改进的自适应大规模邻域搜索算法求解动态需求的混合车辆路径问题*

南丽君, 陈彦如†, 张宗成
(西南交通大学 经济管理学院, 成都 610031)

摘 要: 为了给物流企业在车辆配送方案制定上提供决策支持,针对电动物流车与燃油物流车混合配送的模式,研究了带时间窗的动态需求车辆路径问题,建立了以配送总成本最小化为目标的两阶段整数规划模型。针对模型特点,设计了改进的自适应大规模邻域搜索(improved adaptive large neighborhood search,IALNS)算法,提出新的删除、修复算子及动态阶段加速策略,分别针对大规模的静态算例与动态算例进行算法性能测试。结果表明,与无改进策略的 IALNS(IALNS-ND)相比,静态问题中在相同的求解时间内 75% 的算例(12 个算例中 9 个)IALNS 得到的最小值和平均值优于 IALNS-ND,动态问题中 95%(60 个算例中 57 个算例)的算例可以得到成本和时间均优于 IALNS-ND 的解;与三种算法——自适应大规模邻域搜索算法(ALNS)、大规模邻域搜索算法(LNS)以及变邻域搜索算法(VNS)相比,静态问题中所有算例 IALNS 获得的总成本的最小值和平均值均优于三个对比算法,动态问题中 58%(60 个算例中 35 个算例)的算例 IALNS 能够以少于三个对比算法 1.5 倍甚至 10 倍的时间获得更优的解。同时随着问题动态度的提高,IALNS 的速度更快,质量更好,证明了该算法在求解时效性要求高的动态需求车辆路径问题的优越性。

关键词: 动态需求; 电动车车辆路径问题; 混合车队; 改进的自适应大规模邻域搜索算法
中图分类号: TP301.6;F506 **文献标志码**: A **文章编号**: 1001-3695(2021)10-008-2926-09
doi:10.19734/j.issn.1001-3695.2021.02.0050

Improved adaptive large neighborhood search algorithm for mixed fleet routing problem of dynamic demands

Nan Lijun, Chen Yanru†, Zhang Zongcheng
(*School of Economics & Management*, *Southwest Jiaotong University*, *Chengdu* 610031, *China*)

Abstract: In order to provide decision support for vehicle scheduling of logistics companies, this paper investigated the routing problem with time windows and dynamic demands considering a mixed fleet of electric and conventional vehicles, and proposed a two-stage integer programming model to minimize the total distribution cost. This paper designed an improved adaptive large-scale neighborhood search algorithm(IALNS), proposed the new deletion and repair operators and acceleration strategy in the dynamic stages. It conducted the extensive large-scale computational experiments with both static and dynamic demands to examine the performance of proposed IALNS. The results show that, compared to IALNS-ND, IALNS performs better in term of the minimum and average values in 75% of the static problems (9 out of 12 cases). In 95% (57 out of 60 examples) of the dynamic cases, IALNS works better than IALNS-ND in terms of the cost and computation time. Moreover, compared to ALNS, LNS and VNS, IALNS performs best in term of the best minimum and average values of the total costs for all static cases. In 58% (35 out of 60 examples) of the dynamic case, the IALNS can achieve a better solution in 1.5 times or even 10 times less computation time than the rest algorithms. Also the larger the degree of dynamism of a experiment is, the better the obtained solution obtained by IALNS in a shorter time. Thus IALNS performs best in solving the time-sensitive dynamic demand vehicle routing problem.

Key words: dynamic demands; electric vehicle routing problem; mixed fleet; IALNS

0 引言

近年来,随着环境问题的日益突出,越来越多物流配送企业开始使用节能环保的电动物流车。但是由于续航里程有限及充电设施布局不完善等问题,电动车并不能完全代替燃油车,所以大多物流企业目前主要采用电动车与燃油车混合配送的过渡模式。与燃油车配送不同,由于电动车需途中进行充电,改变了原有配送系统的配置参数,并对配送时间窗产生影响;此外,实际配送过程中配送中心会在车辆离开后继续接收新的需求,导致配送系统的状态不断发生变化,需要对已有配送方案重新进行调整。如果不能合理快速地规划与调整配送路线,不仅会大幅增加企业的运营成本,还会降低顾客服务水平。因此,研究电动车与燃油车混合配送模式下的动态需求车辆路径问题具有重要的现实意义。

车辆路径优化(vehicle routing problem,VRP)一直都是物流配送领域中的热点问题,从 1959 年 VRP 被 Dantzig 和 Ramser 提出后就受到了国内外学者的广泛关注,目前已有非常丰硕的研究成果[1]。结合各种现实场景,研究人员开始对基础的

收稿日期: 2021-02-09; **修回日期**: 2021-04-02 **基金项目**: 国家重点研发计划项目(2018YFB1601402);国家自然科学基金项目(71771190)
作者简介:南丽君(1997-),女,山西忻州人,硕士,主要研究方向为物流系统建模与优化;陈彦如(1974-),女(通信作者),内蒙古包头人,教授,博导,博士,主要研究方向为物流系统建模与优化、机器学习等(chenyanru@swjtu.cn);张宗成(1994-),男,安徽马鞍山人,博士研究生,主要研究方向为物流系统建模与优化、机器学习等。

VRP 进行扩展研究,如有载重约束的 VRP(CVRP)[2]、带时间窗约束的 VRP(VRPTW)[3]、多车型 VRP(HVRP)[4]等。使用电动物流车的 VRP 称为电动车车辆路径问题(EVRP),Conrad 等人[5]首次对 EVRP 进行研究,在允许车辆途中充电的条件下,建立使用车辆数目最少以及行驶成本、服务时间成本和充电成本最小的多目标模型;Schneider 等人[6]进一步提出了带时间窗的 EVRP,建立使用车辆数最小和总行驶成本最小的优化模型,并设计了基于变邻域搜索和禁忌搜索的混合算法;葛显龙等人[7]考虑到充电时间对时间窗的违反会存在影响,提出带软时间窗的 EVRP,建立了行驶成本、路径成本以及车辆使用成本为目标函数的数学模型,并设计了节约里程算法加改进的禁忌搜索算法进行求解;Keskin 等人[8]针对带时间窗的电动车车辆路径问题,考虑电动车在充电站采取部分充电的情况,设计了自适应大规模邻域搜索算法进行求解;Pelletier 等人[9]考虑电动车耗电为非线性的情况,研究天气、道路情况以及司机行为等不确定因素背景下的电动车车辆路径问题,通过将问题定义为鲁棒混合整数线性规划模型,并设计一种基于大规模邻域搜索的两阶段启发式算法进行求解;Alesian 等人[10]提出了可以多次访问充电站的电动车车辆路径问题,并采用一种带有学习策略的进化遗传算法找到最小化成本(与行驶时间、充电时间、能源消耗相关)的车辆配送路线;Yang 等人[11]考虑了分时电价的情况,采用可学习的遗传算法实现对车辆路径以及充电时间的同时优化。如果在同一配送系统中同时使用不同类型的车辆进行配送,考虑因素更多,问题更加复杂。Goeke 等人[12]首次对带时间窗的电动车与燃油车的混合配送问题(EVRPTWMF)进行研究,并设计了自适应大规模邻域搜索算法进行求解;Hiermann 等人[13]考虑带时间窗的同时使用传统燃油车、插电式混合动力车以及电动车三种车型的车辆路径问题,并设计基于遗传算法以及局部和大规模邻域搜索算法的混合启发式算法进行求解。

以上研究均属于静态 VRP,即所有客户需求可以事先确定。但在实际配送过程中,配送中心会在车辆运行途中继续接收新客户的配送需求,并对已有车辆路线重新进行规划,此类问题即动态需求的 VRP(DDVRP)。Hong[14]研究带硬时间窗的 DDVRP,并将动态问题分为一系列的静态问题,设计了改进的大规模邻域搜索算法对该问题进行求解;De Armas 等人[15]将车辆工作时间不同、顾客有多个时间窗、顾客之间存在优先级等现实因素考虑到动态车辆路径问题中,并用变邻域搜索算法求解;Chen 等人[16]进一步采用自适应大规模邻域搜索算法进行 DDVRP 的求解;张文博等人[17]针对带时间窗的 DDVRP 设计了遗传算法以及模拟退火结合的两阶段算法;李阳等人[18]针对带载重约束的动态需求车辆路径问题,提出了一种延迟服务机制,且采用混合变邻域人工蜂群算法进行多阶段求解。

