Journal of Computer Applications
计算机应用,2017,37(8):2387 - 2394
ISSN 1001-9081
CODEN JYIIDU
2017-08-10
http://www.joca.cn

文章编号:1001-9081(2017)08-2387-08
doi:10.11772/j.issn.1001-9081.2017.08.2387

基于密度聚类构建物流配送问题的毁灭移除算法

阳　旺*,何国超,吴　雁
(中南大学 信息科学与工程学院,长沙 410083)
(*通信作者电子邮箱 yangwang@csu.edu.cn)

摘　要:研究多车型大规模物流配送问题,针对企业配送门店规模大且聚集的特点,在自适应大规模邻域搜索(ALNS)框架下提出一种新的邻域映射方式:基于密度聚类的毁灭移除算法。ALNS 包含毁灭与重建两个阶段,通过不断对当前解进行破坏和重建得到更好解。在毁灭阶段,随机选择一条路线进行密度聚类得到簇集合,然后按簇对路线上的门店进行移除;重建阶段随机选择贪婪插入法或 Regret-2 插入法将移除的门店插入到合适的路线上得到新配送方案。通过国际基准测试案例验证了所提算法的有效性,与已有算法对比,基于密度聚类的毁灭移除算法的 ALNS 算法求解结果比案例已知最优解平均误差更低,求解质量更优;应用于实际场景中,该算法能在有限时间内求得较好的配送方案。

关键词:新零售;车辆路径问题;固定车辆数的多车型车辆路径问题;毁灭与重建;密度聚类;自适应大规模邻域搜索
中图分类号:TP399;TP301.6　　**文献标志码**:A

Density clustering based removal heuristic for vehicle routing problem

YANG Wang*, HE Guochao, WU Yan
(*School of Information Science and Engineering, Central South University, Changsha Hunan* 410083, *China*)

Abstract: Focusing on large-scale vehicle routing problem with heterogeneous fleet, a new neighborhood mapping method, namely density clustering based removal heuristic algorithm, was proposed under the Adaptive Large Neighborhood Search (ALNS) frame work. ALNS includes two phases: destruction and reconstruction, which provides optimized solution by destroying and reconstructing current solution. In the destruction phase, a routine was randomly selected to get clusters by density clustering, and then the stores were removed from the routine according to the clusters. In reconstruction, Greedy or Regret-2 insert algorithm was randomly chosen to place those removed stores into proper routine. Test results on benchmark instances validate the effectiveness of the proposed method. Compared with other existing algorithms, the ALNS density clustering based removal heuristic algorithm has lower rate of error and better quality of solutions; in real situations, the proposed algorithm can provide optimized solution in limited time.

Key words: new retail; Vehicle Routing Problem (VRP); Heterogeneous Fixed Fleet Vehicle Routing Problem (HFFVRP); ruin and recreate; density clustering; adaptive large neighborhood search

0　引言

随着电子商务的发展,未来将进入“新零售”时代,线上线下和物流结合在一起,产生新的零售方式[1]。在“新零售”方式下,物品由企业统一配送到各个门店(便利店、超市等),客户的需求由离客户最近的门店进行服务,以实现“当日达”,甚至“小时达”,能极大地提高客户的物流体验。对企业而言,门店数量将增加,门店每天需求量将急速增加,如何使用一组车辆对大规模门店进行货物配送,如图 1 所示,是亟待解决的问题。

大规模门店物流配送本质上是解决一个车辆路径问题(Vehicle Routing Problem, VRP),最早由学者 Dantzig 等于 1959 年提出[2],指一组车辆从物流总仓出发对一系列在地理上无规律分布的客户进行服务,要求每个客户的需求必须被满足且只能由一辆车提供服务,每辆车的路线开始于物流总仓,并且最终结束于物流总仓。VRP 已被证明是 NP-hard 问题,使用动态规划法、分支界限法等精确算法只适用解决 100 个客户规模以内的 VRP[3],不适用于大规模物流配送问题。根据约束条件不同,如车辆最大容量约束、门店服务时间窗约束等,可将 VRP 分为不同类型的扩展问题[3]。

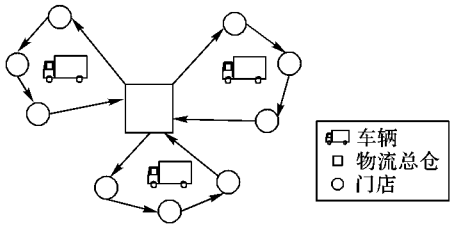

图 1　物流配送问题
Fig. 1　Logistics distribution problem

在本文研究的企业案例中,企业拥有 8 000 多家门店,配送车队拥有多达 37 种车型。目前企业仍处于人工调度派车阶段,线上和线下的订单分开处理,存在车辆空载率高、配送成本偏高、客户物流体验差等问题,制约了企业的进一步发

收稿日期:2017-02-21;**修回日期**:2017-04-18。　**基金项目**:国家科技支撑计划项目(2015BAH05F02)。
作者简介:阳旺(1982—),男,湖南湘潭人,副教授,博士,CCF 会员,主要研究方向:计算机算法;　何国超(1991—),男,广东湛江人,硕士研究生,主要研究方向:车辆路径问题;　吴雁(1992—),女,湖南涟源人,硕士研究生,主要研究方向:车辆路径问题。

展。企业急需进行配送调度自动化,配送算法要能在限定时间内计算可行的配送方案,同时充分利用企业自身的配送资源。因此,本文考虑以下 VRP 要素:多种车型,每种车型车辆数量有限,不同车型固定费用、可变费用和最大装载容量不同,每个门店的订货需求必须被满足且只能由一辆车服务。根据要素确定本文研究的问题是一个具有固定车辆数的多车型车辆路径问题(Heterogeneous Fixed Fleet Vehicle Routing Problem, HFFVRP)[3]。同时,本文提出的算法需在限定的时间内得到较好的解。

1 相关研究

大规模物流配送问题具有网络拓扑结构复杂、数据处理规模大等[4]特点,难以直接解决。现实中企业一般将门店按行政区域进行划分,从而将大规模 VRP 转化为多个小规模 VRP 进行解决。但按行政区域进行划分并不是最合理的划分方式,有学者提出改进的基于基地启发式算法(Location Based Algorithm)解决门店分区问题[4]。近年来,随着大数据处理技术的成熟,聚类分析被广泛应用到各个领域,不少学者将聚类分析应用于 VRP 中。Ewbank 等[5]使用无监督模糊聚类技术对客户点进行聚类分组,然后构建分组的旅行商问题(Travelling Salesman Problem, TSP)路线;Shin 等[6]提出一种基于几何中心(centroid-based)聚类的三阶段算法,先围绕 g 个种子客户点进行聚类,然后根据类的几何中心和车辆容量约束对客户进行聚类再分配,最后构建每个类内部的 TSP 路线。但是,上述算法需指定聚类个数,而在实际应用中,聚类个数往往难以确定。本文拟采用 DBSCAN(Density-Based Spatial Clustering of Applications with Noise)密度聚类算法,该算法根据用户输入的密度参数定义稀疏数据和稠密数据,只将稠密数据集作为聚类[7]。该算法可以克服其他基于距离的聚类只能发现类圆形类簇的缺点,并且无需指定聚类个数,适合用于对配送路线进行聚类分析。

