文章编号: 1001-0920 (2017) 01-0039-08 DOI: 10.13195/j.kzyjc.2015.1230

离散型帝国竞争算法在仓储订单调度中的应用

颜 波†, 刘 巳, 黄燕红

(华南理工大学 经济与贸易学院，广州 510006)

摘 要: 针对自动化立体仓储的订单批量处理过程中的优化调度问题, 采用离散型帝国竞争算法对订单调度流程进行求解, 构建出库订单优先调度和复合订单调度两种对比流程来进行优劣势的探讨以及算法比较. 调度实例求解结果表明, 所提出算法的优化质量更高, 且在处理大批量复合订单时更具有优越性, 从而验证了所提出算法的有效性和优越性.

关键词: 自动化立体仓储；帝国竞争算法；订单调度流程

中图分类号: TP18 **文献标志码:** A

Application of discrete imperialist competitive algorithm in warehouse order scheduling

YAN Bo†, *LIU Si*, *HUANG Yan-hong*

(School of Economics and Commerce，South China University of Technology，Guangzhou 510006，China)

Abstract: The discrete imperialist competitive algorithm is presented to solve the optimization scheduling problem of automated warehouse, when it is in the batch processing of orders. Outbound order priority scheduling and compound orders scheduling have been built and are solved by using discrete imperialist competitive algorithm(ICA). Finally, several examples are given and the computational results show that the presented approach can obtain a better optimal solution and has advantages in dealing with large quantities of orders scheduling.

Keywords: automated warehouse；imperialist competitive algorithm；order scheduling

0 引 言

自动化立体仓库是一种用于存放货物的多层高架仓储系统, 系统处理多批量订单时采取的调度策略、建模分析方法以及求解算法在一定程度上影响着现代仓储调度控制与决策的发展方向. 路径优化、订单任务排序等是自动化立体仓库调度的重点, 国内外学者采用各种元启发式算法对此进行了应用研究. Sobeyko 等[1]、Lorenzo 等[2]、Oliveira[3]分别以加权拖期和加权拖期订单任务数最少、路径和流量调度最优、总任务拖期最短为目标函数建立调度模型, 用遗传算法进行求解, 表明其优化策略能有效解决优化调度问题; Jalilvand-Nejad 等[4]将遗传算法与模拟退火算法相结合, 解决了循环调度作业问题; Mati 等[5]提出一种基于贪心算法和遗传算法的调度算法, 对多资源调度问题进行了相关研究; 马丁等[6]提出了一种带启发式规则的遗传模拟退火两阶段算法, 对相关生产调度实例进行了研究.

目前, 多数学者在建模时通常考虑单个货物出入库的最短时间, 然而实际运作中, 企业常以顾客订单为执行对象; 同时, 订单的完成时间为订单内所有出/入库任务全部完成的时间, 其中单个出/入库任务的提前完成对总订单的最终完成时间影响不大, 但订单中某个任务的延误则会造成整个订单的拖期. 因此, 本文以最小化出入库订单的完成时间和出库订单的总拖期为优化目标, 建立多目标调度模型.

在自动化立体仓储订单调度求解算法的选择上, 由于订单在批量、客户重要程度以及时间等方面不同, 调度问题的求解复杂性远大于传统问题. 目前的算法大多局限于启发式方法和智能优化方法. 启发式方法因其原理简单、求解速度快的特点而被人们广为

收稿日期: 2015-10-09；**修回日期:** 2016-03-03.

基金项目: 广东省软科学研究计划项目(2013B070206013, 2015A070704005)；广东省科技计划项目(2013B040500007, 2013B040200057)；中央高校基本科研业务费专项资金项目(2015XZD14, 2015KXKYJ02).

作者简介: 颜波 (1970−), 男, 教授, 博士生导师, 从事供应链优化与协调等研究；刘巳 (1992−), 男, 硕士生, 从事运营与供应链管理、优化与规划的研究.

†通信作者. E-mail: yanbo@scut.edu.cn

接受, 尤其是动态调度方面[7]. 帝国竞争算法在解决大规模组合优化问题上具有一定的优越性, 并且, 与其他算法相比, 帝国竞争算法还同时考虑到求解问题内部和外部的所有约束, 将研究对象外部资源的可用性和成本以及内部的资源约束都考虑在内. 对于自动化立体仓库系统而言, 鉴于立体仓库的自动化处理与订单排序的耦合性, 对于不同订单调度流程问题, 帝国竞争算法在此类经济批量调度问题的分解策略与相关规则的研究中具有较大的优越性.

近年来, 越来越多的学者对帝国竞争算法进行设计, 用于求解车间调度问题[8]、集成产品的混合外包优化[9]、肤色检测[10]等问题. 帝国竞争算法在求解连续型或离散型问题上均具有良好的搜索效果和适用性[11], 同时, 相比较其他算法而言, 帝国竞争算法是一种已在连续优化问题上取得较好效果的新型社会政治算法. 而订单调度问题属于典型的离散型连续优化问题和NP-Hard问题. 为了充分发挥帝国竞争算法在组合目标连续优化领域的寻优能力, 在上述研究的基础上, 本文对帝国竞争算法进行离散化设计, 通过引进“汉明距离”提出一种新型的求解离散型问题的帝国竞争算法, 对多批量的不同订单调度流程分别求解. 实验结果表明, 所提出算法在求解组合目标连续优化问题时求解质量更高, 在处理大批量订单时更具有优越性.

