计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(2):281−291 [doi: 10.15888/j.cnki.csa.009739] [CSTR: 32024.14.csa.009739]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

具有时间窗约束松弛的混合蚁群算法求解 VRPTW①

骆　维[1,3], 陈仕军[2], 吴华伟[1,3]

[1](湖北文理学院 湖北隆中实验室, 襄阳 441053)
[2](湖北文理学院 数学与统计学院, 襄阳 441053)
[3](湖北文理学院 纯电动汽车动力系统设计与测试湖北省重点实验室, 襄阳 441053)
通信作者: 陈仕军, E-mail: csj@hbuas.edu.cn

摘　要: 为求解带时间窗的车辆路径问题, 以最小化总行驶里程为目标建立混合整数规划模型, 提出了一种具有时间窗约束松弛的混合蚁群算法. 首先, 提出改进的蚁群算法与“TSP-Split 编码与解码”相结合的方法, 来构建允许违反时间窗约束的解路径, 以提高算法的全局寻优能力. 然后, 利用“及时返回”原则和惩罚函数方法, 提出基于变邻域搜索的修复策略来修复不可行解. 最后, 对 56 个 Solomon 和 12 个 Homberger 基准算例进行试验计算, 结果表明该算法的求解质量优于文献中的对比算法, 且在 50 个测试实例上获得了已知最优解, 其余实例也能在可接受计算时间内获得准最优解, 验证了所提算法的有效性.

关键词: 蚁群算法; 带时间窗的车辆路径问题 (VRPTW); 时间窗约束松弛; TSP-Split; 及时返回; 变邻域搜索

引用格式: 骆维,陈仕军,吴华伟.具有时间窗约束松弛的混合蚁群算法求解 VRPTW.计算机系统应用,2025,34(2):281–291. http://www.c-s-a.org.cn/1003-3254/9739.html

Hybrid Ant Colony Optimization with Time Window Constraint Relaxation for Solving VRPTW

LUO Wei[1,3], CHEN Shi-Jun[2], WU Hua-Wei[1,3]

[1](Hubei Longzhong Laboratory, Hubei University of Arts and Science, Xiangyang 441053, China)
[2](School of Mathematics and Statistics, Hubei University of Arts and Science, Xiangyang 441053, China)
[3](Hubei Key Laboratory of Power System Design and Test for Electrical Vehicle, Hubei University of Arts and Science, Xiangyang 441053, China)

Abstract: To solve the vehicle routing problem with time windows (VRPTW), this study establishes a mixed-integer programming model aimed at minimizing total distance and proposes a hybrid ant colony optimization algorithm with relaxed time window constraints. Firstly, an improved ant colony algorithm, combined with TSP-Split encoding and decoding, is proposed to construct a routing solution that allows time-window constraints to be violated, to improve the global optimization ability of the algorithm. Then, a repair strategy based on variable neighborhood search is proposed to repair infeasible solutions using the principle of return in time and the penalty function method. Finally, 56 Solomon and 12 Homberger benchmark instances are tested. The results show that the proposed algorithm is superior to the comparative algorithms from references. The known optimal solution can be obtained in 50 instances, and quasi-optimal solutions can be obtained in the remaining instances within acceptable computing time. The results prove the effectiveness of the proposed algorithm.

Key words: ant colony optimization (ACO); vehicle routing problem with time windows (VRPTW); time window constraint

① 基金项目: 湖北文理学院科研能力培育基金科技创新团队项目 (2020kypytd006); 湖北文理学院研究生创新计划 (YCX202421)
收稿时间: 2024-06-23; 修改时间: 2024-07-18; 采用时间: 2024-07-25; csa 在线出版时间: 2024-11-15
CNKI 网络首发时间: 2024-11-18

relaxation; TSP-Split; return in time; variable neighborhood search (VNS)

带时间窗的车辆路径问题 (vehicle routing problem with time windows, VRPTW) 最初由 Solomon[1]提出, 是车辆路径问题 (vehicle routing problem, VRP) 最重要的扩展问题之一. 与传统 VRP 不同, VRPTW 不仅要考虑容量约束, 还要服务时间必须满足各个客户点预先规定的时间窗 (time window) 约束, 即车辆必须在时间窗口内才能对客户点进行服务. VRPTW 在车辆调度[2]、电商超市[3]、冷链配送[4]中具有广泛应用, 是各大物流运输企业所面临的重要问题. 一个求解 VRPTW 的高效算法, 能为物流运输企业提供合理高效的配送路径规划, 并能节省巨额运营成本, 具有重要的现实价值.

VRPTW 属于 NP-hard 问题, 求解该问题的主要方法包括精确算法和启发式算法. 理论上, 精确算法能确保找到问题的最优解, 但由于计算时间复杂度高, 在处理大规模 VRPTW 问题时, 不具有实用性. 启发式算法可粗略分为基于贪婪策略的简单启发式算法和基于启发式的智能优化算法, 合理高效的启发式算法能够在可接受时间内求得大规模问题的满意解. 因此大多数针对 VRP 的研究仍然集中在开发或改进现有的启发式算法. 在 VRPTW 研究中, 常用的启发式算法包括禁忌搜索算法 (tabu search algorithm, TSA)[5]、自适应大邻域搜索 (adaptive large neighborhood search, ALNS)[6]、遗传算法 (genetic algorithm, GA)[7]、蚁群算法 (ant colony optimization, ACO)[8]等.

蚁群算法[9]作为一种模拟自然界蚂蚁觅食行为的群体智能算法, 最初由 Dorigo 等提出来用于求解旅行商问题 (travelling salesman problem, TSP). 由于蚁群算法具有分布式、自适应和鲁棒性强的特点, 适于求解复杂的离散优化问题, 已被部分学者用于求解 VRPTW 问题. 但传统蚁群算法过于依赖信息素的引导, 存在容易陷入局部最优解的缺点. 为此, 金淳等[10]使用分布式 agent 系统结合 ACO 进行并行运算, 并在算法中添加邻域搜索策略, 在满足求解精度的同时能大幅度缩小计算时间. Shen 等[11]将蚁群系统与头脑风暴算法相结合, 在算法中添加局部搜索算子来提高寻优能力. 张雄等[12]对蚁群算法求得的最优解再进行强化邻域搜索, 以弥补蚁群算法在局部寻优方面的不足, 在求解“Solomon-C 类”算例上达到了最优解. 何美玲等[13]改进了蚂蚁状态转移概率公式, 设计自适应信息素挥发系数, 最后将插入算子和交换算子嵌入变邻域搜索来跳出局部最优解, 在“Solomon-C 类”算例上验证了有效性. 雷金羡等[14]通过在信息素更新阶段增加奖励值来降低信息素的积累速度, 同时动态调整信息素挥发因子, 并加入了一些局部搜索算子来增加算法的局部搜索能力, 利用部分 Solomon 实例验证了所提算法的有效性.