基于邻域搜索的算法解决 CVRP(Capacitated VRP)取得了较好的效果,一方面在多个国际基准测试案例上求得已知最优解,另一方面可扩展多种约束条件解决复杂的现实问题。但是,随着问题规模的增大,已有算法容易陷入局部最优。邻域搜索求解质量取决于邻域映射的方式。传统的邻域映射方式主要分为路径间交换或路径内交换,如Shift(1,0)[8]、Swap(1,1)[8]、2-opt[8]等,算法每次迭代时对配送方案中 1 个或者 2 个节点进行操作,邻域搜索范围较小,当问题的解空间较大时,求解结果容易陷入局部最优。自适应大规模邻域搜索(Adaptive Large Neighborhood Search, ALNS)[9]扩展了传统邻域搜索的邻域概念,设计的邻域结构对多个节点进行操作,从而扩大了邻域搜索的范围,更有可能求得全局最优的解。目前,国内外学者基于自适应大规模邻域搜索解决 HFFVRP 的研究很少,并且在问题规模较大、车型种类较多、算法求解时间有限的情况下,现有自适应大规模邻域搜索算法不能有效解决实际问题。

目前,ALNS 中的毁灭移除算法分为 workday、route 和 customer[11]三个层次,workday 按工作日对配送方案进行移除,route 对配送方案中的配送路线进行移除,customer 对配送路线上的门店进行移除。本文研究和实现 customer 层的毁灭移除算法。Random Removal[9]是最简单的毁灭移除算法,可在解空间中产生任意的邻域集合;该算法易于实现但随机性较大,不能保证 ALNS 求解质量。Shaw Removal[9]定义了相关度函数,邻域映射方式由相关度函数决定;由于相关度函数计算复杂,导致该移除算法执行时间较长,容易造成 ALNS 算法整体执行时间过长的问题。Worst Removal[12]定义了一个门店从当前配送方案移除的开销函数,开销值越大,门店越需被移除,毁灭时移除开销值最大的 q 个门店;该算法移除效果好,但时间复杂度高。在门店规模大且聚集的应用场景下,对配送方案进行门店移除时,已有毁灭移除算法每次移除一个门店,门店可能来自不同的聚集域,这造成移除的节点插入原聚集域代价最低,在重建阶段该移除门店将被重新插入原聚集域。由于执行一轮毁灭和重建操作后,配送方案没有发生变化,ALNS 算法需执行多轮毁灭和重建操作才能移除整个聚集域,从而增加了 ALNS 算法的迭代次数和执行时间,面对大规模门店配送时,ALNS 算法无法在有限时间内求得较优解。

因此,本文在 ALNS 算法中设计了一种基于密度聚类的毁灭移除算法。在毁灭阶段使用密度聚类算法对配送路线进行聚类分簇,移除时以簇为单位,能在一次迭代中达到其他移除算法多次迭代的效果,而且密度聚类算法实现简单,时间开销较少,在执行一轮毁灭与重建操作上该算法时间开销比其他移除算法少。在国际基准测试案例上进行实验,结果表明采用密度聚类的毁灭移除算法执行相同的迭代次数只需更短时间,而且能得到更优的解,验证了基于密度聚类的毁灭移除算法的有效性。应用于本文研究的企业案例中,与人工调度派车相比,本文设计的 ALNS 算法减少了 1/3 的车辆使用,总开销不到其 1/3,极大地降低了企业的配送成本。

2 问题描述及定义

在企业实际物流配送的车队中,费用计算复杂,包括司机津贴、车辆加油费、过路过桥费、车辆维修费、停车费等,为方便研究,拟将车辆费用分为可变费用和固定费用,可变费用由每公里油耗与行驶里程数决定,固定费用是除可变费用外的其他一切费用。本文考虑多种因素形式化为 HFFVRP 数学模型,该模型目标是确定一组车辆对门店进行配送服务,包括每辆车的车型,每辆车应服务的门店、访问门店的顺序和路线。模型目标函数由两部分费用组成:固定成本和可变成本。固定成本为所有车辆的固定费用累加和,可变成本为所有车辆可变费用累加和。目标函数约束条件为本文考虑的现实因素,配送方案优劣通过目标函数值大小进行判定。

HFFVRP 定义如下:设无向图 $G=(V,E)$, $V=\{0,1,\cdots,n\}$ 代表图中所有的节点,其中物流总仓编号为 0,所有车辆必须从 0 节点出发最终回到 0 节点;其余编号为各个门店,每个门店对货物的需求量为 d_i。$E=\{(i,j)\mid i,j\in V, i\neq j\}$ 是任意两个节点的边,权值为实际门店之间的距离 d_{ij}。假设企业拥有的 m 种车型 $M=\{1,2,\cdots,m\}$,第 k 种车型拥有的车辆集合为 φ_k,固定费用为 f_k,每公里油耗费用为 v_k,该车型最大装载量为 Q_k。规定每个门店的需求必须被满足且只能由一辆车进行服务,问题的目标是:确定一组车辆对 n 个门店进行配送服务,使得总开销最小。

本文定义 HFFVRP 模型使用的参数和决策变量如表 1 所示。建立具有固定车辆数的多车型车辆路径问题数学模型如下所示:

$$\min f(x) = \sum_{k\in M}\sum_{l\in\varphi_k}\left(f_k\sum_{i,j\in V/\{0\}}x_{ijkl}\right) + \sum_{i\in v}\sum_{j\in V}\sum_{k\in M}\sum_{l\in\varphi_k}x_{ijkl}d_{ij}v_k \tag{1}$$

$$\text{s.t.}\quad \sum_{k\in M}\sum_{l\in\varphi_k}\sum_{i\in V}x_{ijkl}\geqslant 1,\ \forall j\in V/\{0\} \tag{2}$$

$$\sum_{i\in V}x_{ipkl}-\sum_{j\in V}x_{pjkl}=0,\ \forall p\in V,k\in M,l\in\varphi_k \tag{3}$$

$$\sum_{k\in M}\sum_{l\in\varphi_k}y_{ikl}=1,\ \forall i\in V/\{0\} \tag{4}$$

$$\sum_{i\in V/\{0\}}d_iy_{ikl}\leqslant Q_k,\ \forall k\in M,\forall l\in\varphi_k \tag{5}$$

$$y_{jkl}=\sum_{i\in V/\{0\}}x_{ijkl},\ \forall j\in V/\{0\},k\in M,l\in\varphi_k \tag{6}$$

