控制与决策
Control and Decision

RMFS补货货品存储分配问题

李儒博, 邓旭东, 刘翱, 任亮

引用本文:
李儒博, 邓旭东, 刘翱, 等. RMFS补货货品存储分配问题[J]. 控制与决策, 2025, 40(2): 528-536.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2023.1142

RMFS补货货品存储分配问题

李儒博[1], 邓旭东[1,2†], 刘 翱[1,2], 任 亮[1,2]

(1. 武汉科技大学 管理学院，武汉 430065；2. 武汉科技大学 服务科学与工程研究中心，武汉 430065)

摘 要: 为提高移动机器人履行系统(RMFS)的订单拣选效率,研究RMFS补货货品存储分配问题. 以最大化所有货架上货品之间的关联度总和为目标构建混合整数规划模型,设计求解问题的大规模邻域搜索算法,采用贪婪算法构造初始可行解,结合问题特征定义破坏算子和修复算子,并利用数值实验验证大规模邻域搜索算法的有效性. 结果表明:在贪婪算法生成初始解的基础上,大规模邻域搜索算法能有效提高解的质量,在中等和大规模算例上平均提高37.4%和21.5%. 相比变邻域搜索算法,所提出算法具有更好的优化效果,在中等和大规模算例上平均提高8.9%和10.3%. 此外,利用参数分析实验研究了货架数量、货位数量以及货品分散程度对目标函数值的影响.

关键词: 仓储；移动机器人履行系统；存储分配；订单拣选；大规模邻域搜索

中图分类号: O221；F253 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2023.1142

引用格式: 李儒博,邓旭东,刘翱,等. RMFS补货货品存储分配问题[J]. 控制与决策, 2025, 40(2): 528-536.

RMFS replenishment items storage assignment problem

LI Ru-bo[1], *DENG Xu-dong*[1,2†], *LIU Ao*[1,2], *REN Liang*[1,2]

(1. School of Management，Wuhan University of Science and Technology，Wuhan 430065，China；2. Research Center for Service Science and Engineering，Wuhan University of Science and Technology，Wuhan 430065，China)

Abstract: To improve the order picking efficiency of the robotic mobile fulfillment systems (RMFS), this paper investigates the RMFS replenishment items storage assignment problem. A mixed integer programming model is developed with the objective of maximizing the correlation degree among items in all pods. To solve the problem, a large neighborhood search algorithm is designed for the solution. The initial feasible solution is generated by a greedy algorithm. Combining the characteristics of the problem, the destroy operator and repair operator are proposed to improve the solution. Numerical experiments show that the large neighborhood search algorithm has good performance. In medium-scale and large-scale instances, the large neighborhood search algorithm can effectively improve the solution quality by 37.4% and 21.5% respectively based on the initial solution generated by the greedy algorithm, and it outperforms the variable neighborhood search algorithm with an average improvement of 8.9% and 10.3%. Moreover, the effects of the number of pods, the number of slots, and the scattered degree of items on the objective value are analyzed by the sensitivity analysis experiments.

Keywords: warehousing；RMFS；storage assignment；order picking；large neighborhood search

0 引 言

随着电子商务的迅猛发展,仓储配送中心每天需要处理的订单量急剧增加. 在所有的仓储作业当中,订单拣选是劳动密集度最高和最为耗时的作业环节,其作业成本约占到整个仓储运营费用的55%[1]. 作为仓储配送中心的核心环节,订单拣选系统的作业效率直接关系到仓储配送中心的运营状况.

基于此,移动机器人履行系统(RMFS)应运而生. RMFS是一种典型的“货到人”订单拣选系统. 在RMFS中,拣货人员固定在拣选站,由移动机器人(AMR)抬起存储货品的移动式货架,并将其搬运至拣选站进行货品拣选作业,作业完成后再由AMR将货架搬运回存储区[2-3]. 该系统由KIVA系统公司在2008年获得美国专利,随后在2012年由电商巨头亚马逊将其收购,因此RMFS也被称为KIVA系统[4]. KIVA系统自2014年左右进入中国市场后,其

收稿日期: 2023-08-12；**录用日期:** 2024-05-09.

基金项目: 湖北省教育厅哲学社会科学研究项目(22D022)；武汉市知识创新专项基础研究项目(2022010801010301)；武汉市知识创新专项曙光计划项目(2022010801020317)；中国物流学会、中国物流与采购联合会面上研究课题(2022CSLKT3-130).

责任编委: 刘士新.

†通讯作者. E-mail: dengxudong@wust.edu.cn.

柔性化表现深受市场欢迎,经过几年的发展已经成为主要的“货到人”拣选系统的存取技术[5]. 菜鸟、京东、快仓和极智嘉等国内公司也已成功应用RMFS完成相关业务[6-8].

货品存储分配作为订单拣选系统优化的要素之一,不仅是订单分批、任务分配以及路径规划的基础,还可以有效降低物料处理成本,提高仓储空间利用率和库存控制[9]. 由于电商仓库中的货品种类繁多、货架数量庞大且分布复杂,如何分配货品的存储位置已成为电商企业急需解决的难题之一. 在传统“人到货”订单拣选系统中,由于货架具有固定的存储位置,拣货人员需要沿着静态货架行走并依次检索订单中的目标货品,因此该系统只需考虑货品在各货架上的存储分配,即货品存储分配问题. 然而对于RMFS而言,不仅需要将货品分配到各货架上(货品存储分配),还需要决策移动式货架返回存储区时的位置(货架存储分配)[4,10].

