控制与决策
Control and Decision

基于混合元启发式算法的订单分批问题

吴仁超, 贺建军, 李欣, 殷泽阳, 陈祖国

引用本文:
吴仁超,贺建军,李欣,殷泽阳,陈祖国. 基于混合元启发式算法的订单分批问题[J]. 控制与决策, 2022, 37(8): 2110−2118.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2021.0364

基于混合元启发式算法的订单分批问题

吴仁超[1], 贺建军[1†], 李 欣[1], 殷泽阳[1], 陈祖国[2]

(1. 中南大学 自动化学院，长沙 410083；2. 湖南科技大学 信息与电子工程学院，湖南 湘潭 411201)

摘 要: 订单拣选是仓库运营管理中一项高劳动强度与高成本的操作,拣货员在仓库中从货位拣选出满足订单需求的货物. 订单分批问题(order batching problem, OBP)是订单拣选中的重要规划问题,该问题以最小化拣选批次路径时长为目标,将用户订单分配至拣选批次中. 首先,为了优化订单分配构造高质量批次,提出一种混合元启发式算法,在自适应大邻域搜索框架中融入基于不可行下降的局部搜索,同时引入自适应惩罚机制和一批基于订单与基于批次的移除启发式以及新的算法组件;其次,为了优化拣选路径进一步降低批次旅行时间,提出单向启发式,利用动态规划优化组合多个路径策略. 实验表明,在合理计算时间内,所提出算法的求解质量优于多重启变邻域搜索(MS-VNS)、混合自适应大邻域搜索及禁忌搜索(ALNS/TS),而且所提出算法的最大路径长度减少率达到22.36 %.

关键词: 订单拣选；订单分批问题；混合元启发式算法；不可行下降；单向启发式

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2021.0364

引用格式: 吴仁超,贺建军,李欣,等. 基于混合元启发式算法的订单分批问题[J]. 控制与决策, 2022, 37(8): 2110-2118.

Hybrid metaheuristic algorithm for order batching problem

WU Ren-chao[1], *HE Jian-jun*[1†], *LI Xin*[1], *YIN Ze-yang*[1], *CHEN Zu-guo*[2]

(1. School of Automation，Central South University，Changsha 410083，China；2. School of Information and Electrical Engineering，Hunan University of Science and Technology，Xiangtan 411201，China)

Abstract: Order picking is one of the most laborious and high-cost processes in the warehouse management and operation. Pickers retrieve goods from their storage locations in order to satisfy orders. The order batching problem (OBP) is a planning problem, which is critical for order picking. Customer orders are grouped into batches in such a way that the total travel distance of all pickers is minimized. First, in order to tackle the assignment of orders to obtain high quality batches, the hybrid metaheuristic algorithm is proposed, which incorporates an infeasible descent procedure into the adaptive large neighborhood search framework. An adaptive punishment mechanism, removal heuristics related to customer orders or batches and additional algorithmic components are also introduced. Second, to address the picker routing and hence reduce batch travel time, the unidirectional heuristic is proposed using a dynamic programming approach to combine routing heuristics. By means of computational experiments, the proposed algorithm is compared to the ALNS/TS and the MS-VNS. It is demonstrated that, in reasonable computing times, the algorithm provides solutions of excellent quality which lead to a reduction of the total travel distance by up to 22.36 %.

Keywords: order picking; order batching problem; hybrid metaheuristic algorithm; infeasible descent procedure; unidirectional heuristic

0 引 言

仓库运营管理包括接受、存储、拣选、装箱与运输等一系列操作. 订单拣选处理来自于内外部供应链中订单检索过程[1-2],占仓库运营成本的55 %以上,因此,订单拣选是高效管控仓库极其重要的环节[3]. 为了降低订单拣选成本与工作强度,人们提出了大量的规划问题及相应的决策方法以协助决策者优化管理订单拣选操作. 其中,重要的规划问题包括:货位分布问题、订单分批问题、拣选路径问题等,拣选时长主要依赖于这3个规划问题的求解. 货位分布问题分配货物给仓库内的货位[4];订单分批问题重组来自于内外部的订单[5];拣选路径问题制定了路径策略拜访所

收稿日期: 2021-03-03；**录用日期:** 2021-06-03.

基金项目: 国家自然科学基金项目(61873282)；中央高校基本科研业务费专项资金项目(2018zzts170).

责任编委: 阳春华.

†通讯作者. E-mail: jjhe@csu.edu.cn.

有包含需求货物的货位[6].

在已知仓库布局、货物分布和用户订单的条件下,订单分批问题的求解分为两步: 1) 订单分配优化,将用户订单分配给可行的批次,每个可行批次中的需求货物量不能超过拣选设备容量限制,其目标通常是最小化拣选路径时间或者距离[7]; 2) 构造批次的拣选路径,专用的启发式算法[6]通常用于构建路径并计算路径时长. 当每个批次中允许的订单数量大于2时,订单分批问题被认为是一个NP-hard问题[8]. 现有的精确求解方法[9]无法完全解决现实存在的大规模案例. 目前,多数订单分批研究在提出的订单分配算法后采用已有的路径启发式来构建批次路径. de Koster等[10]提出了两种基于节省的启发式算法: cw(I)和cw(II)优化订单分配,随后,使用遍历策略和最大间隔策略计算批次时长. Albareda-Sambola等[11]提出了变邻域搜索算法,随后,利用遍历策略、最大间距策略和组合策略计算批次时长. Henn等[12]采用基于属性的爬山算法和禁忌搜索算法分配订单,使用遍历策略和最大间距策略计算批次时长. Öncan[13]引入混合禁忌阈值的迭代局部搜索算法,并提出3种路径策略: 遍历策略、前端返回策略和中点策略的混合整数线性规划模型. Menéndez等[14]提出了多重启变邻域搜索(MS-VNS)算法,采用交换移动和转移移动[12]探索可行域,再使用后优化策略降低批次数量,并提出了新的拣选路径启发式. Žulj等[15]提出了混合自适应大邻域搜索与禁忌搜索(ALNS/TS). ALNS/TS算法在可行域内利用了自适应大邻域搜索的多样性能力和禁忌搜索的强化搜索能力. 基于标准案例[12]与遍历策略和最大间距策略,ALNS/TS算法优于基于属性的爬山算法[12]和混合禁忌阈值的迭代局部搜索算法[13]. 拣选路径问题通常被视为特殊的旅行商问题[16-17]. 精确路径算法很少应用于实际仓库运营,而启发式的路径策略则因其可生成清晰且易于理解的路径结构而得到广泛应用[18-19]. 遵循同样的原因,在订单分批问题中,路径启发式算法被广泛应用于构造批次路径. Hall[19]提出了遍历策略、中点策略和最大间隔策略以解决单块仓库中拣选路径问题. Petersen[18]提出了返回策略和复合策略. Roodbergen等[20]延伸了最大间隔策略和遍历策略,提出了组合启发式. Menéndez等[14]提出了遍历策略与最大间距策略的一种组合.

