948.1-	3131.2	213924.6-	1430.2	199504.4	7180.1	**197163.4+**	1174.1	198039	1459.4
mdHefei-2	850	1212	242	9000	367128.2-	6932	361173.2-	3122.2	**337258.9+**	6447.6	341564.4	3092.7	340290.4	2886.3
mdHefei-3	850	1212	364	9000	494900.5-	10173	476446.0-	3946	**439926.3+**	7981.2	450729.7-	3724.9	449327.2	2822.2
mdHefei-4	850	1212	485	9000	655931.3-	15893.3	600753.8-	6830.5	**556329.9+**	10827.5	570733.6-	3362.2	567116.9	3839.8
mdHefei-5	850	1212	606	9000	828839.8-	17028.2	725769.9-	7536.1	688032.6	16707.2	689826.2-	4265.3	**684101.2**	4406.5
mdHefei-6	850	1212	727	9000	992164.4-	25607.8	856229.0-	10351.7	806942.9	15375.8	811526.8-	3872.7	**804170.6**	4140.2
mdHefei-7	850	1212	848	9000	1161062.8-	24107.7	991237.9-	8885.1	968303.5-	21183.5	940167.5-	4506.5	**929021.2**	5165.7
mdHefei-8	850	1212	970	9000	1334204.3-	28438.8	1111462.9-	10623	1098585.1-	25668.5	1048765.5-	7328.2	**1030250.3**	6356.1
mdHefei-9	850	1212	1091	9000	1499142.8-	29525.5	1233504.9-	11079.7	1249425.5-	28119.9	1170442.7-	7926.5	**1151295.9**	7105.7
mdHefei-10	850	1212	1212	9000	1629957.8-	30778.1	1344466.7-	11601.4	1356206.9-	31212.8	1271809.9-	6855.1	**1250149.3**	8516
Mean					916728.0		791496.9		770051.6		749273.0		740376.2	
B-C-W					10-0-0		10-0-0		4-3-3		8-1-1			

表 4 测试集 mdHefei 上的最好结果
(Table 4 The best results on the mdHefei dataset)

Name	HGA	MDMA	UHGS	RoCaSH2	MDCSbA
mdHefei-1	197030	211265	**193672**	195099	195589
mdHefei-2	353758	356198	**328573**	336588	336052
mdHefei-3	480745	468156	**429730**	442998	442391
mdHefei-4	628501	588594	**541227**	564435	558934
mdHefei-5	786867	707425	**663840**	681047	674024
mdHefei-6	952604	830420	**781846**	803637	796254
mdHefei-7	1112599	971732	918686	925918	**915830**
mdHefei-8	1275628	1090041	1041716	1033022	**1015196**
mdHefei-9	1441614	1207747	1177297	1156027	**1136448**
mdHefei-10	1571223	1317617	1301609	1257560	**1233825**
Mean	880056.9	774919.5	737819.6	739633.1	730454.3
BSN	0	0	6	0	4

表 3 中, MDCSbA 在所有 10 个算例上, 均取得了较 HGA 和 MAMD 更好的平均值. 并且与 RoCaSH2 比较, 后者仅在规模最小的算例 mdHefei-1 上的平均值较好, 而在其余 9 个算例上的平均值均较 MDCSbA 差. 值得注意的是, UHGS 在 3 个较小规模算例上取得较好的结果, 然而在其他 7 个算例上均劣于 MDCSbA, 其中包括 6 个规模较大算例 (mdHefei-5–mdHefei-10). 测试集 mdHefei 上的平均值, MDCSbA 对应值最小, 为 740376.2, 其他算法的值分别为 HGA: 916728.0, MDMA: 791496.9, UHGS: 770051.6, RoCaSH2: 749273.0. 秩和检验的结果表明 MDCSbA 仅在 mdHefei-1 测试算例上劣于 RoCaSH2, 在 mdHefei-2 上, 两者可比较, 其他 8 个算例上, 前者明显好于后者. 相较于 UHGS, 仅在 3 个较小规模算例 (mdHefei-2–mdHefei-4) 上 MDCSbA 表现弱, 而在较大规模算例 (mdHefei-7–mdHefei-10) 上, 明显优于 UHGS. 而对于 MDMA 以及 HGA, MDCSbA 在所有算例上均明显好.