货架存储分配考虑货架在返回存储区时的位置分配问题,不同于传统仓库的静态货架,RMFS中的货架可以根据需要及时调整存储位置. Weidinger等[11]将RMFS中的货架存储分配问题转化成一种特殊的区间调度问题,以此构建混合整数规划模型,并提出一种自适应规划方法进行求解;徐翔斌等[12]在进一步考虑仓库布局连续变化的情况下,研究了动态货架存储分配问题,同时利用改进的模拟退火算法进行问题求解;李珍萍等[13]和袁瑞萍等[14]均假设货架在返回存储区时仍存放到原来的储位,其等价于考虑货架定位存储策略;Yuan等[15]在考虑货品分散存储的基础上,提出了一种流体模型来分析基于速度的货架存储策略性能;在此基础上,Li等[16]针对自动导引车局部作业拥堵问题,提出了一种新的基于周转率的货架分散存储策略,并利用仿真研究验证了该存储策略的有效性;Zhuang等[17]和庄燕玲等[18]将RMFS中货架存储分配与机器人调度问题进行联合考虑,并提出了数学启发式求解方法.

货品存储分配则考虑货品在货架内的存储分配问题,由于RMFS中的货架是可以动态调整存储位置的移动式货架,即使货品分配到特定货架,该货品在存储区域的位置也会因货架位置的调整而随之改变,因此货品在分配时不需要针对特定货架进行存储[11]. 针对RMFS货品存储分配的研究,Xiang等[19]以所有货架的货品相似度之和最大化为目标构建了混合整数线性规划模型,并利用优化求解器Cplex进行求解;袁瑞萍等[14]以极大化货架上货品之间的相关性总和为目标构建货品存储分配问题的整数规划模型,设计了启发式算法进行求解;李珍萍等[13]以搬运货架总时间最短为目标建立了整数非线性规划模型,设计了贪婪算法和单亲进化遗传算法进行求解;Li等[16]在考虑不同销售时段的情况下,根据关联关系对货品之间的距离进行刻画,以此构建0-1整数规划模型,并通过聚类算法进行模型求解;Kim等[20]则通过设计一种重优化启发式算法研究了动态需求情况下的RMFS货品存储分配问题;胡祥培等[21]以货品关联度总和最大化为目标,提出了一种基于关联网络的三阶段货位分配方法;Mirzaei等[22]同时考虑货品周转率和关联性,提出了一种集成的聚类分配策略,同时利用一种贪婪构造启发式算法进行求解;Ma等[23]进一步综合考虑货品分类、货品关联性和库存水平,提出了一种新的货品分散存储策略.

以上文献均假设所有货架的初始状态是空的. 然而,在现实场景中仓库的补货和拣选过程往往交替进行,在配送时限普遍很短的情况下,相比于拣选作业,补货作业通常选择在订单低峰期进行[24],此时仓库中仍存放部分剩余货品,针对存量不足或者缺货的货品进行补货. 基于此,在文献[19]的基础上,李珍萍等[25]进一步考虑补货阶段货架上存在部分空余货位的情形,以此构建货品存储分配问题的混合整数规划模型. 然而,上述两者均限制了单个货架上存储货品的种类数量.

综上所述,本文在考虑实际补货的情形下,研究RMFS补货货品存储分配问题,其主要贡献在于: 1) 放宽了补货货品存储限制,构建了RMFS补货货品存储分配问题的混合整数规划模型; 2) 基于问题特征设计了一种大规模邻域搜索算法,采用贪婪算法构造初始可行解,定义破坏算子和修复算子对解进行迭代优化; 3) 与文献[25]提出的变邻域搜索算法进行对比分析,利用数值实验验证模型的正确性以及大规模邻域搜索算法的有效性.

1　RMFS补货货品存储分配问题模型

1.1　问题描述

某电商企业的仓储配送中心总共存储I种货品,已知每种货品需要占用的总货位数,所有货品存储在RMFS中的M个货架上,每个货架有C个货位. 在经历一个订单拣选周期后需要对RMFS进行补货,如何为需要补货的货品分配空余货位,以提高未来订单拣选的效率.

由于货品存储分配是在订单拣选作业之前进行,需要根据历史订单信息分析货品之间的关联关系. 两个不同种类的货品之间的关联度越高,其表示出现在同一订单中的概率越大,将关联度高的不同

货品存储在同一货架上,以在订单拣选过程中减少AMR搬运货架的次数,提高订单拣选效率.

为方便研究上述RMFS补货货品存储分配问题,本文做出如下假设: 1) RMFS采用标准货架,且货架上的每个货位只能存放一种货品; 2) 货品采取分散存储,即每种货品可以存放在多个货架上; 3) 货品的数量用货位数代替,且仓库中的总货位数与所有货品需要的总货位数相等; 4) 补货货品需要补货的货位数之和等于所有货架的空余货位数,即每次补货都将货架上的空余货位补满; 5) 货品与货位之间都能匹配; 6) 考虑固定需求情形,每个时段的订单信息基本一致,即可以根据历史订单信息预测未来订单的统计信息[14,21]. 因此货品之间的关联度可以根据历史订单信息计算获得. 其中: 假设1)和假设2)是现实场景中RMFS的实际情形[26]; 假设3) ~ 假设5)是为了方便建模和算法设计而提出的,并不会影响到实际补货场景中货品存储分配的决策结果[24-25].

1.2 符号说明

为建立RMFS补货货品存储分配问题的数学模型,首先定义基本符号,如表1所示.

表 1 基本符号定义

符号	含义
I	货品种类总数
M	货架总数量
C	单个货架的总货位数
i,j	货品索引$i,j=1,2,\dots,I$
m	货架索引$m=1,2,\dots,M$
I_1	需要补货的货品集合
q_m	货架m的空余货位数
D_i	货品i需要的总货位数
d_i	补货货品i需要补货的货位数
r_{ij}	货品i与j的关联度
n_i	包含货品i的历史订单数量
n_{ij}	同时包含货品i和j的历史订单数量
b_{mi}	若补货前货架m中已存放货品i则为1,否则为0
x_{im}	决策变量,为1则表示将补货货品i存放在货架m中,否则为0
y_{im}	决策变量,补货货品i在货架m上所占空余货位数

在经历一个订单拣选周期后,需要根据D_i对拣选过的货品进行补货. 仓库中的总货位数为$C\cdot M$个,并且等于所有货品需要的总货位数,即$C\cdot M=\sum\limits_{i=1}^{I}D_i$,每个货架的空余货位数必须满足$q_m\leqslant C$.

