最小化拆单率的在线零售商多仓商品摆放优化策略研究

李建斌，李乐乐，黄日环
（华中科技大学管理学院，武汉 430074）

摘要：拆单是指一张订单需要从多个仓库独立进行配送，或由一个仓库在不同时间分别配送，而这会额外增加商家的物流成本。本文对在线零售商多个仓库的商品摆放策略进行了研究，目的是降低由于商品摆放策略不同所引起的拆单。文中考虑了总体不缺货情况下，基于品类和数量的两类拆单情况，并建立数学模型，改进了热销品算法（Bestseller）得到了环形算法（Loop），对相应仿真问题进行求解，并与原热销品算法进行对比。结果表明环形算法能够在兼顾各配送中心负载的同时进一步降低品类拆单率，进而降低总体拆单率。备货量的分析表明，在线零售商可以通过增加备货量来降低甚至消除数量拆单的影响。对品类约束的研究表明，在单仓商品品类较少时，品类拆单对总的拆单率影响更大；单仓商品品类较多时，数量拆单影响更大。效率方面，由于环形算法不需要计算相关性矩阵，效率相比热销品算法要高。

关键词：在线零售；运营管理；拆单；启发式算法
中图分类号：F274; C931 **文献识别码：**A **文章编号：**1004-6062(2017)03-0167-007
DOI：10.13587/j.cnki.jieem.2017.03.021

0 引言

考虑一张含有多种品类（Stock Keeping Unit，SKU，最小存货单位），其中每种品类又含有多个数量（Unit）的订单，订单被下达后，执行系统会将订单分配给配送中心（Distribution Center，DC，本文将仓库与配送中心都作 DC）来完成配送。如果在同一时间没有一个配送中心含有订单所需的全部商品，或者某些 SKU 由于自有属性（如预售产品、产品需要上门服务等）无法在同一时间配送的，那么这张订单将会需要被拆分为多个订单，分别由不同的配送中心来完成配送，或由同一配送中心在不同时间段来完成配送。

拆单无疑会增加企业的运营成本、人工成本和配送成本等。在线零售业逐步成熟的今天，商家的竞争越来越激烈，消费者也对在线零售商提出了更多的要求，包括高质量、高性价比的商品，更优质的配送体验等。不仅如此，随着消费者个性化需求的日益走高，急速配送、指定时间段配送等，都对在线零售商的库存配置及配送能力提出了巨大挑战。Xu[1]指出，产品能否及时交付，已经成为消费者衡量在线零售商的一个因素。从供应链的角度来看，配送作为在线购物的最后一环，显得尤为重要，全方位提高顾客配送环节的体验需要高额的配送成本，而这也成为影响在线零售商发展的一个重要因素。

报告指出[2]，亚马逊 2013 年的运输费用为 66 亿美元，占总运营成本的 77.3%，是当年净利润的 24.2 倍，而亚马逊 2011 的配送成本仅为当年净利润的 6.3 倍[3]，配送成本的增幅远远高于利润的增长幅度。Xu[4]等人分析了一家大型在线零售商的订单，结果发现该企业在配送中分拆了订单量的 10%到 15%，由此增加了 3.75%到 6%的配送量。对于运输成本占据过半运营成本的在线零售商而言，如果 2013 年的亚马逊也有如此高的拆单率，在假设每张订单运输费用相同的前提下，订单被拆分所产生的额外配送成本为 2.6 亿到 4.2 亿美元，这些额外的配送成本相当于当年净利润的 96%到 156%。如果能够降低这部分额外配送成本，无疑会大幅提高在线零售商的利润率，进一步降低商品售价，增强企业竞争力。因此，对拆单问题进行研究显得极为重要。对国内某在线零售商的订单分析发现，该在线零售商配送的多于 80%的包裹重量低于 4.5 千克（10 磅）。对于 10 磅以下的包裹，在线零售商可以通过减少配送次数来降低运输成本[4]，进而降低运营成本。

订单被拆分一般有三个因素。一是 SKU 在配送中心中是如何摆放的（品类拆单，下同）：如果一个配送中心不能够容纳所有的 SKU，那么对于一张含有多个 SKU 的订单，就有可能需要被拆分。二是配送中心中 SKU 的当前库存水平（数量拆单，下同）：尽管某配送中心中包含了订单所需全部的 SKU，但如果其中某个 SKU 当前缺货，或者当前库存量不及订单的需求量，那么这张订单也将被拆分。除此之外，个别 SKU 会因为自有属性而导致拆单（标记拆单）：一旦订单中含有该类 SKU，则订单必须被拆单，如上文提到的预售产品、需要上门服务的产品等。但从运营管理的角度来看，前两个因素更值得关注。

尽管拆单对于在线零售商的利润影响非常大，但针对拆单问题的研究却很少。回顾相关文献发现，学者的研究内容主要集中在并行分区拣货系统的优化策略[5, 6]、配送中心选

收稿日期：2014-05-14 **修回日期：**2015-01-11
基金项目：国家自然科学基金资助项目（71171088、71131004）；中央高校基本科研业务专项基金资助项目（2014YQ002）； 湖北省科技计划软科学类项目（2014BDF112）；教育部新世纪优秀人才基金资助项目（NCET-13-0228）
作者简介：李建斌（1980—），男，江西上饶人；华中科技大学管理学院教授，博士生导师；研究方向：物流与供应链管理、库存控制与优化、电子商务。

