doi: 10.3969/j.issn.1007-7375.220254

多车程多时间窗车辆路径问题研究

吴廷映，鲁佳琪，夏　洋

（上海大学 管理学院，上海 200444）

摘要: 针对末端物流中配送车辆多趟次运输、客户对配送服务时间的多样化需求，研究多车程多时间窗车辆路径问题。构造该问题的最小化车辆数量和总运输成本的双目标混合整数规划模型，设计改进的自适应大邻域搜索算法对其求解；构建了基于路径、车程及客户点 3 个层级上的多种高效的破坏算子和修复算子来扩大解的搜索空间；使用自适应策略选择高效的搜索算子，以及引入模拟退火新解接受准则避免陷入局部最优解来提高搜索效率。通过多种规模算例实验结果分析，验证了改进的自适应大邻域搜索算法的优越性，并分析了考虑多车程的模型对总运输成本的影响。

关键词: 多车程；多时间窗；车辆路径问题；自适应大邻域搜索

中图分类号: F407.472; U121　　**文献标志码:** A　　**文章编号:** 1007-7375(2024)02-0147-11

Vehicle Routing with Multiple Trips and Multiple Time Windows

WU Tingying, LU Jiaqi, XIA Yang

(School of Management, Shanghai University, Shanghai 200444, China)

Abstract: Focusing on the multi-trip transportation of terminal logistics and the diverse demand of customers for service time, a vehicle routing problem with multiple trips and multiple time windows is studied. A bi-objective mixed integer programming model is established to minimize the number of vehicles and the total transportation cost, while an improved adaptive large neighborhood search algorithm is designed to solve the problem. In this algorithm, a variety of efficient destroy and repair operators based on three levels of routes, travel distances and customer points are constructed to expand the search space of solutions. An adaptive strategy is used to select efficient search operators and a simulated annealing rule is introduced to avoid the solution from falling into local optimum and improve the search efficiency. By analyzing the experimental results of instances with various scales, the advantages of the improved adaptive large neighborhood search algorithm are verified, and the positive impact of the model considering multiple trips on the total transportation cost is analyzed.

Key words: multiple trips; multiple time windows; vehicle routing problem; adaptive large neighborhood search

伴随城市人口密度的增加和快递配送需求的增长，交通堵塞和环境污染问题日益严重，末端快递配送成为城市物流中的一大挑战。为减轻末端物流配送车辆对城市内交通和空气环境造成的负面影响，国家已出台了一系列政策，例如禁止大型货运车辆进入城市中心区域，鼓励采用小型载货摩托车进行最后一公里配送等。目前末端物流配送系统中，物流公司通常采用大型货运车辆将货物从位于郊区的配送中心配送至末端网点，再使用小型载货摩托车将货物从末端网点配送至客户。由于小型载货摩托车容量较小，配送过程中通常需要其多次往返于末端网点和客户之间。此外，随着生活节奏的加快，公众时间日趋碎片化，客户对配送服务时间要求越来越高，希望在自己较为方便的多个时间窗内为其提供服务。在市场竞争的驱动下，各大电商企业已将“精准配送、按时配送”作为提高客户满意度的主要手段。然而，配送时间的精准化将导致配送成本的增加，如何在满足送货时间要求的同时降低配送成本，成为当前物流企业亟待解决的问题。基于此，本文针对多趟次配送和客户对配送服务时

收稿日期： 2022-12-19

作者简介： 吴廷映 (1982—)，男，苗族，贵州省人，讲师，博士，主要研究方向为运筹优化、鲁棒优化、智能优化算法。

间的多样化需求，以最小化车辆使用数量和总运输成本为目标对末端网点货物配送制定配送方案。根据末端物流配送的特点，将本文所研究的问题提炼成多车程多时间窗车辆路径问题 (vehicle routing problem with multiple trips and multiple time windows, VRPMTMTW)。

目前，文献中尚未有同时考虑多车程和多时间窗的车辆路径问题的研究，只有少量学者分别针对多时间窗车辆路径问题 (vehicle routing problem with multiple time windows, VRPMTW) 和多车程车辆路径问题 (vehicle routing problem with multiple trips, VRPMT) 进行研究，而且国内关于此类问题的研究极少。VRPMTW 和 VRPMT 均为车辆路径问题 (vehicle routing problem, VRP) 的扩展，为严格的 NP 难问题，精确算法在求解大规模的问题时求解速度较慢。国内外学者针对这两类问题构建了相应的数学规划模型并设计了不同的启发式算法来获得问题的近似最优解。在 VRPMTW 方面：李珍萍等[1] 构建以最小化总运输成本为目标的优化模型，并设计智能水滴算法进行求解；Belhaiza 等[2] 提出一种混合变邻域禁忌搜索启发式算法对其进行求解；Beheshti 等[3] 基于一个实际应用的启发，研究多目标的 VRPMTW，并采用一种协同进化多目标量子遗传算法进行求解；闫芳等[4] 将多时间窗模糊化，在考虑顾客满意度的前提下，构建最小化总成本的多模糊时间窗的数学模型，并采用粒子群算法加以求解；Mao 等[5] 研究带时间窗和多种充电策略的电动车车辆路径问题，设计改进的蚁群算法对其进行求解，并分析充电策略对成本的影响；李博威等[6] 针对带软时间窗的同时取送货车辆路径问题，构建多目标混合整数规划模型，采用理想点法将多目标转化为单目标优化问题，并设计相应的 LINGO 的精确算法求解该问题；张瑾等[7] 研究带容量和时间窗约束的车辆路径问题，构建以最小化车辆使用成本、运输成本和违反时间窗惩罚成本的目标函数，并设计一种改进的离散蝙蝠算法求解该问题；闫军等[8] 研究带时间窗的同时取送货车辆路径问题，构建总成本最小化的数学模型，并设计蚁群算法对其进行求解；Fang 等[9] 针对绿色制造的配送物流规划，研究多个时间窗的异构车辆路径问题，并设计基于禁忌搜索的混合蚁群优化算法对其进行求解。在 VRPMT 方面：Fleischmann[10] 首次对该问题进行研究，并设计平行节约算法对其进行求解；Wassan 等[11] 采用两阶段的邻域搜索算法求解最小化总运输成本的 VRPMT；张晓楠等[12] 从实际出发，研究一种基于模糊需求的 VRPMT，并采用种群进化算法与随机模拟算法进行求解；宋强[13] 探讨基于发货时间限制的 VRPMT，采用改进的混合遗传算法加以求解；刘虹等[14] 研究一种同时取送货的 VRPMT，考虑“实时柔性点”的调整机制，并设计一种变邻域禁忌搜索混合算法来获取最优路径方案；Rahmani[15] 研究随机模糊环境下回程多行程车辆路径问题，基于 Hurwicz 准则构建最小化总成本的模型，并设计基于单纯形法、模糊仿真和萤火虫算法的混合智能算法；Li 等[16] 研究订单可配送时间的多行程车辆路径问题，并构建混合整数线性规划模型，设计自适应大邻域搜索算法对问题进行求解；更多国外相关研究可参考 Cattaruzza 等[17] 的多行程车辆路径问题综述。

