浙江理工大学学报，2019，42(6)：627-636
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673－3851(s).2019.06.005

基于最小集合覆盖的电商订单拆分及配送方式

韩曙光[a]，章园园[b]
（浙江理工大学，a.理学院；b.经济管理学院，杭州 310018）

摘　要：订单履约的效率是影响自营型电商平台运营成本与顾客满意度的一个重要指标。传统的拆单方式会导致订单积压，订单履约效率低下。基于电子商务订单的特性，对订单的拆单方式以及后续的配送路径进行研究，建立了非线性整数规划模型，并采用两阶段法的求解思想，将模型分为覆盖订单部分与路径规划部分，以经典最小集合覆盖思想与改进的蚁群算法的联合方法为模型求解算法，对该模型进行求解分析。算例表明，该模型与算法具有有效性和适用性。

关键词：订单拆分；订单分配；最小集合覆盖；改进蚁群算法
中图分类号：F224.5　　**文献标志码**：A　　**文章编号**：1673-3851 (2019) 12-0627-10

E-commerce order splitting and allocation method based on minimal set covering problem

HAN Shuguang[a], *ZHANG Yuanyuan*[b]
(a. School of Sciences; b. School of Economics and Management,
Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: Efficiency of order fulfillment is an important indicator which affects the operation cost of self-owned E-commerce platform and customers' satisfaction. Order backlog and low efficiency of order fulfillment are caused by the traditional order splitting method. Based on the characteristics of E－commerce orders, the research on the method of order splitting and the subsequent delivery path was conducted, and a non-linear integer programming model was established. Based on the solving method of two－stage approach, the model was divided into order covering part and route planning part, and the model was solved by the combination of minimal set covering problem with the improved ant colony algorithm. Finally, the validity and applicability of the model and algorithm were verified by a numerical simulation.

Key words: order splitting; order allocation; minimal set covering problem; improved ant colony algorithm

电子商务的高速发展，催生了众多自营型电商平台。为了满足客户多样化的需求，提高订单的响应度以及增加配送服务的鲁棒性，自营型电商平台不仅在区域内设立物流配送中心，也在区域的不同方位配置多个前置仓，且这些前置仓的职能各不相同。由于前、后台存在库存偏差，会出现订单过大的情况，部分订单需要被拆分配送。现有的拆单方式对顾客而言存在一定的缺陷：订单被拆分成的包裹数越多，会导致顾客的取件次数越多、包裹的到达时间不稳定且易丢件、漏件，使得配送满意度下降。当

收稿日期：2018－12－27　　**网络出版日期**：2019－05－15
基金项目：国家自然科学基金项目(11571013，11471286，11701518)
作者简介：韩曙光(1977－　)，男，江苏建湖人，副教授，博士，主要从事物流与供应链管理、算法设计等方面的研究。

前多数自营型电商平台的产品差异化不明显,平台间竞争激烈。因此,着力提高客户对配送服务的满意度,进而增加用户粘性,是电商平台提升盈利空间的重要举措之一。

订单交付是网络购物流程的最后一步,也是直接影响电商平台配送成本和客户配送满意度的关键环节。它包括两个部分,第一部分是订单的拆分与分配,第二部分是订单的配送过程。订单的拆分与分配是对订单与仓库之间匹配关系的描述,订单的配送过程是对订单与所属仓库间路径的刻画,二者相辅相成。以往的文献大多将订单拆分和订单配送作为两个相互独立的问题单独进行研究,而事实上拆单的结果会影响订单配送路径的生成,路径的配送成本也会制约订单与仓库之间的匹配结果,故需对订单的拆分与配送应统筹考虑。考虑订单的拆分配送问题与需求可拆分的多配送中心车辆路径问题(Split delivery multi-depot vehicle routing problem, SDMDVRP)[1]相近,是多配送中心的车辆调度问题(Multi-depot vehicle routing problem, MDVRP)[2]与需求可拆分的车辆路径问题(Split delivery vehicle routing problem, SDVRP)[3]的结合。已有许多国内外学者对多配送中心车辆调度和需求可拆分的车辆路径问题进行了广泛的研究。

对于多配送中心车辆调度问题,已有研究多数是将多配送中心转化为单配送中心,用车辆路径问题去求解[4-5],但这种既定的分配规则不符合实际配送要求。因此,学者们开始对动态配送区域进行研究,即在一次配送循环完成之后,根据一定规则形成新的配送区域。肖正中等[6]采用边界分配法打破了配送区域界限,对车辆跨区域调度进行研究。跨区域车辆调度是在既定的配送区域上实现配送路径的优化,其配送范围不再唯一,更贴近实际。除此之外,也有对MDVRP问题整体进行研究,寻求全局意义上的优化[7]。在解决MDVRP问题的算法层面,也有相应的研究。邢鹏[8]提出了以量子理论来解决云配送模式下的多配送中心车辆调度问题,对多配送中心车辆调度问题进行整体优化,利用量子之间的叠加性和相干性,可以并行搜索,且能很好地保持算子的交叉与变异,提高算法鲁棒性。金涛[9]针对标准化差分进化算法易早熟的缺陷,提出了一种改进的差分进化算法应用于MDVRP问题,采用动态自适应规则和高斯扰动的方式对交叉变异过程进行改进,从而获得更好的寻优效果。

需求可拆分的车辆路径问题通常是指在单个仓库的配送区域内,将订单按规则拆分,以达到更高的装载率,减少资源浪费。以往文献研究多数是对同类物品或同质物品按数量级拆分,不符合电子商务订单的特性[10-12]。符卓等[13]考虑到订单内的商品可能无法被拆分配送,提出了需求按订单拆分的车辆路径问题,将整个订单作为被拆分的最小单元,保证了订单的独立性与多样性。Archetti等[14]只允许对商品种类进行分割,不对数量进行拆分,即不同车辆对同一客户点配送不同的商品,建立C-SDVRP(Commodity constrained split delivery vehicle routing problem)模型,以分支-价格-切割法求解。文献[3]是需求可拆分的车辆路径问题的综述。更多的文献集中于算法研究,提出各类适用于该问题的智能算法,鲜有研究涉及订单品类与数量的拆分[15-17]。