其中:$f(x)$ 是目标函数;约束(2)表示各门店至少被一辆车服务一次;约束(3)确保一辆车对一个门店服务后,离开这个门店;约束(4)表示每个门店只能由一辆车提供服务;约束(5)表示被同一辆车服务的门店需要货物总和不能超过该车型最大的装载量;约束(6)表示门店 j 只能由一辆来自其他门店的车辆提供服务。

表 1　HFFVRP 数学模型的参数和决策变量

Tab. 1　Parameters and decision variables of HFFVRR model

标识符		说明
参数	V	物流总仓、所有门店的集合
	M	企业车辆类型的集合
	ϕ_k	k 车型的车辆集合
	f_k	k 车型的固定费用
	v_k	k 车型每公里的油耗费用
	Q_k	k 车型的最大装载量
	d_{ij}	节点 i 到节点 j 的距离
	d_i	门店 i 的需求量
决策变量	x_{ijkl}	若 k 车型的第 l 辆车服务完 i 门店后直接访问 j 门店,x_{ijkl} 为 1,否则为 0
	y_{ikl}	若门店 i 是由 k 车型的第 l 辆车服务,y_{ikl} 为 1,否则为 0

3　算法设计

设 I 为组合优化问题的实例,满足 I 中所有约束条件的可行解为 $S(I)$,衡量一个解好坏的开销函数为式(1),目标是求解式(1)中最小值。对任意一个解 X 上的映射:

$$N:X\in S(I)\rightarrow N(X)\subseteq S(I) \tag{7}$$

称为一个邻域映射,又叫邻域结构。其中:$N(X)$ 为解 X 的邻域集合,称 $X'\in N(X)$ 为 X 的一个邻居。在传统的邻域搜索中,邻域集合的映射方式主要对当前解中一个或者两个元素进行操作。ALNS 是传统邻域搜索的一种扩展,通过对多个元素进行操作,从而扩展了邻域的空间,在解空间中更大的范围内搜索最好解[9]。ALNS 算法求解的质量和速度主要受邻域映射的方式影响,对不同的问题实例,最优的邻域映射方式不同。对于同一个问题实例,设计不同的邻域映射方式,算法求解质量也不相同。ALNS 邻域映射方式由毁灭阶段算法决定,重建阶段则在邻域结构对应的邻域解集合中搜索最佳可行解。ALNS 在一次迭代中主要需要经历毁灭与重建两个阶段:毁灭阶段选择一个简单的移除算法,从当前配送路线中移除 q 个节点,将 q 个节点插入配送路线形成的所有可行解为邻域集合;重建阶段选择一个构造算法从邻域集合中搜索最好解形成新解,最后由接受准则决定是否采纳新解。

设计一个 ALNS 算法需要注意以下三点:1)一组简单有效的毁灭移除算法;2)一组能快速构造可行解的重建算法;3)决策层选择合适的接受准则。

在毁灭阶段本文根据实际情况提出并实现了基于密度聚类的毁灭移除算法(Cluster Removal, CR),这是一种新的邻域映射方式。同时还实现了 Shaw Removal (SR)[9]、Worst Removal(WR)[12]和 Random Removal(RR)[9]三种毁灭移除算法。在重建算法上本文实现了贪婪插入法[11]和 Regret-2 插入法[11]。本文借鉴模拟退火算法[13]设计和实现接受准则。

3.1　算法框架

算法 1　ALNS 框架。

输入　迭代次数 *IterMax*,周期 R,毁灭变量个数 q;

输出　当前最优解 X^*。

过程:

```
1)  初始化参数,构造初始解决方案 X, 令 X* = X
2)  for j = 1 to IterMax do                    //执行设定的迭代次数
3)    for i = 1 to R do                        // R 次迭代一个周期
4)      使用轮盘赌选择法从 L 中选择一个算法组合 a_i
5)      使用 a_i 中的毁灭算法移除 X 方案中 q 个变量
6)      使用 a_i 中的重建算法为 q 个变量重新赋值得到新方案 X'
7)      if 满足接受准则 then
8)        X* = X'
9)        记录 a_i 改善当前方案的情况
10)     end if
11)   end for
12)   根据本周期各算法组合 a_i 改善情况更新其分数 π_i
13) end for
14) 输出 X*
```

算法 1 中 ALNS 框架基本流程包括:产生初始解,自适应选择算法组合,毁灭当前解,重建可行解,根据接受准则判断是否采纳新的可行解。设毁灭与重建算法的全组合为 $L=\{a_1,a_2,\cdots,a_n\}$,相对应的分数集合为 $P=\{\pi_1,\pi_2,\cdots,\pi_n\}$,每种组合 a_i 包括一种毁灭算法和一种重建算法,对应分数为 π_i,π_i 将决定组合 a_i 在轮盘赌选择法被选中的概率。算法 1 中,第 1)行构造初始方案时应选择简单的 VRP 构造算法,本文实现插入法[12]求得初始可行路线集合 X,并设为当前能找到的最好解 X^*。第 2)~12)行是主循环,算法执行指定的迭代次数。在循环内部,算法每执行 R 次迭代将对 P 进行更新,以保证对当前解改善较多的算法组合能被更大概率地选中。在每次迭代中(第 4)~9)行),毁灭算法根据移除规则破坏当前解 X,使得 q 个门店未被服务,重建算法将 q 个门店重新插入到路线集合中得到可行的新路线集合 X',如果 X' 满足接受准则,将成为当前能找到的最好路线组合。当执行次数达到最大迭代次数,输出 X^*。

3.2　基于密度聚类的毁灭移除算法

在本文研究的企业案例中,每天需要处理来自各地 300 多家门店的订单,问题规模较大。在地理分布上,同一个行政区域内门店分布相对密集,但企业物流总仓距离大部分门店较远,基本在 30 km 以上,因此配送路线上的门店具有聚集的特点,适合应用聚类算法。

在图 2 中,假设毁灭阶段选择路线 2 进行移除。对于路线 2 中距离较近的门店 j、k、l 和 m,使用其他毁灭算法进行移除时,算法随机移除其中一个门店,例如门店 k。然后在重建阶段将 k 重新插入到配送路线中,由于在门店 j 和门店 l 之间

插入代价最低，k 将被重新插入到路线 2 中。算法执行一轮毁灭与重建操作后，路线 1 和路线 2 并未发生任何变化，从图 2 左边的配送方案变换为右边的配送方案，算法将需要执行多次迭代。若使用聚类技术，将对路线 2 通过聚类得到 (g,h,i,n) 和 (j,k,l,m) 两个门店簇，每个簇被移除概率为 0.5。被移除的门店簇在重建阶段被重新插入配送路线时可得到图 2 右边的配送方案，这是一个比图 2 左边更优的配送方案，并且只执行一轮毁灭与重建操作，因此，使用聚类技术的邻域映射方式更适合应用于门店规模大且聚集的场景，能减少 ALNS 算法执行的迭代次数，降低 ALNS 算法执行的时间，提高 ALNS 求解质量。

