一种基于动态决策块的超启发式跨单元调度方法

田云娜[1,2] 李冬妮[1] 刘兆赫[1] 郑 丹[1]

摘 要 对运输能力受限条件下的跨单元调度问题进行分析, 提出一种基于动态决策块和蚁群优化 (Ant colony optimization, ACO) 的超启发式方法, 同时解决跨单元生产调度和运输调度问题. 在传统超启发式方法的基础上, 采用动态决策块策略, 通过蚁群算法合理划分决策块, 并为决策块选择合适的规则. 实验表明, 采用动态决策块策略的超启发式方法比传统的超启发式方法具有更好的性能, 本文所提的方法在最小化加权延迟总和目标方面有较好的优化能力并且具有较高的计算效率.

关键词 动态决策块, 超启发式, 蚁群算法, 跨单元调度

引用格式 田云娜, 李冬妮, 刘兆赫, 郑丹. 一种基于动态决策块的超启发式跨单元调度方法. 自动化学报, 2016, **42**(4): 524−534

DOI 10.16383/j.aas.2016.c150402

A Hyper-heuristic Approach with Dynamic Decision Blocks for Inter-cell Scheduling

TIAN Yun-Na[1,2] LI Dong-Ni[1] LIU Zhao-He[1] ZHENG Dan[1]

Abstract In this paper, the inter-cell scheduling problem with a transportation capacity constraint is analyzed. An ant colony optimization (ACO)-based hyper-heuristic with dynamic decision blocks is proposed, which selects appropriate heuristic rules for production and transportation simultaneously. On the basis of traditional hyper-heuristics, a dynamic decision block strategy is proposed, in which several entities are grouped into a decision block under the guidance of pheromones, and appropriate heuristic rules are selected for each decision block. Comparisons between the proposed method and other hyper-heuristics with different decision block strategies are conducted. Computational results show a satisfying performance of the proposed method in minimizing total weighted tardiness with less computational costs.

Key words Dynamic decision block, hyper-heuristic, ant colony optimization (ACO), inter-cell scheduling

Citation Tian Yun-Na, Li Dong-Ni, Liu Zhao-He, Zheng Dan. A hyper-heuristic approach with dynamic decision blocks for inter-cell scheduling. *Acta Automatica Sinica*, 2016, **42**(4): 524−534

单元制造系统 (Cellular manufacturing system, CMS) 的思想是将成组技术应用在生产制造业领域中, 通过构建功能相对独立的生产单元, 将具有相同性质的一批工件集中在单元内加工, 这种生产方式有效地降低了运输时间、响应时间等生产成本[1]. 然而在实际生产中, 受到市场需求多样化和生产经济成本方面的约束, 具有复杂工艺的工件需要通过多个单元协作加工, 形成了跨单元协作的生产模式, 由此产生了跨单元调度问题[2].

在现有的跨单元问题研究中, 由于元启发式算法具备较强的优化能力, 大部分学者采用这种方法解决跨单元调度问题. 其中 Solimanpur 等[3] 提出一种嵌套的禁忌搜索算法解决跨单元工件调度问题. Tang 等[4] 和 Tavakkoli-Moghaddam 等[5] 采用分散搜索算法解决工件在多个单元间的跨单元调度问题. Zeng 等[6] 采用遗传算法和局部搜索结合的方法解决单元内工件排序问题, 以最小化最大完工时间为启发式信息, 采用轮盘赌方法解决单元间工件分派问题. Elmi 等[7] 采用模拟退火方法解决跨单元调度问题. Li 等[8] 采用蚁群优化算方法解决具有柔性路径的跨单元调度问题. 在当前被验证过的最大问题规模下 (60～80 个工件/25～40 台机器/6～8 个单元), 元启发式算法一般需要耗费数百秒 (371 s[8]、840 s[5]) 才能得到调度解. 然而, 在装备制造业复杂产品的 CMS 中, 通常有十余个单元、数百个工件、数百台机器、数千道工序, 元启发式算法将无法承受如此大规模问题的计算压力, 因此从计算效率的角度考虑, 不能直接使用元启发式算法.