1 模型说明

单一订单类型作业模式是指, 在执行一组出入库订单任务时, 出库订单和入库订单分开完成. 本文根据企业的实际情况, 采取出库任务优先调度模式.

复合订单调度模式是指, 当堆垛机有出库任务时, 查找对应的入库台是否有入库任务, 若有入库任务, 则可把一组出入库任务“捆绑”, 堆垛机执行入库任务后不是返回出口, 而是直接移库执行出库任务, 待出库货物运送至出库台完成出库时完成一次复合作业模式.

根据上述理论基础, 可以构建如下数学模型:

$$\min\ \omega\sum_{l\in O}\max\ C_l+(1-\omega)\sum_{l\in O_1}\max\ \{0,C_l-d_l\}. \tag{1}$$

s.t.

$$\sum_{g\in M_k}u_{i,k,g}=1,\ \forall i\in I,\ \forall k\in\{1,2\}; \tag{2}$$

$$v_{i',i,k,g}+v_{i,i',k,g}\leqslant\frac{1}{2}(u_{i,k,g}+u_{i',k,g}),$$
$$\forall i\in I,\ \forall g\in M,\ \forall k\in\{1,2\}; \tag{3}$$

$$v_{i',i,k,g}+v_{i,i',k,g}\geqslant u_{i,k,g}+u_{i',k,g}-1,$$
$$\forall i\in I,\ \forall g\in M,\ \forall k\in\{1,2\}; \tag{4}$$

$$w_{i,i',k,g}\leqslant v_{i,i',k,g},$$
$$\forall i\in I,\ \forall g\in M,\ \forall k\in\{1,2\}; \tag{5}$$

$$\sum_{i,i'\in I}w_{i,i',k,g}=\sum_{i}u_{i,i',k,g}-1,$$
$$\forall i\in I,\ \forall g\in M,\ \forall k\in\{1,2\}; \tag{6}$$

$$c_{i,1}-p_{i,1}\geqslant b_{L_i},\ \forall i\in I,\ \forall l\in O; \tag{7}$$

$$c_{i,2}-p_{i,2}-c_{i,1}\geqslant b_{L_i}-(1-w_{i',i,k,g})B,$$
$$\forall i\in I,\ \forall l\in O; \tag{8}$$

$$c_{i',g}-p_{i,g}-s_{i,i',k,g}\geqslant c_{i,g}-(1-w_{i,i',k,g})B,$$
$$\forall i\in I,\ \forall g\in M_k; \tag{9}$$

$$u_{i,1,g}=A_{i,x},\ \forall i\in I,\ \forall g\in M_1; \tag{10}$$

$$C_l=\max_{L_i=l,i\in I}c_{i,2},\ \forall i\in I,\ \forall l\in O. \tag{11}$$

本模型中使用的数学符号定义如下:

1) 订单调度任务分为两个阶段: 叉车作业阶段和堆垛机作业阶段, k为执行阶段的索引. 因为出库订单是先堆垛机操作继而叉车操作, 而入库订单恰好相反, 所以出/入库订单的操作阶段分成4种标识: $k=1$代表出库订单的堆垛机作业阶段, $k=2$代表出库订单叉车作业阶段, $k=3$代表入库订单的叉车作业阶段, $k=4$代表入库订单堆垛机的作业阶段, $k\in\{1,2,3,4\}$.

2) i为货物出库作业任务的集合, $i\in I$为出库任务的索引; j为货物入库作业任务的集合, $j\in J$为入库任务的索引.

3) O为出库订单和入库订单的集合, $l\in O$为订单的索引. 其中: O_1为出库订单集合, O_2为入库订单集合.

4) b_l为订单l下达的时间; d_l为订单l的期望交付的时间, 入库订单的交付期为$+\infty$; C_l表示订单l的实际完成时间; $C_{i,k}$、$C_{j,k}$分别表示出库任务i或入库任务j的第k阶段的完成时间.

5) M为所有堆垛机和叉车(设备)的集合, $g\in M$为设备的索引, $M_{1,4}$为巷道堆垛机的数量, $M_{2,3}$为叉车的数量.

6) L_i表示出库任务i对应的订单编号, $L_i\in O$; L_j表示入库任务j对应的订单编号, $L_j\in O$.

7) $P_{i,k}$和$P_{j,k}$分别表示出库任务i和入库任务j在第k阶段的操作时间, 操作时间的长短与设备的运行速度和货物存放位置有关.

8) $S_{i,k,g}$和$S_{j,k,g}$表示出库任务i或入库任务j在g设备上的准备时间.

9) A表示货物存放的位置, $A\in\{x,y,z\}$表示货

物存放在立体货架的 x 行、y 列和 z 层.

10) B 表示一个极大的常数.

11) ω 为目标函数权重.