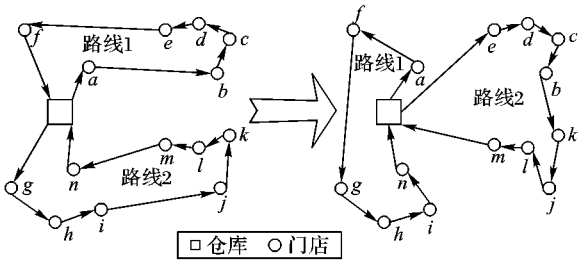

图 2 基于密度聚类的毁灭移除例子

Fig. 2 Example of density clustering based removal heuristic

毁灭阶段需要设计一个简单的移除算法从当前配送方案中移除 q 个门店，若移除算法过于复杂将影响 ALNS 算法整体性能。在 customer 层实现基于聚类的移除算法时，聚类对象为每辆车的配送路线。文献[5]使用的模糊聚类技术和文献[6]使用的基于几何中心的聚类技术需事先指定聚类的个数 g，g 由车辆最大装载量和客户需求量总和决定，最终配送方案将派 g 辆车进行配送，应用于单条配送路线聚类时，k 将难以确定。由于 DBSCAN 聚类算法具有实现简单、无需指定聚类个数、发现任意形状的空间聚类等特点，因此，适合应用于配送路线聚类。在 eclipse 平台下实现该算法时可使用开源数学包 org. apache. commons. math3. ml. clustering，需指定邻域半径参数 *EpsDistance* 和核心对象个数参数 *MinPts*。本文在文献[7]的基础上实现参数的设置方式，*EpsDistance* 由式(8)决定，其中 d_{avg} 为随机选取的 10 个门店之间的平均距离，d_{min} 为 10 个门店之间的最短距离，控制因子 ξ 在本文使用的测试案例中均取值 0.8，*MinPts* 每次随机取值{2,3,4}。由于本文研究的企业案例只提供门店的经纬度信息，门店之间的距离计算需要借助高德地图开发者平台进行查询，因此需要自定义门店间的距离计算方式，并把实现类作为参数传入开源数学包的 DBSCANClusterer 函数中。

$$EpsDistance = (d_{avg} - d_{min}) * \xi \tag{8}$$

基于密度聚类的毁灭移除算法基本思想是随机从当前配送路线集合中选择一条路线，对该路线进行密度聚类得到簇集合，然后随机选择一个簇集合进行移除，直到移除了 q 个门店。设门店集合 $S = \{s_1, s_2, \cdots, s_n\}$，对一条配送路线使用 DBSCAN 聚类后得到簇集合 $C = \{c_1, c_2, \cdots, c_p\}$，每个簇集合拥有的门店 $c_i = \{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\}$，其中 i 取值范围从 1 到 p，查询一个门店 s_i 所在路线使用操作 ShopInRoute $[s_i]$，伪代码如算法 2 所示。

算法 2 Cluster Removal。

输入 门店集合 S，移除门店个数 q；

输出 v；移除门店集 *RemovalShops*。

过程：

```
1) while q > 0 do
2)   if targetRoute == NULL then
3)     从 S 中随机选择一个门店 s_i
4)     targetRoute = ShopInRoute [s_i]
5)   else
6)     从 LastRemoval 中随机选择一个门店 s_i
7)     其他门店到 s_i 距离从小到大排序
8)     选择离 s_i 最近的门店 s_ij 且该门店所属路线未执移除操作
9)     targetRoute = ShopInRoute [s_ij]
10)    清空 LastRemoval = {}
11)  end if
12)  C = DBSCANClusterer (targetRoute, MinPts, EpsDistance)
13)  随机从 C 中选择一个簇 c_i
14)  for 每个 s_il ∈ c_i，其中 l 从 1 到 k do
15)    若 q 为 0，跳到 21)
16)    把 s_il 加入 LastRemoval 中
17)    把 s_il 加入 RemovalShops 中，q 自减 1
18)  end for
19)  把 C_i 加入 RemovalRoutes 中
20) end while
21) 输出 RemovalShops
```

算法分两种情况选择聚类的目标路线 *targetRoute*，通过变量 *LastRemoval* 记录最近被移除的门店。刚开始 *targetRoute* 为空时，随机选择第一条路线进行聚类移除操作，时间复杂度是 $O(1)$。当 *targetRoute* 不为空且已移除门店数量不足 q 个门店时，在第 5) ~10)行，将从最近移除的门店集合中随机选择一个门店 s_j，假设其他门店到 s_j 按距离从小到大排序后为集合 $N = \{s_{j_1}, s_{j_2}, \cdots, s_{j_{n-1}}\}$，从中选择一个未被移除的门店且该门店所属路线未曾执行移除操作，将 *targetRoute* 设置为该门店所属的路线。在具体实现时，由于其他模块也需要 N 集合，在 Cluster Removal 算法中无需重新计算 N，并且选择门店时最多遍历 N 中 q 个门店，所以第 5) ~10) 的时间复杂度与常量 q 相关，为 $O(q)$。在算法第 12)行对 targetRoute 路线进行密度聚类，DBSCAN 聚类算法在最优的情况下时间复杂度为 $O(n \log n)$，在最坏情况下时间复杂度为 $O(n^2)$。因此，Cluster Removal 算法时间复杂度取决于 DBSCAN 聚类算法，在最优情况下时间复杂度为 $O(q \cdot n \log n)$，在最坏情况下时间复杂度为 $O(q \cdot n^2)$。

3.3 重建算法

重建算法将毁灭阶段移除的 q 个门店重新插入到配送路线中，使每个门店的需求都被满足，且不违反各车型车辆数量有限的约束与车辆最大装载容量的约束，形成可行的配送方案。本文根据 HFFVRP 的特点设计了插入代价的计算方式，实现贪婪插入和 Regret-2 插入两种常用的构造算法。

3.3.1 贪婪插入法

算法基本思想是在每次迭代中为一个移除门店寻找插入代价[12]最低的路线进行插入。根据车型可变成本，本文设计了如式(9)所示的插入代价函数：

$$\Delta f_{i,k} = (C^k_{ai} + C^k_{ib} - C^k_{ab}) - \gamma(C^k_{0i} + C^k_{i0}) \tag{9}$$

式(9)表示对 k 车型配送路线上的门店 a 和门店 b 之间插入 i。$\Delta f_{i,k}$ 的值由两部分代价组成，分别是 i 插入 a 与 b 之间的开销和新增一辆车服务 i 的开销。通过控制因子 γ 可以在插入当前路线或新增一辆车上取得平衡。当门店距离仓库较