从上述的研究可以看出，国内外学者对多时间窗、多车程的车辆路径问题分别进行了研究，但尚未有同时考虑多时间窗和多车程的研究。因此，本文在已有研究的基础之上，针对末端物流中配送车辆多趟次配送、客户对配送服务时间的多样化需求，对多车程多时间窗车辆路径进行优化。构建以最小化车辆使用数量和总运输成本的双目标混合整数规划模型，同时考虑多车程、多时间窗等约束条件。设计基于自适应大邻域搜索算法 (adaptive large neighborhood search, ALNS) 的改进的自适应大邻域搜索算法 (improved ALNS, IALNS) 对其求解，为末端网络配送中的车辆路径规划提供解决方案。

本文的创新点如下。

1) 以最小化车辆使用数量和总运输成本为优化目标，对配送车辆数量及运输路径同时进行优化。首次建立 VRPMTMTW 的混合整数规划模型，并通过算例实验证明模型的有效性。

2) 结合模拟退火的思想，设计 IALNS 算法，基于模型特性分别针对路径、车程及客户点 3 个层级上设计多种破坏和修复算子以扩大解的搜索空间，包括随机破坏、最差破坏和相关性破坏 3 类破坏算子，以及贪婪修复、最优修复、后悔值修复和贪婪随机自适应修复 4 类修复算子。使用自适应策略选择高效的搜索算子，以及引入模拟退火新解接受准则避免陷入局部最优解来提高搜索效率。通过

算例实验结果分析，验证了 IALNS 算法的优越性，分析了考虑多车程的模型对总运输成本的积极影响。

1　问题及模型

1.1　问题描述

本文所研究的问题可描述为：某配送中心拥有同类型的一组载货摩托车为一组客户需求提供配送服务，每个需求对应一个配送点和若干个可供选择的时间窗，载货摩托车需在其中一个时间窗为其提供服务。每辆载货摩托车均从配送中心出发，在满足工作时间和发车次数的情况下，车辆在完成一个车程上所有的服务后需要返回配送中心补货后再进行下一车程的服务。每个客户点的需求量和服务时间，客户点之间的路程和平均行驶时间，每辆车的固定成本、装载能力以及单位运距成本均已知。

模型中的符号说明及决策变量定义如表 1 所示。

表 1　符号说明

Table 1　Symbols and descriptions

符号	符号说明
K	车辆集合
0	配送中心，车辆配送的起点
$n+1$	配送中心，车辆配送的终点
C	客户点集合，$C=\{1,2,\ldots,n\}$
N	车场在内的点集合，$N=C\cup\{0,n+1\}$
q_i	客户 i 的货物重量
s_i	客户 i 的卸货时间
w_i	客户 i 的时间窗集合
E_i^v	客户 i 的第 v 个最早开始服务时间
L_i^v	客户 i 的第 v 个最晚开始服务时间
A	节点之间的弧集合，$A=\{(i,j)\mid i,j\in N, i\neq j\}$
d_{ij}	节点 i 到节点 j 的行驶距离
Q	车辆的最大装载量
S	每个工作日里车辆工作的时长限制
P	每辆车每天可以完成的车程集合
L	车辆在每个车程上的时长限制
α	装货系数，即车辆在车场装货时的单位货物重量装货时间
b	运距系数
M	任意无穷大的数
x_{ij}^{kp}	车辆 k 在第 p 车程经过弧 (i,j) 为 1，否则为 0
r_{ij}	车辆从节点 i 到节点 j 的载重量，非负实数
u_i^v	客户 i 在其第 v 时间窗被服务为 1，否则为 0
a_i	节点 i 的开始服务时间
ε^{kp}	车辆 k 第 p 车程的装货时间
ρ^{kp}	车辆 k 第 p 车程的开始时间
σ^{kp}	车辆 k 完成第 p 车程的时间

1.2　VRPMTMTW 数学模型

结合带时间窗的 VRP 模型[1] 和多车程的 VRP 模型[14]，构建如下 VRPMTMTW 模型。

$$\min\left(\sum_{k\in K}\sum_{j\in N}x_{0j}^{k1},\sum_{k\in K}\sum_{p\in P}\sum_{(i,j)\in E}d_{ij}x_{ij}^{k}b\right)。\tag{1}$$

s.t.

$$\sum_{k\in K}\sum_{p\in P}\sum_{j\in N,j\neq i}x_{ij}^{kp}=1,\forall i\in C;\tag{2}$$

$$\sum_{i\in N}x_{ih}^{kp}-\sum_{j\in N,j\neq h}x_{hj}^{kp}=0,\ \forall h\in C,k\in K,p\in P;\tag{3}$$

$$\sum_{j\in N}x_{0j}^{kp}\leqslant 1,\ \forall k\in K,p\in P;\tag{4}$$

$$\sum_{j\in N}x_{0j}^{kp}-\sum_{j\in N}x_{0j}^{k(p+1)}\geqslant 0,\ \forall k\in K,p\in P\backslash\{p_{\max}\};\tag{5}$$

$$\sum_{j\in N}x_{0j}^{kp}-\sum_{j\in N}x_{0j}^{(p+1),1}\geqslant 0,\ \forall k\in K\backslash\{p_{\max}\};\tag{6}$$

$$\sum_{i\in C}\sum_{j\in N,j\neq i}q_ix_{ij}^{kp}\leqslant Q,\forall k\in K,p\in P;\tag{7}$$