本模型中使用的决策变量的定义如下:

1) $u_{i,k_1,g}$ 和 $u_{j,k_2,g}$ 值为 1 分别表示出库任务 i 的第 k_1 阶段或入库任务 j 的第 k_2 阶段的操作是在设备 g 上完成的; 为 0 表示其他. $i,i'\in I$, $j,j'\in J$, $k_1\in\{1,2\}$, $k_2\in\{3,4\}$, $g\in M_g$.

2) $V_{i',i,k_1,g}$ 和 $V_{j',j,k_2,g}$ 值为 1 分别表示出库任务 i 和 i' 的第 k_1 操作阶段和入库任务 j 和 j' 的第 k_2 操作阶段都是在设备 g 上完成的, 且 i'、j' 分别先于任务 i、j 执行 (不一定连续); 为 0 表示其他.

3) $W_{i',i,k_1,g}$ 和 $W_{j',j,k_2,g}$ 值为 1 分别表示出库任务 i 和 i' 的第 k_1 操作阶段和入库任务 j 和 j' 的第 k_2 操作阶段都是在设备 g 上完成的, 且 i'、j' 分别先于任务 i、j 执行 (连续的); 为 0 表示其他.

本文订单调度模型以订单的总完成时间最小及出库订单的拖期最短为优化目标. 约束 (2) 规定每个出/入库订单货物的每个阶段在且仅在一台堆垛机/叉车上执行; 约束 (3)、(4) 规定了两个任务同在一台设备上执行时的条件: 若两个任务都在同一台设备执行, 则两个任务有先后关系; 约束 (5) 为两个任务连续在同一台设备执行的条件; 约束 (6) 表示当设备有 n 个任务时, 该设备有 $n-1$ 个连续的任务; 约束 (7) 表示出库订单任务的开始时间不得早于该任务对应订单的下达时间; 约束 (8) 规定了第 1 阶段的任务执行结束后才能开始执行第 2 阶段的任务; 约束 (9) 规定了当堆垛机或叉车执行两个顺序任务时, 第 2 个任务需等前一个任务完成才能开始; 约束 (10) 规定出入库订单堆垛机的操作只能由货物存储货位对应的巷道堆垛机完成; 约束 (11) 规定了出库订单中所有货物的出库完成时间为订单的完成时间.

2　基于离散型帝国竞争算法的模型求解

2.1　帝国竞争算法离散化设计

仓库订单调度问题属于离散型组合优化问题, 离散型帝国竞争算法的最大变动在于帝国对殖民地的同化过程. 帝国同化的目的是使殖民地更靠近帝国, 而且为了保证国家的多样性, 需要进行一定程度的革命. 因此, 借鉴遗传算法的交叉操作可以实现殖民地向帝国的移动, 从而帝国所代表的解中包含的优势片段能传播到殖民地中, 增加对殖民地的影响力, 最终增强整个帝国集团的势力.

为了使殖民地更好地被同化, 本文引入“汉明距离”作为选择交叉点的约束. “汉明距离”是指在一个码组中, 任意两个码字之间对应位置上码元取值不同的位的数目, 即 $d(x,y)=\sum x[i]\oplus y[i]$, 这里的 $i=1,2,\cdots,n$, 且 x 和 y 都是 n 位的编码, $\oplus$ 表示求异. 例如 $x=[1,0,2,3,1,3,0,1]$, $y=[1,1,2,0,1,3,2,1]$, 则 $d(x,y)=3$, 而“汉明距离”值的设定可以在多次实验中确定. 本文设定殖民地与帝国之间交叉编码数 n_{cross} 必须大于等于编码长度 n_{code} 与“汉明距离” $d(c_{\text{imp}},c_{\text{ncol}})$ 的差, 即

$$n_{\text{cross}}\geqslant n_{\text{code}}-d(c_{\text{imp}},c_{\text{ncol}}). \tag{12}$$

通过交叉后, 为了保证殖民地所代表解的多样性, 防止帝国竞争算法过早陷入局部最优, 殖民地需要进行革命. 帝国竞争算法的革命过程可以借鉴遗传算法的变异操作, 即对交叉后的殖民地进行自身交换、插入和单点变异.

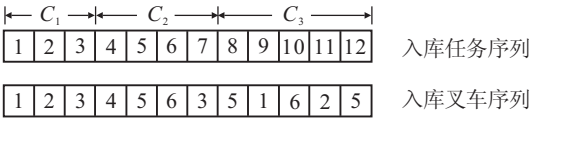
2.2　帝国竞争算法应用设计

2.2.1　初始化编码

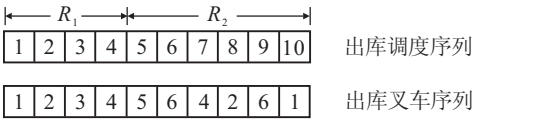
每个任务由其库位对应的堆垛机执行, 堆垛机调度不需要进行编码, 任务顺序调度和叉车调度组成帝国竞争算法初始化的双重编码或三重编码, 一个编码代表一个国家.

帝国竞争算法初始化的每个国家表示订单调度问题的一个解, 订单调度编码的每个解包括两部分: 货物调度顺序部分和叉车调度部分. 货物调度顺序部分表示所有订单货物的执行顺序, 叉车调度部分表示每个货物对应的执行叉车编号.