综上所述,目前 DDVRP 已有一定的研究成果,但是这些研究都是基于传统燃油车。邵赛等人[19]首次将电动车引入动态需求车辆路径问题中,研究了不考虑客户时间窗的电动车配送的 DDVRP。通过国内外的文献检索,目前还没有发现其他针对电动车的 DDVRP 的研究,此外也未有针对电动车与燃油车混合配送模式下的 DDVRP 的研究。不同于单一燃油车或电动车 DDVRP,混合配送模式将电动车与燃油车这两种具有不同配送特点的车辆纳入同一配送系统同步考虑,限制因素更多,尤其需同时考虑动态需求和客户时间窗,模型构建与求解将更为复杂。考虑到该类问题在企业日常运营中会越来越普遍,本文研究了电动车与燃油车混合配送模式下带时间窗的动态需求车辆路径问题(electric vehicle routing problem with time windows and mixed fleet considering dynamic demands, EVRPTWMF-DD)。此外,考虑到动态需求对实时性要求较高,而本文研究的 EVRPTWMF-DD 又属于 NP-hard 问题,用精确算法求解较为困难且时间成本高。因此,本文根据所建模型的特点设计了改进的自适应大规模邻域搜索算法,以期在较短的时间内获得较优的配送方案。

1　问题描述

EVRPTWMF-DD 可描述为某物流企业使用电动车和燃油车为 N 个有时间窗要求的顾客提供配送服务。首先,对上一工作日的预约静态客户制定配送路线并进行送货,在配送过程中,若出现新的客户需求,需要对正在配送的车辆路径进行重新规划或派出新车进行服务。此外,电动车由于续航里程有限,在配送过程中需要到公共充电站充电才能继续进行配送任务。该问题的求解目标是寻找车辆行驶总成本最小化的最优路径。模型的假设条件如下:a)每个客户点只能被一辆车服务,每辆车可以服务多个客户点,车辆在完成配送服务后要驶回配送中心;b)电动车从配送中心出发时电池为满电,并且允许在行驶过程中采用满充的方式进行多次充电,充电站可以被多次访问。为了更好地描述问题,给出如图 1 所示的简单示例。本文建立了 EVRPTWMF-DD 的两阶段 0-1 整数规划模型,包括初始阶段优化模型和动态阶段优化模型。

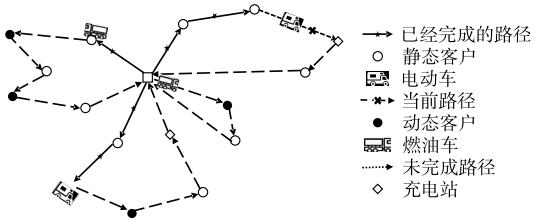

图1　问题描述示例
Fig.1　Sample diagram of problem

2　模型构建

2.1　初始阶段优化模型

初始阶段主要对已经预约的静态客户点进行路线设计,得到初始的配送路径优化方案。模型所用参数与变量如表 1 所示。初始阶段优化模型构建如下:

$$\min Z=\sum_{i\in V_1}\sum_{j\in V_2}\sum_{k\in K_e}x_{ijk}\times d_{ij}\times\lambda_e+\sum_{i\in V_1}\sum_{j\in V_2}\sum_{k\in K_c}x_{ijk}\times d_{ij}\times\lambda_c \tag{1}$$

$$\sum_{i\in V_1}\sum_{k\in K_{1e}}x_{ijk}+\sum_{i\in V_1}\sum_{k\in K_{1c}}x_{ijk}=1\quad\forall j\in N \tag{2}$$

$$\sum_{i\in V_1}x_{ijk}=\sum_{i\in V_1}x_{jik}\quad\forall j\in V_2\backslash\{n+1\},\forall k\in K_1 \tag{3}$$

$$\sum_{j\in V_2}x_{0jk}\leqslant 1\quad\forall k\in K_1 \tag{4}$$

$$\sum_{i\in Nj\in V_2}q_i\times x_{ijk}\leqslant Q_e\quad\forall k\in K_{1e} \tag{5}$$

$$\sum_{i\in Nj\in V_2}q_i\times x_{ijk}\leqslant Q_c\quad\forall k\in K_{1c} \tag{6}$$

$$a_i\leqslant t_{ik}^1\leqslant b_i\quad\forall i\in N,\forall k\in K_1 \tag{7}$$

$$t_{ij}=d_{ij}/v\quad\forall i\in V_1,\forall j\in V_2 \tag{8}$$

$$s_k^i=(B-E_{ik}^1)/v_1\quad\forall i\in F',\forall k\in K_{1e} \tag{9}$$

$$t_{ik}^2=t_{ik}^1+s_{ik}\quad\forall i\in N\cup F',\forall k\in K_1 \tag{10}$$

$$t_{jk}^1\geqslant t_{ik}^2+t_{ij}-M(1-x_{ijk})\quad\forall i\in V_1,\forall j\in V_2,\forall k\in K_1 \tag{11}$$

$$E_{ik}^2=E_{ik}^1\quad\forall i\in N,\forall k\in K_{1e} \tag{12}$$

$$E_{ik}^2=B\quad\forall i\in F',\forall k\in K_{1e} \tag{13}$$

$$E_{jk}^1\leqslant E_{ik}^2-v_2\times d_{ij}\times x_{ijk}+B(1-x_{ijk})\quad\forall i\in V_1,\forall j\in V_2,\forall k\in K_{1e} \tag{14}$$

$$E_{ik}^1>0\quad\forall i\in V_3,\forall k\in K_{1e} \tag{15}$$

$$x_{ijk}\in\{0,1\}\quad\forall i,j\in V_3,\forall k\in K_1 \tag{16}$$

式(1)为目标函数,最小化包含电动车与燃油车的运输成

本;式(2)表示每个客户只能被访问一次;式(3)为流守恒约束;式(4)表示每辆车最多服务一条路径且从配送中心出发;式(5)表示电动车配送的任务总量不能超过其最大载重;式(6)表示燃油车的配送任务总量不能超过其最大载重量;式(7)表示要满足客户点 i 的服务时间窗要求;式(8)表示从点 i 到 j 的行驶时间为两点之间距离与速度的比值;式(9)表示电动车在充电站充电时间的计算;式(10)表示车辆在到达和离开点 i 的时间关系;式(11)表示车辆到达点 j 的时间为前边行驶时间的累计;式(12)表示电动车到达和离开客户点的电量不发生改变;式(13)表示电动车在充电站满充;式(14)表示电动车从点 i 行驶到点 j 的电量消耗关系;式(15)表示电动车到达每一个点的电量都要大于0;式(16)为0-1变量约束条件。

表1 参数以及变量符号说明

Tab. 1 Description of parameters and variable symbols

符号	符号说明	符号	符号说明
0	作为起点的配送中心	$n+1$	作为终点的配送中心
N	顾客集合	F	充电站集合
K_1	车辆集合	F'	充电站访问集合,充电站集合 F 的虚拟点(一个充电站被访问 n 次,即可看成1个充电站和 $n-1$ 个虚拟充电站)
K_{1e}	电动车集合		
K_{1c}	燃油车集合		
$V_1=N\cup F'\cup\{0\}$	客户点、充电站、起点集合	Q_e	电动车最大载重
$V_2=N\cup F'\cup\{n+1\}$	客户点、充电站、终点集合	Q_c	燃油车最大载重
q_i	客户点 i 的需求	t_{ik}^1	车辆 k 到达节点 i 的时间
t_{ik}^2	车辆 k 离开节点 i 的时间	s_{ik}	车辆 k 在客户点(充电站)的服务(充电)时间
$[a_i,b_i]$	客户点 i 的服务时间窗	t_{ij}	从节点 i 到的行驶时间
λ_e/λ_c	电动车/燃油车的单位行驶成本	v	车辆在配送过程中的行驶速度
d_{ij}	节点 i 与 j 之间的距离	v_2	电动车的耗电速率
v_1	电动车在充电站的充电速率	E_{ik}^2	车辆 k 离开节点的电量
E_{ik}^1	车辆到到达节点 i 的电量	x_{ijk}	0-1决策变量,表示车辆 k 是否经过弧 (i,j)
M	为一个无穷大的数		

2.2 动态阶段优化模型

当车辆配送途中出现了新客户的需求,则进入动态优化阶段。在一定的更新时刻将上一阶段未服务的顾客和新顾客作为模型输入,进行路线的重新规划使得总成本最优。该阶段与初始阶段的主要不同在于在更新时刻各出发车辆处于配送途中或客户点、充电站等位置而不是配送中心,因此增加了模型的复杂性,将更新时刻车辆所处位置称为虚拟点。动态阶段模型所用参数以及变量如表2所示。