$$\sum_{i\in C}\sum_{j\in N,j\neq i}r_{ji}x_{ji}^{kp}-r_{ij}x_{ij}^{kp}=q_i,\forall k\in K,p\in P;\tag{8}$$

$$\varepsilon^{kt}=\alpha\sum_{i\in C}\sum_{j\in N,j\neq i}q_ix_{ij}^{kp},\forall k\in K,p\in P;\tag{9}$$

$$\rho^{kp}+\varepsilon^{kp}+p_{0j}-a_j\leqslant M(1-x_{0j}^{kp}),\forall k\in K,j\in N,p\in P;\tag{10}$$

$$a_i+s_i+p_{i,n+1}-\sigma^{kp}\leqslant M(1-x_{i,n+1}^{kp}),\forall k\in K,i\in N,p\in P;\tag{11}$$

$$\sigma^{k,p-1}+\varepsilon^{kp}+p_{0,i}-a_i\leqslant M(1-x_{0i}^{kp}),\forall k\in K,i\in N,p\in P;\tag{12}$$

$$a_i+s_i+p_{ij}-a_j\leqslant M(1-\sum_{k\in K}\sum_{p\in P}x_{ij}^{kp}),\forall k\in K,i\in N,p\in P;\tag{13}$$

$$\sigma^{kp}\leqslant\rho^{k(p+1)},\forall k\in K,p\in P\backslash\{p_{\max}\};\tag{14}$$

$$\sum_{v\in w_i}u_i^v=1,\forall i\in N;\tag{15}$$

$$\sum_{v\in w_i}u_i^vE_i^v\leqslant a_i\leqslant\sum_{v\in w_i}u_i^vL_i^v,\forall i\in N;\tag{16}$$

$$0\leqslant\sigma^{kp}-\rho^{kp}\leqslant L,\forall k\in K,p\in P;\tag{17}$$

$$\sum_{p\in P}(\sigma^{kp}-\rho^{kp})\leqslant S,\forall k\in K;\tag{18}$$

$$x_{ij}^{kp} \in \{0,1\}, \forall i,j \in V, i \neq j, k \in K, p \in P; \quad (19)$$

$$u_i^v \in \{0,1\}, \forall i \in N, v \in w_i; \quad (20)$$

$$a_i \geqslant 0, \forall i \in N; \quad (21)$$

$$0 \leqslant r_{ij} \leqslant Q x_{ij}^{kp}, \forall k \in K, p \in P; \quad (22)$$

$$0 \leqslant \varepsilon^{kp} \leqslant \alpha Q, \forall k \in K, p \in P; \quad (23)$$

$$E_0 \leqslant \rho^{kp} \leqslant L_0, \forall k \in K, p \in P。 \quad (24)$$

目标函数 (1) 为最小化车辆数量和总运输成本；式 (2) 要求任一客户点都必须服务；式 (3) 要求车辆能够在车程上按照顺序为客户点提供服务；式 (4) 要求每个车程只能往返车场一次；式 (5) 要求车辆在行驶路径中，所有车程按照顺序执行；式 (6) 要求车辆按照编号顺序从小到大被调用；式 (7) 表示车辆的容量约束；式 (8) 为客户需求守恒约束；式 (9) 为车辆在各个车程上的装货时间；式 (10) 为车辆在每一车程上的开始服务时间与第一个客户点到达时间之间的关系；式 (11) 为车辆在每一车程上完成服务的时间与最后一个客户点开始服务时间之间的关系；式 (12) 为车辆在上一车程上完成服务的时间与下一车程第一个客户点开始服务时间的关系；式 (13) 为车辆在每一车程上相继到达的两个客户点的开始服务时间之间的关系；式 (14) 要求车辆在上一车程上的完成时间必须早于车辆在该车程上的开始时间；式 (15) 要求客户点只能在一个时间窗内接受服务；式 (16) 为客户的开始服务时间范围；式 (17) 为车辆在每一车程上的工作时长约束；式 (18) 为车辆在其路径上的工作时长约束；式 (19) ~ (24) 为变量约束。

2 改进的自适应大邻域搜索算法

VRPMTMTW 为 VRP 问题的扩展，属于 NP 难问题，精确算法和 CPLEX 等优化求解器难以在可接受的时间内求解该问题的大规模实例，因此，本文针对问题设计了 IALNS 算法。

IALNS 算法框架如图 1 所示。首先，采用贪婪算法的思想生成较优的初始解；其次，使用轮盘赌法自适应地选择针对路径、车程及客户点 3 个层级上的多个破坏算子和修复算子以生成新解，依据模拟退火的接收准则更新当前解，并根据对解的优化情况更新算子权重；最后，依据最大迭代次数和最长求解时间作为算法的终止条件。

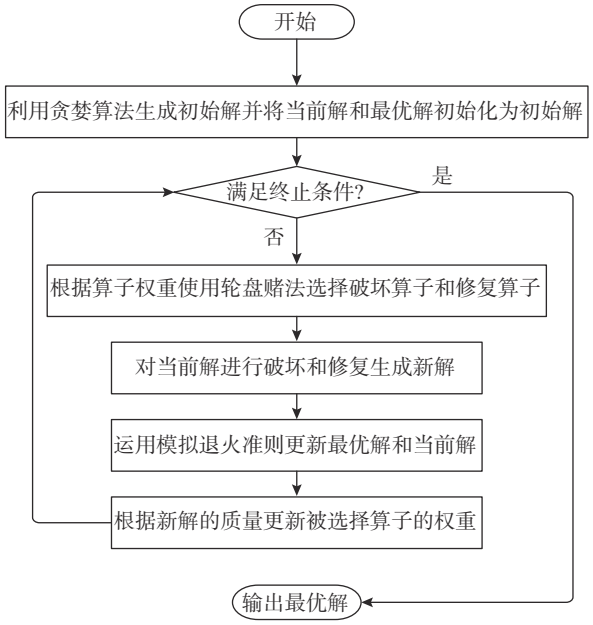

图 1 IALNS 算法流程图

Figure 1 A framework of the IALNS algorithm

2.1 初始解的构造

初始解作为后续算法迭代的基础，对后续解的优化有较大的影响，较优的初始解有助于算法更快地找到更优可行解。初始解构造的具体步骤见算法 1，其中 curRoute 为当前路径，curTrip 为当前车程，C_s 为记录可插入客户的集合，C_{nl} 为未插入路径的客户集，cost 为插入客户点所增加的成本值。