为了说明初始化编码规则, 先假设仓库叉车数量为 4 辆, 对应的编号分别为: 1, 2, 3 和 4; 再假设 3 个入库订单和 2 个出库订单: 入库订单 1 (3 个任务)、入库订单 2 (4 个任务)、入库订单 3 (5 个任务)、出库订单 1



(a)　出库订单优先调度流程出库任务编码方法



(b)　出库订单优先调度流程入库任务编码方法

(c)　复合订单调度流程出入库任务编码方法

图 1　初始化编码规则

(4 个任务)、出库订单 2 (6 个任务).

出库订单优先调度模型与复合订单调度模型出库订单的编码规则一样, 两种订单调度模型具体的初始化编码规则如图 1 所示.

出库优先调度根据出入库任务不同分开编码, 复合调度同样对出库任务和入库任务分开编码, 但是入库任务的编码依据出库任务编码, 采取“搭顺风车”的方式.

2.2.2 适应度函数及解码

本文以订单的总完成时间和出库订单总拖期最小化为优化目标, 算法中各个国家的成本值均可以通过下式计算:

$$\min\ g(l) = \varpi \sum_{l\in O} \max\ C_l + (1-\varpi)\sum_{l\in O} \max\ \{0, C_l - d_l\}. \quad (13)$$

帝国竞争算法中的国家成本越小, 则势力越大, 因此, 本文帝国竞争算法的国家势力(适应度函数) c_n 与数学模型的目标函数成反比. 为了避免分母为 0 造成算法失效, 将分母加上 1, 即适应度函数为

$$c_n = \frac{1}{g(l)+1} = \frac{1}{\varpi \sum_{l\in O} \max C_l + (1-\varpi)\sum_{l\in O_1} \max\{0, C_l - d_l\} + 1}. \quad (14)$$

为了计算各个国家的适应度函数, 需要对初始化的编码进行解码. 解码是将国家编码转化为调度解的过程. 根据构建的订单调度模型, 解码时用到的符号及含义如下所示:

M_k 表示各个阶段 k 的设备的集合; I_k 表示任务 i 的第 k 阶段; g 表示在任务阶段 I_k 可供选择设备的数字编号; $P_{i,k,g}$ 表示任务阶段 I_k 在设备 g 上的执行时间, 且 $P_{i,k,g} > 0$; $\mathrm{SS}_{i,k}$ 表示任务阶段 I_k 的最早允许开始时间; $S_{i,k}$ 表示任务阶段 I_k 的最早开始时间; $C_{i,k}$ 表示任务阶段 I_k 的最早结束时间, 且 $C_{i,k} = S_{i,k} + P_{i,k,g}$.

根据调度数学模型的约束条件, 本文采用的解码步骤如下.

Step 1: 根据每个出入库任务和堆垛机/叉车序列决定出入库时每个阶段对应的操作设备.

Step 2: 对于每一个执行阶段, 有 $\mathrm{SS}_{i,k} = C_{i,k-1}$, 其中 $C_{i,k-1}$ 表示任务 i 早于任务阶段 I_k 执行的结束时间.

Step 3: 查找执行过程 I_k 对应叉车或堆垛机的空闲时间, 记录设备的一组空闲时间段 $[t_1, t_2]$. 对于当前的时间段, 若 $\max(\mathrm{SS}_{i,k}, t_1) + p_{i,k,g} < t_2$, 则有 $S_{i,k} = \max(\mathrm{SS}_{i,k}, t_1)$; 否则, 检查下一个时间段. 如果所有时间段都不满足要求, 则该阶段的执行需要等待, 等待后的开始时间为 $S_{i,k} = \max[(S_{i,k}, C(I_k - 1))]$, 其中 $C(I_k - 1)$ 是某台堆垛机或叉车前一个任务的结束时间.

Step 4: 每个执行阶段的结束时间为 $C_{i,k} = S_{i,k} + P_{i,k,g}$.

Step 5: 计算叉车或堆垛机执行阶段的开始时间以及结束时间, 根据出入库执行环节的不同分别计算某个货物总的执行时间.

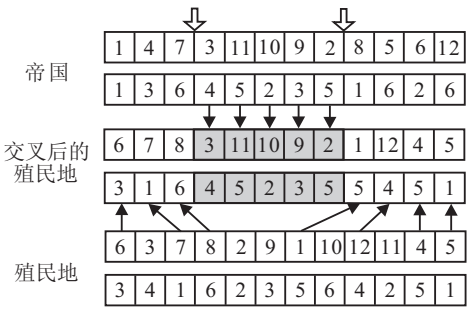
Step 6: 订单总的完成时间为该订单最后一个任务的完成时间, 计算每个订单总的完成时间; 对于出库订单, 计算每个出库订单的拖期.

2.2.3 帝国集团同化及更新

帝国初始化 N 个国家, 选取 M 个势力最大的国家作为帝国, 其余 $N - M$ 个国家按照一定的概率分配给 M 个帝国作为殖民地. M 个帝国及其所属的殖民地组成 M 个帝国集团. 帝国会对其殖民地进行同化, 本文的殖民地同化包括两部分: 帝国与殖民地之间的确定性交叉以及殖民地自身的确定性变异.

1) 帝国与殖民地之间的交叉.