由于电子商务订单的特性及存在“一地多仓”的情况[18],研究订单的拆单原则以及统筹考虑订单拆分与配送很有必要。现存的拆单方式是以“最短距离”为分配原则,指定距离最近的仓库优先与订单进行匹配,若订单没有被完全分配,再指定次近的仓库对订单剩余内容进行匹配,直到订单被匹配完。因订单与各个仓库之间的匹配程度各异,不同的拆单方式会导致包裹数量不同,而减少拆单的数量可以降低路径成本,减少客户取件次数,提高客户的配送满意度。在上述的文献中,均未考虑到匹配程度对拆单方式的影响。本文在多配送中心订单可任意拆分的基础上,对订单与仓库间的匹配程度进行研究。首先,提出以“最小拆单数”为原则的需求可分割多配送中心车辆路径问题;然后,以最小集合覆盖思想[19]确定订单与仓库间的匹配程度,并建立以配送成本最小化为目标的非线性整数规划模型;最后,设计了两阶段法求解,并用算例仿真的形式验证模型的合理性与有效性。

一、订单拆分数学模型

(一)问题描述

本文根据电子商务订单的特性,对电商订单的拆分及配送进行优化,旨在为客户解决订单子包裹繁多、到达时间不一致等问题,在保证客户满意度的基础上对配送成本进行优化。以某自营电商平台为例,其订单分配过程如图1所示:已知仓库的数量与内容,多个仓库共同向若干客户点配货,每个订单具有相应的时间窗信息、地理位置信息、数量和内容信息。车辆从仓库出发,遍历每个订单,按照订单的时间窗进行配送服务。由于电子商务订单具有品类

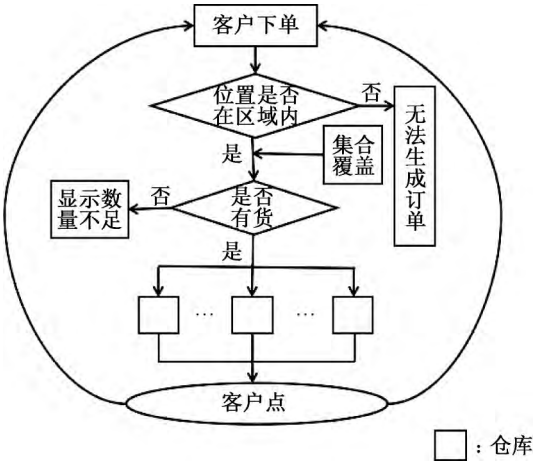


图 1　订单分配过程

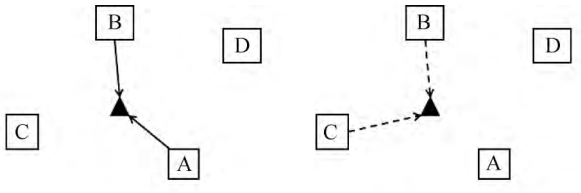
多、数量大的特点，且区域前置仓的职能各不相同，在订单履约过程中不可避免地遇到拆单问题。现有的拆单方式是以仓库与订单之间的距离为拆分原则，而为了保证配送满意度，还应尽量减少拆单数量，以防止出现漏发商品、客户丢件和包裹到达时间不一致等情况。因此，现有以“最短距离”为原则的拆单方式无法起到减少拆单包裹的作用，需要设置新的拆单方式。此时，考虑到订单与仓库之间的匹配程度各不相同，将匹配程度作为设置拆单原则的依据，以“最小拆单数”原则为首要拆单目标，有效减少订单的拆分数量。如表 1 所示，两种拆单方式在平台处理订单的时效性和客户收取包裹的便捷性等方面各有优劣，前者在平台处理订单时较有优势，时效性较高，但易导致客户取件次数多的情况；后者客户取件次数较前者少，但平台处理订单的时效性较差。

表 1　两种拆单方式的优劣

拆单方式	原则	优势	劣势
按“最短距离”拆分	距离优先	平台系统处理订单迅速	包裹数量多，客户取件频繁
按“最小拆单数”拆分	包裹数优先、距离选择次之	客户收件次数减少	对平台系统要求高，决策时间长

下面举例对两种不同的拆单方式具体说明。假设现有 1 号订单为被拆分订单，区域内共有 A、B、C、D 四个仓库提供服务，其地理位置的情况是：与 1 号订单最近的仓库是 A 仓，次近的是 B 仓，再次是 C 仓，D 仓相距最远。根据订单的多样性以及订单与仓库之间的匹配关系，采用不同的拆单方式会造成以下两种不同的匹配结果。

第一种：设 1 号订单的商品无法由 A、D 仓满足，可以由 B 仓或 C 仓满足，此时，若采用“最短距离”的拆单方式，则 1 号订单的商品与 A 仓重合的部分由 A 仓配送，余下的部分由 B 仓送出（图 2(a)）；若采用“最小拆单数”的拆单方式，则 1 号订单直接归入 B 仓或 C 仓的配送范围内，再由后续的路径成本择仓配送（图 2(b)）。

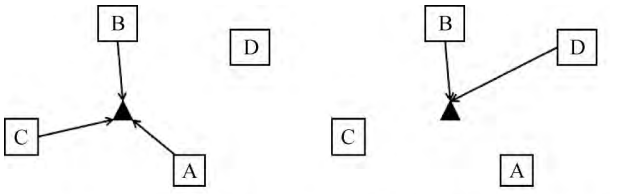


(a) 订单按“最短距离”拆分　(b) 订单按“最小拆单数”拆分

----→可选配送 ——→共同配送 □表示仓库 ▲表示被拆分订单

图 2　第一种情况拆分结果图示

第二种：设 1 号订单的商品无法由 A、B、C、D 任何一个仓库满足，但可以由 A、B、C 三个仓库组合以及 B、D 仓库组合来满足。此时，若采用“最短距离”的拆单方式，1 号订单会被拆分成三个包裹：先找到与 A 重合的部分并由 A 仓配送，再找到与 B 仓重合的部分由 B 仓配送，最后剩余部分由 C 仓配送（图 3(a)）；若采用“最小拆单数”的拆单方式，1 号订单被拆分成两个包裹，分别由 B、D 仓来配送（图 3(b)）。