近且插入当前路线代价较高时,应新增一辆车对门店进行服务;当门店距离仓库较远时,应避免新增一辆车。控制因子 γ 在迭代中每次随机取值 $\{0.00,0.05,0.10,\cdots,1.50\}$。当出现违反车辆容量约束而无法将门店 i 插入路线 k 的情况时,先判断该车型是否还有可用车辆,若存在可用车辆,则 $\Delta f_{i,k}$ 为新增一辆车服务该门店的代价,否则 $\Delta f_{i,k}=\infty$。算法每次迭代寻找式(10)中门店与路线的组合:

$$(i,k)=\arg\min_{i\in S,k\in R}\Delta f_{i,k} \tag{10}$$

然后,将门店 i 插入 k 路线中,更新车辆的可用容量。迭代一直进行,直到 q 个门店全部被插入配送路线中。该算法实现简单,计算 $\Delta f_{i,k}$ 操作最多需要进行 n 次,因此算法的时间复杂度为 $O(q\cdot n)$。但是,对于插入开销最大的门店,算法在最后一次迭代进行处理,此时每条路线对应的车辆可用容量低,插入任何路线都容易违反车辆的最大装载容量约束,唯有新增一辆车对该门店进行服务,从而增加了车辆数,总开销变大。

3.3.2　Regret-2 插入法

Regret-2 算法解决了贪婪插入法存在的问题。假设 Δf_i^1 代表门店 i 插入最优路线的代价,Δf_i^2 代表门店 i 插入次优路线的代价,两者之间的差值越大说明该门店越不适合插入次优的路线中。Regret-2 插入法在每次迭代中寻找式(11)差值最大的门店 i 进行插入处理:

$$i=\arg\max_{i\in S}(\Delta f_i^2-\Delta f_i^1) \tag{11}$$

式(11)保证了门店 i 插入到最合适的路线上,算法迭代后期处理的是最优路线与次优路线之间差值不大的门店,能有效避免贪婪插入法在迭代后期容易出现违反车辆容量约束的问题。

3.4　决策层

经过毁灭与重建两个阶段,算法产生一个新的配送方案,决策层需决定是否接受新的方案。在本文实现的 ALNS 算法中,决策层做了两件事情:1)采用模拟退火接受准则决定是否接受新的配送方案;2)自适应选择下一次迭代的算法组合。

3.4.1　接受准则

模拟退火算法应用在车辆路径问题上取得了很不错的效果[13],本文借鉴模拟退火算法实现接受准则,算法接受比当前解更好的新解,也以一定概率接受比当前解更差的新解。

在 ALNS 算法第 7)行,接受新解 X' 的概率为:$e^{\frac{f(x')-f(x)}{T}}$。其中:$f(x)$ 为式(1)定义的总开销函数;T 代表温度,由 T_{start} 开始降温,降温公式为 $T=T\cdot c$,冷却速率为 c,取值范围 $0<c<1$。初始温度 T_{start} 的设置对算法影响较大,本文根据文献[13]提供的方法,修改总开销函数,然后通过计算插入法产生的初始方案的总开销对 T_{start} 进行设置。

3.4.2　自适应选择算法

毁灭与重建阶段各需一个简单的算法,在本文实现的 ALNS 算法中将有 8 种算法组合,不同的组合适用于不同的数据集,为使算法能自适应地选择合适的算法组合进行毁灭与重建,对每种组合设计了相应的分数,根据分数采用轮盘赌选择法[9]进行选择。假设第 i 种组合分数为 π_i,有 H 种组合,则第 i 种组合被选择的概率为:$\pi_i\Big/\sum_{i=1}^{H}\pi_i$。分数在每个迭代周期结束后进行更新,假设 $\pi_{i,j}$ 代表第 i 种组合在迭代周期 j 的分数,更新规则为:

$$\pi_{i,j+1}=\pi_{i,j}(1-r)+r\frac{\theta_i}{R} \tag{12}$$

其中:r 为选择因子,取值 $0<r<1$,如果历史分数对下一周期分数较为重要,r 设置为接近 0 的数字,否则设置为接近于 1 的数字;R 代表一个周期进行迭代的次数;θ_i 为通过组合 i 计算得到的可行方案在周期 j 内被采纳的次数。

4　实验与分析

本文算法在 eclipse 平台下采用 Java 语言编程实现,测试环境为 PC,配置为 Intel Core i5-3470 @ 3.2 GHz,8.00 GB 内存,32 位 Windows 7 操作系统。为方便描述,毁灭阶段使用 Cluster Removal 机制的 ALNS 算法简称 C_ALNS,未使用该机制的算法简称 ALNS。仿真实验使用的国际标准测试案例可以从网站 http://neo.lcc.uma.es/vrp/vrp-instances/[14] 下载。

4.1　基准测试案例验证有效性

本文提出的 C_ALNS 适用于具有大规模客户,且客户在地理位置上具有聚集特点的场景。在网络上公开的 HFFVRP 数据集中客户点并没有聚集特点,不符合本文需要研究的场景。因此,本文采用符合场景应用的 CVRP(国际基准测试案例进行测试,CVRP 可看成只有一种车型的 HFFVRP。该测试案例为 Augerat 等提供的 B 类型的数据集[14],客户点具有聚集的特点,规模从 31 到 78 不等。

算法参数设置如下:在 C_ALNS 和 ALNS 中,每种毁灭与重建算法组合初始分数 π_i 都设为 0.5,以保证每种组合被选中的概率相同;分数更新选择因子 r 设为 0.2。根据文献[12]给出的建议:在小规模问题中,将毁灭阶段需要移除的客户数量 q 设置为 $\min\{0.1n,30\}$,其中 n 代表客户总数;大规模问题则设置为 $\min\{0.4n,60\}$。迭代次数 *IterMax* 设为 2 000。

Augerat 数据集的客户规模较小,能在较短时间内得到稳定解,对每个测试案例均进行 5 次实验,分别记录 C_ALNS 算法和 ALNS 算法取得的解及花费的时间,记录各移除算法产生的满足接受准则解的次数,结果如表 2 所示。在表 2 中:第一列为测试案例编号;第二列为该案例目前已知的最优解;*cost* 列为相应算法取得的解;*time* 列为相应算法的执行时间(单位为 *s*);*gap*/% 列表示相应算法取得的解与公认最优解之间的误差,计算公式为:

$$gap=(Z_{\text{C\_ALNS/ALNS}}-Z_{best})/Z_{best}*100\% \tag{13}$$

其中:$Z_{\text{C\_ALNS/ALNS}}$ 代表 C_ALNS 算法或者 ALNS 算法取得的解,Z_{best} 代表该案例已知最优解。SR、RR、WR、CR 列为本文实现的移除算法的贡献率,各移除算法贡献率计算方式为:ALNS 执行 2 000 次迭代,由该移除算法产生满足接受准则的解的次数除以所有满足接受准则的解的总次数。测试案例移除算法贡献率越高,对应的邻域映射方式越适合应用于解决该问题实例。从表 2 中可知,两种算法取得的最好解与案例已知最优解之间误差均在 1% 以内,误差较小。在 23 个测试案例上,C_ANS 算法需要更短的时间,所需时间加黑表示该案例下 C_ALNS 算法求解质量比 ALNS 算法更优,占 19 个案例,说明在相同迭代次数下,C_ALNS 算法能在更短的时间内求得更优的解,比 ALNS 更高效。在 4 个例外的案例中,结合各毁灭移除算法贡献率可知,SR 或者 WR 毁灭移除算法更适合应用于这些案例,但 C_ALNS 产生的解与 ALNS 产生的解相差不超过 1%,C_ALNS 需要时间更短。在 C_ALNS 表现更好的 19 个案

