考虑运输能力限制的跨单元调度方法

刘兆赫 [1] 李冬妮 [1] 王乐衡 [1] 田云娜 [1,2]

摘 要 工件在生产单元之间频繁转移产生了跨单元调度问题. 本文结合我国装备制造业的生产实际, 提出考虑运输能力的跨单元调度方法, 设计了一种基于离散蜂群与决策块结构的超启发式算法. 针对传统超启发式算法的局限性提出动态决策块策略, 同时改进传统蜂群算法的侦查蜂策略, 使之具有更好的优化性能. 实验表明, 动态决策块具有比静态决策块更好的性能, 算法在优化能力和计算效率的综合性能上优势显著, 并且问题的规模越大, 优势越明显.

关键词 跨单元调度, 运输能力, 超启发式, 决策块, 人工蜂群算法

引用格式 刘兆赫, 李冬妮, 王乐衡, 田云娜. 考虑运输能力限制的跨单元调度方法. 自动化学报, 2015, **41**(5): 885−898

DOI 10.16383/j.aas.2015.c140498

An Inter-cell Scheduling Approach Considering Transportation Capacity Constraints

LIU Zhao-He[1] LI Dong-Ni[1] WANG Le-Heng[1] TIAN Yun-Na[1,2]

Abstract The issue of inter-cell scheduling arises due to some exceptional parts having to be processed and transported frequently in different cells. This work is inspired by the equipment manufacturing industry of China. Aimed at the inter-cell scheduling problem with transportation capacity constraints, a hyper-heuristic based on discrete artificial bee colony approach and decision block is proposed, in which a dynamic decision block strategy is developed and the scout bee strategy is improved. Experimental results show that the proposed approach outperforms the traditional static decision block strategies, and has a better performance with respect to the overall performance of optimization capability and computation efficiency, which is especially more suitable for large dimension scheduling problems.

Key words Inter-cell scheduling, transportation capacity, hyper-heuristic, decision block, artificial bee colony algorithm

Citation Liu Zhao-He, Li Dong-Ni, Wang Le-Heng, Tian Yun-Na. An inter-cell scheduling approach considering transportation capacity constraints. *Acta Automatica Sinica*, 2015, **41**(5): 885−898

在我国, 多数装备制造企业已经建立了现代化生产线, 但同时大量传统设备与先进设备共存, 多品种混线生产长期存在. 工件的加工路径跨多个单元, 产生跨单元协作的生产模式[1].

跨单元调度问题的研究不仅仅存在于我国装备制造企业中, 在国外制造业中也会面临同样的问题. 据统计, 有 72% 的企业实施了单元制造模式, 其中工件的平均跨单元率达 20% 以上, 且只有 10% 的单元无需其他单元协作[2−3].

在我国, 以综合传动装置为代表的复杂产品生产中, 有 50% 以上的工件需要跨单元协作完成[4]. 然而, 由于单元间缺乏高效的协同运作管理方法, 导致大量的跨单元工件不能按期完工. 据统计, 在延期工件中跨单元工件的比例高达 70% 以上. 以中国兵器内蒙古第一机械集团有限公司为例, 2011 年由于跨单元工件延期造成的直接经济损失高达 1.3 亿元人民币, 由此产生的在制品库存管理费用接近 5 000 万元人民币. 由此可见, 建立有效的跨单元调度优化方法已成为一个亟待解决的问题.

关于跨单元调度的研究可以分为两类, 即跨流水单元调度 (Flow shop scheduling)[5−7] 和跨作业单元调度 (Job shop scheduling)[4,8−10].

在上述研究中, Mosbah 等[5]、Solimanpur 等[7] 忽略了跨单元转移时间, Yousef 等[6]、Tang 等[8]、Li 等[4,10] 均考虑了跨单元转移时间, 以上研究都隐含着一个假设, 即假设单元间具有充足的运输能力, 因此工件在一个单元内完成相应的工序后, 不需任何等待就能向下一个单元转移. 然而在装备制造业中, 单元的布局分散, 工件的体积、重量无法忽略, 工件的跨单元转移需要由运输工具来完成. 一方面, 在实际生产中, 由于小车的数量有限, 工件在跨单元时需要等待直至小车空闲, 另一方面, 各个工件的目的单元不同. 因此, 有必要在分析跨单元调度问题时考虑

收稿日期 2014-07-21 录用日期 2014-12-04
Manuscript received July 21, 2014; accepted December 4, 2014
国家自然科学基金 (71401014), 北京市自然科学基金 (4122069) 资助
Supported by National Natural Science Foundation of China (71401014), and Beijing Natural Science Foundation (4122069)
本文责任编委 王红卫
Recommended by Associate Editor WANG Hong-Wei
1. 北京理工大学计算机学院智能信息技术北京市重点实验室 北京 100081 2. 延安大学数学与计算机科学学院 延安 716000
1. Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081 2. College of Mathematics and Computer Science, Yan'an University, Yan'an 716000

小车运输能力的限制以及运输过程的路径决策. 跨单元调度问题面向的是由多个单元组成的平面结构, 与传统单元制造系统调度问题相比, 不但具有更高的复杂性, 同时问题规模也急剧增加.

在实际生产中, 启发式规则由于其简单、易实施的特点而得到广泛应用. 然而, 由于使用哪些规则都是根据人为经验事先指定, 不具备优化能力, 因此性能上无法满足复杂问题的需要.

相比之下, 元启发式算法 (如蚁群优化、模拟退火等) 具有更强的优化能力, 适于解决复杂的优化问题. 然而, 由于跨单元调度将问题规模扩展至多个单元组成的单元链, 实际生产中往往需要对数百种工件、上百台机器的规模进行在线调度, 导致元启发式算法的搜索空间激增, 无法在可接受时间内得到调度结果.

基于此, 为了同时获得较好的优化性能和较高的计算效率, 本文采用近年发展起来的超启发式算法解决跨单元调度问题. 超启发式算法是 "搜索启发式的启发式" (Heuristics to search heuristics)[11−12], 即超启发式算法的搜索空间不是调度问题的解, 而是适合于求解调度问题的启发式规则. 这样既避免了人工指定启发式规则的主观性、提高了优化能力, 同时也大幅缩小了搜索空间、提高了计算效率.

如果说启发式规则虽然具有很高的计算效率, 但由于优化能力差而不适用于跨单元调度这样复杂的问题, 元启发式算法虽然优化能力较强, 但由于计算效率低而不能满足大规模问题在线调度的需求. 那么超启发式算法则相当于综合了二者的优势, 即用优化能力较强的元启发式算法选出合适的启发式规则, 再用得到的规则进行高效率的在线调度. 将超启发式的思想应用于跨单元调度的研究目前还很少见, 本文进行了这方面的尝试. 可以预期, 超启发式将为实际生产中大规模复杂优化问题的在线求解提供一条有效途径.

"决策块" 是超启发式算法的决策单位, 即, "决策块" 体现了为多大范围的决策对象选择同一个启发式规则. 在决策块的处理上通常有两类方法: Yang 等[13] 在混合流水车间 (Hybrid flow shop, HFS) 的问题模型下采用超启发式算法, 为每个阶段的所有机器选择同一种启发式规则. Vázquez-Rodríguez 等[14]、Fayad 等[15] 也基于类似的思想. 而 Li 等[16] 则认为, 即使是同一个阶段的机器也有状态的差异, 因此针对 HFS 问题, 通过超启发式算法为每一台机器选择一个启发式规则.

在这两类方法中, 决策块的范围都是事先指定的, 本文称之为 "静态决策块策略". 它们的局限性在于: 前者无法为每一台机器都找到合适的规则, 而后者虽然关注到每台机器, 但却付出更大的计算开销.

基于静态决策块策略的局限性, 本文提出一种 "动态决策块策略", 即由超启发式算法动态生成合适的决策块, 再对生成的决策块应用合适的启发式规则.

在动态决策块策略的基础上, 本文基于改进的人工蜂群 (Artificial bee colony, ABC) 算法设计了一种超启发式算法. ABC 算法是一种新型的群智能算法, 由 Karaboga 于 2005 年提出[17]. 在 ABC 算法中, 寻优过程的划分更为细致, 分为雇佣蜂、观察蜂和侦查蜂三种角色. 这种划分更有利于在搜索的扩张 (Exploration) 和解的利用 (Exploitation) 之间找到平衡, 并且已经有许多成功的应用[18−19]. 因此, 作者基于对 ABC 算法的改进设计了本文的超启发式算法, 针对现有侦查蜂策略容易陷入局部最优的特点进行了改进, 使算法具有更好的性能.