两种货品同时被订购的频次越多,它们之间的关联性就越强. 根据文献[23, 25],货品i与货品j之间的关联度r_{ij}计算公式为$r_{ij}=n_{ij}/(n_i+n_j-n_{ij})$,由此可得货品关联度矩阵$R=(r_{ij})_{(I\times I)}$. 按关联度计算得出每种货品与自身的关联度$r_{ii}$为1,然而同一种货品全部存放在一个货架上并不会减少AMR搬运货架的次数,甚至会增加AMR搬运货架的次数从而降低订单拣选效率. 基于此,本文令每种货品与自身的关联度r_{ii}为0.

1.3 混合整数规划模型

综上所述,RMFS补货货品存储分配问题的混合整数规划模型可表示为

$$\max\ z=\sum_{m=1}^{M}\sum_{i\in I_1}\sum_{j\in I_1}r_{ij}x_{im}x_{jm}+\sum_{m=1}^{M}\sum_{i\in I_1}\sum_{j=1}^{I}r_{ij}x_{im}b_{mj}. \tag{1}$$

$$\text{s.t.}\ \sum_{i\in I_1}y_{im}=q_m, m=1,2,\dots,M; \tag{2}$$

$$\sum_{m=1}^{M}y_{im}=d_i,\ \forall i\in I_1; \tag{3}$$

$$\sum_{i\in I_1}\sum_{j=1}^{I}(x_{im}+b_{mj})\leqslant C,\ m=1,2,\dots,M; \tag{4}$$

$$\sum_{m=1}^{M}x_{im}\geqslant 1,\ \forall i\in I_1; \tag{5}$$

$$x_{im}\leqslant y_{im},\ \forall i\in I_1,\ m=1,2,\dots,M; \tag{6}$$

$$y_{im}\leqslant x_{im}q_m,\ \forall i\in I_1,\ m=1,2,\dots,M; \tag{7}$$

$$x_{im}\in\{0,1\},\ \forall i\in I_1,\ m=1,2,\dots,M; \tag{8}$$

$$y_{im}\geqslant 0,\ \forall i\in I_1,\ m=1,2,\dots,M. \tag{9}$$

其中: 式(1)为目标函数,表示最大化所有货架上货品之间的关联度总和,第1项表示补货货品之间的关联度,第2项表示补货货品与货架上已存放货品之间的关联度. 式(2) ~ (9)为约束条件,式(2)表示所有补货货品在每个货架上所占的空余货位数等于该货架的空余货位数; 式(3)表示每种补货货品在所有货架上所占的空余货位数等于该货品所需要的补货货位数; 式(4)表示对于每个货架,补货货品和已存放在该货架上的货品种类数之和不超过单个货架的货位数; 式(5)表示每种补货货品至少存放到一个货架中; 式(6)和(7)表示如果补货货品i存储在货架m上,则其至少需要占用一个空余货位数,但其所占空余货位数不超过货架的空余货位数; 式(8)和(9)为决策变量取值约束.

文献[19, 25]限制了存储在每个货架上的补货货品种类数,本文对该约束进行松弛,在考虑单个货架的总货位数的情况下,放宽了补货货品存储限制. 见约束条件(4),即存储在同一货架上的补货货品种类数只与该货架上的空位数量相关. 因此,本文所构建问题模型的解空间更大,复杂度更高,也更难求解.

2　基于贪婪初始解的大规模邻域搜索算法

RMFS的货品存储分配问题是典型的NP-hard问题[21],对于大规模货品存储分配问题,精确方法难以求解,需要设计启发式算法. 大规模邻域搜索算法(LNS)是一种元启发式算法,因其具有受初始解影响小、全局寻优能力强等特点,近些年被广泛用于求解组合优化问题[27-29]. 因此,本文结合RMFS补货货品存储分配问题的特点,设计大规模邻域搜索算法进行优化求解.

2.1　大规模邻域搜索算法总体框架

大规模邻域搜索算法思想:利用贪婪算法生成初始可行解,在迭代优化过程中,首先“破坏”解,然后将破坏后的解进行“修复”,最终获得更高质量的解. 大规模邻域搜索算法伪代码如算法1所示.

算法1　大规模邻域搜索算法.

输入:贪婪算法生成初始可行解S_{init};

输出:全局最优解S_{best}.

1) $S_{\text{curr}} \leftarrow S_{\text{init}}$, $S_{\text{best}} \leftarrow S_{\text{init}}$ // 初始化当前解和全局最优解

2) while gen $\leqslant$ MAXGEN // 迭代优化

3) $S_{\text{destroy}} \leftarrow \text{destroy}(S_{\text{curr}})$ // “破坏”解

4) $S_{\text{repair}} \leftarrow \text{repair}(S_{\text{destroy}})$ // “修复”解

5) if $f(S_{\text{repair}}) > f(S_{\text{curr}})$ // 根据目标函数值更新当前解和最优解

6) $S_{\text{curr}} \leftarrow S_{\text{repair}}$

7) $S_{\text{best}} \leftarrow S_{\text{curr}}$

8) return S_{best}

2.2　解的编码

采用货架-货位矩阵$S(C$行M列)的编码方式,每一列对应一个货架,每一行对应一个货位,货位数总共有$C \cdot M$个. 每个货位的状态用0或i表示,0表示该货位没有存储货品,即为空货位,i表示货品i存储在该货位上.

例如,在补货之前,8种货品在5个货位数量为4的货架上的存储分配状态S_{rep},如图1所示.

$C\backslash M$	1	2	3	4	5
1	4	0	4	5	3
2	0	5	2	0	0
3	1	2	6	7	7
4	0	0	3	8	1

图1　解的编码

在图1中:空货位的数量(深色方格0的数量)总共有6个,第1个货架(第1列)存储的货品有4和1,还有2个空货位,以此类推. 因此,在补货之前可以得知各个货架上仍存放的货品信息以及空余的货位情况q_m,并且根据D_i可以计算出需要补货的货品种类I_1以及每种补货货品所需的补货货位数d_i.