本文提出一种混合元启发式(hybrid metaheuristic algorithm, HMA)算法处理订单分配,并提出单向启发式(unidirectional heuristic, UH)以构造更短的拣选路径. HMA算法在自适应大邻域搜索[21]的框架中融入了不可行下降以提升整体算法的多样性能力与强化搜索能力.

本文的主要创新点在于:

1) 在HMA算法中引入自适应惩罚机制[22],同时探索可行域与不可行域以提高算法全局搜索能力. 不可行下降探索局部区域中的可行解与非可行解并逐步定位到高质量可行区域,增强局部探索能力. 同时引入多样性保留、重启机制、不可行下降的调用机制以提升性能.

2) 提出4种新的基于订单的移除启发式: 带惩罚的最差移除启发式、相对最差移除启发式、复合最差移除启发式以及一种基于新的相关性度量的相关移除启发式,同时还提出两种基于批次的移除启发式: 随机批次移除启发式和相关批次移除启发式,进而又添加了随机贪婪插入启发式.

3) 针对当前优秀路径策略与精确路径算法之间依然存在着高达10%的差距[18],提出一个新的路径策略: 单向启发式,进一步优化拣选路径. 单向启发式在提出两类返回方式后,利用动态规划优化组合遍历策略、最大间距策略、前端返回与后端返回策略.

1　问题描述与模型建立

1.1　问题描述

矩形单块仓库[8]平行地排列着一组相同长度的拾取过道,货位摆放在拾取过道两侧的货架上,拾取过道与位于仓库前端和后端的两个横排过道相交,交点分别被称为拾取过道的前端端点和后端端点,交付点位于最左拾取过道前端端点不远处. 货位上存放着不同货物,每个订单至少由一种货物组成,每个订单分配给一个拣选批次,所有货物对应的固定货位已知. 每个订单的拣选不允许分离,即不可在不同的批次中检索同一订单,因为订单分离会带来大量后续整理工作. 图1所示为一个单块矩形仓库结构的案例,仓库包括8个平行拾取过道,每个货架包含10个货

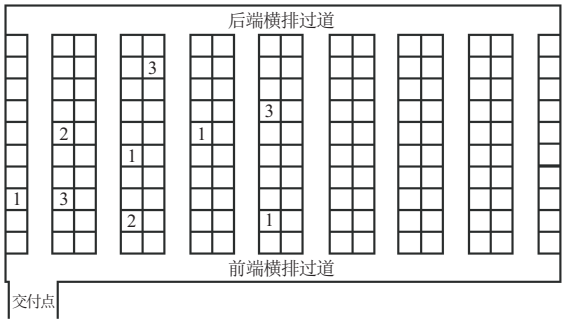

图1　单块矩形仓库的布局

位,并给出了订单1、订单2、订单3的需求货物的货位位置.

1.2 模型建立

本文采用文献[8]中订单分批问题的整数规划模型. $J=\{1,2,\dots,n\}$为订单集合,c_j为订单$j\in J$的货物容量. Q为拣选批次最大容量,限定拣选批次中货物数量. X_{ij}是一个二元变量,如果该变量为1,则订单j分配给批次i;否则该变量设置为0. 定义B为所有可行拣选批次的集合,可行批次集合中的批次容量约束为

$$\sum_{j\in J}c_jX_{ij}\leqslant Q,\ \forall\, i\in B. \tag{1}$$

定义d_i为可行批次$i\in B$的拣选路径时间. 拣选路径以及拣选路径时间由相对应的拣选路径策略决定. 二元变量Y_i表示可行批次i是否被选中:如果可行批次i被选中,则Y_i为1;否则Y_i为0. 订单分批问题建模如下所示:

$$\min\ \sum_{i\in B}d_iY_i. \tag{2}$$

$$\text{s.t.}\ \sum_{i\in B}X_{ij}Y_i=1,\ \forall\, j\in J; \tag{3}$$

$$Y_i\in\{0,1\},\ \forall\, i\in B. \tag{4}$$

其中:目标函数(2)最小化所有选中的可行批次的拣选时间之和,约束(3)限制每个订单只能被分配给一个选中的可行批次,约束(4)限定决策变量Y_i为二元变量.

2 混合元启发式

HMA算法在自适应大邻域搜索的基本框架中融入自适应惩罚机制与基于不可行下降的局部搜索,并添加算法组件以增强算法性能. 组件包括:不可行下降调用机制、重启机制与多样性保留.

HMA算法主体结构如算法1所示. 算法开始于由cw(I)算法生成的初始解S_0,且初始解S_0总是为可行解. 因此,记录当前最优解S^*与当前最优可行解S_f^*均为初始解S_0. 在随后的每次迭代中,依据随机选取的移除启发式D_i,通过移除一些订单或一些批次破坏当前解S. 随后,依据随机选取的插入启发式R_i将已移除订单插入已有批次或者构建新的批次来修复当前解,形成候选解S.

算法1 混合元启发式算法.

1) 利用cw(I)算法生成初始解S_0.

2) $S^*\leftarrow S_0, S_f^*\leftarrow S_0, S\leftarrow S_0, l_{\text{div}}\leftarrow 0, \text{imp}\leftarrow 0, l\leftarrow 0$.

3) while $l<d_{\max}$ and $l-l_{\text{imp}}<d_{\text{imp}}$ do

4) 依据启发式的权重值(W_i^D,W_i^R)选择移除与插入启发式(D_i,R_i).

5) $S'\leftarrow R_i(D_i(S))$.

6) if $l-l_{\text{div}}>d_{\text{div}}$ and $f(S')<f(S^*)(1+\delta_d)$ or $\text{imp}=1$ then

7) $S'\leftarrow$ 不可行下降(S',α).

8) $l_{\text{div}}=l,\ \text{imp}=0$.