本文的贡献在于: 1) 将新近发展起来的超启发式算法用于跨单元调度问题, 以获得更好的优化能力和更高的计算效率; 2) 在传统超启发式算法的基础上, 提出动态决策块策略, 进一步探求寻优能力和计算开销的平衡; 3) 改进传统蜂群算法的侦查蜂策略, 增强跳出局部最优的能力.

1 问题模型

本文基于跨单元调度的生产实际设计了问题模型, 同时为了便于讨论, 在不影响问题关键特征的基础上, 对模型进行了一定的简化. 本文的问题模型由多个生产单元组成, 其中每个单元均有一台小车, 负责将本单元需要跨单元转移的工件运送至其他单元. 问题模型的特点在于以下两个方面: 1) 工件的加工具有柔性路径, 即工件的每道工序可选机器不唯一; 2) 由于考虑了有限的运输能力, 从而产生了运输维度的子问题, 即工件必须等到小车空闲时才能被运输, 并且由于小车上同一批次的工件目的单元不唯一, 因此要考虑小车的路径规划问题. 由上述分析可见, 生产和运输在跨单元调度中互相影响, 因此, 跨单元调度的实质是跨单元生产和跨单元运输的协同.

1.1 问题假设

本文考虑的运输能力受限的跨单元调度问题基于以下假设:

1) 所有工件在零时刻释放.

2) 每台机器同一时刻只能处理一个工件.

3) 工件的工艺路径由多道工序组成, 至少需要在两个不同的单元里完成.

4) 由于机器的加工能力重叠, 工件存在柔性路径. 但对于任一工序, 在同一单元内至多存在一台可加工的机器.

5) 考虑跨单元转移, 忽略工件在单元内的转移.

6) 所有小车均是一致的, 每个单元有且仅有一辆小车, 且小车存在容量限制.

7) 小车一次运输由多个工件组成的一个批次, 同一批次中各个工件的目的单元可能不同. 小车按照一定顺序访问各个目的单元并卸载对应的工件.

8) 小车仅在本单元装载工件, 从本单元出发后不在沿途各单元装载工件, 仅在工件被运送至目的单元后才能卸载该工件. 装载和卸载工件的时间忽略不计.

9) 一旦所有工件均完成运输, 小车立刻返回起始单元.

10) 其他条件, 例如机器抢占、机器损坏等不在本文考虑范围内.

1.2 符号列表

问题模型中用到的符号定义如下.

1) 索引:

$i=1,\cdots,N$: 工件索引

$j=1,\cdots,o_i$: 工件 i 的工序索引

$c=1,\cdots,C$: 单元及所属小车索引

$m=1,\cdots,M$: 机器索引

$b=1,\cdots,B_c$: 小车 c 上批次索引

$t=1,\cdots,T$: 时间

2) 系统变量:

$o_{i,j}$: 工件 i 的第 j 道工序

o_i: 工件 i 的工序总数

$p_{i,j,m}$: $o_{i,j}$ 在机器 m 上的加工时间 ($p_{i,j,m}=0$ 表示 $o_{i,j}$ 不能在机器 m 上加工)

sz_i: 工件 i 的体积

w_i: 工件 i 的权重

d_i: 工件 i 的交货期

v: 小车容量

$d_{c,c'}$: 单元 c 和 c' 之间所需转移时间

$t_{i,j}$: $o_{i.j}$ 完工后所花费的转移时间

$dc_{i,j}$: $o_{i,j}$ 完工后转移的目的单元

$s_{i,j}$: $o_{i,j}$ 开始时间

$p_{i,j}$: $o_{i,j}$ 实际加工时间

$f_{i,j}$: $o_{i,j}$ 完工时间

$$l_{m,c}=\begin{cases}1, & \text{机器 } m \text{ 位于单元 } c\\ 0, & \text{其他}\end{cases}$$

$$e_{i,j}=\begin{cases}1, & o_{i,j+1} \text{ 和 } o_{i,j} \text{ 被分派至不同单元}\\ 0, & \text{其他}\end{cases}$$

3) 决策变量:

$$x_{i,j,m}=\begin{cases}1, & o_{i,j} \text{ 被分派至机器 } m\\ 0, & \text{其他}\end{cases}$$

$$y_{i,j,r}=\begin{cases}1, & o_{i,j} \text{ 开始加工的时刻为 } r\\ 0, & \text{其他}\end{cases}$$

$$z_{c,b,t}=\begin{cases}1, & \text{小车 } c \text{ 的批次 } b \text{ 在 } t \text{ 时刻运输}\\ 0, & \text{其他}\end{cases}$$

$$u_{i,j,c,b,q}=\begin{cases}1, & o_{i,j} \text{ 完工后被分派至小车 } c \text{ 的批次 } b \text{ 进行运输}\\ 0, & \text{其他}\end{cases}$$

1.3 目标函数及约束条件

本文的调度目标是最小化加权延迟总和 (Total weighted tardiness, TWT). 目标函数的数学表达式如式 (1) 所示.

$$\min\sum_{i=1}^{N}w_i\max\{f_i-d_i,0\} \tag{1}$$

实际生产问题中存在许多特性和约束, 本文采用数学约束进行描述.

$$\sum_{m=1}^{M}x_{i,j,m}=1,\quad \forall i,j \tag{2}$$

$$\sum_{m=1}^{M}x_{i,j,m}p_{i,j,m}>1,\quad \forall i,j \tag{3}$$

$$p_{i,j}=\sum_{m=1}^{M}x_{i,j,m}p_{i,j,m},\quad \forall i,j \tag{4}$$

$$\sum_{t=1}^{T}y_{i,j,t}=1,\quad \forall i,j \tag{5}$$

$$s_{i,j}=\sum_{t=1}^{T}y_{i,j,t}t,\quad \forall i,j \tag{6}$$

$$f_{i,j}=s_{i,j}+p_{i,j},\quad \forall i,j \tag{7}$$

$$x_{i,j,m}y_{i,j,t}\sum_{i'=1}^{N}\sum_{j'=1}^{O_i}\sum_{t'=t}^{t+p_{i,j}}x_{i',j',m}y_{i',j',t'}=0,\quad \forall i,j,t,m \tag{8}$$

$$\sum_{c=1}^{C}\sum_{b=1}^{B}\sum_{t=1}^{T}z_{c,b,t}=1 \tag{9}$$

$$\sum_{c=1}^{C}\sum_{b=1}^{B}\sum_{q=1}^{Q}u_{i,j,c,b,q}=e_{i,j},\quad \forall i,j \tag{10}$$

$$\sum_{b=1}^{B}\sum_{q=1}^{Q}u_{i,j,c,b,q}=\sum_{m=1}^{M}x_{i,j,m}l_{m,c},\quad \forall i,j,c,e=1 \tag{11}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{Q_i}\sum_{q=1}^{Q}u_{i,j,c,b,q}sz_i\le v,\quad \forall c,b \tag{12}$$

$$u_{i,j,c,b,q}u_{i',j',c,b,q}dc_{c,j}=u_{i,j,c,b,q}u_{i',j',c,b,q}dc_{i',j'},\quad \forall i,j,i',j',c,b,q \tag{13}$$

$$dc_{i,j}=\sum_{m=1}^{M}l_{m,c}x_{i,j,m}c,\quad \forall i,j \tag{14}$$

$$t_{i,j}\ge u_{i,j,c,b,q+1}u_{i',j',c,b,q}(t_{i',j'}+d_{dc_{i,j},dc_{i',j'}}),\quad \forall i,j,i',j',c,b,q \tag{15}$$

$$t_{i,j}\ge u_{i,j,c,b,q}d_{dc_{i,j},c},\quad \forall i,j,c,b \tag{16}$$

$$\sum_{t=1}^{T}z_{c,b,t}t\ge u_{i,j,c,b,q}f_{i,j},\quad \forall i,j,c,b,q \tag{17}$$

$$s_{i,j+1}\ge \max\{f_{i,j},e_{i,j}(\sum_{c=1}^{C}\sum_{b=1}^{B}\sum_{q=1}^{Q}u_{i,j,c,b,q}\times\sum_{t=1}^{T}z_{c,b,t}t+t_{i,j})\},\quad \forall i,j \tag{18}$$

$$\sum_{t=1}^{T}z_{c,b+1,t}t\ge\sum_{t=1}^{T}z_{c,b,t}t+t_{i,j}+u_{i,j,c,b,q}d_{dc_{i,j},c},\quad \forall i,j,c,b,q \tag{19}$$