表2 动态阶段参数以及变量符号说明

Tab. 2 Description of parameters and variable symbols in the dynamic phase

符号	符号说明	符号	符号说明
C_{0i}	上阶段结束后从起点到虚拟点的成本	N_1	上阶段未服务的顾客与新顾客总和
$K_2=K_{2e}+K_{2c}$	新增车辆(电动车\燃油车)	D	该阶段所有车辆集合
$L_1=N_1\cup F'\cup\{0\}$	顾客、充电站、起点集合	M_1	虚拟点在路中
$L_2=N_1\cup F'\cup M$	顾客、充电站、虚拟点集合	M_2	虚拟点在客户点处
$L_3=N_1\cup F'\cup\{n+1\}$	顾客、充电站、终点集合	M_3	虚拟点在充电站处
$L=L_2\cup\{0\}\cup\{n+1\}$	所有点的集合	E_{jk}^0	开始阶段车辆在充电站虚拟点的电量
Q_{1e}	上阶段电动车的剩余载重	Q_{1c}	上阶段燃油车的剩余载重
t_t	更新时刻		

动态阶段优化模型构建如下:

$$\min Z=\sum_{i\in M}C_{0i}+\sum_{k\in K_{1e}}\sum_{i\in L_2}\sum_{j\in L_3}x_{ijk}\times d_{ij}\times\lambda_e+\sum_{k\in K_{1c}}\sum_{i\in L_2}\sum_{j\in L_3}x_{ijk}\times d_{ij}\times\lambda_c+\sum_{k\in K_{2e}}\sum_{i\in L_1}\sum_{j\in L_3}x_{ijk}\times d_{ij}\times\lambda_e+\sum_{k\in K_{2c}}\sum_{i\in L_1}\sum_{j\in L_3}x_{ijk}\times d_{ij}\times\lambda_c \tag{17}$$

$$\sum_{i\in L/\{n+1\}}\sum_{k\in K_{1e}\cup K_{2e}}x_{ijk}+\sum_{i\in L/\{n+1\}}\sum_{k\in K_{1c}\cup K_{2c}}x_{ijk}=1\quad\forall j\in N_1 \tag{18}$$

$$\sum_{j\in L_3}x_{ijk}=1\quad\forall k\in K_1,\forall i\in M \tag{19}$$

$$\sum_{j\in N_1}x_{0jk}\leqslant 1\quad\forall k\in K_2 \tag{20}$$

$$\sum_{i\in N_1\cup F'}x_{ijk}=\sum_{i\in N_1\cup F'}x_{jik}\quad\forall j\in N_1\cup F',\forall k\in D \tag{21}$$

$$\sum_{i\in L_2}x_{i(n+1)k}=1\quad\forall k\in D \tag{22}$$

$$\sum_{i\in N_1\cup M_2}\sum_{j\in L_3}q_i\times x_{ijk}\leqslant Q_{1e}\quad\forall k\in K_{1e} \tag{23}$$

$$\sum_{i\in N_1\cup M_2}\sum_{j\in L_3}q_i\times x_{ijk}\leqslant Q_{1c}\quad\forall k\in K_{2e} \tag{24}$$

$$\sum_{i\in N_1}\sum_{j\in L_3}q_i\times x_{ijk}\leqslant Q_e\quad\forall k\in K_{2e} \tag{25}$$

$$\sum_{i\in N_1}\sum_{j\in L_3}q_i\times x_{ijk}\leqslant Q_c\quad\forall k\in K_{2c} \tag{26}$$

$$a_i\leqslant t_{ik}^1\leqslant b_i\quad\forall i\in N_1,\forall k\in D \tag{27}$$

$$t_{ij}=d_{ij}/v\quad\forall i\in L/\{n+1\},\forall j\in L_3 \tag{28}$$

$$s_{ik}=(B-E_{ik}^1)/v_1\quad\forall i\in F',\forall k\in K_{1e}\cup K_{2e} \tag{29}$$

$$s_{jk}=(B-E_{ik}^0)/v_1\quad\forall i\in M_3,\forall k\in K_{1e} \tag{30}$$

$$t_{ik}^2=t_{ik}^1+s_{ik}\quad\forall i\in N_1\cup F',\forall k\in D \tag{31}$$

$$t_{ik}^2=t_t+s_{ik}\quad\forall i\in M,\forall k\in K_1 \tag{32}$$

$$t_{jk}^1\geqslant t_{ik}^2+t_{ij}-M(1-x_{ijk})\quad\forall i\in L_1,\forall j\in L_3,\forall k\in K_2 \tag{33}$$

$$t_{jk}^1\geqslant t_{ik}^2+t_{ij}-M(1-x_{ijk})\quad\forall i\in L_2,\forall j\in L_3,\forall k\in K_1 \tag{34}$$

$$E_{ik}^2=E_{ik}^1\quad\forall i\in N_1\cup M_1\cup M_2,\forall k\in K_{1e}\cup K_{2e} \tag{35}$$

$$E_{ik}^2=B\quad\forall i\in F',\forall k\in K_{2e} \tag{36}$$

$$E_{ik}^2=B\quad\forall i\in F'\cup M_3,\forall k\in K_{1e} \tag{37}$$

$$E_{jk}^1\leqslant E_{ik}^2-v_2\times d_{ij}\times x_{ijk}+B(1-x_{ijk})\quad\forall i\in L/\{n+1\},\forall j\in L_3,\forall k\in K_{1e}\cup K_{2e} \tag{38}$$

$$E_{ik}^1>0\quad\forall i\in L,\forall k\in K_{1e}\cup K_{2e} \tag{39}$$

式(17)为目标函数,最小化包含上一阶段车辆从初始配送中心到虚拟点的成本(得到第一阶段的解后求得)、上一阶段剩余车辆以及新派出燃油车与电动车的运输成本;式(18)表示所有客户点被服务一次;式(19)保证配送途中的车辆从虚拟点出发;式(20)表示新的车辆从配送中心出发;式(21)为流守恒约束;式(22)表示所有车辆最后均返回配送中心;式(23)~(26)表示车辆服务的客户需求不可超过电动车或燃油车最大载重量;式(27)表示要满足客户点 i 的服务时间窗要求;式(28)表示从点 i 到 j 的行驶时间为两点之间距离与速度的比值;式(29)表示电动车在充电站充电时间的计算;式(30)

表示上一阶段电动车停在充电站时充电时间的计算;式(31)表示车辆到达和离开点 i 的时间关系;式(32)表示上一阶段车辆离开虚拟点的时间计算;式(33)(34)表示车辆从点 i 到点 j 的时间关系;式(35)表示电动车到达和离开客户点的电量相等;式(36)(37)表示车辆离开充电站时电量为满充;式(38)表示电动车从点 i 行驶到点 j 的电量消耗关系;式(39)表示电动车到达每一个点的电量均大于 0。

3　算法设计

3.1　问题求解策略

针对上述两阶段模型,本文采用定期更新的求解策略,即每隔 T 时段就进行一次路径更新,将上阶段未服务顾客以及新顾客均作为待配送客户输入优化模型中。与动态事件更新策略(即接收到新的需求就进行路径更新)相比,定期更新策略计算量小,同时更新时刻可自行调整,具有较高的柔性。问题求解流程如图 2 所示。

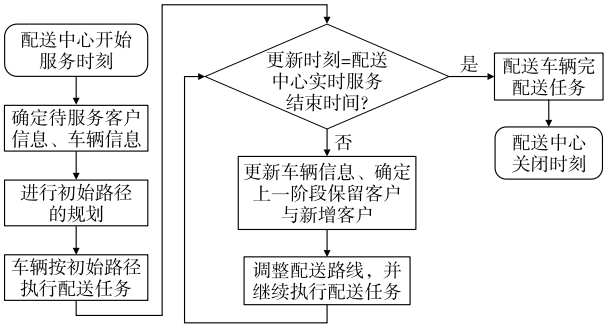

图2　问题求解流程

Fig.2　Process of problem solving

3.2　改进的自适应大规模邻域搜索算法

由于 DDVRP 对于时效性要求较高,且客户规模较大时,精确算法难以在有效时间内获得最优解方案,现有文献大多采用启发式算法,包括遗传算法(GA)[17]、蚁群算法(ACO)[20]、大规模邻域搜索算法(LNS)[14]、变邻域搜索算法(VNS)[15]、自适应大规模邻域搜索算法(ALNS)[16]等。