(a) 订单按“最短距离”拆分　(b) 订单按“最小拆单数”拆分

——→共同配送 □表示仓库 ▲表示被拆分订单

图 3　第二种情况拆分结果图示

综上，以“最小拆单数”作为订单拆分的原则，能够有效减少订单拆分后的包裹数，进而减少客户的取件次数，提高配送满意度，对多品类多数量的订单尤其有效。

（二）模型建立

1.模型假设

a）仓库的数量及位置已知，每个客户点的位置已知，订单上的商品及数量已知，且需求量能被配送中心满足；

b）调度中心的车辆能满足最大配送要求，车辆类型相同；

c）每辆车所装载的商品数量不能超过最大限制量；

d）每条配送路径的长度不超过车辆的最大限制距离；

e)每辆车具有相应的工作时间;

f)每个需求点的需求都可以被拆分,但其时间窗不改变;

g)每个订单都允许被拆分成若干个子订单;

h)每个订单允许被多辆车服务,但每辆车只能服务一次。

2.符号说明

$E=\{e_1,e_2,\cdots,e_h\}$:电商平台的商品种类;

$W=\{1,2,\cdots,N\}$:仓库编号;

$O=\{1,2,\cdots,P\}$:订单编号;

S_w^r:第 w 个仓库中第 r 种商品的供给量;

Q_i^r:第 i 个订单中第 r 种商品的需求量;

$K=\{1,2,\cdots,k\}$:车辆集合;

L:车辆的最大行驶距离;

V:车辆的最大装载量;

λ:车辆行驶单位距离的成本消耗;

d_{ij}:点 i 到点 j 的距离,$d_{ji}=d_{ij}$;

t_0:司机法定工作时间;

$[a_i,b_i]$:第 i 个订单的时间窗;

T_{wk}:第 k 辆车从第 w 个仓库出发的时间;

v:车辆的平均速度;

α:车辆早到的等待成本系数;

β:车辆迟到的惩罚成本系数;

C:一个充分大的数;

π:车辆的启动成本。

3.决策变量

D_w^{ri}:表示第 w 个仓库分配给第 i 个订单的第 r 种商品的数量;

$$y_{ijk}=\begin{cases}1,\text{第 } k \text{ 辆车直接从第 } i \text{ 个节点驶向第 } j \text{ 个节点}\\0,\text{其他}\end{cases};$$

$$y_k=\begin{cases}1,\text{第 } k \text{ 辆车被使用}\\0,\text{其他}\end{cases};$$

T_{ik}:表示第 k 辆车到达第 i 个节点的时刻。

4.建立以配送成本最小化为目标,车辆装载限制、时间窗等为约束条件的数学规划模型:

$$\min z=\sum_{i\in O\cup W}\sum_{j\in O\cup W}\sum_{k\in K}(\pi y_k+\lambda d_{ij}y_{ijk})+\sum_{k\in K}\{\alpha\max[(a_i-T_{ik}),0]+\beta\max[(T_{ik}-b_i),0]\} \tag{1}$$

s.t.

$$\sum_{w\in W}D_w^{ri}=Q_r^i,\forall i\in O,\forall r\in E \tag{2}$$

$$\sum_{i\in O\cup W}\sum_{k\in K}y_{ijk}\geqslant 1,\forall j\in O \tag{3}$$

$$\sum_{w\in W}\sum_{j\in O}\sum_{r\in E}D_w^{rj}y_{wjk}\leqslant V,\forall k\in K \tag{4}$$

$$\sum_{i\in O\cup W}\sum_{j\in O\cup W}d_{ij}y_{ijk}\leqslant L,\forall k\in K \tag{5}$$

$$\sum_{i\in O}\sum_{r\in E}Q_r^i\leqslant V\sum_{k\in K}y_k \tag{6}$$

$$\sum_{i\in O}D_w^{ri}\leqslant G_w^r,1\leqslant r\leqslant h,\forall w\in W \tag{7}$$

$$\sum_{i\in O\cup W}\sum_{k\in K}y_{ijk}-\sum_{i\in O\cup W}\sum_{k\in K}y_{jik}=0,\forall j\in O \tag{8}$$

$$T_{jk}\geqslant T_{ik}+wt_{ik}+\frac{d_{ij}}{v}-C(1-y_{ijk}),\quad\forall i\in O\cup W,\forall j\in O,\forall k\in K \tag{9}$$

$$wt_{ik}=\max[(a_i-T_{ik}),0],\forall i\in O,\forall k\in K \tag{10}$$

$$\sum_{j\in O}T_{jk}\leqslant t_0,\forall k\in K \tag{11}$$

$$y_{ijk}(T_{ik}+wt_{ik}-T_{jk})\leqslant 0,\forall i,j\in O,\forall k\in K \tag{12}$$

$$\sum_{i\in O\cup W}y_{ijk}\leqslant 1,\forall j\in O,k\in K \tag{13}$$

$$a_j\leqslant T_{jk}\leqslant b_j,\forall j\in O,k\in K \tag{14}$$

$$y_{ijk}\in\{0,1\},\forall i,j\in W\cup O,k\in K \tag{15}$$

式(1)为多配送中心配送路径的最小化;

式(2)表示每个订单的各种商品需求都能被满足;

式(3)表示每一个订单都被派送到且可分割;

式(4)—(5)表示车辆装载量与行驶距离不允许超过最大限制;

式(6)表示被选中的车辆总运力能将所有的需求配送完毕;

式(7)表示该地区所有的需求能被满足;

式(8)表示车辆的进出平衡约束;