各约束表示的意思分别为:

约束式 (2) 和式 (3) 表示每道工序仅能被分派至一台机器且该机器可以加工该工序;

约束式 (4) 给出了工序实际加工时间的定义;

约束式 (5) 表示每道工序都需要加工且只能加工一次;

约束式 (6) 和式 (7) 分别给出了工序开始时间和完工时间的定义;

约束式 (8) 表明一台机器同一时间仅能加工一道工序;

约束式 (9) 表示每个批次均需被运输且只能被运输一次;

约束式 (10) 表明每道工序完工后仅能被分派至一个小车上的一个批次;

约束式 (11) 表示每道工序完工后只能被所在单元所属的小车运输;

约束式 (12) 说明同一批次内工件的体积总和不得超过小车容量;

约束式 (13) 表示仅有目的单元相同的工件可以被同时卸载在同一单元 (即在同一批次的同一顺序被运输);

约束式 (14) 给出了一次运输的目的单元的定义;

约束式 (15) 表示任一工件在小车完成其所属批次之前所有的工件的运输, 且到达其目的单元之后才能被卸载 (批次中的首个工件除外);

约束式 (16) 表明任一批次的首个工件只有在小车到达其目的单元后才能被卸载;

约束式 (17) 说明同一批次中所有工件均完成前一道工序的加工后才能开始运输;

约束式 (18) 表明任一工序只有在其前一道工序完成加工, 并被运输至对应单元 (如有跨单元转移发生) 后才能开始加工;

约束式 (19) 表示小车运输完一个批次中所有工件并返回起始单元后才能开始下一批次的运输.

2 基于离散蜂群与决策块结构的超启发式算法

由于本文提出的问题较为复杂、规模往往较大, 所需算法应当兼具优化性能与计算效率, 因此本文采用超启发式算法进行求解. 采用 "动态决策块策略", 即在算法的迭代中动态生成决策块并为每一个决策块选择合适的启发式规则. 同时, 考虑到 ABC 算法在搜索范围的扩张和解的利用之间具有更好的平衡能力, 因此设计了一种基于离散蜂群与决策块结构的超启发式算法 (DABC-based hyper-heuristic approach with decision block, DHD), 解决考虑运输能力限制的跨单元调度问题.

2.1 候选规则集

本文将食物源的编码结构设计成三段, 即工件段、机器段和运输段, 分别对应了工序分派、工序排序以及运输问题.

1) 工序分派规则

Earliest finish time (EFT): 选择加工该工序后将具有最早完成时间的机器.

First available (FA): 选择最先可用的机器.

Shortest processing time (SPT): 选择具有最短加工时间的机器.

Most available (MA): 选择缓冲区内待加工工件个数最少的机器.

2) 工件排序规则

Critical ratio (CR): 选择具有最小关键度的工件.

Shortest remaining processing time (SRPT): 选择具有最短剩余加工时间的工件.

First in first out (FIFO): 选择最早释放到机器的工件.

Shortest processing time (SPT): 选择具有最短加工时间的工件.

Earliest due date (EDD): 选择具有最早交货期的工件.

Weighted shortest processing time (WSPT): 选择具有最小加权加工时间的工件.

Weighted earliest due date (WEDD): 选择具有最小加权交货期的工件.

Minimum slack (MS): 选择具有最小松弛时间的工件.

3) 运输调度决策

Earliest due date (EDD): 小车按工件交货期升序排序后组批运输.

Shortest remaining processing time (SRPT): 小车按工件剩余加工时间升序排序后组批运输.

Shortest processing time × Total operation time (SPT × TOT): 小车按工序的加工时间与该工件剩余加工时间之积升序排序后组批运输.

Operation time + Transfer time (TransOperAndTrans): 小车按工序的加工时间与该工件转移到目的单元时间之和升序排序后组批运输.

Release time + Transfer time (TransOpersAndFIFO): 小车按工序到达缓冲区的时间与该工件转移到目的单元时间之和升序排序后组批运输.

2.2 动态决策块策略

"决策块" 是指在超启发式算法中, 一个规则所决定的决策范围大小. 在传统的超启发式方法中普遍采用的 "静态决策块策略" 无法在考虑问题本身以及计算效率的前提下动态对决策对象范围进行划分. 因此, 本文提出了 "动态决策块策略", 即通过超启发式算法的优化, 找到合适的决策块划分方法, 并采用超启发式算法选择合适的启发式规则应用到相应的决策块.

动态决策块的基本思想是: 在算法开始时, 以问题规模的大小为依据初始化每个决策块的大小, 并为每个决策块随机选择启发式规则. 此后, 采用基于 DABC 的超启发式算法对决策块的划分以及每个决策块上启发式规则的选择, 同时进行优化和迭代.

动态决策块策略的步骤如下:

步骤 1. 运行次数 $run = 0$;

步骤 2. 如果 run 超过实验最大运行次数, 转步骤 8;

步骤 3. 初始化工件、机器和小车的决策块编码;

步骤 4. 为工件、机器和小车的每个决策块随机选择启发式规则, 生成工件、机器和小车的规则编码;

步骤 5. 合并决策块编码与规则编码, 生成决策与规则相对应的最终编码;

步骤 6. 执行基于 DABC 蜂群的超启发式算法, 对初始化决策块及规则进行迭代优化;

步骤 7. $run = run + 1$, 转步骤 2;

步骤 8. 输出平均最小化目标函数值, 算法结束.

以机器的排序决策为例, 举例说明决策块编码与规则编码的生成方式, 如图 1 所示: 第 1 步表示一共有 4 个决策块, 每个决策块内决策的数量分别是 3, 2, 1, 3; 第 2 步为针对 4 个决策块分别搜索到的 4 个启发式规则; 第 3 步是用于机器调度的启发式规则序列.

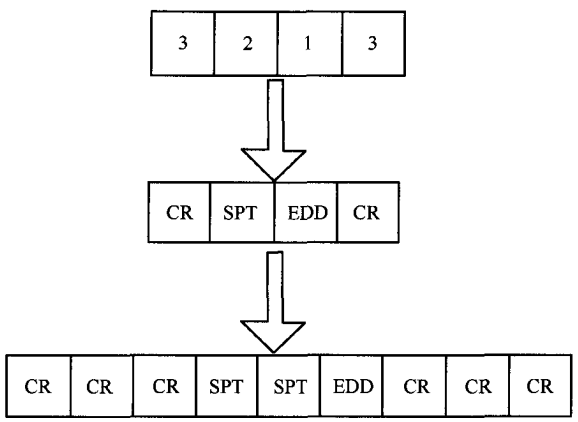

图 1　编码示意图

Fig. 1　Encoding scheme

2.3 解码方案

在本文所提出的考虑运输能力受限的跨单元调度问题中, 所述问题可分解为工序分派、工序排序以及运输调度三个子问题. 因此, 本文针对以上三个子问题分别进行解码.

2.3.1 工件分派

在工件完成一道工序的加工后, 首先对可选的加工机器进行扫描, 判断是否可在本单元完成加工. 如果可以则将零件加入该机器的缓冲区, 如果工件需要跨单元完成, 则采用以下策略选择机器.

算法 1. 工件分派启发式算法

步骤 1. 在对某道工序进行机器分派的决策时, 从编码中选取该决策对应的启发式规则;

步骤 2. 通过启发式规则计算该工序在候选机器上加工的适应度函数值;

步骤 3. 通过计算, 工序选择适应度最高的机器;

步骤 4. 将工序加入单元的等待运输的缓冲区中, 算法结束.

2.3.2 工件排序

当机器空闲时, 如果机器当前缓冲区内存在等待加工的工件, 则采用工件排序启发式算法选择一个工件进行加工.

为了便于描述排序算法, 定义可调度工件集.

定义 1. 把已分配到机器 m 的工件集定义为可调度工件集 SES_m.

令 $SequencingRulem$ 表示机器 m 的排序规则, e 表示待处理的可调度工件. 工件排序的启发式算法描述如下:

算法 2. 工件排序启发式算法

步骤 1. 更新 SES_m;

步骤 2. 对于所有属于 SES_m 的工件 e, 根据 $SequencingRulem$ 选择最佳的工件 e^*;

步骤 3. 机器 m 调度工件 e^*;

步骤 4. $SES_m = SES_m - \{e^*\}$, 并从机器 m 的缓冲区删除 e^*, 算法结束.