上述文献所提的蚁群算法, 在构造解路径时都要求不违反问题约束, 算法搜索的解空间相对较小. 同时, 部分文献测试的 Solomon 算例也不全面, 并且实验结果与最优解存在一定差距. 为进一步研究 VRPTW 的高效求解方法, 本文提出一种具有时间窗约束松弛的改进混合蚁群算法 (hybrid ant colony optimization with time window constraint relaxation, HACO-TWCR). 在基于蚁群算法的构造路径阶段, 对时间窗约束进行松弛, 利用“TSP-Split 编码与解码方法” (下文简称“TSP-Split 方法”) 构造车辆路径, 特别是允许路径不满足时间窗约束, 以扩大解搜索空间, 从而提高算法的全局优化能力. 针对蚁群算法所求得的解, 再利用变邻域搜索 (variable neighborhood search, VNS) 框架和多种局部搜索算子相结合来修复不可行解, 同时对目标成本值进行优化. 此外, 针对蚁群算法, 提出了包括状态转移概率公式、信息素更新、信息素重置等算法策略的改进. 实验阶段, 利用 56 个 Solomon 和 12 个 Homberger 基准算例验证了本文算法的有效性.

1 带时间窗的车辆路径问题及其数学模型

基本的 VRPTW 问题可以描述为如下: 已知有若干车辆位于货物配送中心, 有多个需要进行配送的客户点, 分布在不同的区域位置. 每个客户点的需求量已知, VRPTW 在于合理规划一系列车辆配送路线 (即车辆路径) 以满足所有客户点的需求, 并使得总配送成本达到最小. 此外, 在考虑最小化 VRPTW 的总成本时, 既有大多数研究主要采用两个目标: 第 1 目标是最小化车辆数 (即车辆路径数), 第 2 目标是最小化车辆的总行驶里程. 由于物流运输企业在做配送决策之前已经购置好配送车辆, 因此也有部分研究假定车辆数给定, 以最小化车辆的总行驶里程为主要目标. 本研究采

用后者, 即假定车辆数给定, 以最小化总行驶里程为优化目标. 相关假设如下.

(1) 每个车辆必须从配送中心出发, 并在执行完配送计划后, 返回配送中心.

(2) 每辆车的承载量都不能超过车辆的最大载重量.

(3) 每个客户都被服务一次, 并且只被服务一次.

(4) 车辆在客户点的服务时间必须满足时间窗约束.

基本的 VRPTW 可以采用完全有向图 $G(V, A)$ 来描述, 其中节点集 $V=\{c_0, c_1, \cdots, c_n\}$, 弧集 $A=\{\langle c_i, c_j\rangle | i\neq j, c_i, c_j\in V\}$. 记 c_0 表示配送中心, c_i $(i=1, 2, \cdots, n)$ 表示客户点. 每个客户点 c_i, 都有其相应的需求量 q_i、服务时间 s_i 和时间窗口$[e_i, l_i]$; 对于 c_0: $q_0=0$, $s_0=0$, $e_0=0$. 记 t_{ij} 表示客户点 c_i 与 c_j 之间的车辆行驶时间, 在假定行驶速度不变条件下, 可用 c_i 与 c_j 之间的欧几里得距离 d_{ij} 表示 ($d_{ij}=d_{ji}$). 若车辆到达客户点 c_i 的时刻早于 c_i 的时间窗左端点 e_i, 则必须等待到 e_i 时才能为其服务; 若晚于 c_i 的时间窗右端点 l_i, 则不能为其服务. 记 t_i^k 表示车辆为客户 c_i 开始服务的时刻, 车辆服务客户点 c_i 时所花费的服务 (卸货) 时间 s_i. 基于上述符号定义, 建立 VRPTW 的数学模型:

$$\min f_{\text{VRPTW}} = \sum_{k=1}^{K}\sum_{i=0}^{N}\sum_{j=0}^{N} d_{ij}x_{ij}^k \tag{1}$$

必须满足的约束:

$$\sum_{j=1}^{N} x_{0j}^k = \sum_{i=1}^{N} x_{i0}^k \leqslant 1,\ \forall k = 1,2,\cdots,K \tag{2}$$

$$\sum_{j=0}^{N} x_{ij}^k = \sum_{j=0}^{N} x_{ji}^k,\ i\neq j,\ \forall i = 1,2,\cdots,n;\forall k = 1,2,\cdots,K \tag{3}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{N} x_{ij}^k = 1,\ i\neq j,\ \forall j = 1,2,\cdots,n \tag{4}$$

$$\sum_{k=1}^{K}\sum_{j=0}^{N} x_{ij}^k = 1,\ i\neq j,\ \forall i = 1,2,\cdots,n \tag{5}$$

$$\sum_{i=0}^{N} q_i \sum_{j=0}^{N} x_{ij}^k \leqslant Q,\ i\neq j,\ \forall k = 1,2,\cdots,K \tag{6}$$

$$\sum_{k\in V}\sum_{j\in N} x_{0j}^k \leqslant |K|,\ \forall j = 1,2,\cdots n;\ \forall k = 1,2,\cdots,K \tag{7}$$

$$\begin{cases} t_i^k + s_i + t_{ij} - t_j^k \leqslant (1-x_{ij}^k)\cdot M, \\ i\neq j,\ \forall i, j = 0,1,\cdots,N; \\ \forall k = 1,2,\cdots,K \end{cases} \tag{8}$$

$$\begin{cases} e_j \sum_{i=0} x_{ij}^k \leqslant t_j^k \leqslant l_j \sum_{i=0} x_{ij}^k, \\ i\neq j,\ \forall j = 0,1,\cdots,N; \\ \forall k = 1,2,\cdots,K \end{cases} \tag{9}$$

$$x_{ij}^k \in \{0,1\},\ i\neq j,\ \forall i, j = 0,1,\cdots,N;\forall k = 1,2,\cdots,K \tag{10}$$

其中, 式 (1) 为目标函数, 即最小化总行驶里程. 式 (2) 表示车辆从配送中心出发, 并最后返回配送中心. 式 (3) 为客户点的流量平衡约束. 式 (4) 和式 (5) 确保客户点只能被一辆车服务且仅被服务一次. 式 (6) 表示车辆在行驶过程中不能超出其最大负载容量. 式 (7) 表示为客户点服务的车辆数最多为 K. 式 (8) 和式 (9) 定义了时间窗约束, M 为大常数. 式 (10) 对应决策变量, 表示车辆 k 的配送路径是否从客户 i 到客户 j; 若是, 则决策变量取 1, 否则取 0.

2 具有时间窗约束松弛的混合蚁群算法

针对 VRPTW, 本文提出具有时间窗约束松弛的混合蚁群算法 (HACO-TWCR). 首先, 蚂蚁在构造路径时采用 TSP-Split 方法 (第 2.1 节) 形成具有时间窗约束松弛的 VRPTW 解 (可能不满足时间窗约束), 然后在变邻域搜索框架下结合多种局部搜索算子, 并基于 Nagata 等[15]提出的“及时返回”策略进行不可行解的修复, 并不断优化目标值. 算法的总流程如图 1, 步骤如下.