式(9)表示车辆 k 到达第 j 个需求点的时刻;

式(10)表示第 k 辆车在第 i 个需求点的等待时间;

式(11)为限制车辆的工作时间为法定工作时间;

式(12)表示车辆 k 服务完第 i 个点后直接驶向第 j 个需求点;

式(13)表示同一辆车对同一个需求点至多只能访问一次;

式(14)表示到达需求点 j 的时间应在时间窗范围内;

式(15)表示 y_{ijk} 为该模型的决策变量,当车辆 k 在配送完订单 i 后配送订单 j 则为1,否则为0。

(三)小规模精确求解

对上述优化模型进行精确求解,采用 Lingo11

求解器，对 9 种商品、4 个订单、3 个仓库进行求解，用时 2 分钟左右，求得仓库与订单之间的匹配关系如表 2 所示。小规模求解的仓库配送路径如表 3 所示。

表 2　仓库与订单之间的匹配关系

仓库编号	订单 1	订单 2	订单 3	订单 4
仓库 1	1	0	1	0
仓库 2	0	0	1	1
仓库 3	1	1	0	1

表 3　仓库的路径

仓库编号	仓库的路径
仓库 1	仓库 1 → 客户 3 → 客户 1
仓库 2	仓库 2 → 客户 3 → 客户 4
仓库 3	仓库 3 → 客户 1 → 客户 4 → 客户 2

进一步扩大求解规模，增加节点数时发现，当订单数量提高到 12 个，商品种类增加到 14 种，仓库数量达到 4 个后，Lingo 运行耗时过长，求解速度慢。因此，应考虑采用启发式算法对模型进行大规模求解，以适应问题的规模。下文对上述模型进行了分割，采用缩小解空间的思想，将模型分为订单拆分与物流配送两个部分，并设计两阶段法求解。

二、订单拆分的算法研究

(一)订单拆分规则的研究

需求可拆分的多配送中心车辆路径问题是 NP 难问题。由于需求可拆分的特性使得求解空间比一般的多 TSP 问题更复杂，优化难度更大。因此，学者们通常对这类问题采用多阶段求解的方式，将复杂的求解空间化简，缩小可行解的搜索范围[20]。由于订单履约环节包括订单的拆分与分配、订单配送路径优化两个部分，且两部分相互影响，故本文拟采用两阶段法的思想，将整个求解流程分为两个阶段进行。将订单拆分及匹配视为第一阶段，首先求得每个订单与对应的仓库的匹配程度，然后依据匹配程度选择仓库集合元素数量最少的组合对订单进行配送；将路径规划视为第二阶段，第一阶段得到的所有仓库集合均需遍历该订单，形成多条可选配送路径并计算路径成本。最后，以配送成本作为选择最优路径的依据，确定最终的配送路径。由于订单拆分及匹配过程涉及到订单内容与仓库商品的匹配程度，而路径规划属于车辆路径问题，因此，本文采用集合覆盖思想求解第一阶段——即求解每个订单的可选配送仓库集合，对订单的商品品类及数量与仓库进行匹配，形成匹配程度表，以此作为后续路径选择的依据之一；将订单配送作为第二阶段的路径规划问题，对第一阶段产生的各种匹配方案采用改进的蚁群算法求解，以成本最优为求解目标遍历每个匹配方案，将可选的匹配方案作为蚁群算法初始解集，在此基础上进行迭代，求得最优成本的路径即为最终路径。

(二)最小集合覆盖思想求解匹配程度

集合覆盖问题(Set covering problem，SCP)是一个经典的组合优化问题，在组合数学和计算机复杂理论中最为常见。它的主要思想是从某个集合中找到能够覆盖它的子集，再从这些子集中找出满足条件的点。集合覆盖问题主要分为经典最小集合覆盖问题、最小加权集合覆盖问题、极小碰集问题以及最大集合 K 覆盖问题。本文所要求解的匹配关系是以最少的仓库数量覆盖每一个订单，将订单总集合视为原集合，各个仓库集合作为订单集合的子集集合簇。因此，将经典最小集合覆盖问题作为模型第一阶段的求解最为适宜。计算每个订单的最小集合覆盖子集数目，找出每个订单的最小集合覆盖问题的相应仓库集合，用 0－1 变量表示订单与仓库的匹配关系，并将所有的 0－1 变量汇总，形成 0－1 矩阵，称之为匹配关系矩阵。

集合覆盖问题的模型如下。

符号说明：

$$f_w^i=\begin{cases}1,\text{仓库 } w \text{ 与订单 } i \text{ 有商品重合时}\\0,\text{其他}\end{cases};$$

$$x_w^i=\begin{cases}1,\text{仓库 } w \text{ 是订单 } i \text{ 的覆盖子集}\\0,\text{其他}\end{cases};$$

模型建立：

$$\min\sum_{i\in O}\sum_{w\in W}x_w^i \tag{16}$$

s.t.

$$\sum_{w\in W}\sum_{i\in O}x_w^i\leqslant\sum_{w\in W}\sum_{i\in O}f_w^i \tag{17}$$

$$\sum_{w\in W}x_w^i*s_w^r\geqslant Q_i^r \tag{18}$$

$$\sum_{w\in W}x_w^i\geqslant 1,\forall\, i\in O \tag{19}$$

$$x_w^i,f_w^i\in\{0,1\},\forall\, i\in O,w\in W \tag{20}$$

式(16)是优化后的目标，表示覆盖订单的仓库数最少，这样订单拆分时就达到了“最小拆单数”的效果；

式(17)表示的是只有当仓库与订单中的商品有交集的时候，该仓库才有可能覆盖该订单；

式(18)表示集合覆盖中被选中的子集合仓库必须要能满足订单商品的需求量；

式(19)是指任意一个订单至少被一个仓库覆盖,即每个订单都必须被覆盖完成;

式(20)是对0—1变量范围的约束。

该模型是上述优化模型被拆分后的第一阶段,旨在缩小求解空间,即在每个订单的集合覆盖范围内去寻找与之匹配度最高的仓库,在后续的路径规划模型中作为备选仓库计算可行路径。