2.3　初始解生成

贪婪算法思想:根据历史订单信息构建货品关联度矩阵R,已知各个货品需要的总货位数D_i和补货前的货品存储分配状态S_{rep},由此计算出需要补货的货品集合I_1以及需要补货的货位数d_i. 对于I_1中的每一种补货货品i,若$d_i > 0$,则在R中寻找与其关联度最大的货品j,判断j是否已存储在货架上. 若是,则将i存放到其货架的空位或者其他货架的空位上;否则将i和j一起或者分别存放到有空位的货架上. 贪婪算法伪代码如算法2所示.

算法2　贪婪算法.

输入:补货前的货品存储分配状态S_{rep}、各个货品需要的总货位数D_i、货品关联度矩阵R;

输出:初始可行解的货品存储分配状态S_{init}.

1) I, M, d_i, $I_1 \leftarrow S_{\text{rep}}$, D_i // 确认货品种类总数、货架数、需要补货的货品集合及补货货位数

2) $S_{\text{init}} \leftarrow S_{\text{rep}}$ // 初始化货品存储分配状态

3) for $i \in I_1$ & $d_i > 0$ do

4) find $j \leftarrow$ item_max R // 在R中找到与补货货品i关联度最大的货品j

5) if j inpod m // 确认货品j是否存储在货架上

6) if m have l_e & $(n_e \geqslant d_i)$ // 如果货品j所在货架上有空位且数量大于等于d_i

7) put d_i i in pod m // 将d_i数量的i存放在货架m上

8) else if m have l_e & $(n_e < d_i)$ // 如果货品j所在货架上有空位且数量小于d_i

9) put n_e i in pod m and $(d_i - n_e)$ i in other pods that have l_e // n_e数量的i存放在货架m上,另外$(d_i - n_e)$数量的i随机存放在有空位的货架上

10) else // 如果货品j所在货架上没有空位

11) put d_i i in other pods that have l_e // 将i随机存放在有空位的货架上

12) else // 如果货品j不在货架上,则表示货品j也为补货货品

13) if m have l_e & $(n_e \geqslant 2)$ // 确认是否存在两个空货位及以上的货架. 如果存在,则比较货架数量n_m、d_i和d_j的大小,此时面临以下情形

14) if $(n_m \geqslant d_i)$ & $(n_m \geqslant d_j)$ & $(d_i \geqslant d_j)$

15) pair d_j i and j put in n_m m, put $(d_i - d_j)$ i in other pods that have l_e

16) else if$(n_m \geqslant d_i)$ & $(n_m \geqslant d_j)$ & $(d_i < d_j)$

17) pair d_i i and j put in n_m m, put $(d_j - d_i)j$ in

other pods that have l_e

18) else if $(d_i > n_m)$ & $(n_m > d_j)$

19) pair d_j i and j put in n_m m, put $(d_i - d_j)$ i in other pods that have l_e

20) else if $(d_j > n_m)$ & $(n_m > d_i)$

21) pair d_i i and j put in n_m m, put $(d_j - d_i)$ j in other pods that have l_e

22) else $(d_i > n_m)$ & $(d_j > n_m)$

23) pair n_m i and j in m, put $(d_i - n_m)$ i and $(d_j - n_m)$ j in other pods that have l_e

24) else // 不存在两个空货位及以上的货架

25) put d_i i and d_j j in other pods that have l_e

26) update S_{init}, d_i // 更新需要补货货品的补货货位数和货品存储分配状态

2.4 破坏算子

破坏算子思想: 旨在打破当前解的货品存储分配状态, 即从当前解中随机移除一定数量的补货货品, 以便后续的“修复”能获得更高质量的解. 破坏算子伪代码如算法3所示.

算法3 破坏算子.

输入: 当前解S_{curr};

输出: 破坏后的解S_{destroy}, 被移除的补货货品集合I_{removed}.

1) $S_{\text{destroy}} \leftarrow S_{\text{curr}}$ // 初始化破坏后的解

2) $L_{\min} \leftarrow 2$ // 设定允许移除的最小补货货品数目

3) $L_{\max} \leftarrow \min(\lceil N/2 \rceil, 20)$ // 设定允许移除的最大补货货品数目, N为补货货品总数目

4) $L = L_{\min} + \lceil (L_{\max} - L_{\min})u \rceil$ // 计算需要移除的补货货品数目, $u \in \{0, 1\}$为均匀随机数

5) $S_{\text{destroy}} \leftarrow$remove L item_rep in S_{destroy} // 随机移除L个补货货品

6) $I_{\text{removed}} \leftarrow L$ item_rep

2.5 修复算子

修复算子思想: 通过“破坏”过程获得I_{removed}和S_{destroy}, 接下来需要对S_{destroy}进行“修复”, 即将I_{removed}中的货品依次重新存放回S_{destroy}中, 以此得到“修复”后的解S_{repair}. 修复算子伪代码如算法4所示.

算法4 修复算子.

1) $S_{\text{repair}} \leftarrow S_{\text{destroy}}$ // 初始化修复后的解

2) while $I_{\text{removed}} \neq \varnothing$ do // I_{removed}非空则执行循环

3) $L \leftarrow$ item number in I_{removed} // I_{removed}中补货货品的数目

4) for $i \in I_{\text{removed}}$ do // 计算I_{removed}中每个补货货品存放回空位的关联度增量, 即“插入增量”

5) $\delta_i \leftarrow i$ reinsert in S_{repair} and sort descending // 计算并降序排列每个补货货品的“插入增量”

6) $\alpha_i \leftarrow \delta_i(1) - \delta_i(2)$ // 计算“遗憾值”, 即将最大的“插入增量”减去次大的“插入增量”

7) $j \leftarrow \max\{\alpha_i\}$ // 找到“遗憾值”最大的补货货品j

8) $S_{\text{repair}} \leftarrow j$ reinsert in S_{repair} $\max\{\delta_i\}$ l_e // 将货品j存放到其“插入增量”最大的空位中

9) $I_{\text{removed}} \leftarrow I_{\text{removed}}$ remove j // 将货品j从I_{removed}中移除

3 数值实验

利用MaltabR2015a编程实现本文提出的大规模邻域搜索算法, 在Win10、64 bit操作系统、8 GB内存的运行环境下进行实验. 所有算法最大迭代次数均设为500次, 同时为避免随机因素, 针对每个算例记录10次运行后的平均值、最小值和最大值.