址规划[7-9]、拣货路径优化[10, 11]等。Xu[4]考虑了订单执行系统的再分配问题，由于前期订单的分配结果会影响后续订单的分配，Xu 利用订单从下达到被满足之间的时间延迟来对订单进行集中再分配，有效降低了拆单率。

随着网上零售逐步成为主流，消费者需求多样性日益广泛，一站式购物理念的提出与追捧，也使得在线零售商持有的 SKU 品类与数量都呈现爆炸式增长，配送中心的面积越来越大，数量也越来越多。配送方面，消费者对配送时间的期望越来越短，这也无形中减少了在线零售商聚集订单的机会，增加了拆单的可能性。另一方面，订单数的增加，尤其是在线零售业中不定时的促销情况，也对配送中心的出库能力提出了挑战，这也促使了配送中心数量的增加，而配送中心数量的增加又不可避免地增加订单被拆分的可能性。如何在尽量降低拆单率的情况下将 SKU 放置在不同的配送中心中，成为在线零售商不得不面临的问题。

Catalan[12]在考虑品类拆单的情况下，建立了最小化拆单率的数学模型，结合一家在线零售商的历史数据，给出了相应的解决方案。文中没有考虑库存水平对拆单率的影响，然而实际中由于各种原因会导致单个配送中心某几类 SKU 缺货，此时无疑会产生第二种拆单情况，即数量拆单。考虑到实际操作中，所有配送中心均缺货时，系统会出现无法响应顾客订单的情况，即订单不会产生。因此，本文将缺货定义为单个配送中心缺货，但是全部的配送中心不缺货（即全部配送中心的库存可以满足需求）。并讨论零售商的 SKU 分配策略以及每个配送中心中每种 SKU 的最大库存量。

1 问题描述

如果将 SKU 视为产地，订单作为销地，配送中心作为中转站，则拆单问题可以视为一个带有中转站的运输问题。该运输问题包含两个阶段：第一阶段，在线零售商在不清楚订单状况的情况下对 SKU 进行分配（SKU 到配送中心的运输）；第二阶段，顾客下单后，执行系统将订单分配到不同的配送中心进行配送（配送中心到订单的运输）。这两个阶段的决策都会影响总的运输次数，如果订单状况随机，此问题即为一个二阶段二维随机优化模型。

为了简化问题规模，结合实际情况，做如下假设：

（a）在未来的某一段时间，在线零售商面临的环境不会出现剧烈变化；

（b）多个配送中心处于同一区域；

（c）本文不考虑第三类拆单，即标记拆单；

（d）对指定的 SKU，至少有一个配送中心来存放；

（e）单个配送中心无法存放 SKU 的全部品类及数量，但全部配送中心可以；

（f）单个配送中心可以缺货，但是全部配送中心不缺货；

（g）订单执行系统采用先到先服务的即时化决策。

假设 a 中，在零售商所面临的销售环境较为稳定的情况下，我们可以利用过往的交易数据来预测未来某一段时间内商品的需求情况。假设 b 中，就国内在线销售环境而言，不同区域之间的顾客消费习性差别较大，这就要求在线零售商只能从当地配送中心来满足顾客需求[12]；另一方面，同一区域配送中心的单张订单配送成本相差很小，此假设可将目标函数由降低运输成本转化为降低拆单率。

2 问题建模

约定如下符号：

I：SKU 品类总量，编号为 i；

J：订单总量，编号为 j；

N：配送中心总数，编号为 n；

K_n：配送中心 n 最多能存放的商品品类；

C_n：配送中心 n 最多能存放的商品数量；

R_{ij}：订单 j 对 SKU i 的需求量；

S_i：零售商根据自身库存管理策略决定的 SKU i 的最大库存水平；

M_j：订单 j 所含 SKU 品类的集合；

决策变量如下：

$$y_{in}=\begin{cases}1, \text{ SKU } i \text{ 存放在配送中心} n \text{ 中}\\ 0, \text{ 否则}\end{cases}$$

U_{inj}：配送中心 n 发出 U 个 SKU i 来满足订单 j；

$$z_{nj}=\begin{cases}1, \text{ 配送中心} n \text{ 发出针对订单} j \text{ 的配送}\\ 0, \text{ 否则}\end{cases}$$

Q_{in}：配送中心 n 中能存放 SKU i 的最大数量。

建立如下模型：

$$\text{Min} \sum_{n\in N}\sum_{j\in J} z_{nj}$$

s.t.

$$\sum_{n\in N} y_{in}\geq 1 \qquad \forall i \qquad (1)$$

$$\sum_{i\in I} y_{in}\leq K_n \qquad \forall n \qquad (2)$$

$$\sum_{i\in I} Q_{in}\leq C_n \qquad \forall n \qquad (3)$$

$$\sum_{j\in J} U_{inj}\leq Q_{in} \qquad \forall i,\forall n \qquad (4)$$

$$\sum_{n\in N} U_{inj}\geq R_{ij} \qquad \forall i\in M_j,\forall j \qquad (5)$$

$$0\leq U_{inj}\leq R_{ij} \qquad \forall i\in M_j,\forall j,\forall n \qquad (6)$$

$$\sum_{n\in N} Q_{in}=S_i \qquad \forall i,\forall n \qquad (7)$$

$$U_{inj}\leq (\sum_{j\in J} R_{ij})y_{in} \qquad \forall i\in M_j,\forall n,\forall j \qquad (8)$$