表 4 中, MDCSbA 在各算例上的最优值均好于 HGA 和 MDMA. 与 ROCASH2 比较, 后者仅在规模较小的 mdHefei-1 上取得较 MDCSbA 优的最好值, 其余 9 个算例上的结果均劣于 MDCSbA. UHGS 在最优解方面表现较好, 在 6 个算例上较其他算法好, 但仅限于 mdHefei-1–mdHefei-6. 在剩余 4 个较大规模算例上, MDCSbA 的最优解好于 UHGS.

表 5, 6 为各方法在测试集 mdBeijing 上的结果, 其中表 5 为平均结果, 表 6 为最好结果. 由表 5 可以看出, 在更大规模测试集上, MDCSbA 在仅在最小规模的 mdBeijing-1 测试算例上平均值劣于 ROCASH2, 其余所有算例上平均结果均最小, 说明所提算法 MDCSbA 在问题规模增大的情况下, 平均性能依然明显. 由表 6 的结果数据可知, MDCSbA 在最好性能方面, 其取得的最好结果明显好于 HGA、MDMA、UHGS 以及 RoCaSH2. 秩和检验的结果显示在所有算例上, MDCSbA 均明显好于 HGA 与 MDMA 以及 UHGS. 然而, 相比较 RoCaSH2, MDCSbA 在 5 个算例上与其可比较, 在 mdBeijing-1 上 MDCSbA 劣于后者. 但是 MDCSbA 在 4 个算例上明显好于 RoCaSH2. 总体而言, MDCSbA 在 mdBeijing 上优于 RoCaSH2.

表 5　测试集 mdBeijing 上的平均结果

(Table 5　The average results on the mdBeijing dataset)

Name	$\|V\|$	$\|E\|$	$\|T\|$	Q	HGA		MDMA		UHGS		RoCaSH2		MDCSbA	
					Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
mdBeijing-1	2820	3584	358	25000	719235.0-	11886.7	686975.3-	2276.8	700976.0-	16440.3	**668445.2+**	2591.4	672494.8	2808
mdBeijing-2	2820	3584	717	25000	1150250.3-	24800	991639.8-	5983.9	1067059.5-	34534.7	967421	2069	**967380.4**	1877.8
mdBeijing-3	2820	3584	1075	25000	1617555.9-	52678.3	1291706.8-	7998.9	1428907.1-	51311.5	1255489.4	5982.7	**1253432.1**	5006
mdBeijing-4	2820	3584	1434	25000	2094669.6-	57565.5	1532919.1-	8790.3	1762055.5-	42994	1483625.4	6238.2	**1481648.1**	6821.2
mdBeijing-5	2820	3584	1792	25000	2509434.8-	84950.3	1765292.4-	12919.3	2091122.9-	41928.8	1713324.3-	8715	**1705276.5**	6066.8
mdBeijing-6	2820	3584	2151	25000	3015946.3-	77773.9	2037539.3-	12574.2	2463738.9-	69502.7	1971272.4	6149.4	**1968618.6**	8543.4
mdBeijing-7	2820	3584	2509	25000	3478082.5-	84965.3	2241324.8-	16682.9	2769506.5-	46033.1	2161050.1-	8318.3	**2151529.9**	9665.6
mdBeijing-8	2820	3584	2868	25000	3870752.7-	114643.1	2411349.9-	15910.6	3050471.7-	52036.6	2332161	11504.3	**2327743.4**	10367.9
mdBeijing-9	2820	3584	3226	25000	4359410.8-	137537.6	2660257.6-	16518.7	3360111.4-	77995.8	2562072.2-	12060	**2550754.1**	10630.4
mdBeijing-10	2820	3584	3584	25000	4769283.9-	144797.7	2856373.8-	16984.4	3622938.6-	83524.4	2745051.6-	13751.3	**2727715.9**	12512.1
Mean					2758462.2		1847537.9		2231688.8		1785991.3		1780659.4	
B-C-W					10-0-0		10-0-0		10-0-0		4-5-1			