ALNS 算法[21]属于一种改进的邻域搜索算法,相对于其他启发式算法,其求解效率高、求解效果好。由于本文所提出的问题中不仅考虑新增客户,同时还需考虑电动车与燃油车的不同配送属性以及电动车的充电行为,所以模型求解复杂度高。考虑到 ALNS 中独特的删除插入算子便于进行新顾客的插入以及路径中充电站的安排,本文设计一种改进的 ALNS 算法——IALNS,改进策略包括根据问题特点设计新的删除修复算子、采取不同的算子选择策略以及设计动态阶段加速策略。

3.2.1　路径生成

在每个阶段初始路径生成时采取以下策略:首先对当前阶段所有待配送的客户点按其右时间窗进行升序排序;然后在车辆(燃油车或电动车)满足客户载重约束的条件下,采用单路径距离贪婪插入的方式将客户点逐一插入形成初始解。若决策处在初始阶段,客户为所有静态客户;若决策处在动态优化阶段,则客户为当前阶段新增客户与上一阶段未完成配送的客户点。在插入客户点后进行充电站的插入操作。

3.2.2　充电站插入

生成的电动车初始路径可能不满足续航里程约束,因此可以进行充电站插入操作改进该路径。找到违反续航里程约束的客户点,将离该客户点最近的充电站插入到该客户点左边或者右边,判断是否可以减少违反里程约束的客户点,可以则输出更改路径;否则保持原路径不变。

3.2.3　编码以及解码

首先生成给定最大车辆数的路径集合,每一路径对应一个车辆;未被使用的车辆对应空路径,表示为[0,0]。例如某问题中包含六台车、十个客户点和两个充电站。编号 0 代表配送中心,编号 1 ~ 10 代表客户点,编号 11 ~ 12 为充电站,编号 13 ~ 18 为车辆编码,其中,编号 13 ~ 15 为电动车,编号 16 ~ 18 为燃油车。

假设按照初始解生成规则产生一个解为[0,2,4,5,0] [0,1,7,11,11,3,9,0] [0,10,0] [0,8,6,12,0] [0,0] [0,0]。解码时按照车辆的编号顺序确定对应的路径,并且对解方案进行如下操作:删除空路径和只包含充电站的路径;如果路径中存在相邻的两个编号相同的充电站,则删除其中一个充电站。对于上述解进行解码得到的配送方案如下:a)路径 1(电动车 13),配送中心 0—客户 2—客户 4—客户 5—配送中心 0;b)路径 2(电动车 14),配送中心 0—客户 1—客户 7—充电站 11—客户 3—客户 9—配送中心 0;c)路径 3(电动车 15),配送中心 0—客户 10—配送中心 0;d)路径 4(燃油车 16),配送中心 0—客户 8—客户 6—配送中心 0。

3.2.4　算子

IALNS 算法包括删除客户间隔算子、删除算子、修复算子三大类。删除间隔算子用来确定删除客户数,删除、修复算子用于对解进行破坏修复操作进而得到邻域解。

1)删除客户间隔算子　该算子借鉴 Hiermann 等人[22]的自适应方式,在每个阶段中都从(0,0.1),(0.1,0.2),(0.2,0.3)三个间隔中选取一个间隔算子,选中的间隔乘以该阶段总客户点数即为该阶段要删除的客户点数 n。

2)删除算子　在产生一个解以后,本文采用如下九个算子进行客户点的删除。

a)删除所有违反时间窗约束的点。该算子是针对本文模型特点所提出的一个删除策略,将对解中违反时间窗约束的客户点进行遍历确认,不考虑自适应删除间隔客户点数量,将所有违反时间窗的客户点全部进行删除。后面的实验证明该策略能大大提高整个算法的求解效率。

b)删除所有违反里程约束的点。该算子也是针对本文模型特点所提出的另一个删除策略。该算子与第一个算子相似,同样不考虑自适应间隔删除客户点数量,将对当前解中所有违反里程约束的点进行遍历确认并全部删除。如果某条路径中违反里程约束的点为终点,则将该条路径删除。该策略也能够有效提高算法求解效率。

c)删除所有违反车辆载重约束的点。该算子与前两个算子相似,对当前解中所有违反车辆载重约束的点进行遍历确认并全部删除。如果某条路径中违反车辆载重约束的点为终点,则将该条路径删除。该策略能够有效提高算法求解效率。

d)随机移除客户点。基于文献[21],首先依据删除客户间隔算子确定删除点数 ,然后随机选择解中的一条路径,再随机选择该路径中的一个客户点进行移除,重复该操作直到删除 n 个点。

e)随机移除客户点和充电站。首先依据删除客户间隔算子确定删除点数 ,然后随机选择解中的一条路径,再随机选择该路径中的一个客户或者充电站进行移除,重复该操作直到删除 n 个点。

f)距离成本客户点移除。基于文献[23]选择降低距离成本的客户点,即将该客户点从当前路径中移除后能最大程度降低方案中的行驶成本。首先依据删除客户间隔算子确定删除点数 n,然后对于每个客户点 i,计算 $d_{i-1,i}+d_{i,i+1}-d_{i-1,i+1}$ 的值,将其进行降序排序,选择前 n 个客户点进行移除。

g)距离充电站最小节点移除。从解方案的电动车路径中

随机选取一个充电站,然后选取离充电站最近的点依次进行删除,直到删除包含该充电站在内的 n 个点。

h)相似度移除(只删除客户点)。相似度移除又称为 Shaw 移除算子,由 Shaw[24] 于 1997 年提出,先随机选取一个客户点进行移除,并对现有方案中其他所有客户点计算:

$$R(i,j)=\beta_1\times\frac{d_{ij}}{\max\limits_{i,j\in V}(d_{ij})}+\beta_2\times\frac{|e_i-e_j|}{\max\limits_{i,j\in V}(|e_i-e_j|)}+\beta_3\times\frac{|q_i-q_j|}{\max\limits_{i,j\in V}(|q_i-q_j|)} \tag{40}$$

其中:d_{ij} 为客户 i 与 j 之间的距离,$|q_i-q_j|$ 为客户点之间的需求差异,$|e_i-e_j|$ 为客户点最早开始服务的时间差异。选取相似度最大的客户点进行移除,即删除 $R(i,j)$ 最小的客户点,重复操作直到删除 n 个客户点。其中 β_1、β_2、β_3 表示各影响因素权重,本文中分别取值 6、5、4。

i)随机路径移除。随机选择当前解方案中的一条包含点数小于等于删除点 n 的路径进行移除,生成新的解方案。

3)修复算子 基于文献[21]提出如下修复算子。本文中解的成本包含里程成本和惩罚成本两部分,当解不满足时间窗、载重、里程约束时,则在目标函数上对应添加一个大的惩罚成本,三种惩罚成本相同。

a)考虑全部成本贪婪插入。该算子将被移除的客户点插入所有路径中最好的一个位置,即插入以后带来的总成本增加最小。对每个客户点 i 计算插入到路径 s 的第 j 个客户点和第 $j+1$ 个客户点中间的成本增量 $\Delta c_{sj}=c_{sj}-c_{s0}$($c_{sj}$代表将客户点 i 插入到路径 s 的第 j 个客户点后的总成本,c_{s0}代表未插入客户点 i 的总成本,Δc_{sj}代表成本增量);对所有路径所有位置进行判断,找到最佳插入位置。重复该步骤直到所有删除的客户点被插入,形成新的解方案。

b)只考虑里程约束的贪婪插入。该算子与第一个算子类似,不同之处在于客户点插入时考虑的是里程的增加而不是总成本的增加,即对每个客户点 i 计算插入到路径 s 的第 j 个客户点和第 $j+1$ 个客户点中间的距离增量 $d_{sj}=d_{ji}+d_{i(j+1)}-d_{j(j+1)}$;对所有路径所有位置进行判断,找到最佳插入位置 $l=\text{argmin}_{s\in S,j\in J}(d_{sj})$,即把客户点插入到该点之后。重复该步骤直到所有删除集的客户点被插入,形成新的解方案。

c)考虑全部成本的带扰动的贪婪插入。该算子是对第一个算子的扩展,对于最优插入位置的成本增量给予一个偏离度,即新成本增量 = 原成本增量乘以随机数,随机数在[0,0.2]内随机产生;然后选取新成本增量最小的位置插入客户点。重复该步骤直到所有删除集的客户点被插入,形成新的解方案。