$$U_{inj}\leq (\sum_{j\in J} R_{ij})z_{nj} \qquad \forall i\in M_j,\forall n,\forall j \qquad (9)$$

$$y_{in},z_{nj}\in\{0,1\} \qquad \forall i,\forall n,\forall j \qquad (10)$$

约束（1）保证至少有 1 个配送中心 n 存放 SKU i；

约束（2）为品类约束，配送中心 n 所能存放的商品品

类最多为 K_n ；

约束（3）为数量约束，即配送中心 n 所能存放的商品数量最多为 C_n ；

约束（4）表明配送中心 n 对 SKU i 的配送量 U_{inj} 不得超过该配送中心中 SKU i 的最大存放量 Q_{in} ；

约束（5）保证所有配送中心对订单 j 中的 SKU i 的配送量之和可以满足其需求量 R_{ij} ；

约束（6）即单个配送中心 n 对订单 j 中 SKU i 的配送量 U_{inj} 不能超过其需求量 R_{ij} ，且需为非负整数；

约束（7）即所有配送中心对 SKU i 的最大存储量之和等于 SKU i 的最大库存水平 S_i ；

约束（8）为 y_{in} 的判定约束，只有当 SKU i 存放在配送中心 n 中时，其才可发出一定量的 SKU i 来满足订单 j ；

约束（9）为 z_{nj} 的判定约束，即配送中心 n 发出 SKU i 来满足订单 j 的前提是，该配送中心有针对订单 j 的配送发生。

定理 1：拆单问题为 NP-难问题。

证明：假如不考虑 SKU 的数量，即只考虑品类拆单情况，则模型简化为决策 SKU 在配送中心之间的摆放策略。考虑 SKU 总数 I 为偶数且每个订单只包含 2 种 SKU 的特殊情况，同时配送中心数量 N 为 2，SKU 容量相等 $k_1=k_2=I/2$。此时若把 SKU 看做顶点，则订单是连结两个顶点的边。此时简化后的模型可以用图表示，并可以转化为二分图问题，即将顶点集分割为两个互不相交的子集，目标是使跨越两个子集的边数最小。该问题已经被证明是 NP-难问题[13-15]。因此更为复杂的拆单问题也是 NP-难问题，难以用精确算法求解。

3 算法设计

订单 j 中所包含的 SKU i 从配送中心 n 来完成配送，需要两个条件：首先，配送中心 n 中需要存放有 SKU i ；其次，配送中心 n 中所存放的 SKU i 的数量可以满足订单 j 的需求量。如此，可以将模型分为两步来求解，第一步决定配送中心 n 所包含的 SKU 品类，第二步决定配送中心 n 中所含 SKU 的数量。在此之前，首先介绍数量拆单发生时，在线零售商为了满足顾客订单所采取的两种基本策略。

假如配送中心 n 中 SKU i 数量不足以满足订单 j ，此时订单需要被拆分，而此时的拆分通常有两种基本策略：其一（部分拆分策略，下同），将配送中心 n 中剩余的全部 SKU i 分配给订单 j ，同时不足的量交由另外的配送中心 m 来满足。其二（全部拆分策略，下同），配送中心 n 不配送订单 j 中的 SKU i ，全部交由另外的配送中心 m 来满足。这两种策略都需要额外增加一次配送，所不同的是：部分拆分策略会一定程度上影响配送中心 n 中 SKU 的多样性。对于全部拆分策略，考虑一张特殊的订单 j ，该订单需要 5 个 SKU i ，而每个配送中心中 SKU i 的剩余量都不足 5 个，但是全部配送中心的总和却可以满足该订单，如此会产生一个现象：即消费者在线可以下单，但是订单执行系统却无法执行该订单。实际情况中，我们认为顾客一般更倾向于一种商品经由一次配送，即全部拆分策略。且相较部分拆分策略，全部拆分策略可以减少配送中心 n 中一次拣货。出于后文的算法设计考虑，本文优先采用全部拆分策略，对库存不足的极端情况下产生的订单采用部分拆分策略。

设想另外一种拆单模型，配送中心只有数量约束而没有品类约束，这种情况下，只要订单执行系统做的足够好，总可以将每种 SKU 按照配送中心的剩余容量的比例来进行分配，这样可以得到一个质量相当好的解，假若备货量也充足，甚至能够让拆单率降到 0%。然而现在的情况是，每个配送中心品类约束和数量约束都有，同时，为了平衡仓库负载，这时在往配送中心中存放 SKU 时就必须同时考虑这两个因素。因此，有必要对订单结构进行分析。图 1 是对国内某大型在线零售商一段时间内某一目录商品的销量排序统计结果。