2.3.3 运输调度

当小车完成当前批次的运输并返回所属单元后, 如果单元内存在已经加工完成并等待跨单元运输的工件, 小车在不超过其容量的前提下, 采用以下方法对工件进行组批与路径决策.

算法 3. 运输启发式算法

步骤 1. 当单元 c 的小车空闲时, 从运输编码中选取该决策对应的启发式规则;

步骤 2. 遍历单元的缓冲区, 如果存在两个或两个以上工件具有相同的目的单元, 转步骤 3, 否则转步骤 4;

步骤 3. 通过启发式规则计算具有相同目的单元的工件的总适应度函数值;

步骤 4. 通过启发式规则计算工件的适应度函数值;

步骤 5. 比较适应度函数值, 根据数值大小对运输顺序排序;

步骤 6. 小车按照排序结果将工件依次运输至各个目的单元;

步骤 7. 清空缓冲区, 算法结束.

2.3.4 食物源解码

由于 DHD 方法搜索的是启发式规则而不是调度解, 因此需要设计一个离散事件模拟器 (Discrete event simulator, DES) 解码食物源中的启发式规则, 然后利用这些启发式规则构建调度解, 从而获得目标函数值. 解码算法描述如下:

步骤 1. 参数初始化, 置 DES 时钟 $t = 0$;

步骤 2. 对于一条给定的食物源的不同分段, 分别把分派规则分配至工件集 (JobSet), 排序规则分配至机器集 (MachineSet), 运输调度规则分配至单元集合 (CellSet);

步骤 3. 如果所有工件都已完工, 转步骤 11;

步骤 4. 对于每一个可分配工件 j, 如果存在单元间的柔性路径, 根据工件分派启发式算法调度一个工件, 否则, 说明其下一道工序的加工机器是确定的, 直接指派到该机器的缓冲队列上;

步骤 5. 如果单元 c 的小车是空闲可用的且有要运输的工件, 则转步骤 6, 否则, 转步骤 7;

步骤 6. 根据运输启发式算法对单元 c 选择出要运输的工件集合的运输路线, 并更新相关工件下一道工序的相关信息;

步骤 7. 如果机器 m 变空闲, 则转步骤 8, 否则, 转步骤 10;

步骤 8. 根据工序排序启发式算法调度一个工件;

步骤 9. 记录该工件调度工序的开工时间、完工时间和对应的加工机器;

步骤 10. $t = t + 1$, 转步骤 3;

步骤 11. 利用步骤 9 的记录信息, 计算相应的目标函数值, 算法结束.

2.4 DHD 算法步骤

在 DHD 算法中, 采用 DABC 算法对工序分派编码、工序排序编码以及运输编码进行整体的更新与优化.

DHD 算法整体流程如下:

步骤 1. 运行次数 $run = 0$;

步骤 2. 如果 run 超过实验最大运行次数, 转步骤 12;

步骤 3. 随机初始化种群;

步骤 4. 进化代数 $cycle = 0$;

步骤 5. 如果 $cycle$ 超过最大迭代次数, 转步骤 11;

步骤 6. 雇佣蜂阶段;

步骤 7. 观察蜂阶段;

步骤 8. 更新最好的解;

步骤 9. 侦查蜂阶段;

步骤 10. $cycle = cycle + 1$, 转步骤 5;

步骤 11. $run = run + 1$; 转步骤 2;

步骤 12. 输出平均最小化目标函数值, 算法结束.

2.4.1 雇佣蜂阶段和观察蜂阶段

在 DHD 方法的雇佣蜂与观察蜂阶段中, 雇佣蜂、观察蜂的数量相同. 首先, 雇佣蜂采取邻域搜索策略对其食物源进行探索; 当雇佣蜂阶段完成后, 每个观察蜂采取锦标赛的策略选择一个食物源, 并采取与雇佣蜂相同的邻域搜索方法对该食物源进行探索.

雇佣蜂阶段的步骤如下:

步骤 1. 种群中的每个雇佣蜂到与之对应的食

物源进行一次步骤 2～5;

步骤 2. 分别针对工件分派、工序排序和运输调度的决策块划分编码进行邻域搜索, 与之对应的启发式规则不变, 并计算目标函数值 $F1$;

步骤 3. 分别针对工序分派、工件排序和运输调度, 决策块编码所对应的启发式规则编码进行变异, 即邻域搜索, 与之对应的决策块大小编码不变, 并计算目标函数值 $F2$;

步骤 4. 将邻域搜索的结果 $F1$, $F2$ 分别与该食物源对比, 进行贪心策略的选择, 替换差的解;

步骤 5. 如果解的性能提升, 则将该个体的尝试次数的变量 ($trail$) 清零, 否则 $trail = trail + 1$.

上述过程中邻域搜索的步骤如下:

步骤 1. 随机选择两个索引位置;

步骤 2. 交换这两个索引位置的数值.

举例如图 2 所示.

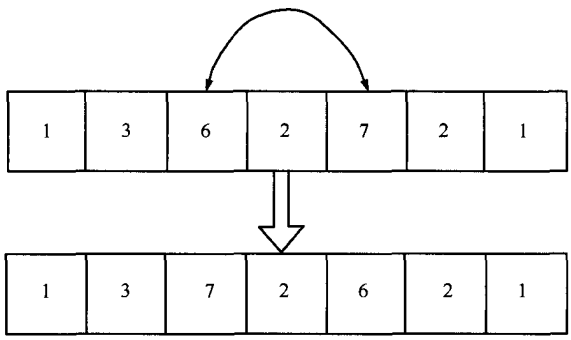

图 2　邻域搜索图例 1

Fig. 2　Example of one-point neighborhood search

2.4.2　改进的侦查蜂策略

在侦察蜂阶段, 如果一个食物源在 r 次连续的迭代中都没有提高 ($trail > r$), 那么它将被抛弃. 在传统 ABC 算法中, 原食物源被抛弃后将产生一个新的食物源, 由于随机产生的食物源更可能产生性能更差的解, 因此这一过程往往降低了搜索的效率.

为了使本文提出的 DHD 算法以较快的速度到达更有效的搜索空间, 本文采用新的方案, 如下:

当食物源在 r 次迭代后性能没有改善时, 为了借鉴该食物源的演化结果, 对其现有的搜索空间进行 T 次全新的邻域变换策略, 并将 T 次变换中所获得的最好的解作为新的食物源. 具体步骤如下.

步骤 1. 邻域变换次数 $t = 0$;

步骤 2. 将即将被抛弃的食物源设置为当前最好的食物源 $SourceBest$;

步骤 3. 如果 t 超过最大邻域变换次数 T, 转步骤 7;

步骤 4. 对 $SourceBest$ 进行邻域搜索;

步骤 5. 如果邻域变换后的解优于 $SourceBest$, 则替换 $SourceBest$;

步骤 6. $t = t + 1$, 转步骤 3;

步骤 7. 采用 $SourceBest$ 作为新的食物源, 算法结束.

为了避免邻域搜索算法陷入局部最优, 新的邻域搜索算法在编码中至少选择 3 个索引位, 并将 3 个索引位与其右侧相邻位进行交换. 举例如图 3 所示.

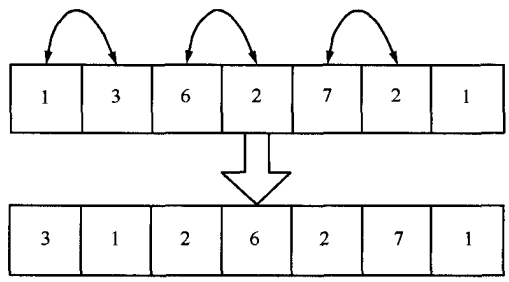

图 3　邻域搜索图例 2

Fig. 3　Example of three-point neighborhood search

3　实验与分析

3.1　实验设计

本文所针对的考虑运输能力限制的跨单元调度问题, 目前没有合适的 Benchmark 可以借鉴, 因此本文设计了多组测试问题 (Test problem) 来验证算法的有效性. 实验使用 Java 语言实现, 运行在 Core i5-2600 2.40 GHz, 4 GB 内存的 PC 机上.