收稿日期 2015-06-25 录用日期 2015-12-28
Manuscript received June 25, 2015; accepted December 28, 2015
国家自然科学基金 (71401014) 资助
Supported by National Natural Science Foundation of China (71401014)
本文责任编委 赵千川
Recommended by Associate Editor ZHAO Qian-Chuan
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

在上述跨单元问题的研究中, 文献 [3, 5] 忽略了跨单元转移时间, 文献 [4, 6−8] 考虑了跨单元转移时间. 但是这些研究都隐含一个假设, 即单元间运输能力充足, 工件在单元间运输无需等待. 而在装备制造业的实际生产中, 由于各个单元分布在不同位置, 工件具有一定的重量和体积, 因此需要由运输工具完成跨单元转移, 但运输工具的数量和容量却有限. 因此如何高效地利用有限的运输工具成为实际生产中的重要问题. 本文考虑运输能力受限的跨单元调度问题, 同时处理工序分派、工序排序和小车运输三个子问题, 问题在复杂度和规模两方面都有所增加, 调度算法需要同时满足优化性能和计算效率两方面的需求.

在实际生产中, 启发式规则在计算效率方面表现出良好的性能, 因其简单、快捷和易实施的特点而被广泛应用[9], 特别适用于复杂、具有动态特性的制造环境[10]. 但启发式规则的缺陷在于它对调度环境和调度目标的依赖性较强, 没有哪个规则能在所有情况下都表现出良好的性能[11], 而且选用规则都是根据人为经验事先指定的, 优化能力较差, 因此无法满足复杂调度问题的优化性能需求.

超启发式算法 (Hyper-heuristic) 作为一种 "搜索启发式方法" (Heuristics to search heuristics)[12], 相当于将启发式规则和其他搜索方法或学习机制的优势相结合, 用优化能力较强的算法搜索或产生高效的规则, 再用得到的规则求解. 在已有的超启发式算法研究中, 遗传算法被广泛应用于选择启发式规则[13−16]. 另外, Li 等[17] 采用基于蚁群优化算法的超启发式方法解决混合流水车间调度问题. 贾凌云等[18] 提出一种基于混合蛙跳算法的超启发式方法, 并通过遗传规划产生的规则扩充超启发式算法的规则集. 刘兆赫等[19] 提出一种基于离散蜂群算法的超启发式方法. 由于超启发式算法的搜索对象是启发式规则而不是问题的解, 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, 同时也缩小了搜索空间, 提高了计算效率.

对于生产调度问题而言, 超启发式算法通常是为生产环境中的 "实体" (比如工件) 搜索底层的启发式规则, 然后对这些实体应用启发式规则进行调度. 本文将超启发式算法中的决策单位称为 "决策块", 决策块的大小决定了为多大范围的实体选择同一个启发式规则. 按照决策块大小为分类标准, 可以将超启发式算法的研究分为两类. 一类是决策块大小均为 1 的情况, 即为每个实体 (机器/工件) 选择一个启发式规则[13−14, 17−18]. 另一类是决策块大小不全为 1 的情况, 即为多个实体选择同一个启发式规则, Yang 等[20] 为每个阶段的所有机器选择一个启发式规则. Vázquez-Rodríguez 等[15−16]、刘兆赫等[19] 将每次决策看作一个决策点, 按时间顺序将所有决策点划分为多个决策块, 为每个决策块选择一个启发式规则.

在上述两类方法中, 决策块大小会直接影响算法的计算效率和优化性能. 当决策块较小时, 能够考虑到不同实体的状态特征, 更有可能为每个实体选到合适的规则, 使得算法的优化性能有保证, 但也会因此付出较大的计算开销, 反之亦然. 因此, 在解决大规模复杂问题时, 决策块的大小成为影响算法计算效率和优化性能的关键因素.

在现有的超启发式算法研究中, 决策块大小一般都是事先指定的[13−15, 17−18], 本文称之为静态决策块策略. 在这类方法中, 决策块大小在算法的运行过程中是确定不变的, 这使算法的适应性受到限制. 动态决策块策略则是通过迭代不断优化决策块的大小, 动态寻找合理的决策块划分方案. 在目前的研究中, 采用动态决策块策略的文献还很少, Vázquez-Rodríguez 等[16] 采用遗传算法动态优化决策块大小和启发式规则序列, 解决多目标作业车间调度问题.