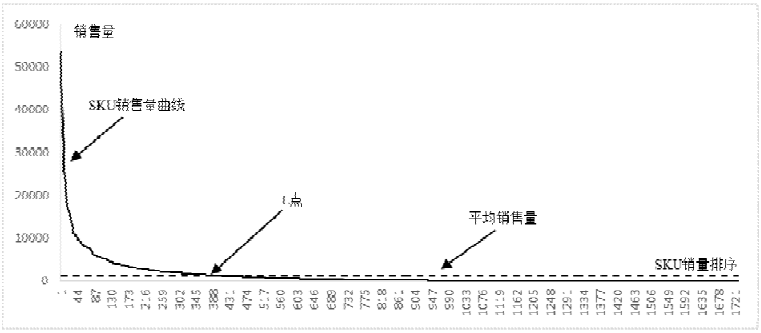

图 1 SKU 销量排序图

图 1 中可以看出，前面为数不多的 SKU 占据了绝大多数的销量，而后面大量的 SKU 销量十分少。进一步的统计发现，前 50%的 SKU 占据了销售量的 95%以上，而前 20%的 SKU 更是占据了销量的 80%左右。一个很简单的设想是，可以通过销量较大的 SKU 来平衡各仓库的负载，而剩余部分来存放其他 SKU，如此在尽量保证负载均衡的情况下来降低拆单率。Catalan 的热销品算法（Bestseller）[12]就体现了这种思想。热销品算法首先选出 B 个热销品，其中 $B=\left\lfloor \frac{\sum_{n\in N} K_n - I}{n-1} \right\rfloor$，将这些热销品分配到所有的配送中心中，如果某个配送中心的品类约束 K_n 小于 B，则分配前 K_n 个热销品，剩余 SKU 则通过相关性矩阵 C 来分配到平均相关性最高的配送中心。其中 $C=R^TR$ ，R 为订单—SKU 矩阵，

若订单 j 中含有 SKU i，则元素 $r(i,j)=1$，否则为 0。在仅考虑品类拆单的情况下，热销品算法结果很好，且各配送中心负载均衡。热销品算法中，每个配送中心中都含有热销品，而对于非热销品 SKU i，只有一个配送中心含有，换句话说，只能从这个配送中心中发出配送，这样无疑会加大被拆单的可能。

我们对热销品算法进行改进，得到了环形算法。环形算法首先将 SKU 依照销量降序排列，计算一段时期内 SKU 的平均销量，将销量高于平均销量的 SKU 记为热销品，分界点记为 L，如图 1 所示。若 $L>B$，则热销品个数为 B，分配 B 个商品到每个配送中心；若 $L\le B$，则热销品个数为 L，分配 L 个商品到每个配送中心。剩余的 SKU 按照销量优先填充某个配送中心，第二个配送中心延续未分配完的部分，如果 SKU 分配完毕后还没有达到品类约束上限，则回过头重新继续分配，其余配送中心顺次分配，类似一个环形，直到全部配送中心分配完毕，如此可以提高持有非热销品的仓库数量，降低由于非热销品带来的拆单。分配过程如图 2 所示，其中浅色部分是待分配 SKU，深色部分是分配到某个配送中心的部分，从上到下依次为 DC1-DC3。

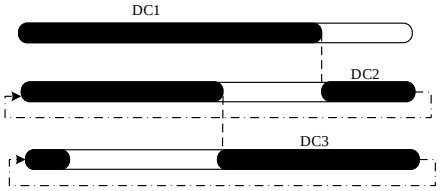

图 2 环形算法非热销品摆放策略示意图

选择销量算法（Sales）作为对比标杆算法，首先依照销售量降序排列后的结果，将 SKU 顺次分配给余下品类数最多的配送中心，SKU 全部分配完后，剩余的空位进行随机指派，完成对 SKU 品类分配。

前面完成了模型的第一步求解，下面需要决定配送中心中每种 SKU 的数量。因为热销品占据了大量的销售量，贸然分配热销品可能会导致配送中心爆仓而无法存放其余的商品，故而，选择按照销量最低的商品依次分配。我们认为，在线零售商出于库存成本和缺货损失考虑，对于销量较高的 SKU 库存保有量较高，销量较低的 SKU 库存保有量较低。因此，非热销品的保有量较低，不至于引起配送中心爆仓，数量分配上选择最简单的按比例均分取整策略。按照品类分配的结果，在分配有 SKU i 的配送中心中依照各自的剩余可存放量（数量）来按比例均分。如此完成第二步求解。

4 算例

依据对某在线零售商销售数据分析所得订单结构，模拟生成几组订单，用不同算法进行求解，并从拆单率和配送中心的负荷均衡两方面来比较算法结果的优劣。订单结构中，Chen 等[16]发现，顾客在单次购买行为中很少有超过 8 种 SKU。因此，设定单张订单所含 SKU 品类在 2-10 种，单种 SKU 所含数量在 1-12 个。实际生成结果中，单张订单最多包含 10 种 SKU，商品数量为 60；最少 2 种 SKU，商品数量为 2；平均为 4.3 个 SKU，14.2 个商品。算法用 Matlab R2013b 实现并运行在 3.7GHz CPU 和 128GB RAM 的服务器上。结合对国内某大型在线零售商销售数据分析结果，设计订单生成算法如下：

订单生成算法：

SKU 处理思路：将 SKU 根据销量划分为 S 类（下标为 s），每类设定数量 A_s、一张订单中的平均需求 $avgD_s$ 和最大需求 $maxD_s$、在订单中出现的概率 P_s。划分数量越多，越有利于贴近实际订单，但计算量也相应增大。

订单处理思路：将订单划分为 O 类（下标为 o），每类设定数量 B_o 与包含 SKU 数量 $Size_o$，订单只根据所含 SKU 数量划分，如在本算例中订单最少包含 2 种 SKU，最多包含 10 种，因此划分为 9 类。