3.1 参数设置及实验数据

由于RMFS补货货品存储分配问题没有基准测试集和标准算例, 本文对参数设置及数据生成说明如下:

1) 补货作业需要根据缺货率μ进行. 缺货率是指补货前货架上的空货位数量占到全部货架总货位数的比例, 即: 当μ为0时, 货架上无空货位, 此时不需要补货; 当μ为1时, 所有货架的货位为空. 选择缺货率较低时进行补货作业, 则需要频繁对仓库进行补货, 而在缺货率较高时进行补货作业, 则会在拣选过程中大概率面临缺货情况. 基于此, 本文设定进行补货作业时的缺货率$\mu = 25\,\%$, 即在补货前货架上的空货位数量占到全部货架总货位数的25 %.

2) 通过设置货品种类总数I、货架数量M以及单个货架的货位数量C生成不同规模的算例, 以验证算法的性能. 其中小规模算例参考文献[25]随机生成, 中大规模的算例参考文献[25,30]以及现实场景中某RMFS的实际参数生成, 算例参数如表2所示. 同时随机生成货品关联度矩阵R、各货品需要的总货位数D_i和补货前的货品存储分配状态S_{rep}.

表 2 算例相关参数

算例规模	I	M	C
小规模	[20, 30]	[10, 30]	5
中等规模	[100, 200]	[50, 100]	7
大规模	[800, 1 000]	[300, 400]	10

3) 求解器和各算法求得的解对应为混合整数规划模型中定义的目标函数值, 即所有货架上货品之间的关联度总和. 此外, 所有实验数据保留一位小数.

3.2　大规模邻域搜索算法性能分析

3.2.1　小规模算例实验

在小规模算例中,利用Gurobi求解器直接求解混合整数规划模型,同时设定Gurobi可接受运行时间为1 800秒. 分别记录大规模邻域搜索算法(LNS)和Gurobi求解器得到的解以及运行时间,同时利用误差Gap值进行比较分析

$$\text{Gap}=\frac{f(\text{LNS})-f(\text{Gurobi})}{f(\text{Gurobi})},$$

其中$f(\text{LNS})$和$f(\text{Gurobi})$分别为大规模邻域搜索算法和Gurobi求解器求得的解.

小规模算例实验结果如表3所示.

表 3　小规模算例下LNS与Gurobi求解结果对比

I-M-C	Gurobi		LNS		Gap/%
	f(Gurobi)	t/s	f(LNS)	t/s	
20-10-5	26.0	0.9	26.0	0.7	0.0
20-12-5	31.0	1.6	31.0	0.8	0.0
20-14-5	35.6	20.8	35.1	1.0	−1.4
20-16-5	40.5	283.7	40.1	0.9	−1.0
20-18-5	50.7	1 800	50.3	1.0	−0.8
20-20-5	52.9	1 800	52.3	1.1	−1.1
30-20-5	53.6	1 800	53.1	1.1	−0.9
30-25-5	66.7	1 800	65.5	1.3	−1.8
30-30-5	87.6	1 800	86.0	1.5	−1.8

在解的质量方面:即使Gurobi在一些很小规模的算例上能够取得最优解,大规模邻域搜索算法得到的解与Gurobi求得的最优解之间的误差Gap仅在0.0%～1.4%之间,在所有小规模算例中也不超过1.8%.

在求解时间方面:由于模型复杂度较高,Gurobi的求解时间随着问题规模的增大而急剧增加,对于含有18个货架以上的算例Gurobi无法在短时间内求得精确最优解,而大规模邻域搜索算法在所有小规模算例中的运行时间均不超过2 s. 因此,大规模邻域搜索算法在求解质量和运行时间上均具有良好的性能.

3.2.2　中大规模算例实验

对于中等规模和大规模算例,分别记录贪婪算法(GA)、变邻域搜索算法(VNS)[25]和大规模邻域搜索算法(LNS)在10次运行后解的平均值Avg、最小值Min和最大值Max,并用加粗字体表示平均值的最优结果. 由于采用贪婪算法生成初始可行解,只记录大规模邻域搜索算法和变邻域搜索算法的运行时间. 此外,计算大规模邻域搜索算法优化后解的平均值相比贪婪算法和变邻域搜索算法提升的百分比,即

$$\text{LNS}-\text{GA}=\frac{\text{LNS(Avg)}-\text{GA(Avg)}}{\text{GA(Avg)}},$$

$$\text{LNS}-\text{VNS}=\frac{\text{LNS(Avg)}-\text{VNS(Avg)}}{\text{VNS(Avg)}}.$$

表4分别记录了在中等规模和大规模算例下各算法的解.