(三)物流配送的路径生成

以下模型是在上述集合覆盖模型的基础上生成的物流配送模型,其主要的功能是对集合覆盖模型中产生的备选配送路径以成本最小化的目标进行优化处理,生成最终匹配结果与配送路径。

$$\min z=\sum_{i\in O_w\cup w}\sum_{j\in O_w\cup w,j\neq i}\sum_{k\in K}(\pi y_k+\lambda d_{ij}y_{ijk})+\sum_{i\in O}\sum_{k\in K}\{\alpha\max[(a_i-T_{ik}),0]+\beta\max[(T_{ik}-b_i),0]\} \tag{21}$$

s.t.

$$W_i=\bigcup_w\{w\mid x_w^i=1\},\forall i\in O \tag{22}$$

$$O_w=\{i\mid x_w^i=1\},\forall w\in W \tag{23}$$

$$\sum_{w\in W_i}D_w^{ri}=Q_r^i,\forall i\in O,r\in E \tag{24}$$

$$\sum_{i\in O_w\cup w}\sum_{k\in K}y_{ijk}\geqslant 1,\forall j\in O_w,i\neq j \tag{25}$$

$$\sum_{w\in W}\sum_{r\in E}\sum_{i\in w\cup O_w}\sum_{j\in O_w}D_w^{rj}y_{ijk}\leqslant V,\forall k\in K \tag{26}$$

$$\sum_{i\in O_w\cup w}\sum_{j\in O_w\cup w}d_{ij}y_{ijk}\leqslant L,\forall k\in K \tag{27}$$

$$\sum_{i\in O}\sum_{r\in E}Q_r^i\leqslant V\sum_{k\in K}y_k \tag{28}$$

$$\sum_{i\in O_w\cup w}\sum_{k\in K}y_{ihk}-\sum_{j\in O_w\cup w}\sum_{k\in K}y_{hjk}=0,\forall h\in O_w \tag{29}$$

$$\sum_{i\in O_w}\sum_{k\in K}y_{wik}-\sum_{j\in O_w}\sum_{k\in K}y_{jwk}=0,\forall w\in W \tag{30}$$

$$T_{jk}\geqslant T_{ik}+wt_{ik}+\frac{d_{ij}}{v}-M(1-y_{ijk}),\quad\forall i\in O\cup W,\forall j\in O,\forall k\in K \tag{31}$$

$$\sum_{j\in O}T_{jk}\leqslant t_0,\forall k\in K \tag{32}$$

$$y_{ijk}(T_{ik}+wt_{ik}-T_{jk})\leqslant 0,\forall i,j\in O_w,k\in K \tag{33}$$

$$wt_{ik}=\begin{cases}a_i-T_{ik},T_{ik}\leqslant a_i\\ o,T_{ik}>a_i\end{cases},\forall i\in O,\forall k\in K \tag{34}$$

$$\sum_{i\in O\cup W}y_{ijk}\leqslant 1,\forall j\in O,k\in K \tag{35}$$

$$a_j\leqslant T_{jk}\leqslant b_j,\forall j\in O,k\in K \tag{36}$$

$$y_{ijk}\in\{0,1\},\forall i,j\in w\cup O_w,k\in K \tag{37}$$

式(21)表示配送成本的最小化;

式(22)表示对每一个订单均从匹配程度表上选取可为其服务的仓库点,作为备选仓库点,由路径成本决定由哪些仓库配送;

式(23)表示匹配程度表上对每个仓库均有相应的订单待服务,并且由路径成本决定每个仓库服务的订单,形成仓库配送路径。

(四)算法描述

传统的蚁群算法因其自身特性具有收敛速度慢且易陷入局部最优的弱点,因此本文采用改进的蚁群算法进行求解,并利用局部和全局搜索混合的搜索方法改进求解结果,使其收敛于较优的解。首先,采用改进的Solomon插入法——时差法[20]选取初始解,以配送时间窗来确定插入的订单及位置,形成较好的初始解;然后,对算法前半段进行全局信息素更新,使其加快收敛;最后,在算法后半段进行信息素局部更新并对最优解进行局部邻域搜索,以防止算法停滞。时差法插入流程如图4所示。

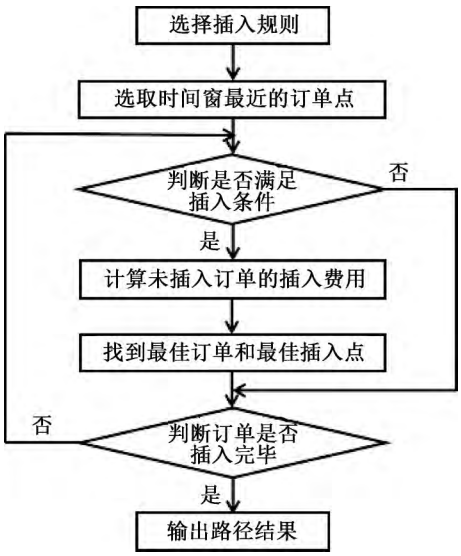


图4 时差法设置初始解流程

算法具体执行以下步骤:

Step1:初始化,令$i=1$,$w=1$,$NC=1$,$m=1$,并设置蚂蚁数量M、指定迭代次数RT以及最大的迭代次数NC_MAX。

Step2:验证i是否超过总订单个数,是,转Step5;否,则执行集合覆盖,令$R_w^i=\dfrac{x_w^i}{\sum_{w\in W}x_w^i}$作为仓库对订单的覆盖率,若$R_w^i=1$,将$i$与$w$置于集合$set_i$,并从订单集合中删去$i$,更新$w$,转至Step4;若$0<R_w^i<1$,将$i$与$w$置于集合$set_i$,并更新$w$与$i$,转Step3;若$R_w^i=0$,记$w=w+1$,转Step2。