算法分为两大步骤，第一步生成订单数据，包括 SKU 与需求量；第二步根据相关性矩阵 T 调整 SKU。T 为 $I\times I$ 的矩阵，SKU i 与另一个 SKU 之间的相关性计算方式为两个 SKU 共同出现的订单数除以 SKU i 出现的订单数。

对于每一个订单 o 有：

Step 1：取得 $Size_o$，对于每一个 SKU 位置有：

Step 1.1：根据概率 P_s，以轮盘赌的方式，产生一个 [0,1] 范围的随机数决定 SKU 类型 s；

Step 1.2：随机从 s 类 SKU 列表中抽取一种 SKU i 加入订单 o 中；

Step 1.3：若新增 SKU i 已存在于订单 o 中，则重复 Step 1.2，否则进入 Step 1.4；

Step 1.4：对于新增的 SKU i，以 $avgD_s$ 为均值生成不超过 $maxD_s$ 的泊松分布随机数作为 SKU i 在订单 o 中的需求数量。

Step 2:对于订单 o 中生成后的每一个 SKU i 有：

Step 2.1：若 [0,1] 随机数 $\le Pc$ 则进入 Step 2.2，否则转到 $i=i+1$，重复 Step 2.1；

Step 2.2：对 SKU i 与订单 o 中的其他 SKU 进行相关性求和，若小于 LB 则进入 Step 2.3，否则 $i=i+1$ 回 Step 2.1；

Step 2.3：将 SKU i 从订单 o 中剔除，得到待选 SKU 列表；

Step 2.4：计算每个待选 SKU 与订单 o 中的 SKU 相关性总和，剔除总和为 0 的 SKU，余下 SKU 降序排序；

Step 2.5：根据相关性总和以轮盘赌的形式选择新的 SKU i^* 替代原 SKU i；

Step 2.6：根据 SKU i^* 所属 SKU 种类，生成需求量。

Pc 与 LB 的设置决定订单生成结果对相关性数据的依赖性，在此均设定为 0.5。另外对于相关性矩阵 T 的获取，本文采取方法为只使用 Step 1 生成 100,000 张或更多随机订单计算相关性矩阵，则之后经过两步骤生成的订单有具有相似的需求趋势。即将一次随机结果作为一定时间内保持稳定的模拟销售环境，影响着新订单的生成，但新订单同时具有一定的随机性（可由 Pc 与 LB 的取值控制），作为需求不确定性的体现。

参数设置方面，对该在线零售商的销售数据进行分析时发现，剔除促销等因素的影响，所选择的单周期数据中，订

单量与 SKU 数量比值在 5:1 左右，对两个周期订单量与 SKU 分别累加，所得比值略低于 10:1，表明新订单对 SKU 需求变化不大，销售环境稳定。因此，算例中订单与 SKU 比取值为 5:1。在不涉及相应参数灵敏度分析时，算例中参数默认环境为配送中心个数 $N=3$，配送中心容量均衡且品类约束 $K_n=0.7I$，数量约束 C_n 为总备货量的一半左右。SKU i 的备货量 S_i 根据 SKU 销售量自动生成，最小为 5 个，总备货量为总销量的 1.2 倍左右。如无特殊说明，下文所提到的拆单率均为原订单计算后的总拆单率。拆单率指需要拆分的订单数占总订单数的百分比，原订单指用于决策 SKU 摆放策略的订单数据。

首先，在 $K_n=0.7I$，$N=3$ 的环境下，考察不同订单规模时算法的稳定性，结果见表 1，此时 B=0.55I，L 与 I 的比值见第一栏最后一行，可以看出，其比值在 0.2 左右，即环形算法计算的热销品占据总 SKU 品类的 20%左右。

表 1 不同订单规模下各算法运行结果比较

（$K_n=0.7I$，$N=3$，B=0.55I）

算法/参数	I=1000		I=2000		I=5000		I=10000		I=20000
	J=5000	J=10000	J=10000	J=20000	J=25000	J=50000	J=50000	J=100000	J=100000
Loop	9.04%	8.38%	10.04%	8.09%	9.30%	8.19%	9.36%	8.04%	9.24%
Bestseller	18.88%	15.84%	17.27%	15.95%	16.89%	15.60%	17.52%	15.75%	17.20%
Sales	27.56%	27.60%	26.61%	27.03%	26.98%	26.85%	27.13%	26.63%	27.03%
L/I	0.2210	0.1980	0.1875	0.1760	0.1814	0.1840	0.2187	0.1726	0.1853

将最大值控制在20,000种SKU, 100,000张订单（200,000张时 Matlab 提示相关性矩阵计算超出内存），这个数量级与 Catalan 所取得数量级一致[12]。在 SKU 总量一定的情况下，订单数量的增加有利于降低拆单率。这个是可以理解的，因为订单数量的增加能更好地降低随机因素对拆单率的影响。另一方面需要注意的是，在订单与 SKU 比值相等的时候，拆单率与 SKU 及订单总量关系并不大，算法结果对于规模大小不敏感。