出库订单优先调度模型中, 先执行出库订单任务, 再执行入库订单任务, 因此, 出库和入库调度的编码是独立的, 其帝国与殖民地之间的交叉也是独立的.



(a) 出库订单优先调度编码交叉

(b) 复合订单调度编码交叉

图 2 调度编码交叉规则

交叉步骤如下.

Step 1: 在帝国编码中随机选择两个交叉点;

Step 2: 将帝国编码中所选交叉点间的片段复制到殖民地对应的位置, 并将原来的片段从殖民地编码中去掉;

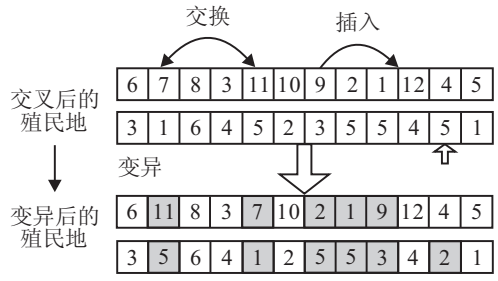
Step 3: 将殖民地编码中剩余的片段按顺序复制到新殖民对应的位置上.

具体如图 2 所示.

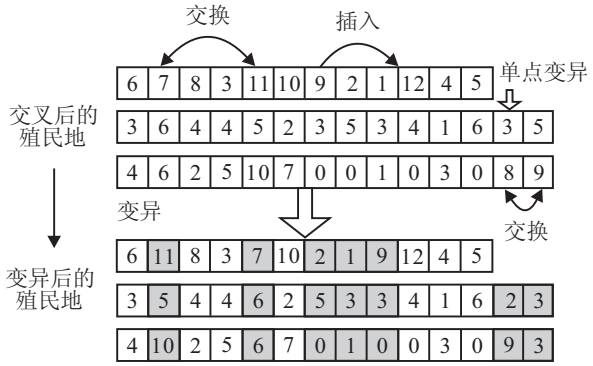
在复合订单调度编码的交叉操作中, 出库订单和入库订单一起进行交叉.

2) 殖民地变异.

本文采取变异的方法包括交换、插入和单点变异, 出库订单优先调度和复合订单调度编码的变异过程分别如图 3 所示. 交换和插入操作: 随机选择复合订单调度编码的出入库和叉车序列片段, 然后交换它们的位置或者将其中一个编码序列插入到另一个编码位置后面; 单点变异在本文仅发生于双重编码/三重编码的第 2 行序列 (叉车序列).



(a) 出库订单优先调度编码变异



(b) 复合订单调度编码变异

图 3　调度编码变异规则

通过计算 "汉明距离" 来实现帝国集团内部的同化效果: 若 "汉明距离" 大于设定值, 则该殖民地需要重新同化; 若 "汉明距离" 小于设定的阈值, 则殖民地同化成功.

当帝国集团内部同化完成后, 对帝国和殖民地进行检验, 符合交换条件的进行更新. 交换条件为: 殖民地经过同化到达一个新的位置后, 若其成本值比所属帝国的成本值小, 则交换殖民地和帝国的位置, 即殖民地成为该集团的帝国, 而原来的帝国则沦为殖民地.

2.2.4　帝国集团竞争

帝国集团竞争前需要计算总成本值, 成本值决定了各个帝国集团势力的大小. 帝国集团的总成本值主要由帝国的成本和殖民地的成本构成, 而且二者的成本对帝国集团的影响程度不相同, 因此, 通过构建目标函数来计算一个帝国集团的势力, 即

$$\mathrm{T.C.}_n = \mathrm{Cost}(\mathrm{imperialist}_n) + \xi \mathrm{mean}\{\mathrm{Cost}(\mathrm{colonies\ of\ empire}_n)\}. \quad (15)$$

其中: $\mathrm{T.C.}_n$ 是第 n 个帝国集团的总成本; ξ 值用来度量殖民地成本在整个帝国集团成本中占的比重, 即 ξ 越大, 殖民地成本占的比重越大, 本文 $\xi = 0.1$.

在帝国集团竞争阶段, 竞争对象的选择方法是: 最弱的帝国释放出其最弱的殖民地, 其他帝国进行竞争并以一定的概率获得该殖民地. 一般而言, 势力越大的帝国越有可能占有该殖民地.

2.2.5　帝国删除及算法终止

由于每次最弱迭代帝国的殖民地会被其他更强的帝国以一定概率夺走, 当这个帝国拥有的殖民地少于预先设定的阈值 α 时 (本文取阈值 $\alpha = 0$), 该帝国将被删除. 由于不断迭代使得最弱的帝国被删除, 当