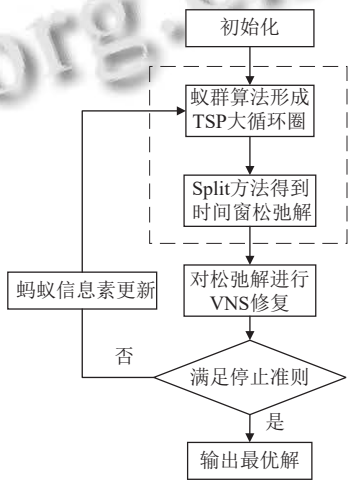


图 1 算法的总流程图

Step 1. 初始化参数, 使用第 2.3.1 节的扫描算法构造初始解, 初始化信息素浓度、信息素边界.

Step 2. 利用蚂蚁构造 TSP 大循环圈. 根据式 (14) 的转移概率公式计算概率, 采用轮盘赌, 从距离当前节

点最近的 1/4 总客户点候选列表中选择下一个客户点, 构建暂时忽略车辆容量约束和时间窗约束的 TSP 大循环圈.

Step 3. 在考虑车辆负载容量约束下, 松弛时间窗约束, 利用 Split 算法分割 TSP 大循环圈, 得到时间窗松弛 VRP 解, 即有可能不满足时间窗约束. 将 Step 2 和 Step 3 合称为“TSP-Split 方法”, 如图 1 的虚线方框内部分.

Step 4. 针对 Step 3 产生的时间窗松弛 VRPTW 解, 利用第 2.4 节的变邻域搜索 (VNS) 进行局部搜索, 在修复不可行解的同时, 不断优化目标成本值.

Step 5. 蚂蚁信息素更新. 与传统蚁群算法不同, 本文使用式 (15) 对蚂蚁形成的 TSP 路径进行信息素更新. 本文的停止准则设置为固定迭代次数, 当达到最大迭代次数时输出最优解, 否则返回 Step 2 进行下一轮迭代.

上述算法步骤中, 利用 Step 3 对 TSP 大循环圈进行分割时, 如果完全不考虑时间窗约束, 会造成 split 算法分割后的子路径违反时间窗的程度可能会很大, 不利于快速修复成可行解. 因此, 在 Step 3 执行 Split 算法时, 将客户 c_i 的右时间窗端点 l_i 进行适当松弛, 使 $l_i=l_i+\delta$, 其中 δ 为松弛系数.

2.1 基于 TSP-Split 编码与解码的求解方法

VRPTW 主要有两种构建路径的方法: 直接编码和间接编码[16]. 直接编码又叫“集群第一路径第二”方法, 该方法将聚集在一起的客户点逐个顺连, 构造可行的车辆路径, 当某个车辆满载后, 再用新车辆继续构造其他不同的车辆路径. 直接编码具有直观和易于实现的优点, 也是目前文献中应用的主流方法[17]. 间接编码又称为“路径第一集群第二”方法, 该方法最早由 Beasley[18] 提出. 应用间接编码时, 先将问题看成旅行商问题 (TSP), 求解出能覆盖所有客户点的 TSP 解 (以下称为“TSP 大循环圈”), 再将 TSP 大循环圈分割成满足车辆容量约束的多条子路径. 与直接编码相比, 间接编码能缩小可行解的搜索空间. 特别是 Prins 等[19]于 2004 年提出一个有效的分割算法 (也称“Split 算法”), 来将 TSP 大循环圈分割成多条子路径, 为间接编码方法带来了便利. Split 算法在考虑车辆容量和返回配送中心的约束条件下, 将 TSP 大循环圈分割问题转化为有向无环图上的最短路径问题, 并提出一个高效的动态规划求解算法, 该算法能得到最优的子路径分割方案. 为了减小搜索空间, 提高寻优效率, 本文也将采用间接编码方法, 并利用 Prins 等所提出的 Split 算法来分割由蚁群算法得到的 TSP 大循环圈. TSP-Split 方法, 如图 2 所示.

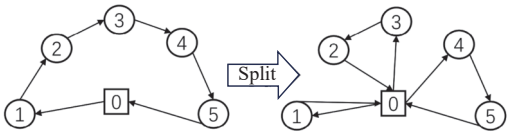


图 2 TSP-Split 方法

本文 Split 算法的输入是由蚁群算法所构造的 TSP 大循环圈, 输出能满足车辆容量约束和松弛时间窗后的多条车辆子路径, 即 VRPTW 解.

2.2 构造时间窗约束松弛的目标函数

为了提高算法的全局优化能力, 上述“TSP-Split 方法”采用了松弛时间窗策略, 因此需要将不满足时间窗约束的解, 修复成可行解. 本文根据不可行解违反时间窗约束的程度, 来定义惩罚成本, 再利用变邻域搜索算法来修复不可行解和优化总成本. 在满足容量约束的前提下, 对时间窗约束进行松弛, 定义车辆子路径 r 的成本函数如下:

$$f_{\text{pen}} = f(r) + \lambda P_{\text{TW}}(r) \tag{11}$$

其中, f_{pen} 是带惩罚项的成本函数, $f(r)$ 是原成本函数; λ 是惩罚系数; $P_{\text{TW}}(r)$ 是违反时间窗约束的惩罚值, 其值等于 $\sum_{i=1}^{u}\max(a_i-l_i,0)$, 其中 a_i 是车辆到达客户点 c_i 的时刻, u 为子路径 r 的客户点数.

此外, 本文采用“及时返回”策略[15]来松弛时间窗约束. 当车辆延迟到达某客户点 c_i 时, 假定车辆可以及时返回, 即车辆在客户时间窗右端点 l_i 开始服务. 采用“及时返回”的时间窗松弛方法有两个优点: (1) 可以使算法搜索的重点集中在可行解的边界上, 有更大概率找到高质量的解; (2) 利用基于邻域算子的局部搜索算法时, 能够更加高效地进行邻域解评估.

2.3 蚁群算法的改进策略

蚁群算法是通过模拟蚂蚁觅食行为, 而提出的群体启发式优化算法, 具有较强的鲁棒性. 但传统的蚁群算法存在收敛到全局最优解速度慢和容易陷入局部最优解的缺点, 因此本文对蚁群算法做了如下几个方面的改进.

2.3.1 构造初始解

本文通过构造好的初始解, 来设置更加合理的初始化信息素值 τ_0, 从而加快算法的收敛速度. 为此, 采

用扫描算法来获得初始解, 相关步骤如下.

(1) 按照与配送中心 c_0 的角度, 对客户集 C 进行扫描, 根据扫描顺序, 以车辆容量 Q 将客户集 C 进行分组, 确定路径数 m.

(2) 每条路径 R_m ($m \leqslant K$) 的客户根据时间窗长度 L 分类. 如果客户 c_i 的时间窗长度 L_i 大于 $L_0/2$ 时, 纳入时间宽裕客户集 A, 否则纳入时间紧迫客户集 U.

(3) 对于每条路径 R_m, 客户集 U 灵活性低, 按照时间紧迫程度依次插入; 客户集 A 灵活性高, 根据带来最小额外距离进行贪婪策略插入, 注意此操作会违反时间窗约束.

(4) 使用步骤 (3) 产生的时间窗约束不可行解, 获得最终可行的 VRPTW 初始解; 若不能修复, 则自动重启程序.

为了使初始解对蚂蚁有更好的引导性, 对初始解中客户点组成的 TSP 大循环圈, 设置初始信息素值 $\tau_0=1/L_s$, L_s 是改进的扫描算法形成的初始路径长度.