考虑到电动车的特殊性,每次在插入完删除集中的客户点后都要进行一次充电站的插入操作。

3.2.5 算子选择策略

对于前三个删除算子,会根据解中三种约束的违约次数选择违约次数最多所对应的算子对当前解进行删除操作,如当前解中违反时间窗约束的次数最多,则选择第一个删除算子,即删除所有违反时间窗约束的点。对于其余算子,则采取自适应的选择策略。算子选择取决于其权重,所有算子初始权重为 1,每连续迭代 ψ 次更新一次权重,更新公式[21]为

$$\omega_d=\begin{cases}w\times a_d/b_d+(1-w)\times\omega_d & b_d\neq 0\\ \omega_d & b_d=0\end{cases} \tag{41}$$

其中:ω_d 代表某个算子的权重,a_d 为算子的得分,b_d 为 ψ 次迭代中该算子的使用次数,w 为一个权重系数。迭代开始时,每个算子的得分为 0,在迭代过程中根据算子的选择情况会对得分进行更新。

3.2.6 动态阶段加速策略

在更新时刻,上阶段未完成服务的老客户和新客户将作为下阶段的输入。对于老客户,首先判断从配送中心派出一辆新车到达该客户的时间是否满足其时间窗约束,如果满足,则该客户所在原有路径上所有未完成服务的客户点以及新客户点作为下一个阶段的客户输入,重新生成配送方案;如果不满足,则在更新阶段的初始解中该客户保留在原有车辆路径中,然后对原配送路径中该客户的下一个客户点进行相同的判断,此过程一直持续直到存在客户点能够由新车服务为止。该策略能够加快动态阶段的求解效率。例如一条路径为[0,3,5,6,7,8,0],在更新时刻该车辆正在前往客户点 6 的途中,首先判断派出新车是否可以服务 6,如果可以,则客户点[6,7,8]作为下一阶段的客户输入,重新进行车辆配送安排;否则,客户点 6 在下一阶段初始解仍然保留在原车辆路线中,然后继续对客户 7 进行相同的判断,直到满足条件为止。

3.2.7 IALNS 算法流程

IALNS 算法中,T 为模拟退火的温度,n 为最大未改进迭代次数,m 为迭代次数,算法终止条件为达到最大迭代次数或达到最大未改进迭代次数,算法具体步骤如下:

a)按照初始解生成规则产生初始解,令当前解 = 当前最优解 = 初始解,并且初始化所有算子的权重和得分,转步骤 b)。

b)对当前解进行判断,如果当前解为不可行解,转步骤 c);否则转步骤 d)。

c)按照算子选择策略从前三个删除算子中选择一个对当前解进行删除操作,然后转步骤 e)。

d)从后六个删除算子中自适应选择一个算子对当前可行解进行破坏操作,转步骤 e)。

e)自适应选择一个修复算子进行插入操作,产生新解,转步骤 f)。

f)如果新解优于最优解则令最优解 = 当前解 = 新解,对所选择的删除和修复算子增加相对应的分数 δ_A,最大未改进迭代次数增加一次,转步骤 i),否则转步骤 g)。

g)如果新解优于当前解,令当前解 = 新解,对所选择的破坏和修复算子增加相对应的分数 δ_B,转步骤 i);否则转步骤 h)。

h)使用模拟退火的策略以相应的概率接受新解,如果接受,令当前解 = 新解,对所选择到的删除和修复算子增加相对应的分数 δ_C,转步骤 i)。

i)判断是否满足更新算子权重的条件,满足则依据算子得分更新权重,将算子得分重置为 0,否则直接转步骤 j)。

j)判断是否满足算法的终止条件,如果满足则输出最优解,否则转步骤 b)。

4 算例分析

本文的数据均取自于文献[6]的研究,该数据是对文献[25]标准案例修改而来的。为了保证实验可信度,本算例同时考虑了客户点的位置服从聚类分布(C)、随机分布(R)和两者混合分布(RC)的情形。参考文献[11]选取如下电动车的参数:电池容量为 60 kW·h,电动车充电速度为(5/3) kW·h/min,单位里程耗电量 0.3 kW·h/km,电动车平均车速为 40 km/h,电动车载重量与燃油车载重量为 10 t,里程费率根据平均电价和里程耗电率计算取 0.18 元/km,燃油车按现有油价取百公里油耗 6 L,平均油价为 7 元/L,所以燃油车里程费率为 0.42 元/km。算法中算子分数取 $\sigma_A=20$,$\sigma_B=12$,$\sigma_C=8$;权重调节反应因子 $w=0.8$;权重更新周期 $\psi=10$,初始温度取 10,温度衰减值取 0.95。算法性能测试中所有算例均为大规模,即 100 个客户点与 20 个公共充电站。

本文实验采用 Python 在配置为 Intel®-Core™ i7-8565U CPU@1.8 GHz 的计算机上求解。

4.1 基础算例求解

本节在文献[6]算例中的 C101 数据集选取 50 个客户和

10 个充电站,车辆数为 15,其中电动车为 6 辆,燃油车为 9 辆,静态客户 30 个,其余 20 个为动态客户,初始各节点的分布如图 3 所示。

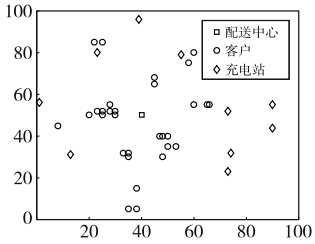

图3　各节点地理位置分布

Fig.3　Geographic location distribution of each node

4.1.1　初始阶段优化

使用 IALNS 算法为初始静态客户规划配送路线,结果如表 3 所示。

表3　初始静态客户路线

Tab.3　Initial static customer route

车辆	路线	车辆	路线
电动车 1	0—7—19—30—0	燃油车 7	0—3—10—21—0
电动车 2	0—13—25—0	燃油车 8	0—6—12—0
电动车 3	0—9—17—28—0	燃油车 9	0—2—24—0
电动车 4	0—4—15—22—0	燃油车 10	0—23—0
电动车 5	0—5—14—20—29—0	燃油车 11	0—8—16—26—0
电动车 6	0—1—11—18—27—0		

4.1.2　动态阶段优化

进入动态阶段,开始出现新的需求,依据 IALNS 算法及动态阶段加速策略对原有路线进行调整或者设计新的配送路线。本文设置更新时间间隔为 120 min。

假设配送中心全天服务,从 0 时开始服务,第一次更新时刻为 2:00,此时没有新的需求产生,车辆仍然按照初始路线进行配送;第二次更新时刻为 4:00,在 2:00 ~ 4:00 配送中心收到 13 个新的客户需求,客户点编号为 31 ~ 43,此时将新的需求与上阶段未服务客户同时作为该阶段输入,进行优化得到新的解方案,如表 4 所示。表中(a,b)表示更新时刻车辆正行驶在客户点 a 与 b 之间,下画线表示加入的新需求。

表4　第二次更新规划路线

Tab.4　The second updated route

车辆	路线	车辆	路线
电动车 1	(19,30)—30—38—0	燃油车 7	(21,0)—33—42—0
电动车 2	(25,0)—0	燃油车 8	(12,0)—34—43—0
电动车 3	(17,28)—28—37—0	燃油车 9	(24,0)—32—41—0
电动车 4	(22,0)—0	燃油车 10	(23,0)—31—40—0
电动车 5	(20,29)—29—39—0	燃油车 11	(26,0)—35—0
电动车 6	(18,27)—27—36—0		

第三次更新时刻为 6:00,在 4:00 ~ 6:00 接收到 7 个新增需求,客户编号为 44 ~ 50,得到的路线如表 5 所示。此时电动车 2 和 4 已经完成配送服务回到配送中心,并且新派出一辆电动车 21 进行服务。

最终,所有车辆的配送路线如图 4 所示。由图 4 可见,当出现新需求时,车辆会继续服务新的客户而不会返回配送中心,以提高车辆的使用率,同时新的配送路线尽量保持原来顾客的配送路线,避免降低顾客的服务水平。