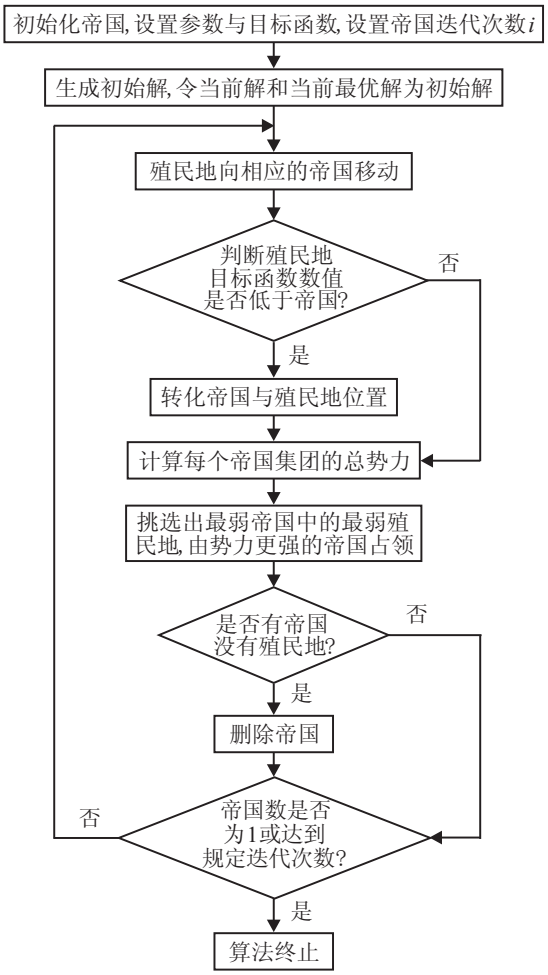


图 4　算法流程

多次迭代后只剩下一个帝国时, 算法终止且输出剩下的帝国作为算法的最优解. 另外, 设定最大迭代次数(本文为500次), 当算法迭代达到最大迭代次数时, 算法终止. 通过计算剩余帝国国家的势力值, 选取势力最大的帝国作为算法的最优输出结果.

为了便于理解算法的过程, 图4给出了利用离散型帝国竞争算法求解的具体流程.

3 实例分析

3.1 实例背景介绍

某从事家电生产及销售的制造企业, 采用帝国竞争算法对所构建的两种流程进行仿真, 参考Nourmohammadia[12]等的文献及通过测试综合考虑帝国竞争算法(ICA)的求解质量和求解速度, 本文最佳参数设置为: 帝国/国家数量是10/300; 殖民地影响因子是0.2; 迭代次数是500, 汉明距离编码长度是0.5.

作为对比, 遗传算法(GA)在选择染色体变异或交叉时, 采取轮盘赌的方式. 根据文献[7], 经多次测试后, 遗传算法用于求解本文订单调度问题的最佳参数为: 迭代次数500, 种群数40, 变异率和交叉率分别为0.4和0.01.

通过模拟该企业某仓库的出入库数据, 得到该仓库的3个入库任务和8个出库任务的订单参数值. 在实际操作中, 根据订单参数及自动化立体仓库布局自动生成各个订单的货位坐标, 采用Matlab保存为.mat文件. 帝国竞争算法求解两种调度流程时均调用该货位坐标.mat文件, 共记录11个订单(307个出入库任务)的货位坐标.

3.2 调度流程仿真结果及分析

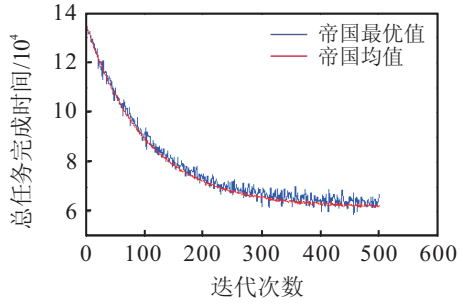
采用帝国竞争算法和遗传算法分别对出库订单优先调度流程和复合订单调度流程进行20次的仿真实验, 分别记录8个出库订单和3个入库订单的最优完成时长、总体执行时间(AT)、拖期(OT)以及仿真的收敛速度. 具体数据记录如表1所示.

表1 订单优化结果

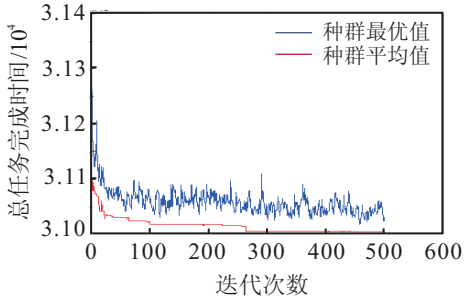
项目		出库订单优先调度			复合订单调度		
		ICA	GA	Δ	ICA	GA	Δ
完成时长	OC1	844	853	9	921	978	57
	OC2	1 184	1197	13	1 236	1 341	105
	OC3	504	511	7	574	681	107
	OC4	1 799	1 814	15	1 902	2 002	100
	OC5	2 390	2 408	18	2 050	2 075	25
	OC6	2 065	2 083	18	1 692	1 775	83
	OC7	3 325	3 353	28	2 991	3 018	27
	OC8	2 766	2 753	−13	2 343	2 389	46
	OR1	2 399	2 413	14	2 220	2 259	39
	OR2	2 203	2 273	70	1 800	1 893	93
	OR3	3 740	3 756	16	3 408	3 447	39
AT		4 940	4 956	16	4 608	4 674	66
OT		3 261	4 026	765	2 542	2 983	441
收敛速度		316	136		390	248	

3.2.1 收敛速度对比

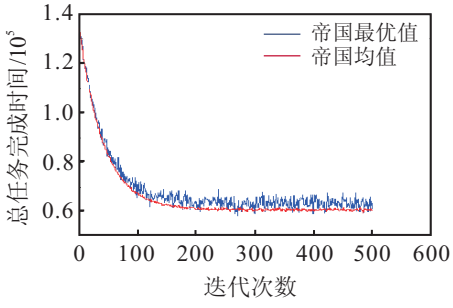
当优化算法运行到一定代数后, 算法的目标值趋于平稳, 多次迭代得到的解的差异小于某一数值, 此时可以认为该算法已开始收敛, 所对应的时间或迭代次数可作为收敛速度的评价标准. 图5分别为对应不同调度的算法迭代曲线.