根据不同的机器与工件的规模, 设计了 17 个不同规模的测试用例, 每个规模下随机产生 10 个算例 (Instance). 产生算例的参数设置如表 1 所示. 每个测试问题按照表 1 所示参数随机产生 10 个不同的算例. 每个算例进行 5 次独立的仿真实验. 10 个算例的目标函数值的均值作为此测试问题的性能指标. 工件的交货期按式 (20) 计算.

$$d_i = r_i + dl \sum_{k=1}^{K} p_{ik} \tag{20}$$

其中, r_i 是工件 i 的到达时间, dl 是交货期因子, 表示交货期的紧张程度, 默认取值为 2; $\sum_{k=1}^{K} p_{ik}$ 表示工件 i 的工序 k 在可选机器上的最小加工时间总和. 不同规模的测试问题包含的工件数从 5 到 400 不等, 工件的工序数从 5 到 19 不等, 机器数从 6 到 100 不等, 单元数从 3 到 15 不等. 每个测试问题采用 $jn_1mn_2cn_3$ 的记法, 例如 $j3m6c3$ 表示该测试问题中包含 3 个单元, 共 6 台机器, 且有 3 个工件需要进行加工.

3.2　参数分析

算法参数的设置对于 DHD 算法的性能具有很大影响, 因此需要对不同的参数组合进行性能评价,

以确定最优算法参数组合. 本文考虑的 DHD 参数共有 4 个, 分别是种群大小、尝试次数限制、侦查蜂迭代次数与进化代数. 对这 4 个参数进行全析因设计 (Full factor design) 实验, 每个参数作为一个因子 (Factor), 每个因子的水平 (Treatment) 如表 2 所示.

在置信度水平 95 % 下, 采用方差分析法 (Analysis of variance, ANOVA) 分析共 108 个不同参数组合的 DHD 的性能表现, 从而确定最优算法参数组合. 对每个 DHD 分别在 10 个测试问题下各运行 5 次.

在以最小化 TWT 为优化目标时, ANOVA 对全析因实验的分析结果如表 3 所示.

从表 3 可以看出, 单因子 PS 和 $cycle$ 是显著的 (P 值 $\leq$ 0.05). 通过比较 F 值大小, 可以识别出对 DHD 影响最大的因子或者交互因子. 每个单因子主效应如图 4 所示, 随着 PS、$cycle$ 的增大, DHD 获得更优的解性能, 而 DHD 在 $PS = 40$ 时获得最优的解性能. 此外, 由于从表 3 可知因子组合 $lim \times PS$, $lim \times T$, $PS \times T$ 间的交互作用存在显著性, 因此还需要分析因子间交互作用对 DHD 解性能的影响. 二阶因子交互作用如图 5 所示. 根据表 3 的 F 值可知, 因子组合 $PS \times T$ ($F = 8.86$) 对解性能的影响最大, 以 $PS \times T$ 为例分析参数的取值. 因为 PS ($F = 259.92$) 较 T ($F = 1.17$) 对解性能的影响更大, 因此先确定 PS 的值. 从图 5 可知 $PS = 40$ 时具有最优解性能, 而 $T = 10$ 时具有最优解性能. 这与图 4 的结果是一致的. 同理, 对于 $lim \times PS$, $lim \times T$ 可知, $lim = 10$, $PS = 40$ 时, DHD 取得最优解性能. 综合以上分析, 在最小化 TWT 目标下, 本文将 DHD 的参数取值定为: 种群大小 $PS = 40$, 尝试次数限制 $lim = 10$, 侦查蜂迭代次数 $T = 10$, 进化代数 $cycle = 150$.

表 1 算例产生的参数表

Table 1 Parameters of test problems

算例产生参数		取值分布
每个单元内机器数		~U [2, 6]
小车容量		~U [2, 10]
工件到达时间 r_i		~U [0, 50]
工件权重 w_i		~U (0, 1]
单元间转移时间 $Trans_{ij}$		~U [6, 50]
工序加工时间	准备时间 st_{ikm}	~U [5, 10]
	处理时间 p_{ikm}	~U [1, 30]

表 2 DHD 参数

Table 2 Parameters in the DHD approach

因子	种群大小 PS	尝试次数限制 lim	侦查蜂迭代次数 T	进化代数 $cycle$
水平	5, 15, 30, 40	5, 10, 20	1, 10, 20	30, 60, 150

表 3 最小化 TWT 目标下的 ANOVA 表

Table 3 Analysis of variance with respect to minimizing TWT

Source	Df	$SeqSS$	$AdjSS$	$AdjMS$	F	P
lim	2	949	949	474	0.05	0.947
PS	3	6 849 492	6 849 492	2 283 164	259.92	0.000
T	2	29 974	29 974	14 987	1.17	0.189
$cycle$	2	6 036 210	6 036 210	3 018 105	343.58	0.000
$lim \times PS$	6	172 330	172 330	28 722	3.27	0.007
$lim \times T$	4	120 283	120 283	30 071	3.42	0.013
$lim \times cycle$	4	14 840	14 840	3 710	0.42	0.792
$PS \times T$	6	467 139	467 139	77 856	8.86	0.000
$PS \times cycle$	6	22 400	22 400	3 733	0.43	0.860
$T \times cycle$	4	11 156	11 156	2 789	0.32	0.865
误差	68	597 325	597 325	8 784		
合计	107	14 322 098				

$S = 93.7240$ $R - Sq = 95.83\%$ $R - Sq$ (调整) $= 93.44\%$

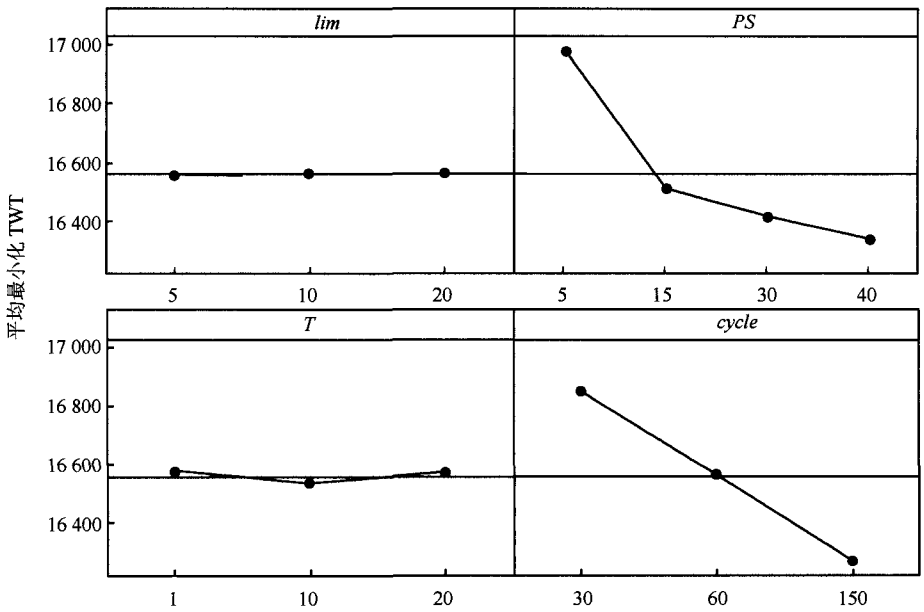

图 4　最小化 TWT 目标下的主效应图

Fig. 4　Effects of single factors with respect to minimizing TWT

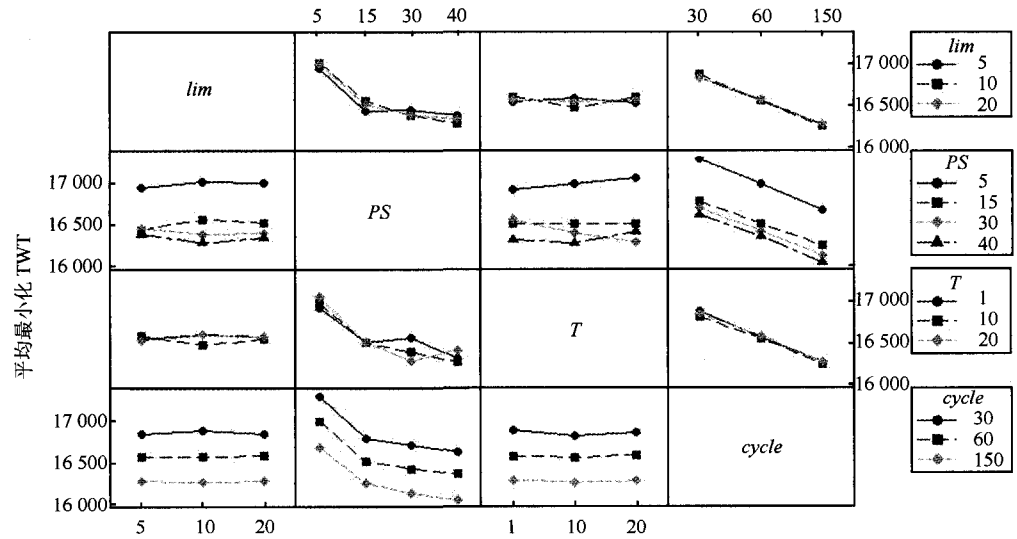

图 5　最小化 TWT 目标下的因子交互作用图

Fig. 5　Effects of two-way interactions among factors with respect to minimizing TWT

3.3　性能对比实验

确定 DHD 算法参数后, 需要对算法的有效性及其效率进行验证.