基于以上分析, 本文设计了一种基于动态决策块和蚁群优化 (Ant colony optimization, ACO) 的超启发式算法 (Dynamic decision block and ACO-based hyper-heuristic, DABH). 在传统超启发式的框架上加入动态决策块策略, 通过 ACO 动态构建决策块并搜索启发式规则. 本文的主要贡献在于以下两点: 1) 动态决策块策略可以合理地缩小搜索空间, 提高算法的计算效率; 2) ACO 算法作为一种构造型优化方法, 通过信息素的引导, 动态生成大小合适的决策块, 从而达到优化性能和计算效率之间的平衡.

1　问题模型

本文根据作业车间跨单元调度问题抽象出问题模型, 以最小化加权延迟总和 (Total weighted tardiness, TWT), 为优化调度目标. 本文问题模型同时考虑跨单元生产和跨单元运输的协同.

1.1　问题假设

本文研究运输能力受限的跨单元调度问题假设描述如下:

1) 所有工件零时刻到达;

2) 每个工件的交货日期已知;

3) 工序在特定机器上的加工时间固定且已知, 准备时间独立于工序次序;

4) 每台机器在同一时刻只能加工一道工序;

5) 工序一旦开始, 便不允许中断和抢占;

6) 单元间存在加工能力重叠的机器, 即工件跨单元加工路径具有柔性, 对于任意一道工序, 单元内最多只有一台可加工的机器;

7) 考虑工件的跨单元转移时间, 忽略单元内转移时间;

8) 小车负责运输由多个工件组成的一个批次, 一个批次内的工件可以有不同的目的单元;

9) 小车仅装载本单元内工件, 从本单元出发后不在沿途各单元装载工件;

10) 小车按照一定顺序访问目的单元并卸载相应工件, 一旦所有工件卸载完毕, 小车立即返回所属单元;

11) 忽略小车装载和卸载工件的时间.

1.2 符号列表

与本文问题相关的符号变量定义如下所示.

索引

i: 工件索引 $(i=1,\cdots,N)$

j: 工件 i 的工序索引 $(j=1,\cdots,J^{(i)})$

m: 机器索引 $(m=1,\cdots,M)$

c: 单元及所属小车索引 $(c=1,\cdots,C)$

b: 小车 c 上批次索引 $(b=1,\cdots,B^{(c)})$

q: 第 b 个批次内工件的运输顺序索引 $(q=1,\cdots,Q^{(b)})$

系统变量

O_{ij}: 工件 i 的第 j 道工序

$J^{(i)}$: 工件 i 的工序总数

p_{ijm}: 工序 O_{ij} 在机器 m 上的加工时间

s_i: 工件 i 的体积

d_i: 工件 i 的交货期

w_i: 工件 i 的权重

v: 小车的容量

$T_{cc'}$: 从单元 c 到单元 c' 所需要的转移时间

t_{ij}: 工序 o_{ij} 完工后所需要的转移时间

D_{ij}: 工序 o_{ij} 完工后转移的目的单元

s_{ij}: 工序 o_{ij} 的开始时间

f_{ij}: 工序 o_{ij} 的完工时间

$$\alpha_{ijm}=\begin{cases}1, & \text{工序 } o_{ij} \text{ 可以被机器 } m \text{ 加工}\\ 0, & \text{其他}\end{cases}$$

$$\beta_{mc}=\begin{cases}1, & \text{机器 } m \text{ 分布在单元 } c \text{ 内}\\ 0, & \text{其他}\end{cases}$$

$$\gamma_{ij}=\begin{cases}1, & \text{工序 } o_{ij} \text{ 需要到另一个单元加工}\\ 0, & \text{其他}\end{cases}$$

决策变量

$$x_{ijm}=\begin{cases}1, & \text{工序 } o_{ij} \text{ 被分派至机器 } m \text{ 上}\\ 0, & \text{其他}\end{cases}$$

$$Y_{ijt}=\begin{cases}1, & \text{工序 } o_{i,j} \text{ 在 } t \text{ 时刻开始加工}\\ 0, & \text{其他}\end{cases}$$

$$Z_{ijcbq}=\begin{cases}1, & \text{工序 } o_{i,j} \text{ 完工后, 工件 } i \text{ 被分派至小车 } c \text{ 的第 } b \text{ 个批次, 且运输顺序为第 } q \text{ 个}\\ 0, & \text{其他}\end{cases}$$

$$\omega_{cbt}=\begin{cases}1, & \text{小车 } c \text{ 的 } b \text{ 批次在 } t \text{ 时刻开始运输}\\ 0, & \text{其他}\end{cases}$$

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. 目标函数的数学表达式如式 (1) 所示.

$$\min\sum_{i=1}^{N}w_i\max\{(f_{iJ^{(i)}}-d_i),0\} \tag{1}$$

根据实际生产中的问题特性和约束, 本文的约束条件描述如下.