4.2　算法性能

为了验证 IALNS 算法的有效性,本文进行了算法对比。由于本文所提出的是带时间窗的混合车辆的动态需求车辆路径问题,就目前国内外文献的检索情况而言,还没有该类问题的研究,所以本文选取了与本研究最为接近的动态需求车辆路径问题的求解算法进行对比,其中代表性的算法有大规模邻域搜索算法(LNS)[14]、变邻域搜索算法(VNS)[15]、自适应大规模邻域搜索算法(ALNS)[16]。此外,为了验证本文策略的有效性,与无改进策略的 IALNS(即去掉 3.2.4 节中前三个删除算子,记为 IALNS-ND)进行了对比。由于本文考虑了电动车与燃油车混合配送场景,所以需将三个对比算法(LNS, VNS, ALNS)进行一定的改动才能应用到本问题。对于 LNS 算法,在文中的 maximum-best first insertion algorithm 中增加了插入充电站的算子;此外,由于该算法仅考虑了燃油车,所以在最坏距离删除算子中,增加删除充电站的考虑。对于 VNS 算法,由于仅考虑燃油车,而且使用所罗门算法产生初始解,同时要求迭代过程中产生的新解必须是可行解,然而本文使用的电动车会涉及到充电站的选择,无法采用所罗门算法产生初始解,所以采用本文 3.2.1 节所提出的策略生成解,然后判断解中分别违反载重、里程、时间窗的客户点的个数,选择违反约束最多的客户点将其全部删除,同时采用 maximum-best first insertion algorithm 插入,直至迭代到可行解。对于 ALNS 算法中,由于该算法也只考虑了燃油车,所以在该算法中增加了插入充电站的算子。本文分别针对大规模的静态问题和动态问题进行算法性能验证。

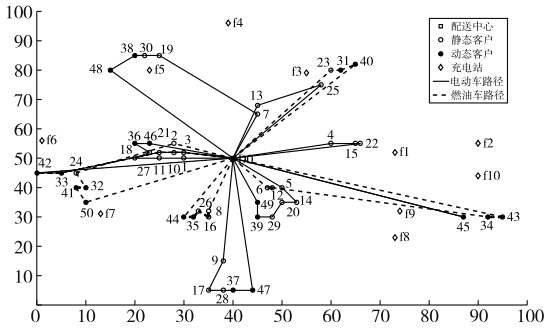

图4　最终客户路线

Fig.4　Final roadmap for all customers

表5　第三次更新规划路线

Tab.5　The third updated route

车辆	路线	车辆	路线
电动车 1	(38,0)—48—0	燃油车 7	(33,42)—42—0
电动车 3	(37,0)—47—0	燃油车 8	(43,0)—0
电动车 5	(29,39)—49—0	燃油车 9	(32,41)—41—50—0
电动车 6	(27,36)—46—0	燃油车 10	(31,40)—40—0
电动车 21	0—45—0	燃油车 11	(35,0)—44—0

4.2.1　静态问题

首先检验算法对于求解静态 EVRPTWMF 的性能,即所有客户均为静态客户。分别从三种客户位置分布中随机选择 4 个客户规模为 100 的数据集进行测试,为了公平对比,将求解时间固定为 200 s,分别应用算法求解每个算例的总成本,每个算例计算 10 次,共进行 600 次实验。结果如表 6 所示。在表中,与 IALNS-ND 相比,加下画线则表示结果更好,12 个算例中 9 个算例 IALNS 的最小值和平均值优于 IALNS-ND,10 个算例 IALNS 的最大值优于 IALNS-ND;就方差来说,两者相差不大,可以看出,在算法稳定性相近的情况下 IALNS 可以获得更优的解,证明本文所提出的加速策略在静态问题中可以提升算法的效果。与 VNS、LNS、ALNS 相比,对结果最好的值进行了加粗,所有算例中,IALNS 的最小值与平均值均优于这三个算法;从最大值来看,12 个算例中有 9 个算例 IALNS 的最大值优于 VNS 和 LNS,12 个算例全部优于 ALNS;就方差而言,尽管 LNS 最好,但在 12 个算例中有 7 个算例 IALNS 的最大值均优于 LNS 算法的最小值,其余 5 个算例两者结果都比较接近,在一定程度也体现了 IALNS 解的质量的稳定性。

表 6 静态问题算法对比结果

Tab. 6 Algorithm comparison results of static problem

算例		R 类				RC 类				C 类			
		R101	R102	R103	R104	RC101	RC102	RC103	RC104	C101	C102	C103	C104
最小值	IALNS	466.1	313.8	282.52	253.54	448.86	397.04	390.17	376.61	204.5	233.37	229.59	238.77
	ALNS-ND	421.64	354.6	316.05	280.54	508.32	521.55	438.03	389.77	200.45	234.77	267.17	251.76
	LNS	557.81	400.34	377.04	278.96	702.45	620.87	541.78	397.16	205.86	280.73	294.86	281.83
	VNS	551.48	399.93	375.04	277.52	701.98	631.29	574.54	397.21	205.01	285.84	300.74	294.12
	ALNS	539.47	483.94	319.2	319.16	574.1	591.25	475.86	430.03	367.4	401.19	293.12	268.03
最大值	IALNS	568.48	386.97	349.33	283.33	649.34	499.61	489.22	397.25	205.86	282.12	295.45	295.57
	IALNS-ND	558.02	405.96	372.83	283.49	705.9	631.29	586.07	397.25	205.86	288.53	302.44	295.4
	LNS	558.02	405.96	380.06	280.54	705.9	631.29	586.07	397.25	205.86	288.53	302.44	295.4
	VNS	558.02	405.96	380.06	280.54	708.93	631.29	586.07	397.25	205.86	288.53	302.44	295.4
	ALNS	870.15	784.03	619.46	397.81	886.89	783.27	734.59	529.73	942.19	832.8	446.88	369.2
平均值	IALNS	510.52	355.3	321.32	277.25	549.98	457.01	430.16	393.61	205.72	254.97	265.51	274.45
	IALNS-ND	471.1	385.92	340.86	280.84	587.26	580.47	553.95	396.5	205.24	258.32	286.68	269.2
	LNS	557.97	403.99	379.46	280.23	704.53	629.03	566.86	397.22	205.86	287.48	301.53	292.87
	VNS	556.78	402.9	379.56	280.24	704.9	631.29	584.82	397.23	205.77	288.26	302.21	295.23
	ALNS	723.02	640.94	518.63	383.4	701.53	669.27	615.17	497.65	616.32	563.43	375.45	341.07
方差	IALNS	32.14	21.95	20.6	8.64	67.51	35.1	26.34	6.67	0.43	15.26	21.04	20.24
	IALNS-ND	39.12	17.51	18.08	0.93	65.67	40.94	49.46	2.36	1.7	21.1	15.4	14.29
	LNS	0.09	1.85	1.27	0.67	1.04	3.77	17.46	0.03	0	2.52	2.35	4.31
	VNS	2.14	2.41	1.59	0.96	1.93	0	3.62	0.02	0.27	0.85	0.53	0.4
	ALNS	100.77	103.95	107	28.63	83.96	76.1	78.45	31.97	159.16	139.66	47.76	37.29

4.2.2 动态问题

为了验证不同的新增客户规模下三种算法的性能和测试本文提出的删除算子策略对算法的影响,本文针对来自不同分布的 12 个数据集的不同动态度的算例进行了测试。动态度 $Dod = (n_d/n) \times 100\%$[14],其中 n_d 表示动态客户的数量,n 表示总的客户数。本测试选取 90%、70%、50%、30% 及 10% 五种不同的动态度,每个算例测试 10 次,共进行 3 000 次实验,求解结果如表 7 ~ 9 所示。其中,算法的终止由迭代次数确定,对于 VNS,采用文献[14]中迭代次数,设置最大迭代次数为 20,每次邻域操作次数为 10,由于 LNS 原文中未给予参考迭代次数,与本文 IALNS 以及 IALNS-ND 设置相同迭代次数,设置最大迭代次数为 100,最大未改进迭代次数为 10。ALNS 中设置算法终止条件最大未改进迭代次数为 5。