实验将从以下 5 个方面验证算法的有效性: 1) 侦察蜂阶段策略的改进前后对比; 2) 决策块的不同划分策略的性能分析; 3) DHD 与传统算法性能对比; 4) 运输决策对跨单元问题的影响分析; 5) 与 CPLEX 对比.

3.3.1　侦察蜂阶段的改进前后对比

在传统 ABC 中, 如果某一食物源在一定迭代次数内没有提高, 侦查蜂将在预定的搜索范围内随机地产生一个新的个体, 这一过程往往降低了搜索的效率, 因为随机产生的个体更可能产生性能更差的解.

而本文在原来个体基础上进行 T 次连续的全新策略的邻域变换, 将 T 次迭代变换中最好的解作为侦查蜂采纳的个体.

本文在进行对比实验时, 将采用传统 ABC 框架解决本文的问题记为 ABC-hyper heuristic (ABC-HH). 本实验按照式 (1) 计算目标函数值 *score*, 以

此作为评价算法性能的指标并按照式 (21) 计算 Gap 值, 记为 Gap_{ABC-HH}. 其中 $score_{ABC-HH}$ 和 $score_{DHD}$ 分别代表以传统 ABC 方法为框架的超启发式算法与本文方法的目标函数值.

$$Gap_{ABC-HH} = \frac{score_{ABC-HH} - score_{DHD}}{score_{DHD}} \quad (21)$$

实验结果如表 4 所示, 在最小化 TWT 目标下, DHD 算法较传统的 ABC 算法平均提升了 3.02%, 表明 DHD 算法比传统的 ABC 算法在侦察蜂阶段跳出局部最优解的能力上有所提高, 从而更加利于全局解的性能的提高.

3.3.2 决策块的不同划分策略的性能分析

本节将两种决策块的划分策略与 DHD 算法中决策块的动态划分方法进行对比.

1) 决策块的范围是调度中的一次决策

这种编码形式是指将在调度中的每一次决策看成一个决策块, 所以编码时每一个决策块的大小都是 1. 将这种方法记为 DHD-ONE.

本实验按照式 (1) 计算目标函数值, 以此作为评价算法性能的指标并按照式 (22) 计算 Gap 值, 记为 $Gap_{DHD-ONE}$. 其中 $score_{DHD-ONE}$ 和 $score_{DHD}$ 分别代表决策块大小为 1 的方法与本文方法的目标函数值.

表 4 最小化 TWT 目标下 DHD 与 ABC-HH 性能比较

Table 4 Comparison between DHD and ABC-HH with respect to minimizing TWT

测试问题	DHD	ABC-HH	Gap_{ABC-HH} (%)
$j5m6c3$	76.2	76.2	0.00
$j15m8c3$	1 153.2	1 183.0	2.58
$j20m11c3$	1 884.5	1 941.7	3.04
$j40m13c5$	6 682.5	6 873.2	2.85
$j50m15c5$	9 539.3	10 322.5	8.21
$j60m16c5$	19 130.6	19 254.2	0.65
$j70m20c7$	22 287.5	22 988.0	3.14
$j80m21c7$	30 588.5	31 875.3	4.21
$j90m21c7$	30 170.3	32 249.0	6.89
$j100m25c9$	35 194.1	36 474.6	3.64
$j120m30c9$	47 657.3	48 359.5	1.47
$j140m35c11$	64 258.2	66 002.0	2.71
$j160m40c11$	76 205.4	79 311.2	4.08
$j180m45c13$	101 389.7	103 847.2	2.42
$j200m50c15$	149 606.2	153 469.8	2.58
$j300m75c15$	271 212.9	275 602.1	1.62
$j400m100c15$	388 897.7	393 579.5	1.20
	73 878.5	75 494.7	3.02

表 5 最小化 TWT 目标下 DHD 与静态决策块策略的性能比较

Table 5 Comparison between DHD and the static decision block strategy with respect to minimizing TWT

测试问题	DHD 动态决策块	DHD-ONE	DHD-All	$Gap_{DHD-ONE}$ (%)	$Gap_{DHD-ALL}$ (%)
$j5m6c3$	76.2	76.2	98.3	0.00	29.00
$j15m8c3$	1 153.2	1 247.7	1 261.3	8.19	9.37
$j20m11c3$	1 884.5	2 183.2	2 180.7	15.85	15.72
$j40m13c5$	6 682.5	7 539.3	6 450.3	12.82	−3.47
$j50m15c5$	9 539.3	11 309.5	9 793.5	18.56	2.66
$j60m16c5$	19 130.6	20 803.4	18 493.0	8.74	−3.33
$j70m20c7$	22 287.5	23 570.0	23 008.0	5.75	3.23
$j80m21c7$	30 588.5	33 786.9	31 141.4	10.46	1.81
$j90m21c7$	30 170.3	33 252.0	32 083.7	10.21	6.34
$j100m25c9$	35 194.1	37 582.3	35 830.4	6.79	1.81
$j120m30c9$	47 657.3	49 477.8	46 283.1	3.82	−2.88
$j140m35c11$	64 258.2	69 030.3	65 517.9	7.43	1.96
$j160m40c11$	76 205.4	82 605.7	83 119.6	8.40	9.07
$j180m45c13$	101 389.7	105 510.9	108 491.9	4.06	7.00
$j200m50c15$	149 606.2	157 207.2	157 844.8	5.08	5.51
$j300m75c15$	271 212.9	284 942.7	292 460.9	5.06	7.83
$j400m100c15$	388 897.7	416 964.5	420 024.4	7.22	8.00
平均值	73 878.5	78 652.3	78 475.5	8.14	5.86

$$Gap_{DHD-ONE}=\frac{score_{DHD-ONE}-score_{DHD}}{score_{DHD}} \tag{22}$$

实验结果如表 5 所示, 在最小化 TWT 目标下, DHD 算法与决策块的大小为 1 的方法相比, 平均性能提升了 8.14 %. 这组实验表明, 当决策块大小为 1 时, 食物源的编码长度大大增加, 虽然搜索空间变大, 但其在相同条件下的计算能力下降, 所以综合求解能力与 DHD 相比较差.

2) 决策块的范围为一个实体 (工件、机器或小车) 在调度过程中的所有决策

本文在进行对比实验时, 将决策块的范围为一个实体的方法记为 DHD-ALL. 本实验按照式 (1) 计算目标函数值, 以此作为评价算法性能的指标并按照式 (23) 计算 Gap 值, 记为 $Gap_{DHD-ALL}$. 其中 $score_{DHD-ALL}$ 和 $score_{DHD}$ 分别代表以实体 (工件、机器或小车) 为单位划分决策块的方法与本文方法的目标函数值.

$$Gap_{DHD-ALL}=\frac{score_{DHD-ALL}-score_{DHD}}{score_{DHD}} \tag{23}$$

实验结果如表 5 所示, 在最小化 TWT 目标下, DHD 动态决策块与以实体 (工件、机器或小车) 为单位划分决策块的方法相比, 平均性能提升了 5.86 %. 两种决策块编码形式上的区别主要在于: DHD 决策块的编码是根据问题规模的大小动态产生的, 并采用基于蜂群的超启发式算法进行迭代优化. 而 DHD-ALL 在划分决策块时, 决策块划分仅仅依赖于实体 (工件、机器或小车) 的数量.

通过本小节的实验分析可以看出, 本文的动态决策块策略方式相比于传统决策块具有一定的优势.