Step3：记 $w=w+1$，检验 w 是否超过总仓库个数，是，跳转 Step4，否，求 R_w^i；若 $R_w^i=1$，将 w 增至集合 set_i，删去 i，更新 w，转至 Step5；若 $0<R_w^i<1$，将 w 增至集合 set_i，并更新 w 与 i，转 Step3；若 $R_w^i=0$，直接转 Step3。

Step4：记 $i=i+1$。

Step5：将 set_i 中的数据化为 0－1 量一一对应写入 w 行 i 列的零矩阵中，抽取每一行元素作为新的集合 $route_w$。

Step6：将 M 只蚂蚁随机放置在 w 个仓库点上，同时建立一张空的禁忌表。

Step7：找出 $route_w$ 中订单的时间窗下限，将该订单作为第一个需求点优先配送，将 $route_w$ 中剩余的订单根据费用函数(38)选择最优订单并寻找最优位置插入。

$$\begin{cases} c_1(i,h,j)=\alpha c_{11}(i,h,j)+\beta c_{12}(i,h,j),\alpha+\beta=1 \\ c_2(i,h,j)=\lambda d_{sh}-c_1(i,h,j),\lambda\geqslant 0 \end{cases} \tag{38}$$

其中：

$$\begin{cases} c_{11}(i,h,j)=d_{ih}+d_{hj}-d_{ij} \\ c_{12}(i,h,j)=b'_j-b_j \end{cases} \tag{39}$$

$c_1(i,h,j)$ 表示将订单 h 插入到 (i,j) 之间所产生的成本，$c_2(i,h,j)$ 表示比单独派车配送订单 h 所节约的成本；$c_{11}(i,h,j)$ 表示插入订单 h 后行驶距离的叠加值，$c_{12}(i,h,j)$ 表示插入订单后对时间差产生的影响值；α,β,λ 为权重参数。利用该费用函数计算出最佳插入订单，并找到最佳插入点，当满足插入条件(车辆装载、时间窗限制等)时，将该订单插入该最佳插入位置，构成初始路径。

Step8：将该订单作为第一个需求点优先配送，并更新禁忌表，按照如下转移概率选择下一个要访问的点，τ_{ij} 为信息素浓度，η_{ij} 为启发式信息，$route_w^m$ 为蚂蚁 m 下一步可以访问的订单集合，lt_{ij} 为迟到时间，wt_{ij} 为等待时间。然后根据信息素浓度与等待时间按以下概率选择依次要访问的订单：

$$j=\begin{cases} \{\operatorname{argmax}_{j\in route_w^m}[\tau_{ij}]^\alpha[\eta_{ij}]^\beta[1/lt_{ij}]^\delta\},q\leqslant q_0 \\ (42),\text{其他} \end{cases} \tag{40}$$

其中：

$$\eta_{ij}=\begin{cases} \dfrac{1}{wt_j}\cdot\dfrac{1}{d_{ij}},wt_j=\max(a_j-T_i-t_{ij}) \\ \dfrac{1}{d_{ij}},wt_j=0 \end{cases} \tag{41}$$

采用轮盘赌时的状态转移概率为：

$$p_{ij}^m=\begin{cases} \dfrac{\tau_{ij}^\alpha\eta_{ij}^\beta[1/lt_{ij}]^\delta}{\sum_{u\in route_w^m}\tau_{ij}^\alpha\eta_{ij}^\beta[1/lt_{ij}]^\delta},j\in route_w^m \\ 0,otherwise \end{cases} \tag{42}$$

其中：

$$lt_{ij}=\min(b_j-T_j,0) \tag{43}$$

Step9：按照上述概率选定下个访问的订单，并将其加入禁忌表，返回 Step7，直至所有的订单全部访问完成。

Step10：记录路径与时刻，并对信息素进行全局更新、时间窗和订单剩余内容，记 $NC=NC+1$，判断 NC 是否小于指定迭代次数 RT，是，转 Step7，进行下一次迭代；否，转 Step11。

Step11：从当前路径中挑选最优的解进行局部更新信息素，更新规则为：

$$\tau_{ij}(NC+1)=(1-\rho)\tau_{ij}(NC)+\rho\Delta(i,j),\quad \rho\in(0,1) \tag{44}$$

Step12：令 $NC=NC+1$，验证 NC 是否小于最大迭代次数 NC_MAX，是，转 Step10，并验证最优解是否更新，若未更新，则变换邻域搜索规则，再次搜索，直到更新最优解或达到最大搜索次数；否，输出结果，结束算法。