算法 1：初始解的构造

```
初始化 Cs, Cnl, cost;
while Cnl ≠ ø
  for (i = 0; i<Cnl.size () ; i++)
    while curRoute 的时长 < S
      开辟一个新的路径;
      while curTrip 的时长 < L
        开辟一个新的车程;
        for curTrip 的各位置 pos
          将 i 插入到 curTrip 的 pos 位置;
          if curTrip.isFeasibleTimeAndLoad ()
            Cs.pushback (curTrip, pos, cost) ;
          end if
        end for
        按 cost 值对 Cs 升序排列并选择 Cs 的第一个方案插入 curTrip;
        清除 Cnl 中被插入的客户点, 清空集合 Cs;
        更新当下车程时间;
      end while
      更新当路径时间 i;
    end while
  end for
end while
```

2.2　破坏算子

破坏算子是指通过一种特定的方式，在某一层级上 (路径、车程、客户点) 对当前解进行破坏的操作。破坏操作能增大解的不确定性，有效跳出局部最优解，深度搜索解空间，为后续修复算子生成新解提供了更大可能性。

1) 路径破坏算子。

依据路径总数随机确定破坏的路径个数 g_1 = rand (routeNum)，破坏当前解 currentSol 中 g_1 条路径，将被破坏路径上的客户点存入未被插入客户集合 C_{nl}。

随机路径破坏算子：随机移除解中的路径，直到满足移除数量为止。

最差路径破坏算子：依次破坏总成本增加最大的路径，直到满足移除数量为止。

相关性路径破坏算子：依次破坏相关性最大的路径，直到满足移除数量为止。

2) 车程破坏算子。

依据车程总数随机确定破坏的车程个数 g_2 = rand (vehicleNum)，破坏当前解 currentSol 中 g_2 个车程，将被破坏车程上的客户点存入未被插入客户集合 C_{nl}。

随机车程破坏算子：随机移除解中的车程，直到满足移除数量为止。

最差车程破坏算子：依次破坏总成本增加最大的车程，直到满足移除数量为止。

相关性车程破坏算子：依次破坏相关性最大的车程，直到满足移除数量为止。

3) 客户点破坏算子。

依据客户点总数随机确定破坏的客户点个数 g_3 = rand (customerNum)，破坏当前解 currentSol 中 g_3 个客户点，将被破坏的客户点存入未被插入客户集合 C_{nl}。

随机客户点破坏算子：对当前解 currentSol 路径上的客户点随机破坏，直到满足移除数量为止。具体步骤见算法 2，其中 currentSol.randomChoose () 函数为随机选择要破坏的路径，currentSol.remove () 函数为删除没有客户节点的路径。

算法 2：随机客户点破坏算子

初始化当前解 currentSol, i := 0;
while $i < g_3$
　route = currentSol.randomChoose () ;
　currentSol.remove (route, currentSol) ;
　将 route 对应的客户点插入待插客户点集合 C_{nl};
　i++;
end while

最差客户点破坏算子：依次破坏总成本增加最大的客户点，直到满足移除数量为止。具体步骤见算法 3，其中 $\Delta cost_i$ 为移除各客户点后路径所节省的成本差值，C_{temp} 为记录客户点 id、位置和 $\Delta cost_i$ 值的集合。

算法 3：最差客户点破坏算子

初始化 $\Delta cost_i$, C_{temp}, $i = g_3$;
while $i > 0$
　for i 属于已经遍历的客户节点
　　计算出移除客户节点插入路径中的 $\Delta cost_i$,并将客户点 id, pos, $\Delta cost_i$ 存储于集合 C_{temp} 中;
　end for
　将 C_{temp} 中的元素按照 $\Delta cost_i$ 降序排列,并将位于 C_{temp} 第 1 位的客户点 c 从 C_{temp} 中移除，插入到 C_{nl};
　if 路径 route 无客户点
　　删除路径 route;
　end if
　清空集合 C_{temp};
　i--;
end while

相关性客户点破坏算子：依次破坏相关性最大的客户点，直到满足移除数量为止。具体步骤见算法 4，其中 C_{re} 为待删除客户点集合，γ 为权重更新参数，θ 为随机性参数，按照不同概率选择集合中的客户节点，R 为两个客户节点之间的关联程度。

算法 4：相关性客户点破坏算子

初始化 C_{re}, C_{temp}, C_{nl}.size () , $0 < \gamma < 1$, $\theta \geq 1$;
从 currentSol 中随机选择一个客户节点存入 C_{re} 集合;
while C_{re}.size () $< g_3$
　从 C_{re} 集合中随机选择一个客户节点;
　for 遍历当前解中不属于 C_{re} 各客户节点集合
　　计算两个客户节点之间的关联程度 R,并存储于 C_{temp};
　end for
　按 R 值对集合 C_{temp} 升序排序;
　将 C_{temp} 第 $[C_{temp}.size () \times \gamma^{\theta}]$ 位的客户节点存入 C_{re} 集合;
　清空集合 C_{temp};
end while
将 C_{re} 集合中的全部客户节点逐个移除;
将客户节点存入未服务客户点集合 C_{nl};
清空集合 C_{re}

2.3　修复算子

将未被插入客户点集合 C_{nl} 中的客户点按修复算子的规则依次插入路径中。修复算子旨在进一步对当前解进行优化，为后续解的优化实现更多选择。

贪婪修复算子：本文使用两个贪婪算子 GI1 和 GI2。GI1 为在每一次迭代中将客户点修复到更优的位置，贪婪算子 GI2 为将客户点修复到最小成本的位置。GI2 是 GI1 的补充，通过引入修复的随机性来解决客户点无法重新被修复到更优位置的问题。以下两种情况需要说明：1) 当客户点未被重新重建到最优位置，且存在可调用的车辆时，则添加新车辆，构造新的路径及车程，将客户点重建最优可行位置；2) 存在一些由于成本过高而被放在最后的客户点，在无法调用新车辆的情况下，无法进行重新重建。具体步骤见算法 5，其中 C_a 为未插入客户的数量。