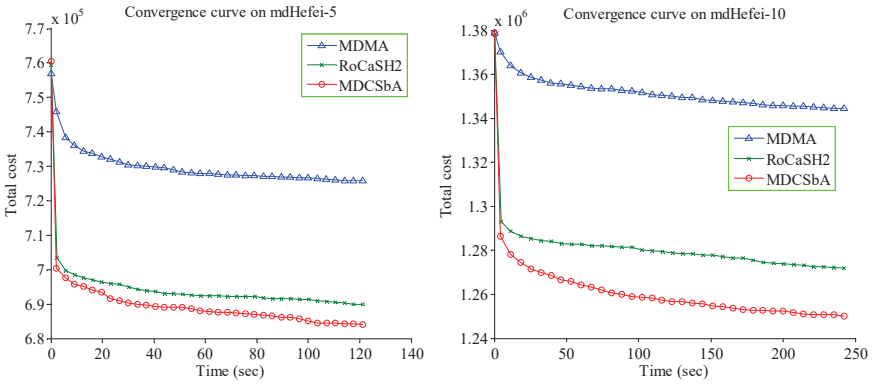
表 6　测试集 mdBeijing 上的最好结果

(Table 6　The best results on the mdBeijing dataset)

Name	HGA	MDMA	UHGS	RoCaSH2	MDCSbA
mdBeijing-1	695942	682251	665930	**660937**	665970
mdBeijing-2	1101333	979308	1002864	962201	**960548**
mdBeijing-3	1518977	1273471	1343156	1244188	**1243614**
mdBeijing-4	2003307	1516397	1673131	1472431	**1466902**
mdBeijing-5	2362875	1735589	2016406	1699213	**1689775**
mdBeijing-6	2891560	2008111	2301315	1958066	**1944238**
mdBeijing-7	3364830	2214531	2672906	2144538	**2126881**
mdBeijing-8	3618989	2370735	2945175	2309496	**2306854**
mdBeijing-9	4083368	2622105	3203359	2541375	**2529961**
mdBeijing-10	4469896	2829829	3493316	2721173	**2706589**
Mean	2611107.7	1823232.7	2131755.8	1771361.8	1764133.2
BSN	0	0	0	1	9

为了更直观地表达 MDCSbA 在各规模问题上的优化性能, 分别从测试集 mdHefei 与 mdBeijing 中选取 4 个有代表性的算例 (mdHefei-5、mdHefei-10、mdBeijing-5 与 mdBeijing-10), 给出各算法在选定算例上平均结果的收敛曲线, 如图 3–4 所示. 图中, X 轴为时间, 单位秒 (sec), Y 轴为总费用. 另外由于 HGA 的结果较差, 作图时会迫使其他算法对应曲线过于挤压, 而 UHGS 则因初始化过程问题, 曲线起始点很高. HGA 及 UHGS 收敛曲线难以与其他算法对应曲线在同一平面画出, 所以图中未描绘它们的曲线.

图 3 中, 由算例 mdHefei-5、mdHefei-10 对应的曲线图可以看出, MDCSbA 对应收敛曲线位于三条曲线的最底层, 且随着问题规模的增大, MDCSbA 收敛特性愈明显.

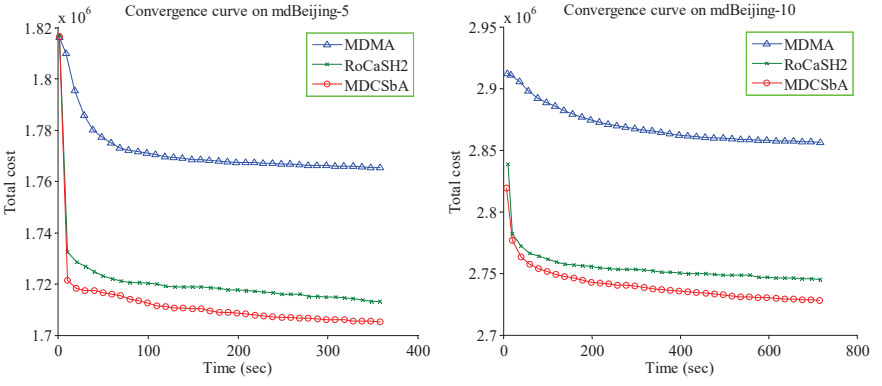


(a) (b)

图 3 算例 mdHefei-5、mdHefei-10 平均收敛曲线图

(Figure 3 The curve graph of mean convergence on mdHefei-5、mdHefei-10)

图 4 中, 对于算例 mdBeijing-5、mdBeijing-10 而言, MDCSbA 的收敛特性明显, 均在其他两条曲线之下. 而且, 在 mdBeijing-10 对应的曲线图中, MDCSbA 与 RoCaSH2 对应收敛曲线随着算法的进行, 之间的间隔呈拉大趋势, 而 mdBeijing-5 对应曲线图中, 两者对应收敛曲线在算法后期, 间距呈缩小状. 上述收敛图中, MDMA 对应曲线均在其他曲线之上, 表现出较差的性能.