9) end if

10) if S' 满足接受规则then

11) $S\leftarrow S'$.

12) 更新超载惩罚系数α.

13) end if

14) if $f(S')<f(S^*)$ then

15) $S^*\leftarrow S',\ l_{\text{imp}}\leftarrow l$.

16) if $1\leqslant l-l_{\text{div}}\leqslant d_{\text{div}}$ then

17) $\text{imp}=1$.

18) end if

19) end if

20) if feasible(S') and $f(S')<f(S_f^*)$ then

21) $S_f^*\leftarrow S'$

22) end if

23) if $\text{modulo}(l,d_r)=0$ then

24) $S'\leftarrow S_f^*$.

25) end if

26) if $\text{modulo}(l,d_{sg})=0$ then

27) 更新启发式的权重值(W_i^D,W_i^R).

28) end if

29) $l\leftarrow l+1$.

30) end while

为了跳出局部最优,探索更广阔的搜索空间,在执行一定数量的搜索迭代后,才考虑调用不可行下降程序,这一段迭代数量称为多样性保留时长d_{div}. 如果在距离上次不可行下降搜索调用后经过了d_{div}次迭代后,发现当前最优解S^*已更新,即当最优解标识符$\text{imp}=1$时,则调用不可行下降程序以提升新的当前最优解S^*,且多样性计数l_{div}设置为当前迭代次数l. 若在随后的迭代中发现候选解S'的目标函数接近或提升当前最优解S^*时,即当$f(S')<f(S^*)(1+\delta_d)$时,则同样使用不可行下降来提升候选解并更新多样性计数l_{div},其中δ_d为最优解的劣化系数.

每当生成一个新的候选解S'时,候选解会被提交给接受规则. 如果被接受,则该候选解成为新的当前解S,并更新超载惩罚系数α;否则,当前解与惩罚系数保持不变. 若出现新的当前最优可行解S^*或新

的当前最优可行解S_f^*,则更新记录. 若当前最优可行解S^*在多样性保留期内出现更新,则最优解标识符$\mathrm{imp}=1$.

为了避免在不可行域做过多的探索,每迭代d_r次后,重置候选解S'为当前最优可行解S_f^*,d_r为重启时长. 当总体迭代次数l达到最大迭代次数$d_{\max}$或当经历最大无提升迭代次数d_{imp}一直没有提升最优可行解S^*时,总体算法终止.

2.1　构造初始解

使用基于节省的启发式算法cw(I)[10]构造订单分批问题的初始解S_0. 首先确定订单j所对应的批次拣选路径时长t_j;然后对可行批次确定路径时长t_{ij},可行批次由订单i与订单j组成,计算订单i与订单j合并后节省时长$\mathrm{tt}_{ij}=t_i+t_j-t_{ij}$. 在迭代中考虑合并最大节省时长所对应的一对订单: 1) 当每个订单都没有被分配给批次时,并且这对订单的需求货物量之和不超过批次容量限制,则合并; 2) 当一个订单分配给了批次,而另一个单独成批且合并可行时,则将单独的订单分配给已有批次; 3) 当两个订单都已有批次时,直接跳过该次合并.

2.2　自适应惩罚

自适应惩罚机制已成功地应用于许多组合优化问题[22-23]. $T(S)$为当前解S中批次的拣选时间之和,$O(S)$为当前解中批次超载量之和,c_j为当前解S中批次j的装载量. 松弛订单分批问题的目标函数为$f(S)=T(S)+\alpha O(S)$,超载量惩罚为$O(S)=\sum_{j\in J}\max\{0,c_j-Q\}$,$\alpha\in[\alpha_{\min}T_0/Q,\alpha_{\min}T_0/Q]$是与案例相关的自适应超载惩罚系数,$T_0$表示初始解$S_0$中拣选时间的总和,$Q$为批次最大容量,$\alpha_{\min}$和$\alpha_{\max}$分别为超载惩罚系数的上界和下界. 超载惩罚系数$\alpha$在初始阶段设定为$\alpha_{\min}T_0/Q$. 候选解被接受规则引入后,若超载量大于0,则$\alpha$更新为$\min\{\alpha u,\alpha_{\max}T_0/Q\}$;否则为$\max\{\alpha/u,\alpha_{\min}T_0/Q\}$,其中惩罚控制参数$u\in[1,2]$.

2.3　接受规则

使用基于模拟退火技术的接受规则来判断候选解S'是否被接受. 当$f(S)>f(S')$时,候选解S'被接受;否则,候选解S'依照概率$P(S,S')$被接受,即

$$P(S,S')=\mathrm{e}^{\frac{-(f(S')-f(S))}{t}},\tag{5}$$

其中t是当前温度值. 初始温度t_0的取值使得落后初始解目标函数值$d\%$的解以50%的概率被接受,d为初始劣化比率. 在后续迭代中,t设定为ηt,其中$\eta\in[0,1]$为冷却因子.

2.4　自适应机制

HMA算法使用轮盘赌依据权重随机选中一个插入启发式R_i. 所有的启发式分配相同的初始权重值W_0. 每产生一个候选解S',依据以下3种情况调整被使用的插入启发式R_i的得分值C_i^R: 1) 如果发现一个新最优解,则得分值C_i^R增加s_a; 2) 如果发现一个优解,则得分增加s_b; 3) 如果发现一个劣化解,但是依旧被模拟退火接受为新的当前解,则得分增加s_c. 在d_{sg}个连续的HMA算法迭代后,更新权重插入启发式R_i的权重W_i^R,即

$$W_i^R=(1-r)W_i^R+rC_i^R/S_i^R.\tag{6}$$

其中:参数$r\in[0,1]$为腐坏因子,S_i^R为插入启发式R_i在迭代段内被选中的次数. 移除启发式D_i得分值与权重W_i^D的调整均采用同样的原理.

2.5　移除与插入启发式

每次迭代中,为了探索较远的邻域,HMA算法随机使用一个移除启发式与一个插入启发式破坏并修复当前解S,生成候选解S'. 移除启发式移除当前解S中q个订单,q在区间$[q_{\min},q_{\max}]$中随机选取. 插入启发式将移除的订单插入到部分解中生成候选解S'.

2.5.1　移除启发式

1) 随机移除启发式. 从当前解中,随机地移除q个订单.