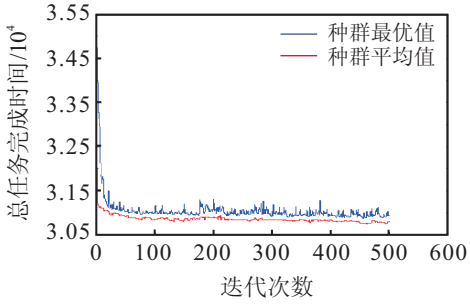
(a) 帝国竞争算法出库订单优先调度迭代曲线



(b) 遗传算法出库订单优先调度迭代曲线



(c) 帝国竞争算法复合调度迭代曲线



(d) 遗传算法复合订单调度迭代曲线

图5 算法迭代曲线

由仿真结果和表 1 可知: 无论哪种调度流程, 离散型帝国竞争算法的收敛相对较慢; 对于出库订单优先调度, 帝国竞争算法的收敛所耗时间约是遗传算法的 2.3 倍; 同样, 在复合调度中所耗时间约是遗传算法的 1.58 倍. 其原因主要在于搜索机制的差异: 遗传算法是基于全局的随机搜索, 通过选择合适的变异率和交叉率可以调整算法的搜索范围, 搜索到更优的解; 帝国竞争算法是基于帝国国家的定向搜索, 初始化后围绕着帝国国家进行同化, 更优的解只能在算法的同化阶段产生并通过帝国更新过程才能获得. 因此, 帝国竞争算法获得新的最优解的过程较为谨慎且间隔比遗传算法长, 造成了帝国竞争算法的收敛速度较慢.

3.2.2　算法求解结果对比

优化算法的求解结果主要有以下参数: 各个出入库订单的执行时间、总体执行时间、出库订单总拖期. 通过表 1 对比帝国竞争算法和遗传算法的求解结果, 可以比较两种算法的优化质量.

可以看出, 对于两种调度流程的仿真, 帝国竞争算法的优化结果比遗传算法的优化结果优越. 当调度相同时, 帝国竞争算法的总体执行时间比遗传算法优越但领先不多, 帝国竞争算法的优越性主要表现在拖期上. 从表 1 的数据可以看出: 在出库任务优先调度中, 帝国竞争算法的拖期比遗传算法少 765 个时间单位; 同样, 在复合订单调度中, 帝国竞争算法的拖期也要少 441 个时间单位; 其减少拖期比率分别约为遗传算法的 19.0% 和 14.8%. 从这些数据可以看出, 使用帝国竞争算法可以略微减少总体执行时间, 但可以大幅度地减少任务拖期, 帝国竞争算法在减少拖期方面相对具有明显的优越性.

相对于遗传算法, 离散型帝国竞争算法在本文的应用能得到更好的求解结果的原因在于:

1) 帝国竞争算法在执行时采取大范围覆盖的“定向搜索 + 随机同化”的方式, 在不破坏原有多个“较优解”的基础上寻找一定范围内可能的更优解. 在该算法的初始化中, 每个帝国国家代表一个“较优解”并在某些阶段分别进行优化, 扩大了算法的搜索范围, 避免陷入局部搜索. 每个殖民地都需要进行随机同化并可能产生更优解. 帝国国家以及同化的殖民地越多, 搜索到更优解的几率越大, 同时, 算法的运行速度越慢.

2) 遗传算法的过早局部收敛, 也称为“早熟现象”, 会导致其在未找到最优解时陷入一个局部最优解, 影响算法的求解质量. “早熟现象”是遗传算法一直存在且难以解决的问题, 产生的原因在于遗传算法的交叉和变异过程中有较强的随机性, 使得其有较好的全局搜索能力但是局部搜索能力较弱. 另外, 遗传算法难以适应由于变异引起的搜索空间的变化, 使得一些优秀个体被过早排除掉, 从而过早陷入局部收敛.

3.2.3　订单调度流程对比

对比表 1 中帝国竞争算法求解得出的出库订单优先调度流程和复合订单调度流程的结果, 可以得到两种调度流程的优缺点:

1) 采用出库订单优先调度时, 出库订单 (OC1、OC2、OC3) 在没有入库订单干扰下可获得与复合调度流程调度下类似甚至更优的解. 当有入库订单需要执行时, 采用出库订单优先调度方案会使插入后的入库订单以及后续的出库订单完成时间大大增加.

2) 采用复合订单调度流程可从整体上合理分配仓库各设备的资源, 提高设备利用率并缩短总体订单的执行时间以及出库订单总拖期. 当出入库订单数量较多时, 复合订单调度流程更能显示出其优势.