(a) (b)

图 4 算例 mdBeijing-5、mdBeijing-10 平均收敛曲线图

(Figure 4 The curve graph of mean convergence on mdBeijing-5、mdBeijing-10)

由表 3–6 的结果及图 3–4 的显示知, MDCSbA 对于小规模问题, 表现较 UHGS、RoCaSH2 弱. 然而, 对于规模较大的问题, MDCSbA 则显示了良好的问题处理能力. 主要原因在于 MDCSbA 对子 CARP 的求解算子是基于分治策略而设计, 如此可以进一步降低问题规模, 加速问题求解. 以上说明了 MDCSbA 中多分治策略的融合, 成功地解决了 LSMDCARP.

5 结 论

大规模 MDCARP 的复杂性在于: 1) 结构的复杂性, 多中心点的结构导致任务需要合适中心点的划分; 2) 大规模性, 致使 MDCARP 的解空间呈爆炸式增长, 难以确定有效的搜索区域;

3) 子 CARP 的复杂性, 由于问题的大规模性, 子 CARP 往往亦处于大规模状态. 本文提出了多分治策略的构架, 形成了基于多分治策略的 MDCARP 求解算法 MDCSbA. 对于 MDCSbA 而言, 在多中心的问题分解阶段、子问题间的交流阶段以及子问题求解阶段, 均使用了分治策略. 在大规模测试集 mdHefei、mdBeijing 上进行了算法性能验证, 结果表明了本文算法 MDCSbA 明显优于其他算法. 所以, MDCSbA 多分治策略的引入, 对于大规模 MDCARP 显示了优良的性能.

参 考 文 献

[1] Golden B L, Wong R. Capacitated arc routing problems. *Networks*, 1981, **11**(3): 305–316.

[2] Corberán Á, Eglese R, Hasle G, et al. Arc routing problems: A review of the past, present, and future. *Networks*, 2021, **77**(1): 88–115.

[3] Bode C, Irnich S. Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, 2012, **60**(5): 1167–1182.

[4] Porumbel D, Coelho I M, Talbi E G. Using an exact bi-objective decoder in a memetic algorithm for arc-routing (and other decoder-expressible) problems. *European Journal of Operational Research*, 2024, **313**(1): 25–43.

[5] Li R, Zhao X C, Zuo X G, et al. Memetic algorithm with non-smooth penalty for capacitated arc routing problem. *Knowledge-Based Systems*, 2021, **220**: 1–18.

[6] Zhang Y Z, Mei Y, Tang K, et al. Memetic algorithm with route decomposing for periodic capacitated arc routing problem. *Applied Soft Computing*, 2017, **52**: 1130–1142.

[7] 张玉州, 刘晓飞, 黄师化, 等. 一种求解多车型 CARP 的有效 memetic 算法. 中国科学技术大学学报, 2017, **47**(7): 583–593.
(Zhang Y Z, Liu X F, Huang S H, et al. An effective memetic algorithm for heterogeneous vehicle CARP. *Journal of University of Science and Technology of China*, 2017, **47**(7): 583–593.)

[8] Zhu Z Y, Li X H, Yang Y, et al. A hybrid genetic algorithm for the multiple depot capacitated arc routing problem. IEEE International Conference on Automation & Logistics, 2007, 2253–2258.

[9] Wang S L, Mei Y, Zhang M J. A multi-objective genetic programming algorithm with α dominance and archive for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 2023, **27**(6): 1633–1647.

[10] Ardeh M A, Mei Y, Zhang M J, et al. Knowledge transfer genetic programming with auxiliary population for solving uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 2023, **27**(2): 311–325.