2) 最差移除启发式[15]. 用移除节省值R_j描述当订单j从当前批次中移除后,该批次拣选时间的下降值. 建立一个有序列表来降序排列移除节省值. 每次迭代中,位于列表位置v的订单被移除,$v=\lfloor|L|\beta^y\rfloor$. 其中:$\beta$从区间$[0,1]$中随机选中,$L$为当前列表长度,$y\geqslant 1$为随机参数.

3) 带惩罚的最差移除启发式. D_k^i为包含订单i的批次k的拣选时间,O_k^i为包含订单i的批次k的超载量,D_k^{i*}为从批次k中剔除订单i后的拣选时间,O_k^{i*}为剔除订单i后的超载量. 订单i的带惩罚的移除节省值$\mathrm{PR}_i=D_k^i-D_k^{i*}+\alpha(O_k^i-O_k^{i*})$,$\alpha$为当前迭代中的超载惩罚因子.

4) 相对最差移除启发式. 相对移除节省值$\mathrm{RR}_i=D_k^i-D_k^{i*}-\gamma t_i$,$t_i$为订单$i$单独成批时的拣选时间,$\gamma\in[0,1]$为时间权重因子.

5) 复合最差移除启发式. 复合移除节省值为$\mathrm{CR}_i=D_k^i-D_k^{i*}+\alpha(O_k^i-O_k^{i*})-\gamma t_i$.

6) 相关移除启发式. Žulj等[15]定义的相似度由订单合并节省值与批次的结构相似性组成. 而本文同时考虑路径相似度和容量相似度. 订单i分配给包含订单j的批次后,拣选时间增大的估计值设定为

$t_{ij}-t_j$. 综合订单i分配给订单j的批次,订单j分配给订单i的批次这两个情况后,订单i与订单j的路径相似度定义为$\mathrm{RD}_{ij}=t_{ij}-0.5t_j-0.5t_i$. 订单$i$与订单$j$的相似度$R_{ij}$可由下式计算:

$$R_{ij}=\varphi\frac{\mathrm{RD}_{ij}}{D_{\max}}+(1-\varphi)\frac{|c_i-c_j|}{Q_{\max}}. \tag{7}$$

其中:$D_{\max}$为初始解S_0中最大的订单拣选时间,$Q_{\max}$为最大订单货物重量,φ为权重参数用于确定时间与重量之间的相对权重,c_i和c_j分别表示订单i和订单j的容量.

7) 随机批次移除启发式. 随机选中批次,累计批次包含的订单数,直到累计订单数首次超过q时迭代停止. 相关批次移除启发式是指:对于每对批次p和批次m,提出批次相似度Z_{pm},由两个批次的订单的最小路径相似度决定,即

$$Z_{pm}=\min_{i\in p,j\in m}t_{ij}-0.5t_j-0.5t_i. \tag{8}$$

2.5.2 插入启发式

1) 贪婪插入启发式. 每次迭代中,计算所有移除订单对每个现有批次或新的批次的插入成本,即订单插入后拣选时间增量与可能的超载量惩罚值的增量之和,将最小插入成本的订单插入到相应批次中.

2) 随机贪婪插入启发式. 为避免小拣选时间订单总是优先插入,依据随机顺序选择插入订单,将其插入到最小插入成本增量的批次中.

3) 后悔插入启发式. 2-后悔插入启发式计算每个移除订单在每个批次上的插入成本,随后使用第2小的插入成本与最小插入成本之差来度量2-后悔值,具有最大2-后悔值的订单被选中并插入到最小成本增量所对应的批次. 本文考虑3-后悔插入启发式,即计算每个订单第2小的插入成本与最小插入成本之差和第3小的插入成本与最小插入成本之差的总和.

2.6 不可行下降

一个当前解S进入不可行下降时,首先使用全域搜索,依据松弛的订单分批问题的目标函数,使用禁忌搜索[24]探索S的邻域$N(S)$;再利用可行域搜索优化探索可行解. 不可行下降使用邻域结构N_{shift}和N_{swap}[12]:N_{shift}邻域尝试从原来的批次中移除一个订单,再将它分配给另一个批次;N_{swap}邻域尝试交换来自于不同批次的订单对. 在每次迭代中,从复合邻域里选取最优邻域解,即$N(S)=N_{\mathrm{shift}}(S)\bigcup N_{\mathrm{swap}}(S)$. 不可行下降继承当前容量惩罚系数$\alpha$,在全域搜索中调整容量惩罚系数$\alpha$. 禁忌搜索中的禁忌任期$\theta$随机从区间$[\theta_{\min},\theta_{\max}]$内抽取. 全域搜索执行$\mathrm{tn}_1$次迭代后,若没有发现可行解,则提交发现的超载量最小的解作为可行域搜索的初始解;若出现了可行解,则提交当前发现的最优可行解. 可行域搜索同样采用禁忌搜索框架:若当前解是可行解,则只允许执行可行的邻域移动;若当前解是不可行解,则执行惩罚值最小的邻域移动. 可行域搜索在迭代tn_2次后,或在可行域搜索中出现当前解是可行解且无法发现可行的邻域解时,不可行下降结束.

3 路径策略

如前文所述,订单分批问题的解包含每个可行批次的结构,需要利用路径策略解决批次内的路径问题,进一步确定分批问题的目标值. 基本的启发式路径策略[18-19]包括:遍历策略、中点策略、最大间隔策略、返回策略及复合策略. 组合启发式[20]拜访每个拾取过道一次,在每个拾取过道中可以完全穿过,也可以使用相同的端点进入与驶出过道,并且依据动态规划从以上策略中作最优的选择. Menéndez启发式[14]先固定包含货物的最左与最右拾取过道,使用遍历策略;随后判断剩余拾取过道是使用最大间距策略还是遍历策略,以获得更短的拣选路径. 因此,组合启发式可视为遍历策略、前端返回策略与后端返回策略的最优组合,而Menéndez启发式则视为遍历策略与最大间距策略的一种组合.

3.1 单向启发式

本文提出的单向启发式同时拓展了组合启发式与Menéndez启发式,从左至右拜访拾取过道,再采用单向返回方式到达交付点,在返回途中拾取剩余货物,利用动态规划实现遍历策略、最大间距策略、前端返回策略、后端返回策略的最优组合. 单向启发式同时考虑两类具有不同返回方式的路径:前端返回路径f在前端横排过道上执行单向返回路径;后端返回路径r则对应后端横排过道,选择其中最短的路径作为最终的单向启发式路径.