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

$$\sum_{t=1}^{T}Y_{ijt}=1,\quad \forall i,j \tag{3}$$

$$s_{ij}=\sum_{t=1}^{T}Y_{ijt}t,\quad \forall i,j \tag{4}$$

$$f_{i,j}=s_{ij}+\sum_{m=1}^{M}p_{ijm}X_{ijm},\quad \forall i,j \tag{5}$$

$$(1-\alpha_{ijm})X_{ijm}=0,\quad \forall i,j,m \tag{6}$$

$$(X_{ijm}s_{ij}\geq X_{i'j'm}s_{i'j'}+X_{i'j'm}p_{i'j'm})\vee (X_{i'j'm}s_{i'j'}\geq X_{ijm}s_{ij}+X_{ijm}p_{ijm}),\ \forall i,i',j,j',m,X_{ijm}X_{i'j'm}=1,i\neq i',j\neq j' \tag{7}$$

$$\sum_{m=1}^{M}X_{ijm}(s_{ij}+p_{ijm})\leq f_{iJ^{(i)}},\quad \forall i,j \tag{8}$$

$$\sum_{t=1}^{T}\omega_{c,b,t}=1,\quad \forall c,b \tag{9}$$

$$\sum_{c=1}^{C}\sum_{b=1}^{B^{(c)}}\sum_{q=1}^{Q^{(b)}}Z_{ijcbq}=\gamma_{ij},\quad \forall i,j \tag{10}$$

$$\sum_{b=1}^{B^{(c)}}\sum_{q=1}^{Q^{(b)}}Z_{ijcbq}=\sum_{m=1}^{M}X_{ijm}\beta_{mc},\ \forall i,j,c,\gamma_{ij}=1 \tag{11}$$

$$D_{ij}=\gamma_{ij}\sum_{m=1}^{M}\beta_{mc}X_{ijm}c,\quad \forall i,j \tag{12}$$

$$t_{ij}\geq Z_{ijcb(q+1)}Z_{i'j'cbq}(t_{i'j'}+T_{D_{i'j'},D_{ij}}),$$

$$\forall i,j,i',j',c,b,q \tag{13}$$

$$t_{ij} \geq Z_{ijcb1}T_{cD_{ij}}, \quad \forall i,j,c,b \tag{14}$$

$$\sum_{t=1}^{T}\omega_{cbt}t \geq Z_{ijcbq}f_{ij}, \quad \forall i,j,c,b,q \tag{15}$$

$$s_{i(j+1)} \geq \max\left\{f_{ij},\gamma_{ij}\left(\sum_{c=1}^{C}\sum_{b=1}^{B^{(c)}}\sum_{q=1}^{Q^{(b)}}Z_{ijcbq}\times \sum_{t=1}^{T}\omega_{cbt}t+t_{ij}\right)\right\}, \quad \forall i,j \tag{16}$$

$$\sum_{t=1}^{T}\omega_{c(b+1)t}t \geq \sum_{t=1}^{T}\omega_{cbt}t+t_{ij}+Z_{ijcbq}T_{D_{ij}c}, \quad \forall i,j,c,b,q \tag{17}$$

$$\sum_{i=1}^{N}\sum_{q=1}^{Q^{(b)}}Z_{ijcbq}S_i \leq v, \quad \forall c,b \tag{18}$$