2.3.2 转移概率公式

传统蚁群算法的转移概率公式只与距离和信息素浓度有关, 在解决 VRPTW 时会导致算法收敛速度变慢, 并且寻优能力不足. 为此, 本文在转移概率公式中引入节约启发式思想和考虑“TSP-Split 方法”求解效率的时间紧迫因子. 节约启发式将客户安排到一条单独从配送中心出发并返回的路径, 然后计算每对客户点之间合并路径的节省距离 S_{ij}. 时间紧迫因子 w'_{ij} 的定义, 见式 (12)、式 (13); 大的时间紧迫因子, 能使得 TSP 路径中客户点的等待时间偏离程度减小, 客户点之间的时间相关性增大, 有助于 Split 算法分割出更优路径.

$$w_{ij} = \max(e_j - s_j - d_{ij} - l_i, 0) \tag{12}$$

$$w'_{ij} = \begin{cases} \dfrac{1}{w_{ij}}, & w_{ij} > 0 \\ 1, & \text{其他} \end{cases} \tag{13}$$

最终改进的转移概率公式, 如式 (14):

$$P_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta} \cdot s_{ij} \cdot w'_{ij}}{\sum\limits_{s \in allow_k} (\tau_{is}^{\alpha} \cdot \eta_{is}^{\beta} \cdot s_{is} \cdot w'_{is})}, & s \in allow_k \\ 0, & s \notin allow_k \end{cases} \tag{14}$$

其中, $allow_k$ 是未访问的客户集, τ_{ij} 是信息素浓度, η_{ij} 是能见度, 其值等于距离的倒数 $1/d_{ij}$.

2.3.3 精英蚂蚁策略与信息素更新策略

传统精英蚂蚁策略没有考虑精英蚂蚁之间的相似度, 因此当精英蚂蚁相似度过高时, 会使得算法收敛到局部最优解. 为此, 本文采用能体现精英蚂蚁差异化的策略, 每次选取固定数量的精英蚂蚁 e 只, 但要使每只精英蚂蚁所构造解的成本差不低于 ω, 来剔除相似解, 从而提高精英蚂蚁群体的多样性. 针对该问题, 信息素的更新策略如式 (15):

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{e} \Delta\tau_{ij}^k(t) + e\Delta\tau_{ij}^{\text{gb}} \tag{15}$$

其中, ρ 为信息素挥发因子, 取值在 0–1 之间; 每次更新除了对精英蚂蚁路径施加 $\Delta\tau_{ij}^k$ 的信息素外, 还会对全局最优路径施加 $e\Delta\tau_{ij}^{\text{gb}}$ 的信息素, 其中 $\Delta\tau_{ij}^{\text{gb}} = 1/L_{\text{gb}}$, L_{gb} 当前最优路径长度. 此外, 对 $\Delta\tau_{ij}^k$ 进行了改进, 如式 (16), 其中 L_k 是精英蚂蚁排名第 k 蚂蚁所构造的路径长度, L_{gb}/L_k 的范围在 0–1 之间, 当 L_{gb} 与 L_k 的差距越小, 信息素增量 $\Delta\tau_{ij}^k$ 就越大. 这样设置既能使较优路径得到更多的信息素加快前期收敛, 又能使每只精英蚂蚁走过的路径上的信息素增加不会过多, 可以减缓信息素的积累, 从而提高全局搜索能力.

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{L_{\text{gb}}}{L_k} \cdot \dfrac{Q}{L_k}, & \text{蚂蚁}k\text{在第}t\text{次迭代时经过节点}(i,j) \\ 0, & \text{其他} \end{cases} \tag{16}$$

2.3.4 其他改进策略

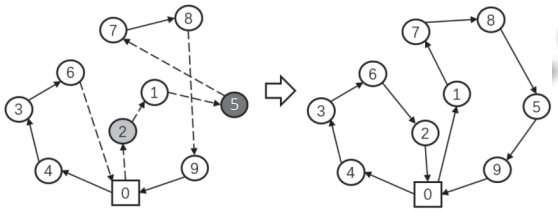
信息素挥发因子决定着路径上残留信息素的浓度, ρ 过小会限制算法的全局搜索能力, 达到局部最优; 过大的 ρ 虽然能提高算法的搜索能力, 但会因为路径上的信息素浓度过低而导致盲目搜索, 进而削弱信息素的引导能力. 本文为 ρ 设置一个动态机制, 如式 (17), 其触发条件是“连续 3 次当前最优解没有改变”.

$$\rho(t) = \begin{cases} 0.96\rho(t-1), & \rho(t) > \rho_{\min} \\ \rho_{\min}, & \text{其他} \end{cases} \tag{17}$$

此外, 本文采用 MMAS 信息素动态重置策略. 信息素动态重置策略将每条路径上的信息素浓度的取值范围限制在一个区间 $[\tau_{\min}, \tau_{\max}]$ 内, $\tau_{\max}=\tau_0=1/L_{\text{gb}}$. 信息素重置策略的触发条件是“当迭代中有 80% 的蚂蚁路径相同时”, 说明此时算法已经收敛到局部最优, 有必要重置信息素矩阵, 使 $\tau_{ij}=\tau_{\max}$.

2.4 基于 VNS 的不可行解修复策略

利用 Relocate、Swap、2-Relocate、2-Swap、2-Opt* 和 2-Opt, 6 种算子构建变邻域搜索框架. Relocate 算子将 1 个客户插入到另一位置; Swap 算子随机交换两个客户位置; 2-Relocate 算子将一条路径中连续 2 个客户插入到同一路径或不同路径的另一位置; 2-Swap 算子交换同一路径或不同路径上连续的两个客户点; 2-Opt* 算子通过在两条不同路径中交换节点段来改进总体路径; 2-Opt 算子针对单条子路径, 随机选择 2 点, 将中间的客户点逆转. 图 3 是 Relocate 算子的示例, 图中客户点 2 插入到另一条子路径的客户点 6 前面, 而客户点 5 插入到同一子路径中的客户点 8 前面.



(a) 原路径图 (b) 改进路径图

图 3 Relocate 算子示例

设定惩罚系数 λ 动态变化, $\lambda\in[\lambda_{\min},\lambda_{\max}]$, 输入蚁群算法得到的解 S, 前期 S 一般是不可行的, 进行变邻域搜索后得到新解 S', 若 S' 仍不可行, 则将惩罚系数 λ 乘以 10, 若当 λ 等于最大惩罚值 $\lambda_{\max}$ 后解仍不可行, 则将不可行解丢弃. 若不可行解修复成功, 则输出修复优化后的解 S. 由于进行 VNS 的客户点是随机选择, 所以算子的使用顺序并不影响最终结果, 相关流程如图 4.