表 2 连续 10 组同规模订单拆单结果比较

（$K_n=0.7I$，$N=3$，$J=10000$，$I=2000$，$B=0.55I$）

算法/参数	组别										均值	标准差
Loop	9.29%	9.71%	9.49%	10.41%	9.77%	9.35%	10.40%	8.96%	9.24%	8.79%	9.54%	0.54%
Bestseller	17.61%	17.08%	16.58%	16.79%	17.43%	17.38%	17.64%	18.91%	16.34%	17.79%	17.35%	0.73%
Sales	25.65%	28.06%	27.38%	29.48%	27.12%	25.70%	28.49%	27.99%	28.58%	25.60%	27.41%	1.37%
L/I	0.1819	0.1789	0.1998	0.1808	0.2098	0.2124	0.1896	0.2112	0.2018	0.1896	0.1956	0.0131

依次计算 10 组同规模订单，将结果置于表 2 中，最后两列为不同算法计算结果的均值和标准差。容易看出同规模订单中，算法结果差异与扰动量均较小，算法性能稳定。基于以上对算法规模稳定性与性能稳定性的结果，后文进一步分析时，出于算法运行时间考虑，默认环境为订单量 $J=10000$，SKU 品类 $I=2000$。

表 3 各算法总拆单率比较

（$K_n=0.7I$，$N=3$，$J=10000$，$I=2000$，$B=0.55I$，$L=0.189I$）

算法	分配 SKU 品类			分配 SKU 数量			DC 配送次数			原订单拆单率	新订单拆单率
	DC1	DC2	DC3	DC1	DC2	DC3	DC1	DC2	DC3		
Loop	1400	1400	1400	59253	57711	56051	3847	3664	3467	9.78%	14.36%
Bestseller	1400	1400	1400	57700	57686	57629	3976	3925	3847	17.48%	20.83%
Sales	1400	1400	1400	57669	57673	57673	4198	4223	4228	26.49%	34.01%

表 3 为默认环境下各算法运行结果比较，无论是原单拆单率还是新订单拆单率，环形算法结果均优于热销品算法，两个启发式算法都优于标杆算法。利用原订单的数据得到的分配方案对于新订单总是差一些。为了分析两类拆单所占的比例，用同样的两组数据，将计算的品类拆单结果列在表 4 中。

表 4 各算法品类拆单率比较

（$K_n=0.7I$，$N=3$，$J=10000$，$I=2000$，$B=0.55I$，$L=0.189I$）

算法	分配 SKU 品类			DC 配送次数			原订单拆单率	新订单拆单率
	DC1	DC2	DC3	DC1	DC2	DC3		
Loop	1400	1400	1400	3362	3349	3307	0.18%	0.95%
Bestseller	1400	1400	1400	3602	3678	3515	7.95%	9.66%
Sales	1400	1400	1400	3909	3906	3998	17.92%	24.71%

表 4 中，在仅考虑品类拆单的时候，环形算法的结果最好，均衡性也最好。对比表 3，在当前设置的备货量前提下，环形算法中，数量拆单占据了总拆单率的一大半，而热销品算法中数量拆单占据了总拆单率的一半左右。可见提高备货量可以降低单仓缺货带来的数量拆单，但考虑到配送中心容量约束，有必要探究合适的备货量水平。在图 3 中，以当前备货量 S_i 为基准，逐步提升备货量来分析总拆单率与备货量的关系。

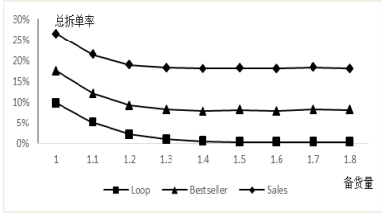

图 3 备货量与拆单率分析

（$K_n=0.7I$，$N=3$，$J=10000$，$I=2000$ $B=0.55I$，$L=0.216I$）

很明显的，随着备货量的逐步增加，拆单率稳步下降，在原备货量的 1.5 倍左右时趋于稳定，此时由于单仓缺货导致的数量拆单基本为 0，总拆单率与品类拆单相等。在线零售商可以结合配送中心的容量大小及自身的库存策略，适度提高备货量以降低数量拆单的影响。

表 5 总拆单率对品类约束 K_n 的灵敏度分析

（$N=3$，$J=10000$，$I=2000$）

算法/参数	0.4	0.5	0.6	0.7	0.8	0.9
Loop	47.86%	23.28%	13.17%	9.02%	6.50%	4.97%
Bestseller	90.77%	50.27%	24.21%	17.33%	7.17%	6.12%
Sales	98.06%	61.54%	38.49%	26.15%	16.14%	11.18%
B/I	0.1	0.25	0.4	0.55	0.7	0.85
L/I	0.191	0.172	0.207	0.184	0.198	0.219

表 5 中，控制单个配送中心的品类约束 K_n 来考察不同算法的拆单率变化。整个过程中，令 3 个配送中心的品类约束 K_n 相同，范围为 I 的 0.4-0.9 倍。注意到 $K_n=0.4I$ 时 $B=0.1I$，而 $L=0.191I$，由环形算法对热销品的定义可得，当前两个算法选取的热销品个数相同，均为 $0.1I$。此时环形算法和热销品算法的区别仅在于对非热销品的分配策略不同，表 5 数据可以看出，此时非热销品的分配策略对算法影响较大。随着 K_n 的增长，拆单率逐步下降。需要注意的是，在 $K_n=0.5I$ 及以后，环形算法选取的热销品均为 L，大小在 $0.2I$ 左右，而热销品算法选取的热销品数量逐步递增。在 K_n=$0.8I$ 及以后，虽然环形算法选取的热销品个数仍为 L，但由于其分配策略，会产生“名义热销品”。以 K_n=$0.8I$ 为例，此时销量前 40%的 SKU 三个配送中心均含有，成为环形算法的“名义热销品”。此时两个算法虽然对“名义非热销品”的分配策略有所不同，但拆单率相近。如果站在“名义热销品”的角度，我们有理由相信是由于“名义热销品”的存在平衡了“名义非热销品”分配策略的影响，因为从销量上看，前 40%的 SKU 占据近 90%的销量，且与热销品算法的热销品销量差距较小；但从算法本身角度分析，则是由于环形算法的非热销品分配策略所产生的“名义热销品”，从而导致的结果。