算法 5：贪婪修复算子

初始化 C_{temp}, $C_a = C_{nl}$.size () , f_i;
while $i < C_a$
  for 集合 C_{nl} 中的各个客户节点 i
    for currentSol 遍历各路径 route 的各位置 pos
      计算并存储插入该位置带来的成本 f_i 并存储在 C_{temp};
      if tempSol 可以插入当前车辆的行驶车程中
        对 C_{temp} 中的元素进行升序排列,并插入最小 f_i 所在的节点;
      else 存在 t_r 条空车程的一个新的路径
        将未插入的客户节点插入到车程中的最优位置中;
      end if
    end for
  end for
  i++;
end while

最优修复算子：从待插入集合中依次选择满足时间窗约束以及载重约束的插入时总成本增值最小的客户点插入到最优位置，直至待插入集合 C_{nl} 为空。具体步骤见算法 6。

算法 6：最优修复算子

初始化当前解 currentSol, i := 0;
计算未插入的客户数量 g_3;
while $i < g_3$
  location = currentSol. minedcostinsert (i , currentSol) ;
  currentSol. inserted (i , location , currentSol) ;
  i++;
end while

后悔值修复算子：基于最大后悔值修复原则进行修复，克服了贪婪修复算子的短视行为。首先计算客户点 i 修复到每条路径的最小修复成本，并将其按照升序的方式进行排列。定义 $f_i^{n_k}$ 为第 n_k 小的修复成本，后悔值为成本的差值，客户 i 的后悔值计算公式为 $\sum_{k=2}^{m}(f_i^{n_k} - f_i^{n_{k-1}})$，$m$ 表示后悔值参数，其作用是避免贪婪修复算子的短视行为，解决客户点难以被重新修复的情况。在每次迭代中，将后悔值最大的客户点插入到一个最优可行位置。需要说明两种情况：1) 若某客户点可行插入的位置数量小于后悔值参数，则在可行修复的客户点中选择修复成本最小的点插入最优可行位置；2) 当客户点未被重新插入到最优位置且存在可调用的车辆时，则添加新的车辆，构造新的路径及车程，将客户点插入到最优可行位置。具体步骤见算法 7。

算法 7：后悔值修复算子

初始化 C_{temp}, $C_a = C_{nl}$.size () , f_i;
while $i < C_a$
  for 集合 C_{nl} 中的客户节点 i
    for currentSol 的各路径 route 的各位置 pos
      计算并存储插入该位置的成本 $f_i^{n_k}$;
      计算出当前客户节点 i 的后悔值 $\sum_{k=1}^{m}(f_i^{n_{k+1}} - f_i^{n_k})$ 并存储在 C_{temp} 中;
      将 C_{temp} 中的元素升序排序,选择后悔值最大的进行插入;
      if 存在客户点无可行插入位置
        添加一条新的路径，将未插入的客户节点插入到路径中的最优位置中;
      end if
    end for
  end for
  i++;
end while

贪婪随机自适应修复算子：计算每个客户点在最优插入位置的总成本增值，按照插入总成本越小权重越大的原则为每一个待插入客户点分配权重，利用轮盘赌法选择插入的客户点，并将该客户点插入到最优位置。具体步骤见算法 8，其中 $\Delta cost_2$ 为插入各客户点后路径所增加的成本值，biggestCost 为最大的成本增加值 $\Delta cost_2 + 0.1$，$costWeight_i$ 为插入 i 的权重值，$standardCW_i$ 为插入 i 的归一化权重值，totalPossibility 为累计的归一化权重值，randDouble 为随机生成的 0 到 1 之间参数。

算法 8：贪婪随机自适应修复算子

初始化当前解 currentSol, i := 0;
计算未插入的对象数量 g_3;
while $i < g_3$
  按 $\Delta cost_2$ 值对集合 C_{temp} 升序排序;

biggestCost = $\Delta cost_{2max}$ + 0.1;
$costWeight_i$ = biggestCost − $\Delta cost_i$;
$standardCW_i = costWeight_i / \sum_{i \in V} costWeight_i$;
轮盘赌法生成概率数 possibility = randDouble ( ) ;
for 集合 C_{temp} 中的各条记录 j **do**
　totalPossibility += $C_{temp}[j]$.standardCW;
　if totalPossibility ≥ possibility **then**
　　$j^* = j$;
　　break;
　end if
end for
选择第 j^*个插入方案,将客户 j 插入;
待修复客户集合中删除 j 并清空集合 C_{temp};
i++;
end while

2.4　解的接收准则

为了避免陷入局部最优解，算法中采用模拟退火新解接受准则。若新解优于当前解，则接受。若新解劣于当前解，则根据概率$e^{-(f(Sol')-(f(Sol))/T}$来判断是否接受。其中 T 为温度，在每次迭代后根据冷却系数降低，$f(\cdot)$ 为目标函数值，Sol 和 Sol'分别表示当前解决方案和新的解决方案。随着温度的下降，算法接受当前非改进解的概率越来越低。

2.5　自适应设计

1) 算子权重的更新。

算子权重表示算子被选中的概率，算子权重越大，说明在以往迭代过程中表现越好，有更大的可能性探索到未出现过的解，其被选中的概率越大。破坏算子和修复算子的初始权重平均分配，迭代过程中依据接受准则传回的算子得分对算子作适应性调整。设ω_n^o和ω_{n+1}^o分别表示算子 o 的原始权重与调整后的权重，ω^o为算子 o 当前阶段权重，其中$\omega^o = W^o \Big/ \sum_{o \in O} W^o$，$W^o$ 为算子 o 当前阶段的得分，即调整为$\omega_{n+1}^o = (1-\lambda)\omega_n^o + \lambda W^o$，$\lambda$ 是控制系数，其取值表示算子下一阶段的权重是侧重于本阶段权重还是过去所有阶段权重，其中 $0 < \lambda < 1$。

2) 轮盘赌法则选择算子。

轮盘赌法则应用于迭代过程中对破坏算子和修复算子的选择，通过随机生成 0 ~ 1 的概率数 Pr，当算子累计权重大于 Pr 时，即选择第 n 个算子。

3　数值实验分析

3.1　算例介绍及参数设置

测试算例分为两部分：1) 客户点数量为 10、30 的 3 种数据类型 (含 C 型聚类分布、R 类随机分布和 RC 类随机聚类分布) 的小规模基准测试算例；2) 依据某城市快递公司的客户点信息特征随机生成客户点数量分别为 5、10、20、50、100 的算例。