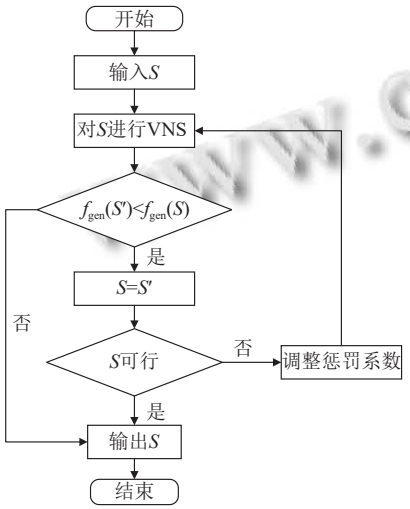


图 4 基于 VNS 的修复策略

此外, 本文采用 Vidal 等[20]提出的邻域修剪方法来适当缩小邻域的范围, 使算法在合理的时间内快速收敛. 该方法与距离和客户时间窗相关联, 给定客户 i 和 j, 其相关度函数为:

$$\gamma(i,j)=d_{ij}+\gamma^{\mathrm{WT}}\max\left(e_j-s_i-t_{ij}-\beta_i,0\right)+\gamma^{\mathrm{TW}}\max(e_i+s_i+t_{ij}-\beta_j,0) \tag{18}$$

相关度函数与距离和时间窗相关, 客户只取前$|\Gamma|$个关联度最高的客户作为邻域客户对进行算子的邻域变换. 相关度函数的基本参数为: γ^{WT}=0.2, 和 γ^{TW}=1.0, $|\Gamma|=|V|\times40\%$, $|V|$为客户规模.

3 仿真实验与分析

3.1 实验设置

(1) 测试算例: 采用 Solomon 数据集和 Homberger 数据集. 其中, Solomon 数据集包含 56 个算例, 每个算例都有一个配送中心和 100 名客户, 这些客户分布在一个 100×100 的直角坐标系内. 根据客户节点的分布特征, Solomon 数据集分为聚集 (C)、随机 (R) 和混合 (RC) 这 3 种类型. 其中, R1、C1、RC1 类的特征为狭窄的时间窗口和较小的车辆载重量; 而 R2、C2、RC2 类则有较宽的时间窗口和较大的车辆载重量. Solomon 数据集是 VRP 中常用和具有代表性的数据集. 为了进一步体现算法的寻优能力, 选取部分 Homberger 数据集进行实验, 该数据集是 Solomon 数据集的扩展, 是采用 200 客户规模的算例, 每类选择 2 个算例进行测试.

(2) 算法参数设置: 蚂蚁数量 m=50, 迭代次数 $maxit$=300, $\lambda\in[50,5000]$, 信息素影响因子 α=3, 启发式影响因子 β=5, 精英蚂蚁的数量 e=5, 精英蚂蚁成本差 ω=3, 信息素强度 Q=0.7, 信息素挥发系数 ρ=0.04, 测试 Homberger 数据集时将最大迭代次数设置为 500 次, 取时间窗松弛系数 δ=80. 所有参数都经过调整, 以平衡求解质量和计算成本.

(3) 对比算法: 本文 HACO-TWCR 算法通过 Solomon 数据集进行测试, 其实验结果与文献[11]算法 (ACS-BSO)、文献[21]算法 (HHHSA)、文献[22]算法 (ScPSO) 进行比较, Homberger 数据集与当前最优解进行比较. 为保证对比的公平性, 本文算法及对比算法均以最小值行驶距离为优化目标. 本文所设计算法采用 C++语言编写, 在 Windows 11 系统, 12th Gen Intel(R) Core (TM) i7-1260P 2.10 GHz (16 GB RAM) 的环境下在 Visual Studio 2022 运行.

实验时每个算例测试 10 次, 实验数据如表 1、表 2

所示. 数据中 BKS 是当前最知名解, TD 是算法得到的最优行驶距离, NV 是对应取得最优距离时的车辆数, Gap 是本文算法与最知名解的相对误差, Std 是本文算法的行驶距离标准差, Avg 是所在列的平均值.

表 1 Solomon 算例实验结果

算例	BKS[11]		ACS-BSO[11]		HHHSA[21]		ScPSO[22]		HACO-TWCR		Gap (%)	Std
	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV		
C101	828.94	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**	10	0.00	0.00
C102	828.94	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**	10	0.00	0.00
C103	828.06	10	**828.06**	10	**828.06**	10	**828.06**	10	**828.06**	10	0.00	0.00
C104	824.78	10	**824.78**	10	846.78	10	852.30	10	**824.78**	10	0.00	0.00
C105	828.94	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**	10	0.00	0.00
C106	828.94	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**	10	0.00	0.00
C107	828.94	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**	10	0.00	0.00
C108	828.94	10	**828.94**	10	**828.94**	10	**828.94**	10	**828.94**	10	0.00	0.00
C109	828.94	10	**828.94**	10	**828.94**	10	854.78	10	**828.94**	10	0.00	0.00
R101	1 642.88	20	**1 642.88**	20	1 656.5	19	1 655.87	20	**1 642.88**	20	0.00	0.00
R102	1 472.62	18	1 479.55	18	1 477.7	18	1 474.75	18	**1 472.62**	18	0.00	0.36
R103	1 213.62	14	1 225.31	15	1 225.84	15	1 227.27	14	**1 213.62**	14	0.00	2.89
R104	976.61	11	1 002.62	12	1 017.5	11	1 012.97	10	**976.61**	11	0.00	4.16
R105	1 360.78	15	1 365.66	15	1 416.7	15	1 375.74	15	**1 360.78**	15	0.00	2.01
R106	1 240.47	13	1 249.51	13	1 269.3	13	1 259.02	14	**1 240.47**	13	0.00	3.57
R107	1 073.34	11	1 091.21	12	1 112.68	12	1 094.74	11	**1 072.92**	11	–0.04	0.59
R108	947.70	10	960.23	10	984.91	10	978.49	11	**941.70**	10	–0.64	0.83
R109	1 151.45	13	1 165.71	12	1 203.7	13	1 185.77	12	**1 151.45**	13	0.00	0.84
R110	1 072.41	12	1 090.92	11	1 139.3	12	1 108.15	12	**1 072.41**	12	0.00	2.89
R111	1 053.50	12	1 063.69	12	1 094.3	12	1 071.86	11	**1 053.50**	12	0.00	1.65
R112	953.63	10	976.28	11	997.19	11	1 001.92	11	**953.63**	10	0.00	3.26
RC101	1 623.58	15	1 643.78	16	1 672.9	16	1 665.27	16	**1 623.58**	15	0.00	4.04
RC102	1 461.23	14	1 464.63	14	1 505.9	14	1 493.58	14	**1 461.23**	14	0.00	7.11
RC103	1 261.67	11	1 275.65	11	1 290.3	12	1 305.40	12	1 266.61	11	0.39	3.61
RC104	1 135.48	10	1 156.92	10	1 153.8	10	1 188.32	11	**1 135.48**	10	0.00	2.49
RC105	1 518.58	15	1 535.78	16	1 564.8	15	1 553.15	15	**1 518.58**	15	0.00	1.39
RC106	1 371.69	13	1 378.45	13	1 417.25	14	1 427.71	14	1 377.00	13	0.39	1.70
RC107	1 212.83	12	1 216.65	12	1 280.28	12	1 280.45	12	**1 212.83**	12	0.00	0.00
RC108	1 117.53	11	1 134.28	11	1 190.15	11	1 210.74	11	1 118.59	11	0.09	5.01
C201	591.56	3	**591.56**	3	**591.56**	3	**591.56**	3	**591.56**	3	0.00	0.00
C202	591.56	3	**591.56**	3	**591.56**	3	**591.56**	3	**591.56**	3	0.00	0.00
C203	591.17	3	**591.17**	3	**591.17**	3	**591.17**	3	**591.17**	3	0.00	0.00
C204	590.60	3	594.89	3	596.55	3	596.55	3	**590.60**	3	0.00	0.00
C205	588.88	3	**588.88**	3	**588.88**	3	**588.88**	3	**588.88**	3	0.00	0.00
C206	588.49	3	**588.49**	3	**588.49**	3	**588.49**	3	**588.49**	3	0.00	0.00
C207	588.29	3	**588.29**	3	**588.29**	3	**588.29**	3	**588.29**	3	0.00	0.00
C208	588.32	3	**588.32**	3	**588.32**	3	**588.32**	3	**588.32**	3	0.00	0.00
R201	1 147.8	8	1 161.20	8	1 162.2	7	1 168.07	7	**1 147.80**	8	0.00	0.58
R202	1 034.35	8	1 058.83	6	1 044.89	5	1 043.68	6	**1 034.35**	8	0.00	4.67
R203	874.87	6	883.42	5	882.29	6	882.44	5	**874.87**	6	0.00	2.25
R204	735.8	5	756.93	4	737.23	4	746.47	5	**735.80**	5	0.00	1.24
R205	954.16	5	978.47	6	975.77	5	962.98	5	957.85	5	0.39	2.01
R206	879.89	5	906.27	4	905.91	6	892.48	5	**879.89**	5	0.00	3.71
R207	799.86	4	812.35	4	818.53	5	817.11	5	**799.86**	4	0.00	3.09
R208	705.45	4	725.05	3	718.54	4	710.30	4	**705.33**	4	–0.02	0.22
R209	859.39	5	879.01	5	866.4	5	871.40	5	**859.39**	5	0.00	4.54
R210	910.7	5	923.43	7	924.87	7	921.42	5	912.47	5	0.19	0.78
R211	755.96	4	776.17	4	776.02	4	782.63	5	756.50	4	0.07	2.51
RC201	1 265.56	9	1 284.71	8	1 283.82	7	1 286.62	8	1 272.94	9	0.58	1.64
RC202	1 095.64	8	1 127.02	7	1 106.85	7	1 116.79	7	1 099.54	8	0.35	0.46
RC203	928.51	5	943.13	6	951.06	5	945.76	5	937.27	5	0.93	3.17