表 4　中大规模算例下的算法结果对比

规模	*I-M-C*	GA			VNS				LNS				LNS-GA/%	LNS-VNS/%
		Avg	Min	Max	Avg	Min	Max	time/s	Avg	Min	Max	time/s		
中等	100-50-7	208.1	203.3	219.9	266.5	262.1	271.5	1.0	**284.8**	282.5	286.1	1.4	36.9	6.9
	100-60-7	240.6	233.5	246.4	322.2	319.4	328.1	1.1	**346.9**	344.0	350.8	1.5	44.2	7.7
	100-70-7	280.9	274.9	288.4	374.6	372.2	377.7	1.3	**410.7**	403.7	414.1	1.8	46.2	9.6
	200-80-7	363.0	358.3	369.1	436.2	431.4	441.3	1.4	**474.4**	472.2	477.6	1.9	30.7	8.8
	200-90-7	396.3	381.4	409.0	488.6	478.9	499.2	1.6	**537.5**	533.5	541.7	2.2	35.6	10.0
	200-100-7	453.6	441.1	466.1	538.4	532.3	545.2	1.8	**593.5**	589.0	598.3	2.5	30.8	10.2
	Average		–			–		1.4		–		1.9	37.4	8.9
大	800-300-10	1 985.1	1 976.2	1 998.2	2 238.5	2 209.3	2 259.0	5.6	**2 478.4**	2 467.1	2 485.1	8.2	24.9	10.7
	800-350-10	2 066.5	2 050.0	2 089.1	2 288.3	2 262.9	2 316.3	6.5	**2 521.7**	2 515.4	2 527.7	10.7	22.0	10.2
	800-400-10	2 078.2	2 069.9	2 095.3	2 290.3	2 274.8	2 307.9	7.0	**2 532.6**	2 521.7	2 540.2	11.4	21.9	10.6
	1 000-300-10	2 059.7	2 043.4	2 075.8	2 266.3	2 249.2	2 281.8	5.7	**2 489.0**	2 484.5	2 496.6	8.0	20.8	9.8
	1 000-350-10	2 116.7	2 099.1	2 129.2	2 287.0	2 268.0	2 300.3	6.5	**2 517.0**	2 504.8	2 531.1	11.1	18.9	10.1
	1 000-400-10	2 101.9	2 071.0	2 124.6	2 303.6	2 286.8	2 324.2	6.9	**2 537.4**	2 523.6	2 548.5	11.7	20.7	10.1
	Average		–			–		6.4		–		10.2	21.5	10.3

在求解质量方面:大规模邻域搜索算法在所有算例中运行得到的解均优于对比算法. 在贪婪算法生成初始解的基础上,大规模邻域搜索算法在中等规模和大规模算例上对解的改进平均提高了37.4%和21.5%,并且相比于变邻域搜索算法优化后的解,大规模邻域搜索算法相应提高了8.9%和10.3%.

在求解时间方面:由于破坏算子和修复算子的设计,大规模邻域搜索算法具有更多样化的邻域结构和搜索空间,因此求解时间比变邻域搜索算法稍长一些,在中等和大规模算例上的平均差值分别为0.5 s和3.8 s. 但总体上大规模邻域搜索算法仍能在12 s内获得比变邻域搜索算法质量更优的解.

3.3 参数分析

3.3.1 货架数量 M 的变化影响

图 2 为大规模邻域搜索算法求解的目标函数平均值随着 M 变化的趋势图.

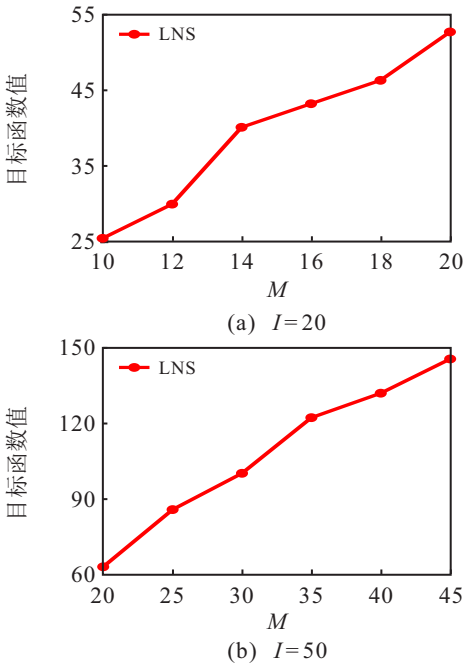

图 2 货架数量 M 的变化影响

在 I(分别为20和50)和 C(为5)不变的情况下,随着 M 的增加,货位的数量和分配给货品的货位数同样会相应增加,因此目标函数值随之相应提高.

3.3.2 货位数量 C 的变化影响

在 I(为100)和 M(分别为40和50)保持不变的情况下,随着单个货架货位数量 C 的增加,目标函数平均值会显著提高,如图3所示

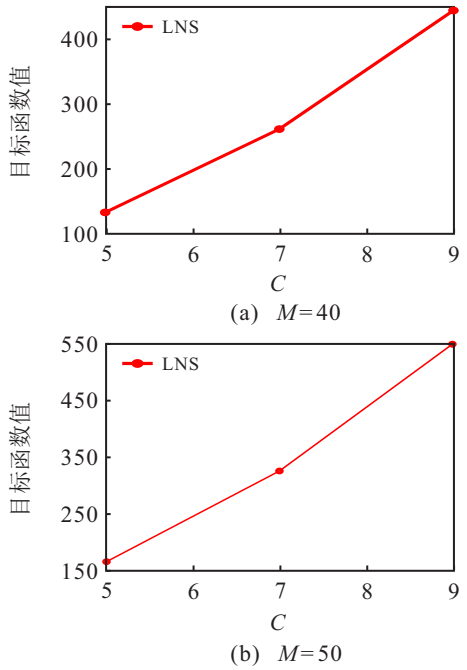

图 3 货架数量 C 的变化影响

一方面,随着 C 的增大,仓库中所有货架的货位数均会增加,分配给货品的货位数同样会相应增加;另一方面,对于单个货架而言,C 的增大意味着可以同时存储更多的货品. 因此,货位数量 C 的变化对目标函数值的影响较大.

3.3.3 货品分散程度 s 的变化影响

文献[31]表示RMFS采取分散存储策略可以有效减少AMR搬运货架的距离. 根据文献[24],本文定义货品的分散程度 s 为平均每个货品需要分配的货位数,即 $s=(C\cdot M)/I$.

货品种类总数不能超过仓库的总货位数,即 $s\geqslant 1$. 当 $s=1$ 时,表明货品种类总数等于总货位数,即平均每种货品对应一个货位,其相当于集中存储策略. 在总货位数确定的情况下,s 越大,表明货品种类数目越少,单个货品所占的货位数越多,即货品更分散地存储在各货架上. 分散存储策略虽然能够有效减少AMR搬运货架的距离,但是随着货品分散程度的增加,同等面积下仓库存储的货品种类数目会相应减少,或者在存储相同种类数目的货品时需要更多的货架.