例中,CR 算法贡献率最高,ALNS 则是 SR 或者 WR 算法贡献率高,这说明 CR 是比 SR 或 WR 时间复杂度更低的毁灭移除算法,定义的邻域映射方式更适合应用于客户具有聚集特点的 VRP 场景下,验证了基于密度聚类的毁灭移除算法的有效性。

表 2 Augerat 数据集测试结果

Tab. 2 Experimental results of Augerat data set

案例编号	最优解	C_ALNS							ALNS					
		cost	time/s	gap/%	算法贡献率/%				cost	time/s	gap/%	算法贡献率/%		
					SR	RR	WR	CR				SR	RR	WR
B-n31-k5	672	676	**1.367**	0.60	20.07	12.41	23.36	44.16	676	1.712	0.60	30.71	26.38	42.91
B-n34-k5	788	789	**1.530**	0.13	24.27	13.81	13.39	48.52	789	2.043	0.13	50.29	33.14	16.57
B-n35-k5	955	956	**1.539**	0.10	18.06	15.16	24.84	41.84	956	1.996	0.10	30.2	27.71	42.92
B-n38-k6	805	807	**1.838**	0.25	23.23	8.41	16.15	52.21	807	2.337	0.25	42.36	18.18	39.45
B-n39-k5	549	553	**1.825**	0.73	20.99	11.07	24.24	43.70	553	2.361	0.73	33.89	21.00	45.11
B-n41-k6	829	833	**2.056**	0.48	23.85	7.88	15.75	52.52	833	2.488	0.48	45.21	12.77	42.02
B-n43-k6	742	746	**2.330**	0.54	20.73	10.78	24.65	43.84	746	2.951	0.54	41.94	15.82	42.24
B-n44-k7	909	914	**2.346**	0.55	24.17	11.51	16.55	47.77	914	2.987	0.55	38.87	20.93	40.20
B-n45-k5	751	753	**2.349**	0.27	21.88	11.29	19.06	47.77	753	3.181	0.27	40.72	18.04	41.24
B-n45-k6	678	682	2.671	0.59	25.89	9.03	34.56	30.52	680	3.301	0.29	29.66	13.28	57.06
B-n50-k7	741	744	**3.219**	0.40	18.89	14.75	28.57	37.79	744	4.096	0.40	28.28	25.25	46.46
B-n50-k8	1312	1318	2.928	0.46	36.62	12.41	20.02	30.94	1317	3.727	0.38	47.08	18.96	33.96
B-n51-k7	1018	1019	**2.763**	0.10	16.05	7.54	22.31	54.09	1019	4.061	0.10	38.02	20.32	41.65
B-n52-k7	747	750	**3.301**	0.40	22.80	10.45	21.62	45.13	750	4.185	0.40	35.76	20.7	43.54
B-n56-k7	707	712	**3.548**	0.71	19.50	12.77	24.82	42.91	712	5.168	0.71	37.63	21.34	41.04
B-n57-k7	1144	1144	**3.792**	0.00	18.69	12.62	28.5	40.19	1144	4.817	0.00	25.73	24.27	50.00
B-n57-k9	1598	1604	**3.740**	0.38	18.87	9.34	23.74	48.05	1604	5.183	0.38	40.58	21.73	37.70
B-n63-k10	1496	1552	4.151	3.75	35.74	5.71	30.00	28.57	1527	6.019	2.07	45.59	15.13	39.29
B-n64-k9	861	868	**4.135**	0.81	14.69	10.17	24.29	50.85	868	6.192	0.81	33.92	17.63	48.55
B-n66-k9	1316	1327	**4.524**	0.84	23.52	5.73	25.49	45.25	1327	6.934	0.84	41.85	12.36	45.79
B-n67-k10	1032	1044	**5.153**	1.16	17.75	8.97	26.24	47.04	1046	6.795	1.36	27.78	17.44	54.78
B-n68-k9	1272	1284	**4.712**	0.94	20.47	9.06	22.64	47.83	1285	7.843	1.02	41.37	15.53	43.11
B-n78-k10	1221	1265	6.490	3.60	24.03	12.06	36.29	27.62	1262	9.566	3.35	32.15	16.16	51.69

表 3 给出 C_ALNS 算法与 Hassani 等的混合蚁群算法 H_ACS[15]、Tasan 等的遗传算法(Genetic Algorithm, GA)[16]、文献[17]公布的粒子群优化(Particle Swarm Optimization, PSO)算法、Ewbank 等的无监督模糊聚类算法[5]、Shin 等的 Centroid-based 3-phase 算法[6]的结果比较,Average 行表示每种算法求解结果与案例已知最优解之间的平均误差,符号“—”表示该算法没有提供对应测试案例求解结果。

从表 3 中可知,C_ALNS 求解 23 个测试案例得到的最好解比两种使用了聚类技术的算法[5-6]求解的结果更优,说明 DBSCAN 聚类算法适合应用于配送路线聚类。与案例已知最优解相比,C_ALNS 算法平均误差为 0.77%,H_ACS 算法平均误差为1.66%,GA 的平均误差为 25.03%,PSO 算法平均误差为 0.30%,Centroid-based 3-phase 算法平均误差为 6.28%,Ewbank 等的无监督模糊聚类算法平均误差为 4.58%。GA 的平均误差较大的原因是算法求解质量取决于测试案例的上界和下界,下界为案例已知最优解,但由于 GA 使用 CPLEX 在 Augerat 数据集上求解的上界较大,故求解结果较差。对比结果表明 C_ALNS 算法求解质量优于H_ACS算法、GA,Centroid-based 3-phase 算法和 Ewbank 等的无监督模糊聚类算法,稍逊于 PSO 算法,但 C_ALNS 算法在给定迭代次数下求得最好解,以满足时间限制要求,PSO 算法以最优解为目标,并未限定时间约束,因此,本文设计的 C_ALNS 算法能有效解决车辆路径问题。

4.2 企业实际案例

本文研究的企业案例每天凌晨前汇总当天所有门店的订单,并在凌晨后处理部分订单,在白天处理大部分订单。企业需要算法在有限时间内求得较好的配送方案,指导货物装车,并使车辆按指定路线进行配送。