算法流程见图 5。

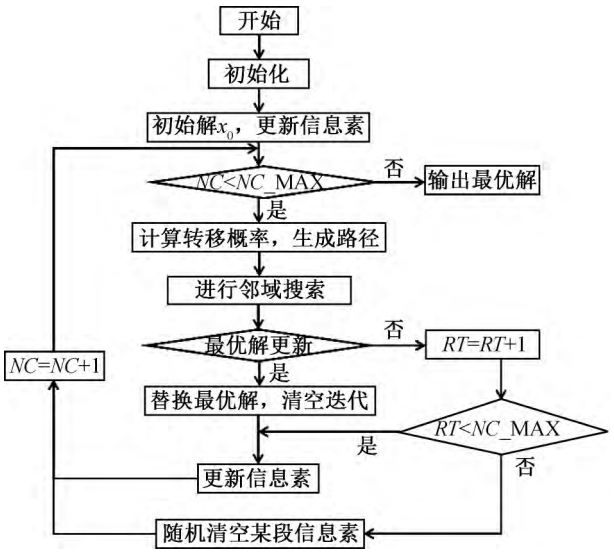


图 5　算法流程图

三、案例仿真

(一)案例数据选取

本文以电商平台 Y 为例，对其在华东某区的仓库数据内容作随机处理。已知平台 Y 在该区域有 1 个物流配送中心、3 个前置仓这 4 个仓库及其坐标位

置。现假设有 500 个订单需要当天配送,而无法被就近的仓库满足的订单占 20%,即有 100 个订单需要拆分。本文研究的对象是因现有的以"最短距离"为拆单原则而产生订单拆分与分配的情况,故对需要拆单的 100 个订单具体考虑,对其余不需要拆单的订单按就近仓库分配,其订单内容不计入仓库库存。随机生成仓库与订单的商品种类(15 种以内)与数量(0～10 件),且订单上的其他内容如地理信息和时间窗信息等也随机生成。每个仓库的车辆为同型车辆,且车辆数充足。假设车辆每行驶一公里所产生的行车费用为 0.8 y,单次行驶距离的限制为 100 km,城区平均行驶速度为 30 km/h,每辆车的司机每天累计工作 8 h。由于订单最多能生成 15×10 个商品,故假设订单均为长订单,并将车辆的载重限制假定为 150 个订单。令车辆早于时间窗到达而产生的等待成本系数为 50 y/h,晚于时间窗到达产生的惩罚成本系数为 80 y/h。以 15 个订单为例,最小集合覆盖求得的各 $route_w$ 如表 4 所示。

表 4 15 个订单的集合覆盖情况

$Route_w$	备选配送路径的订单集合
$Route_1$	(3,8,9,12,14)
$Route_2$	(1,4,5,6,10,15)(1,5,6,10,15)(1,5,10,15)(1,4,5,10) (1,4,5,10,15)(1,4,5,6,10)(1,4,6,10)(1,4,6,10,15)
$Route_3$	(2,4,6,9,11)(2,4,9)(2,4,6,11)(2,4,9,11)
$Route_4$	(6,7,9,11,13,15)(7,9,11,13,15)(7,11,13,15)(7,11,15) (6,7,11,13,15)(6,7,9,11,13)(6,7,11,13)(6,7,9,11,15)

(二)求解结果

由最小集合覆盖思想得到与每个订单相匹配的仓库集合,其中有部分订单是由两个及以上的仓库共同配送的,因此这类订单具有多种匹配关系。此时,考虑各种仓库组合与订单之间的路径成本不同以及时间窗的限制等因素,以配送成本最小化的角度选择相应的仓库组合对其进行配送。通过执行上述改进的蚁群算法步骤,得到改进蚁群算法的迭代进化与传统蚁群算法的区别(图 6)。改进的蚁群算法在迭代初期的收敛速度比传统蚁群算法快,在迭代次数达到 80 次后,采用邻域搜索策略进行局部最优解的更新,使算法寻优能力更强,且于迭代次数为 150 次左右趋向收敛。

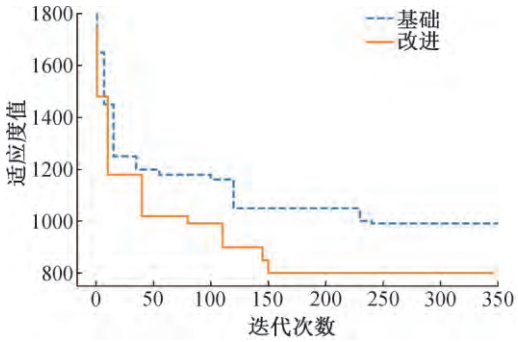


图 6 蚁群算法迭代过程图

表 5 为求得的最终配送路径,每个仓库具有相应待配送的客户点(订单编号),由表 5 可知配送到每个客户点的时刻以及仓库的配送费用。

表 5 最终配送路径

仓库编号	路径	成本/元	到达时间
$w=1$(配送中心)	w(1)→12→29→41→3→14→8→81 w(1)→42→73→62→20→17→19	67.6	7:00→8:35→9:17→9:40→11:20→13:35→14:47 10:40→12:05→13:21→14:10→14:44→15:26
	w(2)→23→18→4→77→56→88→5→36→51→93→67		7:00→7:42→8:37→9:23→10:50→13:20→14:45→15:10→15:48→16:50→17:30→18:13
$w=2$	w(2)→11→49→27→38→76→80→61→35→32	101.9	7:20→8:11→8:46→9:18→9:55→10:30→11:25→13:14→14:22→15:07
	w(2)→6→15→30→54→75→45→43→69→1→10→34→63→50→48→71→66		8:00→8:26→8:53→9:23→9:55→10:35→11:20→12:30→13:00→13:47→14:35→15:05→15:40→16:15→16:58→17:55
$w=3$	w(3)→11→7→28→64→31→44→58→74→82→95→91→79→60→22	89.7	6:40→6:59→7:33→8:23→8:53→9:25→9:59→10:26→10:47→11:00→11:38→12:25→13:05→13:44→14:36
	w(3)→78→86→52→33→16→87→56→94→98→72→13		8:10→8:45→9:54→10:21→10:55→11:24→12:35→12:57→13:20→13:48→14:25→14:57
$w=4$	w(4)→11→90→47→100→39→49→83→9→2→24	124.8	6:30→7:10→7:43→8:26→8:55→9:37→9:59→10:28→10:53→11:33
	w(4)→6→70→25→21→37→46→85→53→40→97→84		7:20→8:15→8:53→9:42→10:11→10:56→11:45→12:15→12:47→13:20→14:25
	w(4)→92→65→38→55→57→68→99→89→96→59→76→26		9:00→9:35→10:18→10:46→11:29→12:03→12:24→13:09→13:47→14:00→14:49→15:25

通过计算得出，该订单配送的路径总成本为 392 元，其中等待成本为 31.7 元，等待总时间为 38 分钟；惩罚成本为 24 元，迟到总时间为 18 分钟。上述给出了 100 个待拆单的算例，通过改变拆单规则，由其他仓库接受完整订单，使得拆单数变少，最终只剩下不到 3%的订单数需要拆分，且拆单的包裹数均在 2 个以内，大大减少了拆单后的包裹数量。本文所运用的两阶段法对该算例进行了验证，结果表明该算法对适应新拆单规则的算例有效。