其中, 式 (2) 表示一个工序只能在一个机器上进行加工; 式 (3) 表示一个工序在同一时刻只能被加工一次; 式 (4) 和式 (5) 分别表示工序的开始加工时间和完工时间; 式 (6) 表示工序只能被分派到具有相应加工能力的机器上; 式 (7) 保证被分派到机器上的工件的顺序性; 式 (8) 表示工件的完工时间大于或等于其任意一道工序的完工时间; 式 (9) 表示小车的一个批次只能被运输一次; 式 (10) 表示一个工件只能被分派到一个小车的一个批次; 式 (11) 表示需要跨单元的工件只能被分派到本单元的小车; 式 (12) 定义小车运输目标单元; 式 (13) 和式 (14) 表示任意批次的所有工件只有在小车到达其目的单元才能被卸载; 式 (15) 表示每个批次必须等到批次内所有工件的前一道工序完成才能开始运输; 式 (16) 表示任意工序只能在其前一道工序完成, 并被运输至目的单元才能开始加工; 式 (17) 表示每个小车运输完一个批次内所有工件并返回所在单元后才能开始下一批次运输; 式 (18) 表示所有被分派到同一小车的同一批次的工件体积总和小于或等于小车容量.

2　DABH 算法

在运输能力受限的跨单元调度问题中, 需要考虑多个子问题以及它们之间的协同. 在解决这一类大规模复杂调度问题时, 需要兼顾算法的优化性能和计算效率. DABH 在超启发式的框架上加入动态决策块策略, 通过合理的缩小搜索空间, 从而达到提高算法计算效率的目的. 同时, DABH 采用 ACO 作为超启发式算法的高层优化方法, 通过信息素的引导动态寻找合理的决策块划分方案, 并为不同的决策块选择合适的启发式规则, 通过得到的启发式规则进行调度产生完整解.

2.1　基于动态决策块的算法框架

本文问题模型包含三个子问题, 其中工序分派为每道工序选择一台加工机器; 工序排序确定机器缓冲队列中待加工工件的加工次序; 小车运输包含对工件的组批和路径决策, 组批确定小车每个批次运送哪些工件, 路径决策确定工件的运输次序. DABH 算法解决以上三个问题的流程描述如图 1.

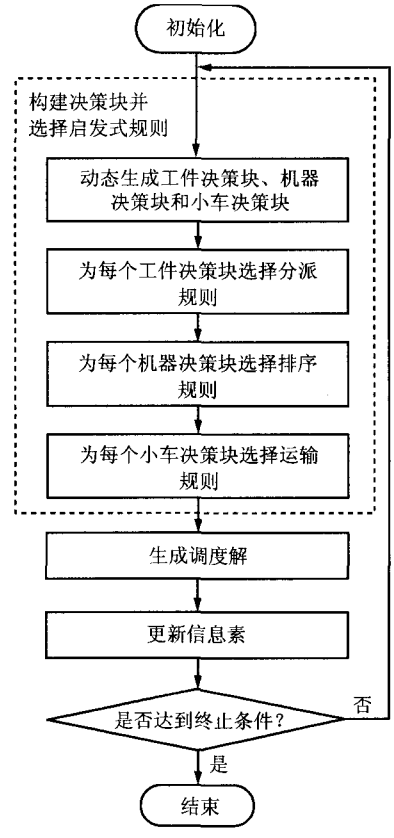

图 1　DABH 算法的整体流程图

Fig. 1　General algorithm of DABH

在图 1 中, 所有决策块大小在初始阶段随机生成, 在迭代过程中根据解的优化性能更新部分最优解的信息素, 然后根据信息素逐步优化决策块的大小, 使决策块趋向于更加合理的划分. 同时, 根据候选规则的信息素浓度为每个决策块选择规则. 在调度过程中, 每个决策块内的实体采用相同的规则, 生成最终调度解.

2.2　基于动态决策块的编码

在 DABH 算法中, 每个决策块的大小代表所包含实体的数量, 这里实体指工件、机器和小车, 决策块的大小决定一个启发式规则所作用的范围, 本节对决策块以及基于决策块的编码进行形式化的描述.

本文将实体集合记为 $E=\{e_1,\cdots,e_N\}$, 其

中 N 表示实体的数量. 将启发式规则集合记为 $H=\{H',H'',H'''\}$, 其中 $H'=\{h'_1,\cdots,h'_n\}$ 代表工序分派规则集合, $H''=\{h''_1,\cdots,h''_p\}$ 代表工序排序规则集合, $H'''=\{h'''_1,\cdots,h'''_q\}$ 代表小车运输规则集合. 在规则选取过程中, 分别从 H'、H''、H''' 中为工件决策块、机器决策块和小车决策块选取规则, 最终得到基于决策块的规则编码.