表 1 Solomon 算例实验结果 (续)

算例	BKS[11]		ACS-BSO[11]		HHHSA[21]		ScPSO[22]		HACO-TWCR		Gap (%)	Std
	TD	NV	TD	NV	TD	NV	TD	NV	TD	NV		
RC204	786.38	4	807.71	4	801.43	4	791.40	4	**786.38**	4	0.00	2.54
RC205	1 157.55	7	1 170.98	8	1 169.22	7	1 159.42	7	1 162.45	7	0.42	2.09
RC206	1 054.61	7	1 093.64	7	1 082.66	6	1 063.57	6	**1 054.61**	7	0.00	5.00
RC207	966.08	6	986.70	6	984.93	6	966.08	6	**966.08**	6	0.00	4.21
RC208	779.31	4	785.60	4	788.9	5	780.72	4	**779.31**	4	0.00	1.58
Avg	977.28	8.63	988.55	8.64	997.41	8.64	994.89	8.61	977.92	8.63	0.06	1.69

注: 加粗数据表示算法获得的最好解

表 2 Homberger 算例实验结果

算例	BKS		HACO-TWCR		Gap (%)	Std
	TD	NV	TD	NV		
C121	2 704.57	20	2 704.57	20	0.00	0.0
C122	2 700.65	20	2 700.65	20	0.00	0.0
C221	1 929.39	7	1 929.39	7	0.00	1.69
C222	1 858.65	7	1 858.65	7	0.00	1.00
R121	4 676.14	23	4 742.07	23	1.39	39.24
R122	3 928.65	20	4 029.51	21	2.50	39.14
R221	3 476.38	13	3 523.68	14	1.34	35.15
R222	3 016.27	11	3 083.88	13	2.19	21.30
RC121	3 524.39	20	3 563.11	20	1.09	14.51
RC122	3 227.97	19	3 271.93	20	1.34	5.70
RC221	2 804.16	10	2 834.61	11	1.07	8.91
RC222	2 489.09	9	2 538.04	11	1.93	19.13
Avg	3 003.14	15.31	3 037.28	15.92	0.99	14.29

从表 1 可以看出: (1) 从行驶距离上看, 结果最好的是 HACO-TWCR, 其次是 ACS-BSO, 最后是 ScPSO、HHHSA. HACO-TWCR 相比于 ACS-BSO、ScPSO 和 HHHSA, 其最优平均行驶距离分别改善了 1.1%、1.7% 和 2%. 在 56 个 Solomon 算例中, HACO-TWCR 与 BKS 的最优平均值相差极小, 仅为 0.07%, 其中 HACO-TWCR 获得了 46 个已知最优解, 其余算例的 Gap 均不超过 1%. HACO-TWCR 在 C1、R1、RC1 类的 29 个算例获得了 89.7% 的最优解, 在 C2、R2、RC2 类的 27 个算例获得了 74.1% 的最优解, 说明在 100 客户规模上, 本文算法更擅长求解时间窗较窄、车辆载重较小的问题, HACO-TWCR 对时间窗宽松、车辆载重较大的算例求解仍有提升空间, 但总体受客户时间窗和车辆载重的影响不大.

(2) 从使用车辆数上看, ScPSO 总体取得了最优车辆数, 比 BKS 平均最优车辆数改善了 0.2%; 其次是 HACO-TWCR, 获得了与 BKS 相同的车辆数; 最后是 HHHSA 和 ACS-BSO, 平均最优车辆数均为 8.64 辆. 本文以行驶距离为第 1 目标, 所以 HACO-TWCR 在车辆数上也取得不错的效果.

(3) 从算法稳定性和运行速度上看, 本文算法求解 Solomon 算例上具有较强的稳定性, 算法标准差 Std 较小, 均在 10 以下. 在运行速度上, 由于运行环境和编程语言的不同, 本文算法之间的运行速度不好比较. 本文算法的迭代次数固定, 虽然使用的算子数量较多, 但每个 Solomon 算例的运行时间在 115–140 s 之间, 大部分算例运行较快, 所以 HACO-TWCR 有着不错的运行效率, 这归功于高效的“及时返回”方法和邻域缩小策略.