（三）两种拆单方式的对比

为了突出两种拆单原则对拆单及匹配结果的影响，下文采用相同的仿真数据对“最短距离”拆单原则与“最小拆单数”拆单原则在包裹数量、派车成本、惩罚成本、等待成本和路径运输成本五个方面进行对比，结果如图 7 所示。通过对比可知，以“最小拆单数”为拆单原则不仅能大幅减少拆单数量，更能降低成本，尤以惩罚成本和等待成本为主，表明在改变拆单原则后，订单的履约效率得到大幅提升。

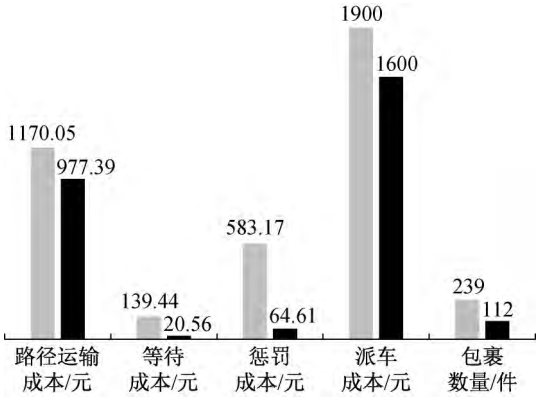


图 7　两种拆单方式的最终路径成本结果对比

四、结论与展望

本文根据电子商务订单的特性，对订单的拆单方式进行研究。首先提出了以“最小拆单数”为原则的拆单方式，对大型订单进行拆分，利用经典的最小集合覆盖模型建立仓库与订单之间的匹配关系，并以最小配送成本为目标建立配送路径规划的模型确定匹配。然后设计了两阶段法进行求解，引入缩小求解空间的思想，采用改进的蚁群算法在该求解空间内寻找最优路径。最后采用仿真算例验证了模型与算法的有效性和可行性。

本文基于客户满意度的视角，提出优先考虑减少客户取件次数的拆单方式，将降低物流配送成本作为次要目标，这符合现阶段电商公司为增加用户粘性而着力提高客户满意度的要求。本文针对物流配送这一环节，仅考虑了客户取件次数与部分包裹延迟配送导致违背时间窗的情况。但在现实中，一方面，对客户满意度的度量应是多元化的，取件次数与送货时间窗只是量化该指标的一部分，因此如何考虑对客户满意度进行更丰富的指标量化是今后研究方向之一，如个性化拆单等。另一方面，由于集合覆盖是一个 NP 完全问题，该两阶段模型至少是 NP 完全问题，其求解难度随着规模的扩大呈指数型增长，尤其是电子商务订单的爆炸式增长特别依赖快速算法，故应对大规模数据的算法也是下一步可研究的方向。

参考文献：

[1] Wang X, Golden B, Wasil E, et al. The min－max split delivery multi-depot vehicle routing problem with minimum service time requirement[J]. Computers & Operations Research, 2016, 71:110-126.

[2] 郎茂祥.多配送中心车辆调度问题的模型与算法研究[J].交通运输系统工程与信息,2006,6(5):65-69.

[3] 刘新宇,符卓,邱萌.需求可拆分车辆路径问题研究:文献综述[J]. 技术经济,2017,36(1):96-109.

[4] 殷脂,叶春明.多配送中心物流配送车辆调度问题的分层算法模型[J].系统管理学报,2014,23(4):602-606.

[5] 马宇红,姚婷婷,张浩庆. 基于分区的多配送中心多车型车辆调度问题与遗传算法设计[J]. 科技导报,2013,31(2):61-67.

[6] 肖正中,谭建,周玉峰,等.跨区域多配送中心车辆调度智能优化研究[J]. 中国烟草学报,2017,23(4):114-120.

[7] 葛显龙,王旭,邓蕾.基于联合配送的开放式动态车辆路径问题及算法研究[J]. 管理工程学报,2013,27(3):60-68.

[8] 邢鹏.基于云平台的多配送中心车辆调度问题研究[D]. 北京:北京交通大学,2013.

[9] 金涛.多配送中心物流车辆调度的改进差分进化算法[J]. 计算机工程与应用,2014,50(3):232-235.

[10] 杨鹏,邹浩,徐贤浩.带时间窗集送货需求可分车辆路径问题的改进蚁群算法[J]. 系统工程,2015,33(9):58-62.

[11] Tang J, Ma Y, Guan J, et al. A Max－Min Ant System for the split delivery weighted vehicle routing problem[J]. Expert Systems with Applications, 2013, 40(18):7468-7477.

[12] Silva M M, Subramanian A, Ochi L S. An iterated local search heuristic for the split delivery vehicle routing problem [J]. Computers & Operations Research, 2015, 53: 234-249.

[13] 符卓,刘文,邱萌. 带软时间窗的需求依订单拆分车辆

路径问题及其禁忌搜索算法[J]. **中国管理科学**,2017,25(5):78-86.

[14] Archetti C, Bianchessi N, Speranza M G. A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem[J]. Computers & Operations Research, 2015, 64: 1-10.

[15] Han F W, Chu Y C. A multi-start heuristic approach for the split-delivery vehicle routing problem with minimum delivery amounts [J]. Transportation Research Part E: Logistics and Transportation Review, 2016, 88: 11-31.

[16] Jin M Z, Liu K, Bowden R O B. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem[J]. International Journal of Production Economics, 2007, 105(1): 228-242.

[17] Gulczynski D, Golden B, Wasil E. The split delivery vehicle routing problem with minimum delivery amounts [J]. Transportation Research Part E: Logistics and Transportation Review, 2010, 46(5): 612-626.

[18] **张源凯,黄敏芳,胡培祥.网上超市订单分配与物流配送联合优化方法**[J]. **系统工程学报**, 2015, 30(2): 251-258.

[19] **王继强.集合覆盖问题的模型与算法**[J]. **计算机工程与应用**,2013, 49(17):15-17.

[20] **熊浩,鄢慧丽. 需求可拆分车辆路径问题的三阶段禁忌算法**[J]. **系统工程理论与实践**,2015,35(5):1230-1235.

(责任编辑:陈丽琼)