4.2.1 数据提取

企业业务部门提供以下数据:每个门店详细地址信息,包括每个门店的经纬度;37 种车型详细数据信息,包括车辆最大装载量、车辆固定开销费用、车辆每公里油耗;一个月的订单历史数据和人工派车调度处理的结果数据。本文对订单历史数据按日进行处理,每天有 350 个左右的门店下单。利用门店的经纬度信息,通过高德开放平台可查询 2 个门店之间的最短距离或用时最少距离,从而得到每 2 个门店之间的距离。

4.2.2 参数设置

本文随机选取一天订单进行处理,涉及 345 个门店,规模较大,因此毁灭阶段需移除门店个数 q 设为 60。对 C_ALNS 和 ALNS 中每种算法组合初始分数设为 0.5,保证每种组合选择概率相同,分数更新选择因子 r 设为 0.2。算法分别进行 1000、2000、4000、8000、16000 次迭代,以说明算法求得的最好解、迭代次数和时间开销三者之间的关系。同时选取 5、10、15、20、25、30、35 辆车进行一组实验,迭代次数为 5000,以说明车辆规模与时间开销的关系。最后选取 C_ALNS 和 ALNS 迭代 5000 次的配送方案与企业人工调度配送方案进行对比。

表 3　算法结果对比

Tab. 3　Comparison of the results of C_ALNS and other algorithms

案例	C_ALNS		H_ACS		GA		PSO		Centroid-based3-phase 算法[6]		Ewbank 算法[5]	
	cost	*gap*/%	*cost*	*gap*/%	*cost*	*gap*/%	*cost*	*gap*/%	*cost*	*gap*/%	*cost*	*gap*/%
B-n31-k5	676	0.60	—	—	736	9.52	672	0.00	700	4.17	—	1.10
B-n34-k5	789	0.13	—	—	865	9.77	788	0.00	851	7.99	—	2.20
B-n35-k5	956	0.10	—	—	1 263	32.25	955	0.00	969	1.47	—	3.40
B-n38-k6	807	0.25	—	—	925	14.91	805	0.00	834	3.60	—	5.40
B-n39-k5	553	0.73	—	—	744	35.52	549	0.00	620	12.93	—	4.90
B-n41-k6	833	0.48	—	—	1 105	33.29	829	0.00	862	3.98	—	3.30
B-n43-k6	746	0.54	—	—	1 047	41.11	742	0.00	857	15.50	—	4.70
B-n44-k7	914	0.55	—		1 126	23.87	909	0.00	963	5.94	—	2.60
B-n45-k5	753	0.27	—	—	—	—	751	0.00	807	7.46	—	4.00
B-n45-k6	682	0.59	—	—	—	—	678	0.00	743	9.59	—	6.10
B-n50-k7	744	0.40	—	—	—	—	741	0.00	772	4.18	—	7.30
B-n50-k8	1 318	0.46	1 317	0.38	—	—	1 315	0.23	1431	9.07	—	4.70
B-n51-k7	1 019	0.10	—	—	—	—	1 038	1.96	1 028	0.98	—	3.00
B-n52-k7	750	0.40	—	—	—	—	747	0.00	754	0.94	—	3.70
B-n56-k7	712	0.71	—	—	—	—	707	0.00	741	4.81	—	3.70
B-n57-k7	1 144	0.00	—	—	—	—	1 162	1.57	1163	1.66	—	7.50
B-n57-k9	1 604	0.38	1 598	0.00	—	—	1 598	0.00	1 673	4.69	—	4.00
B-n63-k10	1 552	3.75	1 541	3.01	—	—	1 496	0.00	1 664	11.23	—	5.00
B-n64-k9	868	0.81	—	—	—	—	864	0.35	910	5.69	—	6.70
B-n66-k9	1 327	0.84	1 372	4.26	—	—	—	—	1 468	11.55	—	2.70
B-n67-k10	1 044	1.16	1 035	0.29	—	—	1 034	0.19	1 108	7.36	—	5.90
B-n68-k9	1 284	0.943	1 301	2.28	—	—	1 273	0.08	1 338	5.19	—	4.90
B-n78-k10	1 265	3.60	1 238	1.39	—	—	1 249	2.29	1 276	4.50	—	8.60
Average		0.77		1.66		25.03		0.30		6.28		4.58

4.2.3　结果对比与分析

图 3 对比了 C_ALNS 与 ALNS 算法配送方案总开销,总开销为式(1)定义的目标函数。从迭代次数与总开销关系中可知,算法在执行相同次数情况下,C_ALNS 算法求解效果更优,说明 CR 算法适用于门店规模大且聚集特点的应用场景,能提高 C_ALNS 求解质量。两条曲线下降的趋势表明两种算法随着迭代次数增加,求解质量更优,并最终达到稳定值,因此,在有限时间内,对大规模问题应尽量增加算法执行次数以取得较优的配送方案。

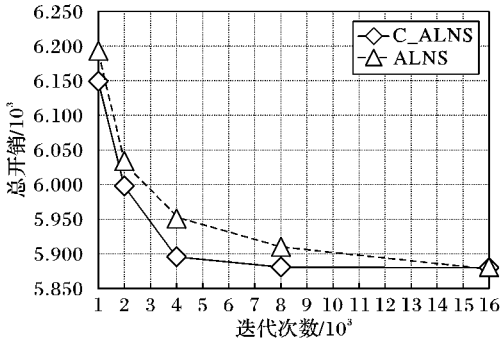

图 3　迭代次数与总开销关系

Fig. 3　Relationship between iterations and total cost

图 4 对比了两种算法的执行时间。在执行相同迭代次数下,C_ALNS 算法需要更少时间,说明 CR 是一个简单算法,相比其他算法组合,使用 CR 算法的组合只需花费更短时间执行一轮毁灭与重建的操作,从而缩短了 C_ALNS 算法整体执行时间。两种算法执行时间曲线走势说明迭代次数与花费时间呈正相关关系,若企业需要在 1 个小时内得到较好的配送方案,在门店规模为 345、车型数目为 37 种的情况下,执行 5 000次迭代是较好的选择。

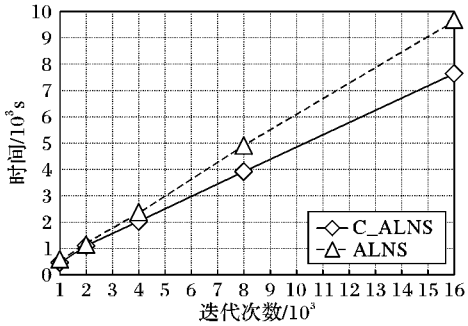

图 4　迭代次数与时间开销关系

Fig. 4　Relationship between iterations and time

图 5 说明车辆规模与算法时间开销的关系。图中曲线走势表明在给定迭代次数下,车辆规模越大,算法执行时间越长。在实际应用中,应根据需要选择合适车型,以减少车队规模。例如,对生鲜商品进行配送,只需考虑具有冷藏功能的车型,从而减少配送车辆规模,降低算法执行时间,在有限时间内增加算法执行次数,以求得更优的配送方案。