3.2 前端返回路径

假定给定的拾取清单包括n个包含货物的拾取过道,定义拾取过道i的前端端点a_i和后端端点b_i. 前端返回路径由两段子路径组成:单向拜访路径ff_n从交付点开始顺序拜访直到拾取过道n后终止于交点a_n;单向返回路径fr_n从a_n出发,在前端横排过道上执行带拾取的单向返回直到交付点. 本文使用部分前端返回路径来表示动态规划中的阶段,部分前端返回路径的等价类则代表着阶段中的状态. 拜访拾取过道i的部分前端返回路径f_i由部分单向拜访路径ff_i与部分单向返回路径fr_i组成. 部分单向拜访路径ff_i指到达拾取过道i的路径;部分单向返回路径fr_i

代表在前端横排过道上拾取了从拾取过道 i 到拾取过道 1 中所有的剩余货物. 定义路径 f_i 的两个等价类 fa_i 与 fb_i: 等价类 fa_i 中, 路径 ff_i 在拜访完拾取过道 i 后, 到达前端端点 a_i; 而等价类 fb_i 中的 ff_i 停止于后端端点 b_i.

每当添加一个新的拾取过道时, 通过确定可行的过道拾取策略完成状态之间的转移. 使用两个步骤延伸 f_i: 1) 添加横排过道路径延伸 ff_i 与 fr_i 至拾取过道 $i+1$ 的端点; 2) 添加拾取过道 $i+1$ 的可行拾取策略, 完成拾取过道 $i+1$ 的拜访. 如图 2 所示, 存在两种横排延伸方式: ff_i 与 fr_i 同时执行前端横排延伸, 延伸时长 $\mathrm{fc}_{a,i+1} = 2d_{a,i+1}$; ff_i 选择后端横排延伸的同时 fr_i 执行前端横排延伸, 时长 $\mathrm{fc}_{b,i+1} = d_{a,i+1} + d_{b,i+1}$. 过道内拾取策略在 3 种可能的策略中选择: 遍历策略、最大间距策略、前端返回策略. 在遍历策略中, 拣货员完全穿过所有包含订单货物的拾取过道, 即由一个端点进入拾取过道, 再由另一个端点驶出. 最大间隔策略考虑最大化节省地分离过道内的上下部分的货物. 首先计算拾取过道中两个货物之间最大的间隔, 最靠近前端货物与前端之间的间隔, 最靠近后端货物与后端之间的间隔; 然后, 定义拾取过道的最大间隔为上述 3 个间隔中最大的一个; 最后, 依据最大间隔划分过道, 即所有过道的最大间隔前部分组成过道的前部分, 剩余为过道的后部分. 拣货员两次进入过道, 一次由前端点进入与驶出拜访过道的前部分, 另一次由后端点进入与驶出拜访过道的后部分. 前端返回策略规定拣货员均由位于前端的端点进入与驶出包含订单货物的拾取过道. 假定 d_i^1+1 为遍历过道 $i+1$ 所需时间, d_i^2+1 为采用最大间距策略拜访过道 $i+1$ 所需时间, d_i^3+1 为前端返回拜访过道 $i+1$ 所需时间. 假定交付点到包含货物的拾取过道 1 的端点 a_1 的横排延伸所需要时间为 d_0. 定义 tfa_i 为等价类 fa_i 中保存的最短部分前端返回路径的拣选时间, tfb_i 对应于等价类 fb_i. 转移的成本为分别来自于添加的横排过道和拾取过道的路径时间之和. 因此, 通过阶段中状态的转移来确定下一个状态, 保留状态内最优成本的值, 迭代地执行动态规划. 计算等价类 fa_1 与 fb_1 中保存的路径拣选时间: $\mathrm{tfa}_1 = 2d_0 + d_1^3$, $\mathrm{tfb}_1 = 2d_0 + d_1^1$. 对每个拾取过道 $i \in (1,n)$ 计算 $\mathrm{tfa}_i = \min\{\mathrm{tfa}_{i-1} + \mathrm{fc}_{a,i} + d_i^3, \mathrm{tfb}_{i-1} + \mathrm{fc}_{b,i} + d_i^1\}$, $\mathrm{tfb}_i = \min\{\mathrm{tfa}_{i-1} + \mathrm{fc}_{a,i} + d_i^1, \mathrm{tfb}_{i-1} + \mathrm{fc}_{b,i} + d_i^2\}$. 最后, 计算 $\mathrm{tfa}_n = \min\{\mathrm{tfa}_{n-1} + \mathrm{fc}_{a,n} + d_n^3, \mathrm{tfb}_{n-1} + \mathrm{fc}_{b,n} + d_n^1\}$. 等价类 fa_n 被认为是前端返回路径的最小拣选时间.

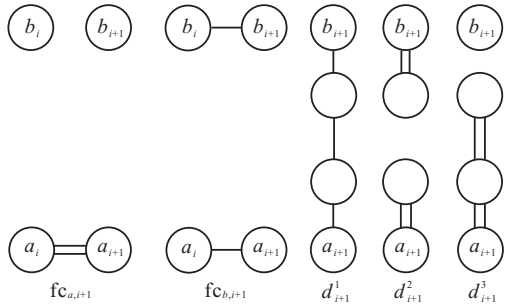

图 2　部分前端返回路径 f_i 的转移

3.3　后端返回路径

定义拜访到拾取过道 i 的部分后端返回路径 r_i 的两段部分子路径: 1) 部分单向拜访路径 rf_i 从交付点开始到达端点 a_2, 顺序拜访直到拾取过道 i, 停止于过道 i 的某个端点; 2) 部分单向返回路径 rr_i 从端点 b_i 出发, 在后端横排过道上返回至端点 b_1 并拾取剩余货物, 遍历拾取过道 1 后返回交付点. 在等价类 ra_i 中, rf_i 停止于交点 a_i; 等价类 rb_i 中, rf_i 停止于 b_i.