在数值实验中，算法总迭代次数 H_1 为 30 000 次，最长运行时间 G 为 10 800 s，片段迭代次数 H_2 为 200 次，权重更新参数 η 为 0.1，获得新全局最优解的分数τ_1为 33，获得更好解的分数τ_2为 13，获得被接受的较差解的分数τ_3为 9，模拟退火初始温度系数 T_{start} 为 0.35，模拟退火冷却系数μ为 0.999 75，相关性破坏算子参数ϕ_1为 0.50，ϕ_2为 0.25，ϕ_3为 0.15，ϕ_4为 0.10，运距系数 b 为 1，车辆最大载重量 Q 为 40 件。

3.2　实验结果与对比分析

本文实验计算机配置为 LAPTOP-1PEO6NHS AMD Ryzen 7-5800H 3.20 GHz、4 核、16 GB 内存，IDE 采用 Visio Studio 2019 和 CPLEX 12.8，使用 C++语言。实验结果见表 2 ~ 表 6，其中，车辆数为 VN (辆)，总成本为 TC (元)，求解时间为 RT (s)。

3.2.1　模型和算法的有效性验证

1) 小规模基准测试算例。

将 IALNS 算法对客户点数量为 10、30 的 3 种数据类型基准算例进行求解，并与最优解进行对比，结果如表 2 所示。其中，ΔObj 为 IALNS 算法求得当前最好解与已知最优解的车辆偏差数，Gap 为 IALNS 算法求得当前最好解与已知最优解总成本偏差率。

由表 2 可见，针对小规模基准测试算例，IALNS 算法的求解结果相对于已知最优解来说，车辆偏差数均为 0，总成本偏差率也均为 0，说明 IALNS 算法求解的结果与已知最优解相等，证明了 IALNS 算法的有效性。

2) 小规模随机算例。

将 IALNS 算法对随机生成的 5、10、20 个客户点的小规模随机算例进行求解，并与 CPLEX 求解结果进行对比，结果如表 3 所示。其中，ΔObj_2 为 IALNS 算法求得当前最好解与 CPLEX 最好解的车辆偏差数，Gap_2 为 IALNS 算法求得当前最好解与 CPLEX 最好解总成本偏差率。

由表 3 可见，针对小规模随机算例，IALNS 算法的求解结果相对于 CPLEX 求解结果来说，车辆偏差数大多为 0，少部分为−1，总成本偏差率在

−7.09% ~ 0.00% 之间。对于 CPLEX 在有限时间内求得最优解的算例，IALNS 可在更短时间内求得相同最优解；对于 CPLEX 无法在有限时间内求得最优解的算例，IALNS 可在更短时间内求得相对 CPLEX 最好解更优的解。IALNS 的运行时间均小于 CPLEX 的求解时间，求解效率更高。

表 2　小规模基准算例的计算结果

Table 2　Calculation results of small-scale benchmark instances

算例	已知最优解		IALNS		ΔObj	Gap/%
	VN/辆	TC/元	VN/辆	TC/元		
R102	17	1 486.12	17	1 486.12	0.00	0.00
R111	10	1 096.72	10	1 096.72	0.00	0.00
C101	10	828.94	10	828.94	0.00	0.00
C106	10	828.94	10	828.94	0.00	0.00
RC104	10	1 135.48	10	1 135.48	0.00	0.00
RC107	11	1 230.48	11	1 230.48	0.00	0.00
R201	4	1 252.37	4	1 252.37	0.00	0.00
R206	3	906.14	3	906.14	0.00	0.00
C201	3	591.56	3	591.56	0.00	0.00
C202	3	591.56	3	591.56	0.00	0.00
RC204	3	798.41	3	798.41	0.00	0.00
RC206	3	1 146.32	3	1 146.32	0.00	0.00

表 3　随机算例 R5、R10、R20 的计算结果

Table 3　Calculation results of random instances R5, R10 and R20

算例	CPLEX			IALNS			ΔObj_2	Gap_2/%
	VN/辆	TC/元	RT/s	VN/辆	TC/元	RT/s		
R5_1	2	**185.29**	1.29	2	**185.29**	0.93	0	0.00
R5_2	1	**145.25**	0.62	1	**145.25**	0.54	0	0.00
R5_3	2	**159.20**	6.73	2	**159.20**	3.90	0	0.00
R5_4	1	**124.54**	3.24	1	**124.54**	1.74	0	0.00
R5_5	2	**163.52**	0.89	2	**163.52**	0.68	0	0.00
R10_1	3	**220.81**	126.71	3	**220.81**	4.62	0	0.00
R10_2	5	**348.39**	157.84	5	**348.39**	3.46	0	0.00
R10_3	3	**259.83**	87.38	3	**259.83**	2.42	0	0.00
R10_4	2	**278.64**	5 568.95	2	**278.64**	50.79	0	0.00
R10_5	6	**340.67**	200.60	6	**340.67**	2.69	0	0.00
R20_1	4	567.81	10 800	4	**564.39**	28.83	0	−0.60
R20_2	5	521.49	10 800	4	**484.53**	6.44	−1	−7.09
R20_3	6	498.26	10 800	5	**474.29**	56.81	−1	−4.81
R20_4	7	516.48	10 800	7	**509.53**	13.44	0	−1.35
R20_5	4	615.03	10 800	3	**611.96**	17.59	−1	−0.50

3.2.2　算法在大规模算例中的表现

将 IALNS 算法对随机生成的 50、100 个客户点的大规模算例进行求解，结果如表 4 所示。其中，ΔObj_3 为 IALNS 算法 5 次求解最好解与 5 次求解平均解的车辆偏差数，Gap_3 为 IALNS 算法 5 次求解最好解与 5 次求解平均解的总成本偏差率。

表 4　随机算例 R50 及 R100 的计算结果

Table 4　Calculation results of random instances R50 and R100

算例	$IALNS_{best}$		$IALNS_{avg}$		ΔObj_3	Gap_3/%
	VN/辆	TC/元	VN/辆	TC/元		
R50_1	5	648.64	5	652.01	0	−0.52
R50_2	5	523.49	5	525.06	0	−0.30
R50_3	4	636.37	4	641.47	0	−0.80
R50_4	5	562.84	5	565.94	0	−0.55
R50_5	5	499.37	5	502.76	0	−0.67
R100_1	8	1 326.12	8	1 331.00	0	−0.37
R100_2	6	1 193.82	6	1 196.28	0	−0.21
R100_3	10	1 107.60	10	1 110.74	0	−0.28
R100_4	6	1 033.12	6	1 037.03	0	−0.38
R100_5	8	1 126.41	8	1 129.98	0	−0.32