定义 1. 将所有实体划分成若干个决策块, 所有决策块构成的集合定义为 B, $B=\{b_1,\cdots,b_i,\cdots,b_L\}$, 集合中元素须同时满足两个条件:

1) $\bigcup_{i=1}^{L} b_i=E$;

2) $\bigcap_{i=1}^{L} b_i=\emptyset$.

在定义 1 中, 条件 1) 表示所有实体都被划分到各个决策块中, 条件 2) 表示一个实体不能重复出现在多个决策块中.

例 1. 假设在工件分派子问题中有 4 个工件, 这些工件被划分成两个决策块. 那么两个决策块的大小之和为 4, 并且决策块之间没有重复的工件出现.

定义 2. 对于决策块集合 $B=\{b_1,\cdots,b_i,\cdots,b_L\}$, 基于决策块的规则编码定义为 $A=\{a_1,\cdots,a_i,\cdots,a_L\}$.

例 2. 假设测试问题有 8 个实体, 包括 2 个工件, 4 台机器和 2 个单元, 即 $E=\{e_1,\cdots,e_8\}$, 启发式规则集合 $H=\{h'_1,h'_2,h''_1,h''_2,h'''_1,h'''_2\}$.

当决策块大小均为 1 时, 即为每个实体选择一个规则. 如图 2 所示, 虚框内为基于 8 个实体的规则编码 $A=\{a_1,\cdots,a_i,\cdots,a_8\}$, 在 H 中分派、排序和运输规则分别有 2 个, 那么对应编码的搜索空间大小为 $2^2\times2^4\times2^2$.

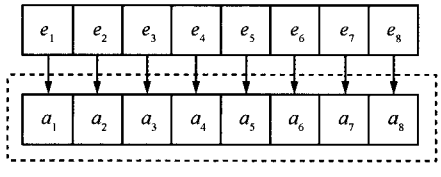

图 2 决策块大小均为 1 的编码

Fig. 2 Representation of decision blocks whose size is 1

当决策块大小不全为 1 时, 即存在为多个实体选择一个规则的情况. 如图 3 所示, 将 8 个实体划分为 4 个决策块, 即 $B=\{b_1,b_2,b_3,b_4\}$. 其中 $b_1=\{e_1,e_2\}$, $b_2=\{e_3\}$, $b_3=\{e_4,e_5,e_6\}$, $b_4=\{e_7,e_8\}$, 它们分别是 1 个工件决策块、2 个机器决策块和 1 个小车决策块.

图 3 中虚框内为基于 4 个决策块的规则编码 $A^*=\{a_1^*,a_2^*,a_3^*,a_4^*\}$, 即将 a_1^* 分派给 e_1 和 e_2, a_2^* 分派给 e_3, a_3^* 分派给 e_4、e_5 和 e_6, a_4^* 分派给 e_7 和 e_8. 类似地, 编码 A^* 的搜索空间大小为 $2^1\times2^2\times2^1$.

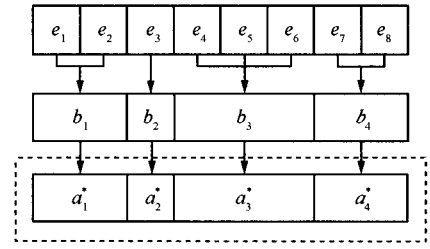

图 3 决策块大小不全为 1 的编码

Fig. 3 Representation of decision blocks with different sizes

由此可见, 当决策块大小不全为 1 时, 决策块策略能够有效缩小搜索空间, 在计算效率方面会有所提高. 但是当决策块大小过大时, 问题求解空间将过于单一, 有可能遗漏掉优质解, 导致算法的优化性能受到限制. 因此确定合理的决策块大小, 将有利于在提高计算效率的同时保证算法的优化性能.

2.3 基于动态决策块的规则选取

在 DABH 算法中, 通过信息素引导动态生成决策块, 以决策块为单位选择启发式规则, 对决策块内的所有实体应用规则得到调度解. 算法根据生成的调度解的优劣更新信息素, 从而达到寻优的目标.