如图 3 所示, rf_i 执行前端横排延伸和 rr_i 执行后端横排延伸后所需时间 $\mathrm{rc}_{a,i+1} = d_{a,i+1} + d_{b,i+1}$. rf_i 与 rr_i 同时执行后端横排延伸所需时间 $\mathrm{rc}_{b,i+1} = 2d_{b,i+1}$. 拾取过道可选择 3 种拾取策略: 遍历策略、最大间距策略、后端返回策略. 遍历策略和最大间距策略所需时间的设定与前端返回路径一致, 后端返回策略规定拣货员均由位于后端的端点进入与驶出包含订单货物的拾取过道. 假定 d_{i+1}^4 为后端返回拜访过道 $i+1$ 的时间. 定义 tra_i 为等价类 ra_i 中保存的路径时间, trb_i 对应于 rb_i. 计算 ra_2 与 rb_2 中保存的路径拣选时间: $\mathrm{tra}_2 = 2d_0 + d_1^1 + \mathrm{rc}_{a,2} + d_2^2$, $\mathrm{trb}_2 = 2d_0 + d_1^1 + \mathrm{rc}_{a,2} + d_2^1$. 当延伸至拾取过道 $i \in (2,n)$ 时, 计算 $\mathrm{tra}_i = \min\{\mathrm{tra}_{i-1} + \mathrm{rc}_{a,i} + d_i^2, \mathrm{trb}_{i-1} + \mathrm{rc}_{b,i} + d_i^1\}$, $\mathrm{trb}_i = \min\{\mathrm{tra}_{i-1} + \mathrm{rc}_{a,i} + d_i^1, \mathrm{trb}_{i-1} + \mathrm{rc}_{b,i} + d_i^4\}$. 后端返回路径的最小时间为 $\mathrm{trb}_n = \min\{\mathrm{tra}_{n-1} + \mathrm{rc}_{a,n} + d_n^1, \mathrm{trb}_{n-1} + \mathrm{rc}_{b,n} + d_n^4\}$.

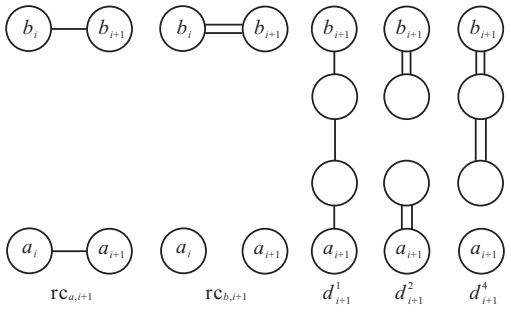

图 3　部分后端返回路径 r_i 的转移

4　计算实验

设计实验在基准案例[12] 上比较了 HMA 算法与 MS-VNS[14]、ALNS/TS[15] 的性能, 使用 C++ 执行计算, 计算硬件配置为 Windows7、i5-4590 处理器、16 GHz

内存.

4.1 基准案例与参数设定

基准案例中的仓库具有10个平行拾取过道,其中每个过道上配置90个货位. 库存管理采用两种货物分配方案:类分配(CBD)与随机分配(UDD). 考虑4种订单数量规模$n = \{40, 60, 80, 100\}$与4种容量限制的拣选设备$Q = \{30, 45, 60, 75\}$. 每个订单包含的货物数量在区间$[5, 25]$中随机抽取. 基准案例考虑两种路径启发式:遍历策略和最大间隔策略. 共有64种不同的案例类型,每个类型包含40个相似案例,最终包含2 560个案例. HMA算法引入新的路径策略,只考虑32种不同的案例类型,共1 280个案例. 随后,依据货物分配方案,将这些案例划分至不同基准集合中. 在每个基准集合内,每个案例类型由订单数量和拣选设备容量决定. 在每个案例类型上,比较HMA算法与使用不同传统路径策略的现有算法.

依据由Ropke等[21]提出的方法设定HMA算法参数. 在有限数量案例中调整参数,从每个案例类型中选中一个案例组成测试案例集用于参数调整. 首先确定得分值与初始权重:$s_a = 100$, $s_b = 80$, $s_c = 70$, $W_0 = 10$;然后得到迭代段长度$d_{sg} = 100$,腐坏因子$r = 0.3$;接下来确定自适应惩罚系数的下界$\alpha_{\min} = 50$,上界$\alpha_{\max} = 500$,惩罚控制参数$u = 1.15$;再确定移除启发式中的随机参数$y = 3$,时间权重因子$\gamma = 0.15$,时间与重量之间的相对权重参数$\varphi = 0.65$. 设定接受规则中初始劣化比率$d = 30$,冷却因子$\eta = 0.999\,75$. 发现移除启发式中,移除订单数量q与订单数量n相关,其下界$q_{\min} = \max\{0.15n, 10\}$,上界$q_{\max} = 0.35n$. 不可行下降中禁忌任期$\theta$的下界$\theta_{\min} = 0.25n$,上界$\theta_{\max} = 1.75n$. 设定最大迭代次数$d_{\max} = \max\{50n, 3\,000\}$,无提升迭代次数$d_{\text{imp}} = 150$,多样性保留时长$d_{\text{div}} = \max\{0.3n, 20\}$,用于控制不可行下降调用的最优解的劣化系数$\delta_d$为0.3,不可行下降迭代中全域搜索时长$\text{tn}_1 = 15n$,可行域搜索时长$\text{tn}_2 = 15n$. 为了避免在不可行域做过多的探索,重启迭代次数$d_r$设定为$\max\{2.5n, 150\}$.

4.2 比较实验

MS-VNS与HMA算法各自提出了新的批次拣选路径策略,而ALNS/TS使用两种不同传统的路径策略:S型策略与最大间隔策略. 因此,在每个基准集合内的每个案例类型上,将使用S型策略的ALNS/TS(ALNS/TS+SS)、使用最大间隔策略的ALNS/TS(ALNS/TS+LG)、MS-VNS算法与HMA算法进行比较,所有算法执行一次. 给出平均偏差值dev.,而dev.定义为在一类案例中,当前算法最优解与所有算法中发现的已知最优解之间的平均百分比偏差值,即$\text{dev.} = 100(\text{BKS} - Z)/\text{BKS}$. 其中:$Z$是ALNS/TS+SS或ALNS/TS+LG、MS-VNS、MHA中发现的最优解的平均值,BKS是已知最优解的平均值,所有方法的最优解中的最优解为已知最优解. #best为每个方法中的已知最优解数量,算法运行时间T的单位为s.