3.3.3 DHD 与传统算法性能对比

1) 与组合规则比较

在所有的组合规则中, 本文共考虑了 160 种组合规则, 在最小化 TWT 的目标下对每种组合规则进行仿真实验. 通过分析实验结果, 选取平均性能最优的 3 种组合规则 (如表 6 所示).

表 6　比对实验的组合规则

Table 6　Combinatorial heuristic rules used

排序规则	指派规则	运输调度规则	记作
WSPT	SPT	EDD	WSSE
EDD	SPT	SPT × TOT	ESST
CR	SPT	SPT × TOT	CSST

本实验按照式 (1) 计算目标函数值, 以此作为评价算法性能的指标并按照式 (24) 计算 Gap 值, 记为 Gap_{rules}. 其中 $score_{rules}$ 和 $score_{DHD}$ 分别代表组合规则方法与本文方法的目标函数值.

$$Gap_{rules}=\frac{score_{rules}-score_{DHD}}{score_{DHD}} \tag{24}$$

实验结果如表 7 所示, 在最小化 TWT 目标下, DHD 较组合规则方法平均性能提升了 42.41 %. 在实际生产中, 企业的数据规模非常庞大, 启发式规则因具有简单高效和灵活性而成为首选的调度方法. 然而制造系统是复杂多变的, 没有任何一个规则能在所有环境和评价标准下性能都优于其他的规则, 而本文采用决策块策略的超启发式算法, 通过算法的更新与迭代, 合理地确定了决策块的划分结果, 同时通过算法更新为每个决策块选择合适的规则, 最终使得问题的每个决策可以获得较为合适的规则.

表 7　最小化 TWT 目标下 DHD 与组合规则的性能比较

Table 7　Comparison between DHD and combinatorial heuristic rules with respect to minimizing TWT

测试问题	DHD	WSSE	ESST	CSST	Gap_{rules} (%)
$j5m6c3$	76.2	147.8	121.8	179.3	96.37
$j15m8c3$	1 153.2	1 563.9	2 549.6	2 912.2	103.08
$j20m11c3$	1 884.5	3 084.1	3 592.7	3 249.6	75.58
$j40m13c5$	6 682.5	7 666.9	9 694.1	9 911.7	36.04
$j50m15c5$	9 539.3	12 803.9	16 904.6	17 589.7	65.27
$j60m16c5$	19 130.6	29 794.6	39 715.6	40 884.4	92.35
$j70m20c7$	22 287.5	29 684.7	31 764.1	31 252.6	38.64
$j80m21c7$	30 588.5	35 792.7	40 314.9	40 875.6	27.48
$j90m21c7$	30 170.3	35 792.7	40 314.9	40 875.6	29.25
$j100m25c9$	35 194.1	40 059.3	41 419.4	43 653.5	18.52
$j120m30c9$	47 657.3	55 955.0	61 301.6	63 237.1	26.24
$j140m35c11$	64 258.2	73 085.7	85 476.3	80 387.5	23.95
$j160m40c11$	76 205.4	88 001.2	91 564.7	95 807.5	20.45
$j180m45c13$	101 389.7	116 889.5	117 305.5	121 363.6	16.90
$j200m50c15$	149 606.2	162 424.7	168 704.6	177 348.7	13.29
$j300m75c15$	271 212.9	292 138.6	323 395.9	332 274.8	16.49
$j400m100c15$	388 897.7	419 015.8	494 468.9	498 830.4	21.05
平均值	73 878.5	82 582.4	92 271.1	94 154.9	42.41

2) 与 HSGA[17] 方法比较

HSGA 算法为基于遗传算法 (Genetic algorithm, GA) 设计的超启发式算法, 用于求解混合流水车间调度问题. 它在基因编码上采用的是以工件与机器为单位来划分决策块, 本文在与 HSGA 算法进行对比时, 将 HSGA 算法应用于本文所提出的问题模型并进行实验.

本实验按照式 (1) 计算目标函数值 $score$, 以此作为评价算法性能的指标并按照式 (25) 计算 Gap 值, 记为 Gap_{HSGA}. 其中 $score_{HSGA}$ 和 $score_{DHD}$

分别代表 HSGA 方法与本文方法的目标函数值.

$$Gap_{HSGA} = \frac{score_{HSGA} - score_{DHD}}{score_{DHD}} \quad (25)$$

实验结果如表 8 所示, 在最小化 TWT 目标下, DHD 较 HSGA 方法平均性能提升了 41.95%. 并且, 随着问题规模的增大, DHD 相比于 HSGA 产生的解越好. 以上实验结果表明 DHD 方法相比于传统算法具有优越性, 并且尤其适合解决大规模的跨单元调度问题.

表 8 最小化 TWT 目标下 DHD 与 HSGA 方法的性能比较

Table 8 Comparison between DHD and HSGA with respect to minimizing TWT

测试问题	DHD	HSGA	Gap_{HSGA} (%)
$j5m6c3$	76.2	68.1	−30.72
$j15m8c3$	1 153.2	1 385.9	9.88
$j20m11c3$	1 884.5	2 670.0	22.44
$j40m13c5$	6 682.5	7 123.4	10.44
$j50m15c5$	9 539.3	11 829.4	20.79
$j60m16c5$	19 130.6	18 796.0	1.64
$j70m20c7$	22 287.5	30 195.9	31.24
$j80m21c7$	30 588.5	43 642.2	40.14
$j90m21c7$	30 170.3	45 677.1	42.37
$j100m25c9$	35 194.1	56 710.0	58.27
$j120m30c9$	47 657.3	72 170.0	55.93
$j140m35c11$	64 258.2	108 451.8	65.53
$j160m40c11$	76 205.4	141 372.6	70.08
$j180m45c13$	101 389.7	179 509.2	65.46
$j200m50c15$	149 606.2	272 697.9	72.76
$j300m75c15$	271 212.9	526 274.6	79.95
$j400m100c15$	388 897.7	827 241.3	96.95
平均值	73 878.5	137 989.1	41.95

3.3.4 运输决策对跨单元问题的影响分析

跨单元调度问题中生产和运输相互影响和制约, 因此运输决策 (即如何进行组批和路由) 是生产和运输之间协同的关键. 为了验证运输决策对跨单元问题的影响, 本文假设另外一种运输策略. 即假设小车在每次运输工件时, 只能将目的单元相同的工件进行组批, 运输至目的单元并返回本单元. 该策略简记为 DHD-single destination (DHD-SD). 本节将本文所提出的运输策略与该策略进行对比. 两种策略在对比过程中采用相同的候选规则.

本实验按照式 (1) 计算目标函数值 Gap_{DHD-SD}, 以此作为评价算法性能的指标并按照式 (26) 计算 Gap 值, 记为 $score_{DHD-SD}$. 其中 $score_{DHD-SD}$ 和 $score_{DHD}$ 分别代表 DHD-SD 方法与本文方法的目标函数值.

$$Gap_{DHD-SD} = \frac{score_{DHD-SD} - score_{DHD}}{score_{DHD}} \quad (26)$$

实验结果如表 9 所示, 在最小化 TWT 目标下, DHD 较目的单元唯一的方法平均性能提升了 21.17%. 本小节实验表明, 考虑运输的路径规划策略与目的单元唯一的策略相比性能较高. 原因在于, 在考虑运输能力受限的跨单元调度问题中, 当小车空闲并等待运输下一个批次时, 单元内等待被运输的工件所去往的目的单元往往不同. 针对这样的现状, 如果小车每次只将目的单元相同的工件进行组批并运输, 会导致大量的工件被滞留, 且在运输过程中耗费了大量时间. 因此, 合理地对需要跨单元的工件进行组批与路径决策, 可有效地减少由于运输能力受限所产生的瓶颈.