在给定 $M=20, C=5$ 的情况下,设定不同的 I 值以生成不同的算例,针对每个算例利用大规模邻域搜索算法独立运行10次取平均值,结果如表5和图4所示. 可以看出,随着货品分散程度的增加,货品种类数目呈现减少的趋势,尤其当 $s\leqslant 10$ 时,货品种类数目随着分散程度的增加而急剧减少.

表 5 货品分散程度 s 的实验结果

s (I-M-C)	LNS	s (I-M-C)	LNS	s (I-M-C)	LNS
1 (100-20-5)	67.2	7 (14-20-5)	60.2	14 (7-20-5)	48.9
2 (50-20-5)	66.2	8 (12-20-5)	57.1	17 (6-20-5)	48.8
3 (33-20-5)	61.8	9 (11-20-5)	57.2	20 (5-20-5)	52.8
4 (25-20-5)	62.5	10 (10-20-5)	46.8	25 (4-20-5)	48.9
5 (20-20-5)	60.6	11 (9-20-5)	51.7	33 (3-20-5)	31.9
6 (17-20-5)	57.4	12 (8-20-5)	51.4	50 (2-20-5)	24.5

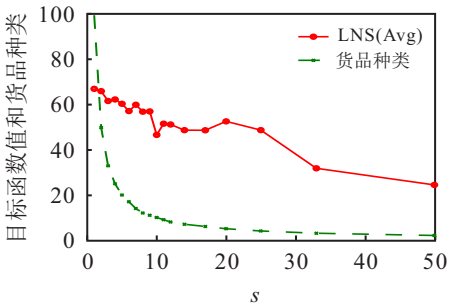

图 4 货品分散程度 s 的变化影响

货品分散程度对目标函数值的影响并不是单调的. 一方面,随着 s 的增加,货品被分配的货位数也相

应增多,同一种货品重复存放在相同货架上的概率增大,导致关联度总和降低;另一方面,每种货品货位数的增加可以让关联度高的两种货品重复组合存储,从而提高关联度总和. 因此,目标函数值会受到补货前货品存储分配初始状态的影响. 然而,当$I < C$时,意味着每个货架肯定有重复存储的货品,并且随着I的减少重复程度增高,当$I = 1$时代表仓库中只存储一种货品,此时货品关联度总和为0. 因此,当$I \leqslant C$时,随着s的增加,目标函数值单调减少.

4 结 论

本文对RMFS补货阶段的货品存储分配问题进行了研究,构建了问题的混合整数规划模型,并结合问题特征设计了大规模邻域搜索算法对不同规模的算例进行求解. 结果表明,在小规模算例上,大规模邻域搜索算法得到的解与Gurobi求得的解之间的误差不超过1.8%,同时大规模邻域搜索算法具有明显的运行时间优势. 其次,在贪婪算法生成初始可行解的基础上,大规模邻域搜索算法能有效提高解的质量,在中等和大规模算例上平均提高37.4%和21.5%,相比变邻域搜索算法[25]具有更好的优化效果,在中等和大规模算例上平均提高8.9%和10.3%. 此外,参数分析实验表明,货架数量和货位数量与目标函数值呈现正相关关系,而货品分散程度对目标函数值的影响并不是单调的.

在实际场景中,由于受到季节性需求变化、产品促销、产品生命周期、消费者偏好等因素的影响,客户需求以及订单结构具有较大波动性和时效性. 因此,对于电商企业而言,考虑动态需求情况下的货品存储分配问题更具现实意义,后续可进行重点研究.

参考文献(References)

[1] de Koster R, Le-Duc T, Roodbergen K J. Design and control of warehouse order picking: A literature review[J]. European Journal of Operational Research, 2007, 182(2): 481-501.

[2] Wurman P, D'Andrea R, Mountz M. Coordinating hundreds of cooperative, autonomous vehicles in warehouses[J]. AI Magazine, 2008, 29(1): 9-19.

[3] Guizzo E. Three engineers, hundreds of robots, one warehouse[J]. IEEE Spectrum, 2008, 45(7): 26-34.

[4] Boysen N, de Koster R, Weidinger F. Warehousing in the e-commerce era: A survey[J]. European Journal of Operational Research, 2019, 277(2): 396-411.

[5] 尹军琪. 物流配送中心的拣选技术与策略分析[J]. 物流技术与应用, 2021(10): 116-119.
(Yin J Q. Picking technologies and strategies analysis of logistics distribution center[J]. Logistics & Material Handling, 2021(10): 116-119.)

[6] 徐翔斌, 马中强. 基于移动机器人的拣货系统研究进展[J]. 自动化学报, 2022, 48(1): 1-20.
(Xu X B, Ma Z Q. Robotic mobile fulfillment systems: State-of-the-art and prospects[J]. Acta Automatica Sinica, 2022, 48(1): 1-20.)

[7] 孙阳君, 赵宁. 多机器人存取系统研究现状[J]. 物流技术与应用, 2021, 26(6): 129-133.
(Sun Y J, Zhao N. Research status of multi robot retrieval system[J]. Logistics & Material Handling, 2021, 26(6): 129-133.)

[8] 李昆鹏, 刘腾博, 贺冰倩, 等. "货到人"拣选系统中AGV路径规划与调度研究[J]. 中国管理科学, 2022, 30(4): 240-251.
(Li K P, Liu T B, He B Q, et al. A study on routing and scheduling of automated guided vehicle in "cargo-to-picker" system[J]. Chinese Journal of Management Science, 2022, 30(4): 240-251.)

[9] Reyes J J R, Solano-Charris E L, Montoya-Torres J R. The storage location assignment problem: A literature review[J]. International Journal of Industrial Engineering Computations, 2019, 10(2): 199-224.

[10] Azadeh K, de Koster R, Roy D. Robotized and automated warehouse systems: Review and recent developments[J]. Transportation Science, 2019, 53(4): 917-945.

[11] Weidinger F, Boysen N, Briskorn D. Storage assignment with rack-moving mobile robots in KIVA warehouses[J]. Transportation Science, 2018, 52(6): 1479-1495.