2.3.1 信息素结构

在构造信息素结构时, DABH 包含两类信息素矩阵, 即决策块划分矩阵和启发式规则选择矩阵.

针对分派、排序、运输三个子问题, 决策块划分矩阵分别为工件、机器和小车三个决策块划分矩阵. 三个矩阵的大小分别为 $N\times D'$、$M\times D''$ 和 $C\times D'''$, 其中 N、M 和 C 分别表示工件、机器和小车的数量, D'、D'' 和 D''' 分别表示工件、机器和小车决策块大小的上限, 上限取值通过参数实验获得, 在第 3.2 节中有详细描述. 矩阵中元素 $\tau_{i,p}$ 表示第 i 个决策块的大小为 p 的信息素浓度.

同上分析, 启发式规则矩阵分别为工件、机器和小车三个分派规则矩阵. 矩阵大小为分别为 $N\times R'$、$M\times R''$ 和 $C\times R'''$, 其中 R'、R'' 和 R''' 分别表示分派规则、排序规则和运输规则的数量. 矩阵中元素 $\tau_{i,q}$ 表示第 i 个决策块选取第 q 个启发式规则的信息素浓度.

2.3.2 候选规则集

针对三个子问题, 候选规则分别对应分派、排序和运输三种规则. 另外, 由于工件交货期 (Due date) 直接影响 TWT, 因此在候选排序规则集中加入与交货期相关的规则, 其中包含公认效果显著的 Apparent tardiness cost、Cost over time、Minimum slack、Slack per remaining processing time 等规则[9].

1) 候选分派规则

First available (FA): 选择最先可用机器.

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

Least utilization (LU): 选择加工该工序后具有最低占用率的机器.

Most available (MA): 选择当前缓冲区内待加工工件最少的机器.

Shortest processing time (SPT): 选择加工时间最短的机器.

2) 候选排序规则

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

Smallest processing time ratio (SPTR): 选择具有最小加工时间比率的工件.

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

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

Time in shop (TIS): 选择在当前机器缓冲队列中等待时间最长的工件.

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

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.

Apparent tardiness cost (ATC) : 选择 ATC 值最大的工件, 具体计算见式 (19).

$$\frac{w_i}{p_{ijm}} \exp\left(-\frac{\max(d_i - p_{ijm} - t), 0}{K\overline{p}}\right) \tag{19}$$

Cost over time (COVERT): 选择 COVERT 值最大的工件, 具体计算见式 (20).

$$\frac{w_i}{p_{ijm}} \max(1 - \max(d_i - p_{ijm} - t), 0), 0) \tag{20}$$

Slack per remaining processing time (S/RPT): 选择 S/RPT 值最小的工件, 具体计算见式 (21).

$$\max\left(\frac{d_i - p_{ijm} - t}{p_{ijm}}, 0\right) \tag{21}$$

3) 候选运输规则

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

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

Smallest processing time ratio (SPTR): 选择具有最小加工时间比率的工件.

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

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

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

Time in shop (TIS): 选择当前机器缓冲队列中等待时间最长的工件.

2.3.3 确定决策块大小和规则

在 DABH 算法中, 采用 ACO 算法来确定决策块的大小, 并为每个决策块选取规则. 每只蚂蚁通过信息素浓度计算选择决策块大小和启发式规则的概率.

1) 当蚂蚁选择决策块大小时, 第 i 个决策块的大小为 p 的概率由式 (22) 计算可得.

$$Pr_{ip} = \frac{\tau_{ip}}{\sum_{k=1}^{D} \tau_{ik}} \tag{22}$$

其中, τ_{ip} 表示第 i 个决策块的大小为 i 的信息素浓度. D 表示决策块大小的上限, 当选择工件决策块大小时, D 为 D'; 当选择机器决策块大小时, D 为 D''; 当选择小车决策块大小时, D 为 D'''.

2) 当蚂蚁选择启发式规则时, 第 i 个决策块选取第 q 个启发式规则的概率由式 (23) 计算可得.

$$Pr_{iq} = \frac{\tau_{iq}}{\sum_{k=1}^{R} \tau_{ik}} \tag{23}$$

其中, τ_{iq} 表示第 i 个决策块的选取第 q 个启发式规则的信息素浓度. R 表示启发式规则的数量, 为工件决策块选择规则时, R 为 R'; 为机器决策块选择规则时, R 为 R''; 为小车决策块选择规则时, R 为 R'''.