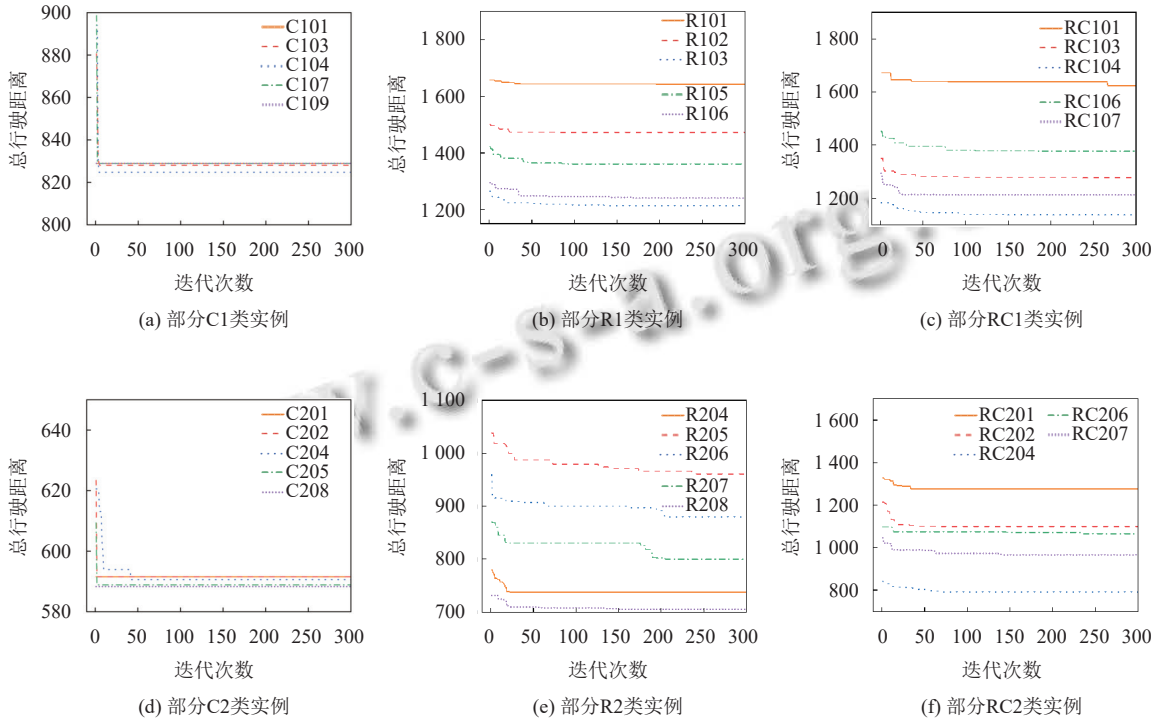
(4) 从算法收敛性上看, HACO-TWCR 可以在大部分 Solomon 算例的测试上快速收敛. 图 5 是 HACO-TWCR 求解 Solomon 算例的部分收敛曲线图, 从图 5 可以看出, 在 C1 和 C2 类, 算法可以快速收敛, 除了 R2 类外, 其余实例大都在 150 次迭代前已经收敛. 图 6 是 HACO-TWCR 求解 Solomon 算例的部分最优配送路径展示, 从图 6 可以看出, 所选取算例的最优配送路径规划合理, 无过多配送路径交叉重叠.

为进一步测试所提算法的性能, 将测试算例规模进一步扩大至 200 客户点, 表 2 是 Homberger 算例的实验结果. 从表 2 可以看出: 当客户规模扩大为 200 客户数量时, 算法的寻优性能相应变差, 但仍有较好的效果. 在行驶距离上, HACO-TWCR 的 C1 和 C2 均得到了最优解, 其余算例相对误差较小, 不超过 3%, 总的平均 Gap 仅有 0.99%. 在车辆数上, HACO-TWCR 的最优平均车辆数与 BKS 相比的误差仅有 3.8%. 在算法稳定性和运行速度上, HACO-TWCR 求解 Homberger 算例的平均标准差仅有 14.29, 每个 Homberger 算例运行时间在 561–645 s 之间, 可以得知当规模进一步扩大后 HACO-TWCR 仍能保持较好的性能.

3.2 优化策略分析

为了检验 HACO-TWCR 中 TSP-Split 方法和时间窗约束松弛方法的有效性, 使用 56 个 Solomon 算例进行测试分析, 取 10 次运行结果的平均值, 然后按类型

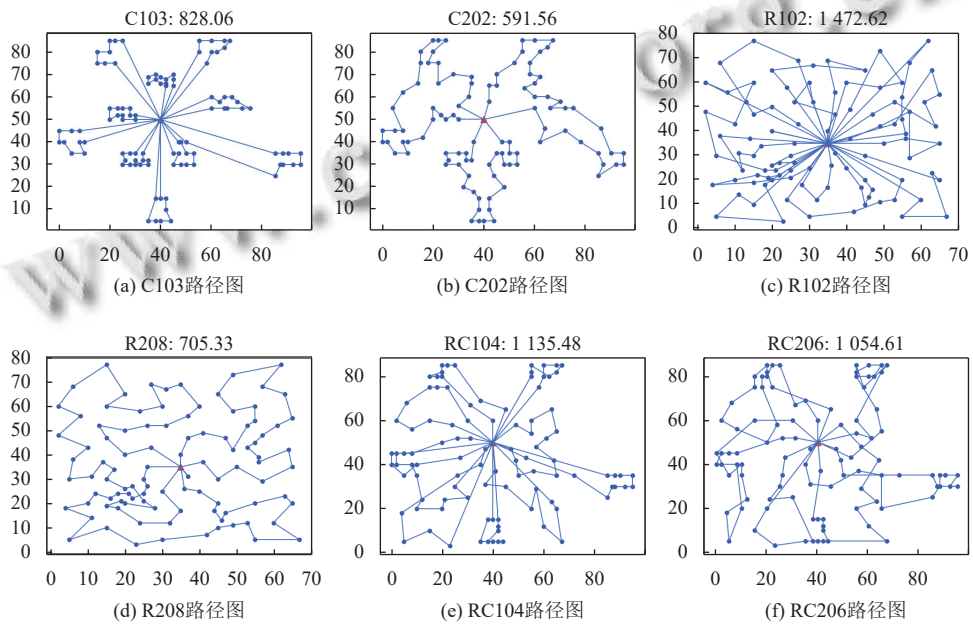
分为 6 类, D-ACO 是使用传统直接编码方法的蚁群算法, I-ACO 是使用 TSP-Split 方法的蚁群算法, HACO-N 是不允许时间窗约束违反的混合蚁群算法, Solomon 数据集测试的平均结果如表 3 所示. 由于两种编码解码方式的差异性, 设置 D-ACO 的信息素挥发系数 ρ=0.3, 信息素强度 Q=5; I-ACO 的信息素挥发系数 ρ=0.1, 信息素强度 Q=5. D-ACO 与 I-ACO 其余基本参数与 HACO-TWCR 一致, 这样各自可以取到比较好的效果.