表 7 R 类数据算法对比结果

Tab. 7 Algorithm comparison results of class R data

算例	*Dod*	IALNS		IALNS-ND		获得可行解次数	LNS		VNS		ALNS	
		成本	时间	成本	时间		成本	时间	成本	时间	成本	时间
R101	90	373.6	20.27	370.5	17.99	10	376.62	235.38	391.97	202.25	370.73	44.38
	70	352.06	20.12	362.09	52.8	10	363.05	244.78	664.54	379.06	191.71	369.93	68.74
	50	331.83	100.06	362.87	111.26	10	367.61	515.81		378.28	218.98	348.94	292.92
	30	324.06	288.17	328.75	467.53	10	462.15	560.64		456.53	222.01	364.52	675.6
	10	324.31	401.58	382.39	450.62	7	506.11			535.84	248.82	364.06	1927.1
R102	90	380.62	30.88	380.56	27.13	10	378.94	113.99		420.83	187.84	394.54	56.36
	70	342.06	88.14	381.58	92.66	10	378.09	145.32		396.32	212.06	374.03	130.75
	50	308.43	119.41	315.84	211.69	10	415.78	370.41		490.3	227.39	347.92	202.62
	30	358.42	318.9	366.59	388.15	8	457.27	507.56		487.17	257.78	339.07	410.15
	10	303.45	384.63	321.82	464.01	8	399.77	920.46		419.66	266.13	324.57	964.86
R103	90	379.01	31.23	428.95	32.16	10	399.72	267.84		445.21	215.75	429.14	162.7
	70	346.65	69.13	368.29	73.35	10	383.78	236.41		420.7	210.4	387.84	225.51
	50	289.71	198.5	291.98	215.3	10	386.39	311.86		460.33	247.7	333.36	322.74
	30	230.12	198.75	238.97	265.48	9	322.43	311		359.08	340.35	251.71	548.55
	10	229.1	705.82	244.73	737.13	8	341.54	1206.9		391.55	452.9	280.41	1217.6
R104	90	323.97	26.11	381.02	56.02	10	388.92	127.09		393.65	587.89	434.19	472.34
	70	300.09	60.29	305.96	87	10	343.59	202.07		382.72	244.74	328.92	299.24
	50	218.55	194.38	249.5	209.94	10	273.79	315.63		294.32	333.12	252.56	269.58
	30	198.97	294.68	208.05	354.68	10	255.07	664.51		253	413.68	216.23	671.56
	10	258.34	100.46	261.83	274.93	10	268.48	562.95		271.23	544.1	244.2	908.42

在表 7 ~ 9 中,“ – ”表示无可行解。首先对比 IALNS 与 IALNS-ND 算法的性能,以验证所提出的改进策略(即 3.2.4 节中的前三个删除算子)的有效性,两者相比更好的结果使用下画线标出。在 IALNS-ND 中,由于去掉了三个删除算子,在算法求解过程中会出现无法得到可行解的情况,所以表中给出了该算法在 10 次测试中获得可行解的次数。从表 7 可以看出,R 类数据中,有 25%(20 个算例中有 5 个)的数据测试中会存在无法获得可行解的情况,10 次均得到可行解的 75%(20 个算例中有 15 个)的数据中,86%(15 个算例中有 13 个)的数据 IALNS 可以在更短的时间内得到成本优于 IALNS-ND 的解,另外两个数据 IALNS 的结果略差于 IALNS-ND。从表 8 可以看出,对于 RC 类数据,IALNS-ND 算法中两个数据出现无法 10 次全部得到可行解的情况;此外在所有算例中,IALNS 均可以在更短的时间内得到成本优于 IALNS-ND 的解。从表 9 中可知,

在 C 类数据中,只有 30%(20 个算例中有 6 个)的数据 IALNS-ND 可以 10 次全部得到可行解,在有可行解的结果中,约 93%(14 个算例中有 13 个)的数据 IALNS 可以在更短的时间内得到成本优于 IALNS-ND 的解,只有 C101 中动态度为 30% 的数据两者成本相同,IALNS 的时间略差于 IALNS-ND。综上可知,本文提出的加速策略能够有效提升求解效果和效率。

表 8　RC 类数据算法对比结果

Tab. 8　Algorithm comparison results of class RC data

算例	*Dod*	IALNS		IALNS-ND		获得可行解次数	LNS		VNS		ALNS	
		成本	时间	成本	时间		成本	时间	成本	时间	成本	时间
RC101	90	417.85	25.41	418.37	38.08	10	454.93	116.95	478.42	198.67	442.38	38.9
	70	410.49	30.2	420.4	59.65	10	418.68	214.53	466.81	217.4	**392.05**	63.18
	50	351.69	165.22	367.53	205.83	10	447.93	238.51	522.72	249.06	352.26	245.07
	30	338.69	425.27	362.32	492.26	10	563.95	**265.21**	618.07	288.64	400.02	**383.57**
	10	370.19	711.24	378.71	723.24	10	634.58	**413.67**	663.88	**322.24**	445.36	1057.7
RC102	90	462.74	26.95	488.93	30.19	10	**452.55**	225	503.4	202.87	**426.88**	63.3
	70	378.11	63.95	433.09	72.96	5	468.67	290.5	557.42	199.34	413.37	101.43
	50	329.93	161.14	–	–	0	488.32	489.89	561.75	239.26	381.73	189.56
	30	317.42	299.02	376.37	330.33	10	554.67	352.51	632.91	**272.32**	378.46	419.28
	10	347.03	473.86	395.75	760.13	10	608.44	538.78	635.54	**270.79**	405.81	801.81
RC103	90	366.39	69.18	447.82	71.91	10	411.9	158.42	523.66	222.27	454.61	82.11
	70	346.22	90.81	383.07	129.08	10	393.41	224.41	434.81	252.68	414.87	142.38
	50	287.26	156.18	298.02	232.19	10	319.12	224.09	447.28	277.06	304.47	203.53
	30	282.55	330.04	296.69	353.77	10	409.58	432.61	501.54	613.19	336.9	894.35
	10	316.07	601.97	342.72	806.27	10	530.31	**502.01**	545.73	**339.49**	380.5	1648.3
RC104	90	340.89	100.54	349.98	102.87	10	406.51	136.08	460.08	247.2	408.02	156.27
	70	286.88	79.77	325.63	82.21	10	314.75	215.17	402.69	269.26	323.11	199.34
	50	259.46	233.37	298.31	239.85	10	291.66	418.26	377.07	332.97	278.88	316.9
	30	301.74	323.67	306.73	330.29	10	343.28	342.89	363.19	417.06	**266.95**	**236.69**
	10	340.4	355.56	343.81	391.18	10	392.91	456.25	393.45	416.55	348.64	1029.9

表 9　C 类数据算法对比结果

Tab. 9　Algorithm comparison results of class C data

算例	*Dod*	IALNS		IALNS-ND		获得可行解次数	LNS		VNS		ALNS	
		成本	时间	成本	时间		成本	时间	成本	时间	成本	时间
C101	90	264.65	16.51	–	–	0	258.67	85.54	262.26	147.94	**256.42**	30.72
	70	250.08	21.17	252.96	22.84	10	**247.09**	107.72	256.52	199.32	285.9	36.02
	50	241.08	27.56	244.94	37.48	10	240.89	199.14	**240.22**	224.61	281.28	91.4
	30	225.08	51.37	225.08	48.19	10	**221.27**	358.89	225.08	244.5	305.84	233.12
	10	209.77	217.98	331.84	218.29	10	211.68	422.75	212.12	255.02	259.04	627.52
C102	90	486.19	23.2	–	–	0	**471.71**	87.84	513.46	276.04	487.08	51.78
	70	468.65	72.44	–	–	0	480.18	136.93	498.87	221.9	514.65	74.75
	50	409.34	127.75	–	–	0	465.37	202.64	491.81	263.44	476.73	143.28
	30	383.51	246.91	–	–	0	434.26	277.94	445.69	291.88	438.79	331.97
	10	227.65	480.5	258.75	849.46	10	294.1	486.54	299.53	**390.82**	276.13	609.53
C103	90	756.24	51.99	784.33	61.89	3	781.2	110.89	777.05	217.63	**738.09**	170.39
	70	736.6	70.31	747.72589.28	102.29	3	743.59	211.19	783.56	234.53	747.36	134.68
	50	573.95	143.06	315.17	290.23	8	630.68	222.84	683.49	270.93	622.5	249.97
	30	286.01	328.98	233.26	617.44	9	361.45	469.29	378.25	366.27	331.51	396.91
	10	215.89	499.19		1520.7	9	313.81	708.42	323.42	560.21	282.33	964.37
C104	90	1134	55.98	–	–	0	1158.1	237.77	1153.9	224.46	**1067.9**	282.87
	70	884.81	45.34	862.06	189.01	10	907.48	383.85	927.28	286.12	**863.64**	195.73
	50	399.71	195.96	438.5	225.35	7	441.99	308.73	497.62	390.71	412.76	216.86
	30	267.9	331.23	293.9	466.46	5	339	605.34	360.91	600.14	278.59	681.58
	10	210.45	469.39	216.69	828.99	3	293.73	813.41	316.13	715.82	256.4	878.01