[12] 徐翔斌, 马中强. RMFS订单拣选系统动态货位再指派研究[J]. 计算机集成制造系统, 2021, 27(4): 1146-1154.
(Xu X B, Ma Z Q. Dynamic location reassignment of RMFS order picking system[J]. Computer Integrated Manufacturing Systems, 2021, 27(4): 1146-1154.)

[13] 李珍萍, 范欣然, 吴凌云. 基于"货到人"拣选模式的储位分配问题研究[J]. 运筹与管理, 2020, 29(2): 1-11.
(Li Z P, Fan X R, Wu L Y. Study on the storage allocation problem under cargo to person picking mode[J]. Operations Research and Management Science, 2020, 29(2): 1-11.)

[14] 袁瑞萍, 王慧玲, 李俊韬, 等. 基于移动机器人的订单拣选系统货位优化模型和算法研究[J]. 系统科学与数学, 2020, 40(6): 1050-1060.
(Yuan R P, Wang H L, Li J T, et al. The slotting optimization model and algorithm in robotic mobile fulfillment systems[J]. Journal of Systems Science and Mathematical Sciences, 2020, 40(6): 1050-1060.)

[15] Yuan R, Graves S C, Cezik T. Velocity-based storage assignment in semi-automated storage systems[J].

Production and Operations Management, 2019, 28(2): 354-373.

[16] Li X W, Hua G W, Huang A Q, et al. Storage assignment policy with awareness of energy consumption in the Kiva mobile fulfilment system[J]. Transportation Research Part E: Logistics and Transportation Review, 2020, 144: 102158.

[17] Zhuang Y L, Zhou Y, Hassini E, et al. Rack retrieval and repositioning optimization problem in robotic mobile fulfillment systems[J]. Transportation Research Part E: Logistics and Transportation Review, 2022, 167: 102920.

[18] 庄燕玲, 孙玉姣, 朱涛, 等. 移动货架系统多机器人“存-取货架”调度优化方法[J]. 系统工程理论与实践, 2023, 43(2): 488-508.
(Zhuang Y L, Sun Y J, Zhu T, et al. Multi-robot scheduling problem of robotic mobile fulfillment systems: An adaptive matheuristic[J]. Systems Engineering — Theory & Practice, 2023, 43(2): 488-508.)

[19] Xiang X, Liu C C, Miao L X. Storage assignment and order batching problem in Kiva mobile fulfilment system[J]. Engineering Optimization, 2018, 50(11): 1941-1962.

[20] Kim H J, Pais C, Shen Z J M. Item assignment problem in a robotic mobile fulfillment system[J]. IEEE Transactions on Automation Science and Engineering, 2020, 17(4): 1854-1867.

[21] 胡祥培, 丁天蓉, 张源凯, 等. 基于关联网络的机器人移动货架系统货位分配方法[J]. 工程管理科技前沿, 2022, 41(1): 56-64.
(Hu X P, Ding T R, Zhang Y K, et al. A method for storage assignment in robotic mobile fulfillment system based on association networks[J]. Forecasting, 2022, 41(1): 56-64.)

[22] Mirzaei M, Zaerpour N, de Koster R. The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance[J]. Transportation Research Part E: Logistics and Transportation Review, 2021, 146: 102207.

[23] Ma Z Q, Wu G H, Ji B, et al. A novel scattered storage policy considering commodity classification and correlation in robotic mobile fulfillment systems[J]. IEEE Transactions on Automation Science and Engineering, 2023, 20(2): 1020-1033.

[24] Weidinger F, Boysen N. Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse[J]. Transportation Science, 2018, 52(6): 1412-1427.

[25] 李珍萍, 贾顺顺, 卜晓奇, 等. 无人仓系统储位分配问题的优化模型与算法[J]. 中国管理科学, 2022, 30(1): 124-135.
(Li Z P, Jia S S, Bu X Q, et al. The optimization model and algorithm for storage location assignment problem in unmanned warehouse system[J]. Chinese Journal of Management Science, 2022, 30(1): 124-135.)

[26] Qin H L, Xiao J, Ge D D, et al. JD.com: Operations research algorithms drive intelligent warehouse robots to work[J]. Informs Journal on Applied Analytics, 2022, 52(1): 42-55.

[27] 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.

[28] 伍国华, 毛妮, 徐彬杰, 等. 基于自适应大规模邻域搜索算法的多车辆与多无人机协同配送方法[J]. 控制与决策, 2023, 38(1): 201-210.
(Wu G H, Mao N, Xu B J, et al. The cooperative delivery of multiple vehicles and multiple drones based on adaptive large neighborhood search[J]. Control and Decision, 2023, 38(1): 201-210.)

[29] 展月, 姜兆勤, 刘振元. 多结构型任务驱动的上门服务调度优化模型与自适应大规模邻域搜索算法[J]. 控制与决策, 2024, 39(3): 947-955.
(Zhan Y, Jiang Z Q, Liu Z Y. Optimization model and revised adaptive large neighbourhood search algorithm for multi-structured on-site service scheduling problem[J]. Control and Decision, 2024, 39(3): 947-955.)

[30] Boysen N, Briskorn D, Emde S. Parts-to-picker based order processing in a rack-moving mobile robots environment[J]. European Journal of Operational Research, 2017, 262(2): 550-562.

[31] Lamballais Tessensohn T, Roy D, de Koster R B M. Inventory allocation in robotic mobile fulfillment systems[J]. IISE Transactions, 2020, 52(1): 1-17.

作者简介

李儒博(1994−), 男, 博士生, 主要研究方向为仓储运营管理与优化, E-mail: rubo_li@wust.edu.cn;

邓旭东(1964−), 男, 教授, 硕士, 主要研究方向为管理优化与决策, E-mail: dengxudong@wust.edu.cn;

刘翱(1987−), 男, 副教授, 博士, 主要研究方向为优化调度, E-mail: liuao@wust.edu.cn;

任亮(1985−), 男, 讲师, 博士, 主要研究方向为物流优化, E-mail: renliang@wust.edu.cn.