表1比较了类分配场景内16个案例类型中,HMA算法、ALNS/TS+SS、ALNS/TS+LG与MS-VNS的性能. 从求解质量上,HMA算法在所有案例

表1 不同的订单分批算法在类分配场景中的计算结果

案例类型		ALNS/TS+SS			ALNS/TS+LG			MS-VNS			MHA		
订单	容量	dev.	#best	T/s	dev.	#best	T/s	dev.	#best	T/s	dev.	#best	T/s
40	30	12.43	0	2.04	4.98	2	5.86	1.52	5	2.30	0.03	38	4.63
	45	9.53	0	2.22	8.74	0	9.31	2.53	0	6.54	0.00	40	7.31
	60	4.72	1	2.85	10.25	0	14.28	4.67	0	10.29	0.00	39	12.47
	75	4.24	0	3,82	15.24	0	19.14	8.28	0	16.57	0.00	40	19.02
60	30	14.53	0	5.57	4.31	0	18.64	2.23	0	6.58	0.00	40	16.64
	45	11.97	0	6.32	10.65	0	29.21	4.83	0	14.94	0.00	40	25.69
	60	8.34	0	7.68	15.87	0	42.55	7.53	0	28.32	0.00	40	42.97
	75	7.35	0	9.06	16.31	0	51.34	10.12	0	43.86	0.00	40	50.32
80	30	15.98	0	13.55	5.62	0	34.26	3.34	0	14.68	0.00	40	30.29
	45	7.67	0	14.71	8.13	0	61.23	7.44	0	33.01	0.00	40	54.21
	60	10.12	0	19.21	15.68	0	89.67	12.33	0	45.52	0.00	40	70.17
	75	10.88	0	20.45	18.43	0	120.33	14.35	0	70.63	0.00	40	110.62
100	30	12.62	0	25.44	5.36	0	70.25	5.44	0	21.54	0.00	40	63.26
	45	11.44	0	27.64	10.26	0	117.64	9.75	0	44.65	0.00	40	108.22
	60	13.22	0	30.24	16.34	0	168.65	14.49	0	84.85	0.00	40	125.04
	75	8.55	0	38.11	18.74	0	229.36	18.72	0	115.94	0.00	40	184.32
average		10.22	0.63	14.31	11.56	0.13	67.61	7.92	0.31	35.01	0.00	39.81	57.82

类别上均优于其他算法. 具体体现为: 在所有案例上, HMA 算法解的平均值比 MS-VNS 算法降低了 7.92 %, 比 ALNS/TS+SS、ALNS/TS+LG 分别降低了 10.22 % 和 11.56 %; 在每个案例类别中, HMA 算法都发现了最多的已知最优解, 总体上: HMA 算法在 640 个案例中发现了 637 个已知最优解, 且 HMA 算法发现的已知最优解中仅仅包括了 5 个与其他算法相同的已知最优解; 虽然与 MS-VNS 相比 HMA 算法的计算时间增大了 65.15 %, 与 ALNS/TS+SS 相比 HMA 算法的计算时间增大了 3 倍, 较 ALNS/TS+LG 减少 14.48%, 但是, HMA 算法的运行时间仍然保持在合理范围内. 因此, HMA 算法在合理的计算时间内可以发现质量更高的解.

表 2 呈现了 HMA 算法、ALNS/TS+SS、ALNS/TS+LG 与 MS-VNS 在随机分配场景中的性能. 在所有案例上, HMA 算法的求解质量优于其他算法. HMA 算法解的平均值较 MS-VNS、ALNS/TS+SS、ALNS/TS+LG 分别降低了 8.56 %、11.07 % 和 13.04 %. HMA 算法在 640 个案例中发现了 637 个已知最优解, 剩余的已知最优解则由 MS-VNS 发现. 在计算时间方面, HMA 算法的求解时间较 ALNS/TS+LG 降低了 14.85 %, 但是较 MS-VNS 增大了 42.37 %, 是 ALNS/TS+SS 的 4 倍. HMA 算法的运行时间同样保持在合理范围内, HMA 算法在合理的计算时间内也可以发现质量更高的解.

表 2　不同的订单分批算法在随机分配场景中的计算结果

案例类型		ALNS/TS+SS			ALNS/TS+LG			MS-VNS			MHA		
订单	容量	dev.	#best	T/s	dev.	#best	T/s	dev.	#best	T/s	dev.	#best	T/s
40	30	15.78	0	2.12	5.16	0	4.94	2.08	3	3.14	0.02	37	4.86
	45	10.95	0	2.36	9.94	0	8.62	2.81	0	7.32	0.00	40	8.03
	60	4.61	0	3.01	11.45	0	16.32	4.39	0	14.42	0.00	40	14.35
	75	4.38	0	3.95	17.35	0	23.68	8.42	0	23.72	0.00	40	22.52
60	30	15.67	0	5.97	5.85	0	18.35	2.67	0	7.31	0.00	40	17.36
	45	12.86	0	6.86	10.34	0	28.42	5.38	0	22.04	0.00	40	27.23
	60	8.67	0	8.12	15.72	0	50.53	7.19	0	44.15	0.00	40	47.96
	75	8.35	0	9.54	18.69	0	57.34	12.32	0	49.43	0.00	40	52.64
80	30	14.67	0	15.49	7.39	0	36.26	5.23	0	15.75	0.00	40	32.16
	45	8.39	0	17.84	9.61	0	65.57	7.74	0	33.01	0.00	40	56.21
	60	10.58	0	19.92	17.14	0	95.21	10.92	0	46.84	0.00	40	72.67
	75	12.93	0	21.68	20.05	0	135.82	16.67	0	73.93	0.00	40	115.98
100	30	12.27	0	26.06	7.94	0	76.61	6.57	0	29.82	0.00	40	69.81
	45	11.46	0	28.14	11.67	0	120.76	8.68	0	64.28	0.00	40	112.63
	60	15.39	0	31.34	17.94	0	178.68	15.75	0	109.25	0.00	40	129.67
	75	10.19	0	39.19	22.36	0	236.47	20.06	0	145.67	0.00	40	198.09
average		11.07	0	15.10	13.04	0	72.10	8.56	0.19	43.13	0.00	39.81	61.39