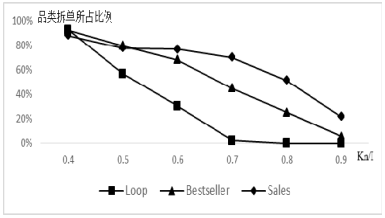

图 4 品类拆单所占比例对 K_n 的灵敏度分析

（$N=3$，$J=10000$，$I=2000$，$B=0.55I$，$L=0.193I$）

图 4 中，随着品类约束 K_n 的增大，品类拆单所占比例逐步降低。其中环形算法下降速度最快，热销品算法次之。在 K_n 较小时（如 $K_n=0.4I$），品类拆单对总拆单率的影响最大。对于环形算法，在 $K_n>0.7I$ 时，主要影响在单仓缺货引起的数量拆单。对于某一指定的配送中心，品类约束 K_n 是较为灵活的，可以通过降低 SKU 的平均备货量来提高。在线零售商在考虑多仓布局时，不能一味追求更大的 K_n，需综合考虑品类和备货量的情况来布局。如，在 K_n/I 较大时，可以适当减少配送中心中商品的品类，同时增大 SKU 平均备货量来减少数量拆单的影响。

表 6 算法运行效率对比（$K_n=0.7I$，$N=3$，单位秒）

计算环节	I=1000		I=2000		I=5000		I=10000		I=20000
	J=5000	J=10000	J=10000	J=20000	J=25000	J=50000	J=50000	J=100000	J=100000
Loop	0.0127	0.0156	0.0193	0.0319	0.0406	0.1041	0.2053	0.3501	0.6705
Bestseller	0.0830	0.0757	0.2046	0.2013	0.9749	0.9469	3.5572	3.6508	14.4472
Sales	0.0666	0.0712	0.1392	0.1449	0.3764	0.4892	0.8016	0.9633	1.9519
相关性矩阵	0.1429	0.2930	0.7110	1.4273	6.6815	12.673	34.9214	79.4503	228.4045

表 6 是不同计算环节中，算法运行效率比较，其中算法一栏对应的时间为各自算法分配 SKU 至配送中心的时间，显然环形算法的分配效率最高。随着 SKU 或订单数量的增长，算法运行时间有不同程度的增长，计算相关性矩阵所需时间呈跳跃性增长。由于热销品算法中需要计算相关性矩阵，这就大大降低了整个热销品算法的运行效率。对于拥有数十万种 SKU、数百万订单的在线零售商，计算相关性矩阵无疑会耗费大量时间，甚至无法在有效时间内完成求解。同时，相关性矩阵的计算过程中需要庞大的内存空间，而环形算法可以节省这部分计算时间和内存。

5 结论

随着在线零售业竞争的进一步加剧，更低的利润率是每个管理者都必须要面对的。对于规模以上的在线零售商而言，厂商之间的进货成本相差无几，如何通过降低运营成本，在不降低消费者体验的前提下提高利润是企业迫切需要解决的问题。本文站在仓储管理的角度对在线零售商多仓商品摆放策略进行了初步研究，以期通过对多个配送中心内部商品的摆放策略进行调整以降低订单的拆单率，从而降低商家的配送成本，进而降低运营成本。文中结合在线零售商的库存管理策略，在同时考虑品类和数量的两类拆单的情况下，建立了最小化拆单率的数学模型。改进了热销品算法得到了环形算法，对拆单问题的仿真结果表明，环形算法在兼顾各配送中心负载的同时可以进一步降低品类拆单率，从而降低总体拆单率。进而本文讨论了备货量对单仓缺货所引起拆单的影响，结果表明，在线零售商可以通过增加备货量来降低甚至消除数量拆单的影响。品类约束的研究表明，在单仓商品品类较少时，品类拆单对总的拆单率影响更大；单仓商品品类较多时，数量拆单影响更大。从算法效率上来讲，环形算法由于不需要计算相关性矩阵，所以效率比热销品算法高，尤其涉及到大规模订单—SKU 矩阵计算时，环形算法优势更为明显。

在实际中，配送中心的 SKU 摆放策略还需要考虑到其他因素，比如 SKU 自身特性以及货架的类型，这是本文后续研究的一个方向。对于 SKU 摆放决策已定的企业，库存水平是影响拆单的一个重要因素，提升库存水平可降低数量拆单的影响，然而却会同时提升企业的库存持有成本，将库存水平与拆单问题相结合，更全面地考虑降低运营成本的问题将是本文后续研究的另一方向。

参 考 文 献

[1] Xu P J. Order fulfillment in online retailing: What goes where[D]. Massachusetts Institute of Technology, 2005.

[2] Amazon.com, Inc. SEC form 10-K for the fiscal year ended December 31, 2013, http://goo.gl/QF8cRQ.