(a) 部分C1类实例 (b) 部分R1类实例 (c) 部分RC1类实例

(d) 部分C2类实例 (e) 部分R2类实例 (f) 部分RC2类实例

图 5 HACO-TWCR 求解 Solomon 算例的部分收敛曲线图



(a) C103路径图 (b) C202路径图 (c) R102路径图

(d) R208路径图 (e) RC104路径图 (f) RC206路径图

图 6 HACO-TWCR 求解 Solomon 算例的部分最优配送路径图

在表 3 中, 我们可以看出: (1) 与传统蚁群算法 D-ACO 相比, I-ACO 除了 R1 类和 RC2 类, 其他类型的结果都更好, 也能看出在处理聚类实例上, I-ACO 效果更好. 一个主要的原因是 TSP-Split 方法信息素只更新 TSP 大循环, 有些算例客户地理位置的分布和时间窗的严格要求, 导致边缘径的总行驶距离中占比过大, 容易造成信息素矩阵和路径不匹配, 从而造成效果不理想. 在 100 客户规模上, 结合了 TSP-Split 方法的蚁群算法更适合求解 VRPTW. (2) 除了 C1 和 C2 类与 HACO-TWCR 算法持平外, HACO-N 其余算例结果都劣于 HACO-TWCR, 证明了时间窗约束松弛再修复的有效性.

表 3 优化策略测试 Solomon 数据集的平均结果

算法	C1	C2	R1	R2	RC1	RC2
BKS	828.38	589.86	1 179.94	878.02	1 337.82	1 004.20
D-ACO	1 258.56	882.71	1 666.33	1 344.42	1 908.75	1 617.31
I-ACO	1 023.64	845.77	1 697.72	1 307.03	1 899.18	1 621.04
HACO-N	828.38	589.86	1 188.95	887.88	1 361.6	1 016.95
HACO-TWCR	828.38	589.86	1 182.91	882.16	1 343.24	1 010.84

4 结论与展望

本文研究了带时间窗的车辆路径问题, 以最小化总距离为目标构建混合整数规划模型, 为此提出一种具有时间窗约束松弛的混合蚁群算法. 首先, 用改进的蚁群算法结合 TSP-Split 方法, 来构建允许违反时间窗约束的初始解, 然后, 使用基于 VNS 的修复策略来到修复和优化的目的. 最后, 利用 Solomon 和 Homberger 基准算例验证了算法的有效性. 所提算法在求解 Solomon 算例时, 总体优于近年的先进算法; 在求解 Homberger 基准算例时, 该算法结果与最优解的相对误差小, 对求解 200 客户规模的 VRPTW 问题也具有较好的性能; 通过实验测试和对比分析, 验证了在 100 客户的 Solomon 测试实例上, 蚁群算法结合 TSP-Split 方法, 总体优于传统蚁群算法; 相比于不违反时间窗约束, 时间窗松弛方法后 HACO-TWCR 的性能更优. 下一步, 将时间窗扩展到软时间窗, 验证 TSP-Split 方法在不同规模算例的有效性, 和设计更加有效的惩罚机制来进一步提升算法的综合性能.

参考文献

1 Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, 1987, 35(2): 254–265. [doi: 10.1287/opre.35.2.254]

2 陈凯, 邓志良, 龚毅光. 离散灰狼优化算法求解 VRPSPDTW 问题. 计算机系统应用, 2023, 32(11): 83–94. [doi: 10.15888/j.cnki.csa.009305]

3 孙一凡, 高更君. 考虑电商超市多仓多品类订单拆分与整合的车辆路径优化. 制造业自动化, 2023, 45(10): 19–24. [doi: 10.3969/j.issn.1009-0134.2023.10.005]

4 李丽滢, 罗继康, 牛莉霞. 多中心冷链共同配送路径优化及利润分配研究. 制造业自动化, 2023, 45(10): 203–210. [doi: 10.3969/j.issn.1009-0134.2023.10.040]

5 贺琪, 官礼和, 崔焕焕. 硬时间窗 VRP 的混合变邻域禁忌搜索算法. 计算机工程与应用, 2023, 59(13): 82–91. [doi: 10.3778/j.issn.1002-8331.2208-0431]

6 Pan BB, Zhang ZZ, Lim A. A hybrid algorithm for time-dependent vehicle routing problem with time windows. Computers & Operations Research, 2021, 128: 105193. [doi: 10.1016/j.cor.2020.105193]

7 Maroof A, Ayvaz B, Naeem K. Logistics optimization using hybrid genetic algorithm (HGA): A solution to the vehicle routing problem with time windows (VRPTW). IEEE Access, 2024, 12: 36974–36989. [doi: 10.1109/ACCESS.2024.3373699]

8 徐英卓, 李凯, 周俊. 基于改进蚁群算法的钻井救援车辆路径规划. 计算机系统应用, 2022, 31(4): 268–272. [doi: 10.15888/j.cnki.csa.008442]

9 Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53–66. [doi: 10.1109/4235.585892]

10 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. 计算机应用研究, 2018, 35(3): 666–670.

11 Shen Y, Liu MD, Yang J, *et al*. A hybrid swarm intelligence algorithm for vehicle routing problem with time windows. IEEE Access, 2020, 8: 93882–93893. [doi: 10.1109/ACCESS.2020.2984660]

12 张雄, 潘大志. 融合邻域搜索策略蚁群算法求解带时间窗口的车辆路径问题. 计算机与现代化, 2022(3): 98–102, 110. [doi: 10.3969/j.issn.1006-2475.2022.03.017]

13 何美玲, 魏志秀, 武晓晖, 等. 基于改进蚁群算法求解带软

时间窗的车辆路径问题. 计算机集成制造系统, 2023, 29(3): 1029–1039. [doi: 10.13196/j.cims.2023.03.029]
14 雷金羡, 孙宇, 朱洪杰. 改进蚁群算法在带时间窗车辆路径规划问题中的应用. 计算机集成制造系统, 2022, 28(11): 3535–3544. [doi: 10.13196/j.cims.2022.11.017]
15 Nagata Y, Bräysy O, Dullaert W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research, 2010, 37(4): 724–737. [doi: 10.1016/j.cor.2009.06.022]
16 Jia YH, Mei Y, Zhang MJ. Confidence-based ant colony optimization for capacitated electric vehicle routing problem with comparison of different encoding schemes. IEEE Transactions on Evolutionary Computation, 2022, 26(6): 1394–1408. [doi: 10.1109/TEVC.2022.3144142]
17 Prins C, Lacomme P, Prodhon C. Order-first split-second methods for vehicle routing problems: A review. Transportation Research Part C: Emerging Technologies, 2014, 40: 179–200. [doi: 10.1016/j.trc.2014.01.011]
18 Beasley JE. Route first-luster second methods for vehicle routing. Omega, 1983, 11(4): 403–408. [doi: 10.1016/0305-0483(83)90033-6]
19 Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 2004, 31(12): 1985–2002. [doi: 10.1016/S0305-0548(03)00158-8]
20 Vidal T, Crainic TG, Gendreau M, *et al*. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & Operations Research, 2013, 40(1): 475–489. [doi: 10.1016/j.cor.2012.07.018]
21 Zhang Y, Li JC. A hybrid heuristic harmony search algorithm for the vehicle routing problem with time windows. IEEE Access, 2024, 12: 42083–42095. [doi: 10.1109/ACCESS.2024.3378089]
22 Wang YF, Chen X, Shuang Z, *et al*. Self-competition particle swarm optimization algorithm for the vehicle routing problem with time window. IEEE Access, 2024, 12: 127470–127488. [doi: 10.1109/ACCESS.2024.3401487]

(校对责编: 孙君艳)