相较于 ALNS/TS+SS 算法、ALNS/TS+LG 算法, HMA 算法的解的平均值在表 1 和表 2 两种分配场景中的所有案例类型中均有所降低, 范围为 4.24 %～22.36 %. 并且相较于 ALNS/TS+LG 算法, 本文算法解的平均值的降低率随拣选设备容量、订单数量的增大而增大. 在订单数量为 100、设备容量为 75 的案例类型中, HMA 算法解的平均值较 ALNS/TS+LG 算法在表 1 和表 2 两种分配场景中分别降低了 18.74 % 和 22.36 %. ALNS/TS+SS 的计算时间均远小于其他算法. 在两种分配场景中, 虽然 HMA 算法的计算时间比 ALNS/TS+SS 高 3～4 倍, 但是, HMA 算法相较于 ALNS/TS+LG 算法分别降低了 14.50 % 和 14.85 %. 相较于 MS-VNS 算法, HMA 算法的解的平均值在表 1 和表 2 两种分配场景中的所有案例类型中均有所降低, 范围为 1.52 %～20.06 %. 同样, 增大拣选设备容量、订单数量, HMA 算法相较于 MS-VNS 算法解的平均值的降低率有所增大. 在表 1 场景中, 订单数量为 100、设备容量为 75 的案例类型中, HMA 算法解的平均值的降低率为 18.72 %, 是平均降低率的 2.4 倍. 在表 2 场景中, 同样在最大订单数量、最大容量所对应的案例类型中, 本文算法取得了最大降低率, 是平均降低率的 2.3 倍. 在表 1、表 2 场景中, HMA 算法的计算时间增大了 65.15 % 与 42.37 %.

5　结　论

本文讨论了订单分批问题. 首先, 设计了混合元启发式算法, 融合了自适应大邻域搜索与不可行下降去优化订单分配; 其次, 为了优化拣选路径, 设计了新的路径策略: 单向启发式; 在新的返回方式上, 优化组

合了多种基本路径策略. 基于标准数据集,将本文算法与已有优秀算法进行了比较实验,在1 280个案例中发现了1 274个已知最优解,其中匹配了6个已知最优解. 本文算法在求解质量上优于已有算法,尤其在面对高订单数量与高设备容量时,本文算法在求解质量上具有极大的优势. 而且,本文算法的计算时间为4.63～198.09 s,保持在合理范围内.

参考文献(References)

[1] Petersen C G I, Schmenner R W. An evaluation of routing and volume-based storage policies in an order picking operation[J]. Decision Sciences, 1999, 30(2): 481-501.

[2] Dyckhoff H, Lackes R, Reese J. Supply chain management and reverse logistics[M]. Berlin: Springer, 2004: 323-347.

[3] Tompkins J A, White J A, Bozer Y A, et al. Facilities planning[M]. NewJersey: John Wiley & Sons, 2010: 385-447.

[4] Yu Y G, de Koster R B M, Guo X L. Class-based storage with a finite number of items: Using more classes is not always better[J]. Production and Operations Management, 2015, 24(8): 1235-1247.

[5] Manzini R. Ware housing in the global supply chain[M]. London: Springer, 2012: 105-137.

[6] Masae M, Glock C H, Grosse E H. Order picker routing in warehouses: A systematic literature review[J]. International Journal of Production Economics, 2020, 224: 107564.

[7] Cergibozan C, Tasan A S. Order batching operations: Anover view of classification, solution techniques, and future research[J]. Journal of Intelligent Manufacturing. 2019, 30(1): 335-349.

[8] Gademann N, Velde S. Order batching to minimize total travel time in a parallel-aisle warehouse[J]. IIE Transactions, 2005, 37(1): 63-75.

[9] Muter İ, Öncan T. An exact solution approach for the order batching problem[J]. IIE Transactions, 2015, 47(7): 728-738.

[10] de Koster M B M, Vander Poort E S, Wolters M. Efficient order batching methods in ware houses[J]. International Journal of Production Research. 1999, 37(7): 1479-1504.

[11] Albareda-Sambola M, Alonso-Ayuso A, Molina E, et al. Variable neighborhood search for order batching in a warehouse[J]. Asia-Pacific Journal of Operational Research, 2009, 26(5): 655-683.

[12] Henn S, Wäscher G. Tabu search heuristics for the order batching problem in manual order picking systems[J]. European Journal of Operational Research, 2012, 222(3): 484-494.

[13] Öncan T. MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem[J]. European Journal of Operational Research, 2015, 243(1): 142-155.

[14] Menéndez B, Pardo E G, Alonso-Ayuso A, et al. Variable neighborhood search strategies for the order batching problem[J]. Computers & Operations Research, 2017, 78: 500-512.

[15] Žulj I, Kramer S, Schneider M. A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem[J]. European Journal of Operational Research, 2018, 264(2): 653-664.

[16] Scholz A, Henn S, Stuhlmann M, et al. A new mathematical programming formulation for the single-picker routing problem[J]. European Journal of Operational Research, 2016, 253(1): 68-84.

[17] 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.

[18] Petersen C G II. An evaluation of order picking routeing policies[J]. International Journal of Operations & Production Management, 1997, 17(11): 1098-1111.

[19] Hall R W. Distance approximations for routing manual pickers in a ware house[J]. IIE Transactions, 1993, 25(4): 76-87.

[20] Roodbergen K J, Koster R. Routing methods for warehouses with multiple cross aisles[J]. International Journal of Production Research, 2001, 39(9): 1865-1883.

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

[22] Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem[J]. Computers & Operations Research, 2004, 31(12): 1985-2002.

[23] Lacomme P, Prins C, Ramdane-Chérif W. Evolutionary algorithms for periodic arc routing problems[J]. European Journal of Operational Research, 2005, 165(2): 535-553.

[24] Glover F. Future paths for integer programming and links to artificial intelligence[J]. Computers & Operations Research, 1986, 13(5): 533-549.

作者简介

吴仁超(1989−), 男, 博士生, 从事系统优化与调度的研究, E-mail: 154601004@csu.edu.cn;

贺建军(1965−), 男, 教授, 博士生导师, 从事建模、控制与优化复杂系统和生产过程等研究, E-mail: jjhe@csu.edu.cn;

李欣(1986−), 女, 博士生, 从事复杂系统建模与优化、机器视觉监控与诊断的研究, E-mail: 154601005@csu.edu.cn;

殷泽阳(1993−), 男, 讲师, 博士, 从事飞行器动力学建模、智能决策及先进制导与控制等研究, E-mail: yinzeyang@csu.edu.cn;

陈祖国(1990−), 男, 讲师, 博士, 从事过程控制、人工智能、网络控制系统和智能优化以及知识驱动知识化等研究, E-mail: zg.chen1@siat.ac.cn.

(责任编辑：李君玲)