由表 4 可见，针对大规模随机算例，在 R50 的 5 个测试算例中，IALNS 算法求得最好解与平均解的偏差比率在−0.80% ~ −0.30% 之间；在 R100 的 5 个测试算例中，偏差比率在−0.38% ~ −0.21% 之间。以上结果表明，IALNS 算法求得最好解与平均解的偏差较小，求解效果稳定。

3.2.3　考虑多车程的模型对总运输成本的影响分析

考虑多车程的模型对总运输成本的影响分析结果如表 5 所示。其中 VRPMTW 为不考虑多车程的模型，VRPMTMTW 为考虑多车程的模型，其中，ΔObj_4 为 VRPMTMTW 模型求解结果与 VRPMTW 模型求解结果的车辆偏差数，Gap_4 为 VRPMTMTW 模型求解结果与 VRPMTW 模型求解结果的总成本偏差率。

表 5　VRPMTW 与 VRPMTMTW 的比较

Table 5　Comparison of results between VRPMTW and VRPMTMTW

算例	VRPMTW		VRPMTMTW		ΔObj_4	Gap_4/%
	VN/辆	TC/元	VN/辆	TC/元		
R5_1	2	185.29	2	185.29	0	0.00
R5_2	1	145.25	1	145.25	0	0.00
R5_3	2	161.51	2	159.20	0	−1.43
R10_1	3	220.81	3	220.81	0	0.00
R10_2	5	350.92	5	348.39	0	−0.72
R10_3	3	262.83	3	259.83	0	−1.14
R20_1	4	567.81	4	567.81	0	0.00
R20_2	5	487.72	4	484.53	−1	−0.65
R20_3	5	474.29	5	474.29	0	0.00
R50_1	5	659.62	5	648.64	0	−1.66
R50_2	5	523.49	5	523.49	0	0.00
R50_3	5	657.1	4	636.37	−1	−3.15
R100_1	9	1 374.42	8	1 326.12	−1	−3.51
R100_2	7	1 265.61	6	1 193.82	−1	−5.67
R100_3	10	1 124.06	10	1 107.60	0	−1.46

由表 5 可见，从车辆使用数量上来看，有 27% 的算例求解结果中多车程与单车程的模型车辆偏差数为−1，其他为 0；在大规模算例中，有 50% 的算例车辆偏差数为−1。从总成本来看，最大偏差率

为 −5.67%，最小偏差率为 0.00%，平均偏差率为 −1.44%，在大规模算例中总成本的差异较为明显，成本偏差率范围在−5.67% ~ −1.46% 之间。说明多车程的模型相对单车程的模型更优，且随着数据规模增大，这种优势更加明显。

3.2.4 IALNS 算法效果分析

为分析 IALNS 算法相较于基本 ALNS 算法改进的效果，在相同参数环境下，用基本 ALNS 算法求解结果并与 IALNS 算法求解结果进行对比，结果如表 6 所示。其中，ΔObj_5 为 IALNS 算法求解最好解与 ALNS 算法求解最好解的车辆偏差数，Gap_5 为 IALNS 算法求解最好解与 ALNS 算法求解最好解的总成本偏差率。

表 6 ALNS 算法与 IALNS 算法求解效果对比分析

Table 6 Comparison analysis of solutions obtained by ALNS and IALNS algorithms

算例	ALNS			IALNS			ΔObj_5	Gap_5/%
	VN/辆	TC/元	RT/s	VN/辆	TC/元	RT/s		
R20_1	4	567.81	158.87	4	567.81	28.83	0	0.00
R20_2	5	503.52	28.55	4	484.53	6.44	−1	−3.77
R20_3	6	487.28	115.28	5	474.29	56.81	−1	−2.67
R20_4	7	518.53	84.92	7	516.53	13.44	0	−0.39
R20_5	4	615.03	208.51	3	611.96	17.59	−1	−0.50

由表 6 可见，在 5 个算例中，有 3 个算例的车辆偏差数为−1，其他为 0，总成本偏差率在−3.77% ~ 0.00% 之间，且 IALNS 算法求解时间远远小于 ALNS 算法求解时间。说明 IALNS 算法优于 ALNS 算法。

4 总结与展望

本文以车辆使用数量、路径行驶成本最小化为目标，研究多车程多时间窗车辆路径问题，构建相应的混合整数规划模型。为高效率地求得模型的满意解，设计了改进的自适应大邻域搜索算法。通过求解 3 种数据类型的基准测试算例及随机生成的 5、10、20 个客户点的小规模算例，并与 CPLEX 求得的最优解进行对比，验证了算法的有效性。随后通过求解随机生成的 50、100 个客户点的大规模算例验证了算法的稳定性。与不考虑多车程的模型对比，充分说明了将多车程引入模型的必要性。最后分别使用普通 ALNS 算法和本文的 IALNS 算法对同一随机算例进行求解，证明了本文提出的 IALNS 算法的高效性。本文的 IALNS 算法在小规模案例上表现较好、在大规模算例的求解中表现稳定，在实际应用中对提升宏观物流网络配送方案规划的效率以及解决末端环节客户数量多、分布广散的实际配送难题提供一定的决策依据。

基于本文的研究结果，后续研究将考虑更符合实际的众多因素，比如多车场、不同车型、不同车速、存在客户优先级等。未来也可以设计基于此算法的改进算法，比如将自适应大邻域搜索算法与禁忌算法进行结合，也可以研究此算法在求解其他问题时的合理性和有效性，高效地解决更符合实际的车路径优化问题，为物流企业配送方案的制定提供更多的参考依据。

参考文献：

[1] 李珍萍, 赵菲, 刘洪伟. 多时间窗车辆路径问题的智能水滴算法[J]. 运筹与管理, 2015, 24(6): 1-10.
LI Zhenping, ZHAO Fei, LIU Hongwei. Intelligent water drops algorithm for vehicle routing problem with multiple time windows[J]. Operations Research and Management Science, 2015, 24(6): 1-10.