表 4 对比企业人工调度、ALNS 算法和 C_ALNS 算法计算的配送方案。车型数为每种配送方案使用车型的种数,车辆数为每种配送方案使用的车辆总数,车辆平均装载率为所使用的车辆装载率之和除以使用车辆总数。相比人工调度计算的配送方案,C_ALNS 算法与 ALNS 算法使用了更多的车型,C_ALNS 算法减少接近 1/3 的车辆使用,平均车辆装载率提

高了 20%，总开销不到其 1/3，有效地降低了企业配送成本。因此，C_ALNS 算法能在有限时间内求得较优配送方案，适合应用在多车型大规模企业物流配送问题中。

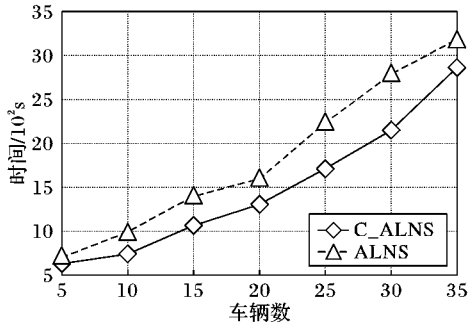

图 5 车辆规模与时间开销关系

Fig. 5 Relationship between vehicle scale and time

表 4 几种算法配送方案对比

Tab. 4 Comparison of delivery solutions calculated by different algorithms

算法	车型数	车辆数	车辆平均装载率/%	总开销
人工调度	6	30	70.36	18312
ANLS	10	23	91.37	5910
C_ALNS	10	21	91.54	5881

5 结语

为解决多车型大规模物流配送问题，本文在 ALNS 算法框架下设计和实现了一种基于密度聚类的毁灭移除算法。使用国际标准测试案例进行仿真实验，通过 C_ALNS 与 ALNS 算法求解结果对比可知，基于密度聚类的毁灭移除算法定义的邻域映射方式更适合门店规模大且聚集的场景，能有效降低 ALNS 算法整体执行时间，提高 ALNS 求解质量，从而验证所提移除算法的有效性。与 Ewbank 等的无监督聚类算法[5]、Shin 等的 Centriod-based 3-phase 算法[6]、H_ACS 算法[15]、GA[16]、PSO 算法[17]相比，C_ALNS 求解质量优于 H_ACS 算法、GA、Ewbank 和 Shin 的算法，稍逊于 PSO 算法，但 C_ALNS 算法在限定时间内求解，PSO 算法以最优解为目标，并未限定时间约束，因此，C_ALNS 算法能有效解决物流配送问题。应用于企业实际数据集的结果表明，C_ALNS 算法能在有限时间内得到较优的配送方案，极大地降低了企业的物流配送成本。在新零售模式下，本文提出的算法能有效解决企业对大规模数量的门店进行货物配送的问题，下一步将研究门店到各顾客之间的物流配送问题。

参考文献（References）

[1] 郝建彬. “新零售”的曙光[J]. 互联网经济, 2016(11): 12 - 15. (HAO J B. The dawn of “new retail” [J]. The Internet Economy, 2016(11): 12 - 15.)

[2] DANTZIG G B, RAMSER J H. The truck dispatching problem [J]. Management Science, 1959, 6(1): 80 - 91.

[3] KUMAR S N, PANNEERSELVAM R. A survey on the vehicle routing problem and its variants [J]. Intelligent Information Management, 2012, 4(3): 66 - 74.

[4] 曹二宝, 赖明勇, 聂凯, 等. 大规模物流配送车辆调度问题研究[J]. 湖南大学学报(自然科学版), 2007, 34(12): 89 - 92. (CAO E B, LAI M Y, NIE K, et al. Research on large-scale vehicle routing problem of logistics-distribution[J]. Journal of Hunan University (Natural Sciences), 2007, 34(12): 89 - 92.)

[5] EWBANK H, WANKE P, HADI-VENCHEH A. An unsupervised fuzzy clustering approach to the capacitated vehicle routing problem [J]. Neural Computing and Applications, 2016, 27(4): 857 - 867.

[6] SHIN K, HAN S. A centroid-based heuristic algorithm for the capacitated vehicle routing problem [J]. Computing & Informatics, 2011, 30(4): 721 - 732.

[7] 谭颖, 胡瑞飞, 殷国富. 多密度阈值的 DBSCAN 改进算法[J]. 计算机应用, 2008, 28(3): 745 - 748. (TAN Y, HU R F, YIN G F. Adapted DBSCAN with multi-threshold[J]. Journal of Computer Applications, 2008, 28(3): 745 - 748.)

[8] VAZ PENNA P H, SUBRAMANIAN A, OCHI L S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem [J]. Journal of Heuristics, 2013, 19(2): 201 - 232.

[9] ROPKE S, PISINGER D. A unified heuristic for a large class of vehicle routing problems with backhauls [J]. European Journal of Operational Research, 2006, 171(3): 750 - 775.

[10] GRANGIER P, GENDREAU M, LEHUÉDÉ F, et al. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization [J]. European Journal of Operational Research, 2014, 254(1): 80 - 91.

[11] AZI N, GENDREAU M, POTVIN J-Y. An adaptive large neighborhood search for a vehicle routing problem with multiple routes [J]. Computers & Operations Research, 2014, 41(1): 167 - 173.

[12] GHILAS V, DEMIR E, VAN WOENSEL T. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines [J]. Computers & Operations Research, 2016, 72: 12 - 30.

[13] YU V F, LIN S-Y. A simulated annealing heuristic for the open location-routing problem [J]. Computers & Operations Research, 2015, 62(C): 184 - 196.

[14] NEO. VRP Instances [EB/OL]. [2017-02-17]. http://neo.lcc.uma.es/vrp/vrp-instances/.

[15] HASSANI A H E, BOUHAFS L, KOUKAM A. A hybrid ant colony system approach for the capacitated vehicle routing problem and the capacitated vehicle routing problem with time windows[M]// Vehicle Routing Problem. [S.l.]: InTechOpen., 2008: 58 - 70.

[16] TASAN A S, GEN M. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries [J]. Computers & Industrial Engineering, 2012, 62(3): 755 - 761.

[17] TEOH B E, PONNAMBALAM S G, KANAGARAJ G. Differential evolution algorithm with local search for capacitated vehicle routing problem [J]. International Journal of Bio-Inspired Computation, 2015, 7(5): 321 - 342.

This work is supported by the National Key Technology R&D Program (2015BAH05F02)

YANG Wang, born in 1982, Ph. D., associate professor. His research interests include computer algorithm.

HE Guochao, born in 1991, M. S. candidate. His research interests include vehicle routing problem.

WU Yan, born in 1992, M. S. candidate. Her research interests include vehicle routing problem.