表 9 最小化 TWT 目标下 DHD 与目的单元唯一方法的性能比较

Table 9 Comparison between DHD and DHD-SD with respect to minimizing TWT

测试问题	DHD	DHD-SD	Gap_{DHD-SD} (%)
$j5m6c3$	76.2	51.2	−32.81
$j15m8c3$	1 153.2	1 243.8	7.85
$j20m11c3$	1 884.5	2 167.2	15.00
$j40m13c5$	6 682.5	7 306.2	9.33
$j50m15c5$	9 539.3	11 272.2	18.17
$j60m16c5$	19 130.6	20 288.9	6.05
$j70m20c7$	22 287.5	26 271.1	17.87
$j80m21c7$	30 588.5	36 952.7	20.81
$j90n21c7$	30 170.3	37 417.3	24.02
$j100m25c9$	35 194.1	45 670.4	29.77
$j120m30c9$	47 657.3	60 059.9	26.02
$j140m35c11$	64 258.2	85 495.0	33.05
$j160m40c11$	76 205.4	103 914.3	36.36
$j180m45c13$	101 389.7	137 971.4	36.08
$j200m50c15$	149 606.2	205 895.9	37.63
$j300m75c15$	271 212.9	370 240.8	36.51
$j400m100c15$	388 897.7	537 410.1	38.19
平均值	73 878.5	99 389.9	21.17

3.3.5 与 CPLEX 的比较

为了进一步评估 DHD 算法, 验证算法的性能, 采用 CPLEX12.4 对本文研究的问题建立约束规划模型, 计算不同问题规模下的最优调度结果, DHD 算法与 CPLEX 进行对比分析, 对比结果见表 10. 按照式 (27) 计算 Gap 值, 记为 Gap_{CPLEX}.

$$Gap_{CPLEX} = \frac{score_{CPLEX} - score_{DHD}}{score_{DHD}} \quad (27)$$

由于问题模型复杂, 本实验设置可以接受的运

表 10　最小化 TWT 目标下 DHD 与 CPLEX 的性能比较

Table 10　Comparison between DHD and CPLEX with respect to minimizing TWT

问题模型	DHD		CPLEX		TWT Gap (%)	Time Gap (%)
	TWT	Time (s)	TWT	Time (s)		
$j6m6c2$	250.3	0.156	191.1	443.3	−23.65	284 066.67
$j7m6c2$	265.5	0.177	352.6	10 800.0	32.8	6 101 594.92
$j7m8c3$	254.6	0.274	—	—	—	—

行时间上限为 6 小时. 如表 10 所示, 在小规模问题 $j7m6c2$ 下, DHD 算法的性能比 CPLEX 算法优秀 32.80 %. 而且算法运行所需的 CPU time 只需要 0.177 s.

由此可见, DHD 算法能够快速有效地得到近优解. 在 $j7m8c3$ 以上的规模问题下, CPLEX 在可接受的时间范围之内, 无法得到最优解, 而 DHD 算法以较短的运行时间获得了可行解. 实验结果表明, DHD 在保证计算效率的同时, 具备较好的寻优能力, 是一种高效的算法.

4　结论

本文提出一种 DHD 方法解决运输能力受限的跨单元调度问题, 通过采用动态决策块策略与改进的侦察蜂策略使得超启发算法的寻优能力得到了提升. 实验结果表明, 本文所设计的动态决策块策略对问题中的决策进行了有效的划分, 与静态决策块策略相比具有一定优势. 其次, 本文针对传统的 ABC 算法进行改进, 并将其作为超启发式方法中的高层算法, 相比于传统算法 (如遗传算法) 具有显著优势. 此外, 本文所提出的运输调度策略能有效地解决生产调度中跨单元运输所造成的瓶颈. 与 CPLEX 对比实验结果显示, DHD 方法能够快速地得到近优解, 兼备了算法的寻优能力与计算效率.

为简化问题, 本文在决策时, 假设小车一旦空闲便对缓冲区内所有等待跨单元的工件进行组批, 并没有对缓冲区的工件进行组批范围的决策, 在未来的工作中, 可以尝试先用启发式规则确定工件是否加入组批, 然后再将此批工件进行路径规划并送至目的单元.

References

1 Li D N, Wang Y. Production scheduling in intercell cooperative production mode. In: Proceedings of the 24th Chinese Control and Decision Conference (CCDC). Taiyuan: IEEE, 2012. 504−506

2 Garza O, Smunt T L. Countering the negative impact of intercell flow in cellular manufacturing. *Journal of Operations Management*, 1991, **10**(1): 92−118

3 Johnson D J, Wemmerlov U. Why does cell implementation stop? factors influencing cell penetration in manufacturing plants. *Production and Operations Management*, 2004, **13**(3): 272−289

4 Li D N, Meng X W, Li M, Tian Y N. An ACO-based intercell scheduling approach for job shop cells with multiple single processing machines and one batch processing machine. *Journal of Intelligent Manufacturing*, DOI: 10.1007/s10845-013-0859-2

5 Mosbah A B, Dao T M. Optimimization of group scheduling using simulation with the meta-heuristic extended great deluge (EGD) approach. In: Proceedings of the 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). Macao: IEEE, 2010. 275−280

6 Yousef G K, Reza T M, Amir K. Solving a multi-criteria group scheduling problem for a cellular manufacturing system by scatter search. *Journal of the Chinese Institute of Industrial Engineers*, 2011, **28**(3): 192−205

7 Solimanpur M, Elmi A. A tabu search approach for group scheduling in buffer-constrained flow shop cells. *International Journal of Computer Integrated Manufacturing*, 2011, **24**(3): 257−268

8 Tang J F, Wang X Q, Kaku I, Yung K L. Optimization of parts scheduling in multiple cells considering intercell move using scatter search approach. *Journal of Intelligent Manufacturing*, 2009, **21**(4): 525−537

9 Elmi A, Solimanpur M, Topaloglu S, Elmi A. A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. *Computers & Industrial Engineering*. 2011, **61**(1): 171−178

10 Li D N, Wang Y, Xiao G X, Tang J F. Dynamic parts scheduling in multiple job shop cells considering intercell moves and flexible routes. *Computers & Operations Research*, 2013, **40**(5): 1207−1223

11 Burke E K, Hyde M, Kendall G, Ochoa G, Özcan E, Woodward J R. A classification of hyper-heuristic approaches. *Handbook of Metaheuristics*. Berlin: Springer, 2010. 449−468

12 Burke E K, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E, Qu R. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 2013, **64**(12): 1695−1724

13 Yang T, Kuo Y, Cho C. A genetic algorithms simulation approach for the multi-attribute combinatorial dispatching decision problem. *European Journal of Operational Research*, 2007, **176**(3): 1859−1873

14 Vázquez-Rodríguez J A, Petrovic S. A new dispatching rule based genetic algorithm for the multi-objective job shop problem. *Journal of Heuristics*, 2010, **16**(6): 771−793

15 Fayad C, Petrovic S. A fuzzy genetic algorithm for real-world job shop scheduling. In: Proceedings of the 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Bari, Italy: IEA, 2005. 524−533

16 Li D N, Meng X W, Liang Q Q, Zhao J Q. A heuristic-search genetic algorithm for multi-stage hybrid flow shop scheduling with single processing machines and batch processing machines. *Journal of Intelligent Manufacturing*, DOI: 10.1007/s10845-014-0874-y

17 Karaboga D. An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Computer Engineering Department, Engineering Faculty, Erciyes University, Turkey, 2005.

18 Pan Q K, Tasgetiren M F, Suganthan P N, Chua T J. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, 2011, **181**(12): 2455−2468

19 Tasgetiren M F, Pan Q K, Suganthan P N, Chen A H L. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Information Sciences*, 2011, **181**(16): 3459−3475

刘兆赫 北京理工大学计算机学院硕士研究生. 主要研究方向为演化计算和生产调度. E-mail: cyuyan888@163.com
(**LIU Zhao-He** Master student at the School of Computer Science, Beijing Institute of Technology. His research interest covers evolutionary computation and production scheduling.)

李冬妮 北京理工大学计算机学院副教授. 主要研究方向为智能优化, 企业计算, 物流管理. 本文通信作者.
E-mail: ldn@bit.edu.cn
(**LI Dong-Ni** Associate professor at the School of Computer Science, Beijing Institute of Technology. Her research interest covers intelligent optimization, enterprise computation, and logistics management. Corresponding author of this paper.)

王乐衡 北京理工大学计算机学院硕士研究生. 主要研究方向为演化计算和生产调度. E-mail: wanglh1991@163.com
(**WANG Le-Heng** Master student at the School of Computer Science, Beijing Institute of Technology. Her research interest covers evolutionary computation and production scheduling.)

田云娜 北京理工大学计算机学院博士研究生. 主要研究方向为演化计算与智能优化方法.
E-mail: ydtianyunna@163.com
(**TIAN Yun-Na** Ph. D. candidate at the School of Computer Science, Beijing Institute of Technology. Her research interest covers evolutionary computation and intelligent optimization approaches.)