[3] Amazon.com, Inc. SEC form 10-K for the fiscal year ended December 31, 2011, http://goo.gl/ifctT.

[4] Xu P J, Allgor R, Graves S C. Benefits of reevaluating real-time order fulfillment decisions[J]. Manufacturing and Service Operations Management, 2009,11(2):340-355.

[5] Jane C, Laih Y. A clustering algorithm for item assignment in a synchronized zone order picking system[J]. European Journal of Operational Research, 2005,166(2):489-496.

[6] Parikh P J, Meller R D. Selecting between batch and zone order picking strategies in a distribution center[J]. Transportation Research Part E: Logistics and Transportation Review, 2008,44(5):696-719.

[7] Küçükaydin H, Aras N, Kuban Altınel I. Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution[J]. European Journal of Operational Research, 2011,208(3):206-220

[8] Sun H, Gao Z, Wu J. A bi-level programming model and solution algorithm for the location of logistics distribution centers[J]. Applied Mathematical Modelling, 2008,32(4):610-616

[9] Awasthi A, Chauhan S S, Goyal S K. A multi-criteria decision making approach for location planning for urban distribution centers under uncertainty[J]. Mathematical and Computer Modelling, 2011,53(1):98-109.

[10] Pan J C, Shih P. Evaluation of the throughput of a multiple-picker order picking system with congestion consideration[J]. Computers & Industrial Engineering, 2008,55(2):379-389.

[11] Parikh P J, Meller R D. Estimating picker blocking in wide-aisle order picking systems[J]. IIE Transactions, 2009,41(3):232-246.

[12] Catalán A, Fisher M L. Assortment allocation to distribution centers to minimize split customer orders. The Wharton School, University of Pennsylvania, Working Paper, 2012.

[13] Feige U, Krauthgamer R, Nissim K. Approximating the minimum bisection size (extended abstract): Proceedings of the thirty-second annual ACM symposium on Theory of computing[Z]. STOC. ACM, 2000: 530-536.

[14] Karp R M. Reducibility among combinatorial problems[M]. Springer, 1972. 85-103

[15] Andreev K, Racke H. Balanced graph partitioning[J]. Theory of Computing Systems, 2006,39(6):929-939.

[16] Chen L, Hsu F, Chen M, et al. Developing recommender systems with the consideration of product profitability for sellers[J]. Information Sciences, 2008,178(4):1032-1048.

Inventory allocation policy to distribution centers with minimum split orders in online retailing

LI Jian-bin, LI Le-le, HUANG Ri-huan

（School of Management, Huazhong University of Science and Technology, Wuhan 430074, China）

Abstract: Order splitting refers to the delivery of a retailing order in multiple shipments, which adds to fulfillment cost. The effect is much more significant while a retailer holds a large number of SKUs. In this paper, SKU allocation policy to multiple DCs in online retailing is explored to minimize the amount of split orders. Split orders due to SKU and due to unit are both considered in our research. An integer formulation of split orders problem is given. A new algorithm Loop is designed and compared with the best practice. The result shows that the algorithm performs well in reducing split orders, workload balance of DCs, and efficiency. Further analysis shows that split orders due to unit can be minimized or even eliminated when an online retailer keeps higher inventory level. Our discussion on the quantity of SKUs points out that more orders are split due to SKUs when distribution centers hold relatively few SKUs while order splitting due to unit becomes significant when more SKUs are held in each one distribution center. As for computation efficiency, the Loop algorithm is more efficient without the requirements for co-appearance matrix**.**

Firstly, based on the literature review we transfer minimizing operation cost to minimizing split orders, and find that three main factors can lead to split orders. The first factor is that a certain order demand or SKUs are not stocked in one distribution center (DC), which is called split orders due to SKU. The second factor is that the inventory level of certain SKUs can not fulfill such an order, which is called split order due to unit. The third factor is that the order will be split because of particular SKUs, such as pre-sale products. Order splitting can be minimized by optimizing SKU allocation policy, which is to decide SKU allocation to DCs and the maximum inventory level of each SKU in each DC that the SKU is allocated. An integer formulation considering both two factors is given and proved as NP-hard.

Secondly, according to the analysis of sales data from an online retailer in China, we found that a small quantity of SKUs accounts for majority sales. Therefore, these SKUs can be defined as the bestsellers and normal SKUs. Allocating bestsellers to all the DCs can minimize split orders and balance the workload. With such an idea, Loop Algorithm is designed.

Thirdly, to test the efficiency of Loop Algorithm, a numerical example is given. We use simulated orders based on the analysis of sales data to decide SKU allocation and calculate split orders. Three algorithms are compared: Loop Algorithm, Bestsellers designed by Catalan and Fisher, and Sales as benchmark. The comparison includes two aspects: split rate and workload balance of DCs. Split rate refers to the percentage of split orders. The result shows that Loop performs the best in both aspects.

Fourthly, to distinguish two kinds of split orders, a further discussion is given. Based on sensitivity analysis, feasible quantities of both SKUs and units are given considering reasonable holding cost. Qualitative conclusion is also given according to numeric results.

In summary, splitting orders is a newly raised problem with the rapid growth of online retailing and it will receive more attention in the future because how to minimize fulfillment cost has been one of the most important issues for online retailers.

Key words: Online retailing; Operation management; Split orders; Heuristics

中文编辑：杜 健；**英文编辑：**Charlie C. Chen