把案例中的出入库订单数量扩展到 15 个出库订单和 5 个入库订单, 采用帝国竞争算法分别对出库订单优先调度流程和复合订单调度流程进行求解, 具体数据如表 2 所示.

通过对比可知: 当订单量较少时, 出库优先调度总执行时间较少; 随着订单量的增加, 复合订单的各订单执行时间和总执行时间都相对较少, 且在减少拖期方面存在明显的优越性, 对比于出库优先调度, 复合订单的减少拖期比率约为 36.4%.

在增加出入库的订单数量后, 由求解结果及调度流程分析可知, 当自动化立体仓库的出库订单和入库订单交替下达时, 采取出库订单优先调度和复合订单调度的优缺点在于:

1) 出库订单优先调度优先执行出库订单, 在仓库

表 2　订单优化数据

	OC1	OC2	OC3	OC4	OC5	OC6	OC7	OC8	OC9	OC10	OC11	OC12	OC13	OC14	OC15	OR1	OR2	OR3	OR4	OR5	OT	AT
出库优先	862	1257	550	1992	2137	1800	3163	2550	2179	2040	2449	2040	1773	1349	1510	2573	3633	2802	2482	1854	3547	7854
复合订单	978	1356	602	2123	2046	1683	3086	2458	1875	1763	2189	1868	1662	1085	1302	2402	2758	2390	2215	1629	2256	7629
时间差	−116	−99	−52	−131	91	117	77	92	304	277	260	172	111	264	208	171	875	412	267	225	1291	225

操作中可以更快释放出库备货区, 容纳新的入库订单, 避免造成多个订单混乱.

2) 从整体上看, 当订单量较多时, 采取复合订单调度比出库订单优先调度更能节省时间和合理调配资源. 从表 2 的第 5 个出库订单起, 复合调度下的出库订单的执行时间都比采取出库订单优先调度下的执行时间短.

3) 对比仿真实验数值可知: 尽管在两种调度方案中, 订单总完成时间相差不大, 但随着需要处理的订单量数量的增加, 采用复合订单调度能大大缩短出库订单的总拖期.

4 结 论

本文在仓储调度研究中提出了一种离散型多目标帝国竞争算法, 在特定的编码方式下, 通过设计 “汉明距离” 和对交叉变异操作的编码规则进行离散化改进, 取得了有效的求解性能, 求解质量优于现有的代表算法, 不仅丰富了多目标多批量连续订单调度优化的求解方法, 而且拓宽了帝国竞争算法的应用领域. 通过实际案例的仿真计算结果验证了该调度求解算法的有效性和合理性以及求解质量上的优越性.

参考文献(References)

[1] Sobeyko O, Mönch L. Grouping genetic algorithms for solving single machine multiple orders per job scheduling problems[J]. Annals of Operations Research, 2015, 235(1): 709-739.

[2] Lorenzo B, Glisic S. Optimal routing and traffic scheduling for multihop cellular networks using genetic algorithm[J]. IEEE Trans on Mobile Computing, 2013, 12(11): 2274-2288.

[3] Oliveira J A. Scheduling the truckload operations in automatic warehouse[J]. European J of Operational Research, 2007, 179(3): 723-735.

[4] Jalilvand-Nejad A, Fattahi P. A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem[J]. J of Intelligent Manufacturing, 2015, 26(6): 1085-1098.

[5] Mati Y, Xie X. A genetic-search-guided greedy algorithm for multi-resource shop scheduling with resource flexibility[J]. IIE Trans, 2008, 40(12): 1228-1240.

[6] 马丁, 陈庆新, 毛宁, 等. 具有交货期约束带准备时间的平行机分批调度[J]. 计算机集成制造系统, 2012, 18(1): 111-117.
(Ma D, Chen Q X, Mao N, et al. Parallel machine batch scheduling for due date constraints and setup time[J]. Computer Integrated Manufacturing Systems, 2012, 18(1): 111-117.)

[7] 王凌, 邓瑾, 王圣尧. 分布式车间调度优化算法研究综述[J]. 控制与决策, 2016, 31(1): 1-11.
(Wang L, Deng J, Wang S Y. Survey on optimization algorithms for distributed shop scheduling[J]. Control and Decision, 2016, 31(1): 1-11.)

[8] Goldansaz S M, Jolai F, Anaraki A H. A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop[J]. Applied Mathematical Modelling, 2013, 37(23): 9603-9615.

[9] Shirkouhi S N, Eivazy H, Ghodsi R, et al. Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm[J]. Expert Systems with Applications, 2010, 37(12): 7615-7626.

[10] Razmjooya N, Mousavib B S, Soleymani F. A hybrid neural network imperialist competitive algorithm for skin color segmentation[J]. Mathematical and Computer Modelling, 2013, 57(3): 848-856.

[11] Atashpaz G E, Lucas C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition[C]. IEEE Congress on Evolutionary Computation. Singapore, 2007: 4661-4667.

[12] Nourmohammadia A, Zandiehb M. An imperialist competitive algorithm for multi-objective U-type assembly line design[J]. J of Computational Science, 2013, 4(5): 393-400.

(责任编辑：李君玲)