[2] BELHAIZA S, HANSEN P, LAPORTE G. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows[J]. Computers & Operations Research, 2014, 52: 269-281.

[3] BEHESHTI A K, HEJAZI S R, ALINAGHIAN M. The vehicle routing problem with multiple prioritized time windows: a case study[J]. Computers & Industrial Engineering, 2015, 90: 402-413.

[4] 闫芳, 王媛媛. 多模糊时间窗车辆路径问题的建模及求解[J]. 交通运输系统工程与信息, 2016(6): 182-188.
YAN Fang, WANG Yuanyuan. Modeling and solving the vehicle routing problem with multiple fuzzy time windows[J]. Journal of Transportation Systems Engineering and Information Technology, 2016(6): 182-188.

[5] MAO H, SHI J, ZHOU Y, et al. The electric vehicle routing problem with time windows and multiple recharging options[J].

IEEE Access, 2020, 8: 114864-114875.

[6] 李博威, 户佐安, 贾叶子, 等. 带软时间窗的同时取送货车辆路径问题研究[J]. 工业工程, 2020, 23(5): 75-81.
LI Bowei, HU Zuoan, JIA Yezi, et al. A research on vehicle routing problem with simultaneous pick-up and delivery and soft time windows[J]. Industrial Engineering Journal, 2020, 23(5): 75-81.

[7] 张瑾, 洪莉, 戴二壮. 求解带容量和时间窗约束车辆路径问题的改进蝙蝠算法[J]. 计算机工程与科学, 2021, 43(8): 1479-1487.
ZHANG Jin, HONG Li, DAI Erzhuang. An improved bat algorithm for the vehicle routing problem with time windows and capacity constraints[J]. Computer Engineering and Science, 2021, 43(8): 1479-1487.

[8] 闫军, 常乐, 王璐璐, 等. 带时间窗的同时取送货车辆路径问题求解算法[J]. 工业工程, 2021, 24(5): 72-76.
YAN Jun, CHANG Le, WANG Lulu, et al. A solution algorithm for the problem of taking delivery vehicle path at the same time with time window[J]. Industrial Engineering Journal, 2021, 24(5): 72-76.

[9] FANG W, GUAN Z, YUE L, et al. Heterogeneous-vehicle distribution logistics planning for assembly line station materials with multiple time windows and multiple visits[J]. International Journal of Industrial Engineering Computations, 2022, 13(4): 473-490.

[10] FLEISCHMANN B. The vehicle routing problem with multiple use of vehicles[R/OL]. (1980-01)[2022-12-19]. https://www.researchgate.net/publication/221704650_The_vehicle_routing_problem_with_multiple_use_of_vehicles.

[11] WASSAN N, WASSAN N, NAGY G, et al. The multiple trip vehicle routing problem with backhauls: formulation and a two-level variable neighbourhood search[J]. Computers & Operations Research, 2017, 78(C): 454-467.

[12] 张晓楠, 范厚明. 带时间窗偏好的多行程模糊需求车辆路径优化[J]. 计算机集成制造系统, 2018, 24(10): 2461-2477.
ZHANG Xiaonan, FAN Houming. Optimization for multi-trip vehicle routing problem with fuzzy demands considering time window preference[J]. Computer Integrated Manufacturing System, 2018, 24(10): 2461-2477.

[13] 宋强. 改进混合遗传算法在 MTVRPTW 中的建模与优化[J]. 重庆交通大学学报 (自然科学版) , 2018, 37(9): 79-86,134.
SONG Qiang. Application of improved hybrid genetic algorithm in the modeling and optimization of multi-trip vehicle routing problem with time windows[J]. Journal of Chongqing Jiaotong University (Natural Science) , 2018, 37(9): 79-86,134.

[14] 刘虹, 傅晓敏. 考虑同时取送随机需求的多行程车辆路径研究[J]. 西安电子科技大学学报 (社会科学版) , 2019, 29(3): 87-95.
LIU Hong, FU Xiaomin. Study on multi-trip vehicle routing problem with stochastic simultaneous pickup-delivery demand[J]. Journal of Xidian University (Social Science Edition) , 2019, 29(3): 87-95.

[15] RAHMANI A. The multiple trip vehicle routing problem with backhauls in random fuzzy environment: using (α, β) -cost minimization model under the Hurwicz criterion[J]. International Journal of Computer Mathematics, 2019, 96(12): 2548-2566.

[16] LI W, WU Y, KUMAR P N R, et al. Multi-trip vehicle routing problem with order release time[J]. Engineering Optimization, 2020, 52(8): 1279-1294.

[17] CATTARUZZA D, ABSI N, FEILLET D. Vehicle routing problems with multiple trips[J]. Annals of Operations Research, 2018, 271(1): 127-159.

(责任编辑: 刘敏仪)

(上接第 106 页)

[17] 邱宁佳, 王晓霞, 王鹏, 等. 结合迁移学习模型的卷积神经网络算法研究[J]. 计算机工程与应用, 2020, 56(5): 43-48.
QIU Ningjia, WANG Xiaoxia, WANG Peng, et al. Research on convolutional neural network algorithm combined with transfer learning model[J]. Computer Engineering and Applications, 2020, 56(5): 43-48.

[18] ZHANG K, ZHANG K, BAO R. Prediction of gas explosion pressures: a machine learning algorithm based on KPCA and an optimized LSSVM[J]. Journal of Loss Prevention in the Process Industries, 2023, 83: 105082.

[19] GRETTON A, FUKUMIZU K, HARCHAOUI Z, et al. A fast, consistent kernel two-sample test[C]//Proceedings of the 22nd International Conference on Neural Information Processing Systems (NIPS'09) . Red Hook, New York, USA: Curran Associates Inc. , 2009: 673–681.

[20] HU J, SHEN L, SUN G. Squeeze-and-excitation networks[C]// 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, Utah, USA: IEEE, 2018: 7132-7141.

[21] 王颖慧, 苏怀智. 基于 PCA-GWO-SVM 的大坝变形预测[J]. 人民黄河, 2020, 42(11): 130-134.
WANG Yinghui, SU Huaizhi. Dam deformation prediction based on PCA-GWO-SVM model[J]. Yellow River, 2020, 42(11): 130-134.

(责任编辑: 孟晓燕)