针对不同算法(IALNS、LNS、VNS、ALNS)的测试,从表 7 可知在 R 类数据中,与 LNS 相比,有 95%(20 个算例中有 19 个)的算例 IALNS 的求解质量和求解时间均优于 LNS,另外一个算例虽然 LNS 获得成本优于 IALNS 0.4%(R102 动态度为 90% 的算例)的解,但求解时间却是 IALNS 的三倍多;与 VNS 相比,有 75%(20 个算例中有 15 个)算例 IALNS 的求解质量和求解时间均优于 VNS;与 ALNS 相比,有 85%(20 个算例中有 17 个)的算例 IALNS 的求解质量和求解时间均优于 ALNS,其余三个算例中最差成本高于 ALNS 0.58%(R104 的动态度为 10% 的算例),但是求解时间为 ALNS 的九倍多。对于 RC 类数据来说,从表 8 中可看出,有 80%(20 个算例中有 16 个)的算例 IALNS 的求解质量和求解时间均优于 LNS、VNS 和 ALNS,只有 1 个数据(RC104 中动态度为 30% 的算例)ALNS 可以以更短的时间得到质量优于 IALNS 的解;从表 9 的 C 类数据对比来看,C 类数据集中有 60%(20 个算例中有 12 个)的算例的解成本 IALNS 优于三个对比算法,其余 25% 的算例 IALNS 所求解与最优解最多相差 5.8%(C104 中动态度为 90% 的算例),具体来看,分别有 85%(20 个算例中有 17 个)、95%(20 个算例中有 19 个)、80%(20 个算例中有 16 个)的成本和时间优于 LNS、VNS、ALNS;从求解时间来看,对于 C101 的算例,IALNS 算法能够以少于对比算法甚至九倍(RC101 中动态度为 70% 算例,与 VNS 相比)的时间找到质量相近的解,

在 C102 到 C104 算例中,IALNS 在求得质量好的解时使用时间可最多优于其他算法三倍多(RC103 的动态度为 70% 的算例)。此外,从表中可以看出,动态度越高,IALNS 求解时间越快,求解性能越好,能够快速响应客户需求,提高服务客户的效率。因此,IALNS 适合求解动态需求的车辆路径问题。

5 结束语

本文考虑了电动车和燃油车混合配送的实际场景,研究了带时间窗的动态需求车辆路径问题,构建了初始和动态两个阶段 0-1 整数规划模型。针对所建模型的特点,设计了改进的自适应大规模邻域搜索算法进行求解,其中包括设计新的删除修复算子、设计动态阶段加速策略等。为了验证算法性能,本文分别对大规模静态需求车辆路径问题和大规模动态需求车辆路径问题进行求解。结果表明,对于静态问题,在相同的求解时间内,与 IALNS-ND 相比,12 个算例中 9 个算例 IALNS 的最小值和平均值优于 IALNS-ND,10 个算例 IALNS 的最大值优于 IALNS-ND;与 VNS、LNS 和 ALNS 相比,所有算例 IALNS 所获得的总成本的最小值与平均值均优于 LNS、VNS 和 ALNS。对于动态问题,与 IALNS-ND 相比较,所有数据 IALNS 均可获得可行解,且 95%(60 个算例中 57 个算例)的算例可以得到成本优于 IALNS-ND 的解;与 LNS、VNS 和 ALNS 相比,58%(60 个算例中 35 个算例)的算例 IALNS 能够以少于 1.5 倍甚至 10 倍的时间获得更优的解;同时求解问题的动态度越高,IALNS 的算法性能和效率越好,表明 IALNS 在求解时效性要求较高的动态车辆路径问题方面具有较为明显的优势,能够为企业快速获得满意的动态配送路径方案。目前本文仅考虑了车辆配送总成本这一目标函数,下一步可以丰富问题的求解目标,如减少碳排放成本、减少车辆使用数等。同时可以细化电动车充电行为特征,使得问题更符合企业实践。

参考文献:

[1] Vidal T, Laporte G, Matl P. A concise guide to existing and emerging vehicle routing problem variants[J]. European Journal of Operational Research,2020,286(2):401-416.

[2] Rabbouch B, Saadaoui F, Mraihi R. Empirical-type simulated annealing for solving the capacitated vehicle routing problem[J]. Journal of Experimental & Theoretical Artificial Intelligence, 2020, 32(3):437-452.

[3] Marinakis Y, Marinaki M, Migdalas A. A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows [J]. Information Sciences,2019,481(5):311-329.

[4] Lai D S W, Demirag O C, Leung J M Y. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph [J]. Transportation Research Part E: Logistics and Transportation Review,2016,86(2):32-52.

[5] Conrad R G, Figliozzi M A. The recharging vehicle routing problem [C]//Proc of Industrial Engineering Research Conference. 2011: 1-8.

[6] Schneider M, Stenger A, Goeke D. The electric vehicle-routing problem with time windows and recharging stations[J]. Transportation Science,2014,48(4):500-520.

[7] 葛显龙,竹自强.带软时间窗的电动车辆路径优化问题[J].工业工程与管理,2019,24(4):96-104,112.(Ge Xianlong, Zhu Ziqiang. Electric vehicles routing problem with soft time window[J]. Industrial Engineering and Management,2019,24(4):96-104,112.)

[8] Keskin M, Çatay B. Partial recharge strategies for the electric vehicle routing problem with time windows[J]. Transportation Research Part C: Emerging Technologies,2016,65(4):111-127.

[9] Pelletier S, Jabali O, Laporte G. The electric vehicle routing problem with energy consumption uncertainty[J]. Transportation Research Part B: Methodological,2019,126(8):225-255.

[10] Alesiani F, Maslekar N. Optimization of charging stops for fleet of electric vehicles: a genetic approach[J]. IEEE Intelligent Transportation Systems Magazine,2014,6(3):10-21.

[11] Yang Hongming, Yang Songping, Xu Yan, *et al*. Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm [J]. IEEE Trans on Smart Grid,2015,6(2):657-666.

[12] Goeke D, Schneider M. Routing a mixed fleet of electric and conventional vehicles[J]. European Journal of Operational Research, 2015,245(1):81-99.

[13] Hiermann G, Hartl R F, Puchinger J, *et al*. Routing a mix of conventional, plug-in hybrid, and electric vehicles[J]. European Journal of Operational Research,2019,272(1):235-248.

[14] Hong Lianxi. An improved LNS algorithm for real-time vehicle routing problem with time windows[J]. Computers & Operations Research,2012,39(2):151-163.

[15] De Armas J, Melián-Batista B. Variable neighborhood search for a dynamic rich vehicle routing problem with time windows[J]. Computers & Industrial Engineering,2015,85(7):120-131.

[16] Chen Shifeng, Chen Rong, Wang Gaige, *et al*. An adaptive large neighborhood search heuristic for dynamic vehicle routing problems [J]. Computers & Electrical Engineering,2018,67(4):596-607.

[17] 张文博,苏秦,程光路.基于动态需求的带时间窗的车辆路问题[J].工业工程与管理,2016,21(6):68-74.(Zhang Wenbo, Su Qin, Cheng Guanglu. Vehicle routing problem with time windows based on dynamic demands[J]. Industrial Engineering and Management,2016,21(6):68-74.)

[18] 李阳,范厚明,张晓楠.动态需求下车辆路径问题的周期性优化模型及求解[J/OL].中国管理科学,(2020-11-10).https://doi.org/10.16381/j.cnki.issn1003-207x.2019.1495.(Li Yang, Fan Houming, Zhang Xiaonan. A periodic optimization model and solution for capacitated vehicle routing problem with dynamic requests[J/OL]. Chinese Journal of Management Science,(2020-11-10). https://doi.org/10.16381/j.cnki.issn1003-207x.2019.1495.)

[19] 邵赛,毕军,关伟.基于电动汽车的动态需求车辆路径问题[J].吉林大学学报:工学版,2017,47(6):1688-1695.(Shao Sai, Bi Jun, Guan Wei. Electric vehicle routing problem with charging and dynamic customer demands[J]. Journal of Jilin University: Engineering and Technology Edition,2017,47(6):1688-1695.)

[20] Schyns M. An ant colony system for responsive dynamic vehicle routing[J]. European Journal of Operational Research,2015,245(3):704-718.

[21] Ropke S, Pisinger D. An Adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows[J]. Transportation Science,2006,40(4):455-472.

[22] Hiermann G, Puchinger J, Ropke S, *et al*. The electric fleet size and mix vehicle routing problem with time windows and recharging stations [J]. European Journal of Operational Research,2016,252(3):995-1018.

[23] Demir E, Bektaş T, Laporte G. An adaptive large neighborhood search heuristic for the pollution-routing problem [J]. European Journal of Operational Research,2012,223(2):346-359.

[24] Shaw P. A new local search algorithm providing high quality solutions to vehicle routing problems[R]. Glasgow, Scotland: University of Strathclyde,1997.

[25] Solomon M M. Algorithms for the vehicle routing and scheduling problems with time window constraints[J]. Operations Research, 1987,35(2):254-265.