[11] Wei C G, Wϕhlk S, Che A. A multi-level capacitated arc routing problem with intermediate facilities in waste collection. *Computers & Operations Research*, 2024, **167**: 106671.

[12] Zhou Y M, Qu C H, Wu Q H, et al. A bilevel hybrid iterated search approach to soft-clustered capacitated arc routing problems. *Transportation Research Part B*: *Methodological*, 2024, **184**: 102944.

[13] Xing L N, Rohlfshagen P, Chen Y W, et al. An evolutionary approach to the multidepot capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 2010, **14**(3): 356–374.

[14] Kansou A, Yassine A. New upper bounds for the multi-depot capacitated arc routing problem. *International Journal of Metaheuristics*, 2010, **1**(1): 81–95.

[15] Chen N, Qiu T, Zhou X B, et al. A distributed co-evolutionary optimization method with motif for large-scale IoT robustness. *IEEE/ACM Transactions on Networking*, 2024, **32**(5): 4085–4098.

[16] Yang M, Gao J, Zhou A M, et al. Contribution-based cooperative co-evolution with adaptive population diversity for large-scale global optimization. *IEEE Computational Intelligence Magazine*, 2023, **18**(3): 56–68.

[17] 李珍萍, 焦鹏博, 韩倩倩, 等. 成品油二次配送库存-路径优化模型与两阶段算法. 系统科学与数学, 2023, **43** (5): 1120–1137.
(Li Z P, Jiao P B, Han Q Q, et al. Inventory routing optimization model and two-phase algorithm for refined oil secondary distribution. *Journal of Systems Science and Mathematical Sciences*, 2023, **43**(5): 1120–1137.)

[18] 鲁健, 曾振柄. 用符号计算证明 Ramsey 定理的一个机械化方法. 系统科学与数学, 2021, **41**(12): 3311–3323.
(Lu J, Zeng Z B. A mechanical proof of Ramsey's theorem via symbolic computation. *Journal of Systems Science and Mathematical Sciences*, 2021, **41**(12): 3311–3323.)

[19] Zhang Y Z, Mei Y, Huang S H, et al. A route clustering and search heuristic for large scale multi-depot capacitated arc routing problem. *IEEE Transactions on Cybernetics*, 2022, **52**(8): 8286–8299.

[20] Zhang Y Z, Mei Y, Zhang H Q, et al. An effective route clustering and search heuristic for large-scale multi-depot capacitated arc routing problem. *IEEE Computational Intelligence Magazine*, 2023, **18**(4): 43–56.

[21] Taillard É D, Vo S. POPMUSIC-partial optimization metaheuristic under special intensification conditions. Essays and Surveys in Metaheuristics. Edited by Ribeiro C C, Hansen P. Berlin: Springer, 2002, 613–629.

[22] Mei Y, Li X D, Yao X. Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 2014, **18**(3): 435–449.

[23] Tang K, Mei Y, Yao X. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 2009, **13**(5): 1151–1166.

[24] Shang R H, Dai K Y, Jiao L C, et al. Improved memetic algorithm based on route distance grouping for multiobjective large scale capacitated arc routing problems. *IEEE Transactions on Cybernetics*, 2016, **46**(4): 1000–1013.

[25] Tang K, Wang J, Li X D, et al. A scalable approach to capacitated arc routing problems based on hierarchical decomposition. *IEEE Transactions on Cybernetics*, 2017, **47**(11): 3928–3940.

[26] Dijkstra E. A note on two problems in connection with graphs. *Numerische Mathematik*, 1959, **1**(1): 269–271.

[27] Zhang Y Z, Mei Y, Zhang B Z, et al. Divide-and-conquer large scale capacitated arc routing problems with route cutting off decomposition. *Information Sciences*, 2021, **553**: 208–224.

[28] Ulusoy G. The fleet size and mix problem for capacitated arc routing. *European Journal of Operational Research*, 1985, **22**(3): 329–337.

[29] Vidal T. Node, edge, arc routing and turn penalties: Multiple problems-one neighborhood extension. *Operations Research*, 2017, **65**(4): 992–1010.

[30] Wϕhlk S, Laporte G. A fast heuristic for very large-scale capacitated arc routing problems. *Journal of the Operational Research Society*, 2018, **69**(12): 1877–1887.