2.3.4 信息素更新

本文采用最大最小蚂蚁系统 (Max-min ant system, MMAS) 的信息素更新方法, 更新信息素的值限定在区间 $[\tau_{\min}, \tau_{\max}]$ 内, $\tau_{\min}$ 取值为 0.1, $\tau_{\max}$ 取值为 5. 在每次循环中, 所有蚂蚁均完成调度解的构造后进行信息素更新, 参与更新的仅为本次循环中的 σ 个最优解. 这种方法既能使蚁群搜索的范围集中在较优解附近, 又不会因使用循环中的单个最优解或全局最优解更新信息素而使收敛速度过慢[21].

信息素更新规则如下:

1) 决策块大小信息素更新规则. 若第 i' 个决策块大小确定为 k, 则决策块划分矩阵中的元素 $\tau_{i'k}$ 根

据式 (24) 进行更新.

$$\tau_{i'k} = (1-\rho)\tau_{i'k} + \rho\Delta\tau \tag{24}$$

2) 启发式规则的信息素更新规则. 若第 i' 个决策块选择第 l 个分派规则, 则启发式规则选择矩阵中的元素 $\tau_{i'l}$ 根据式 (25) 进行更新.

$$\tau_{i'l} = (1-\rho)\tau_{i'l} + \rho\Delta\tau \tag{25}$$

在式 (24) 和 (25) 中, $\Delta\tau = Q/Score$, Q 为信息素更新量影响因子, $Score$ 为更新信息素的调度解对应的目标函数值.

2.4 基于动态决策块的解码

DABH 算法通过 ACO 算法为决策块选择启发式规则, 得到一组规则编码序列, 系统将其转换成具体的调度解, 即解码过程. 解码算法描述如下:

步骤 1. 参数初始化, 置离散事件模拟器 (Discrete event simulation, DES) 时钟 $t = 0$.

步骤 2. 分别把排序规则分配至各机器, 分派规则分配至各工件, 运输规则分配至各单元小车.

步骤 3. 如果所有工件都已完工, 转步骤 11.

步骤 4. 对于可分派工件, 若存在单元间柔性路径, 则根据分派规则选择加工机器, 更新被选机器的缓冲队列; 否则, 说明其下一道工序的加工机器是确定的, 直接分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. 根据运输规则计算单元 c 中待运输工件的优先级, 在不超过小车容量的情况下根据优先级进行组批并确定小车的运输路径.

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. 根据排序规则调度一个工件.

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. $t = t + 1$, 转到步骤 3.

步骤 11. 利用步骤 9 的记录信息, 根据式 (1) 计算相应的目标函数值 TWT. 算法结束.

3 实验与分析

为了验证 DABH 算法的优化性能和计算效率, 本文进行了多组对比实验, 仿真实验采用 JAVA 语言实现, 运行在 3.10 GHz Core i5-2400 CPU, 4 GB RAM 的 PC 机上.

3.1 实验设计

在运输能力受限的跨单元调度问题研究中, 目前尚没有相关的 Benchmark, 因此设计了不同规模下的多组测试用例. 测试用例的性能评价指标为目标函数值 TWT, 按照式 (1) 计算可得, 与 TWT 相关的工件交货期 d_i 按式 (26) 进行计算.

$$d_i = r_i + k\sum_{j=1}^{n} p_{ij} \tag{26}$$

其中 r_i 是工件 i 的到达时间, 由于本文问题假设所有工件零时刻到达, 因此 $r_i = 0$; $\sum_{j=1}^{n} p_{ij}$ 是工件 i 在可选机器上的平均加工时间 p_{ij} 的总和, k 为交货期因子, 表示交货期的紧急程度, 默认取值为 6.

实验包括 20 个不同大小的规模, 机器和工件相关属性的取值如表 1 所示. 每个规模下根据表 1 的参数随机产生 10 个不同测试用例, 对每个不同的测试用例进行 5 次独立的仿真实验. 每个测试问题采用 $pn_1mn_2cn_3$ 的记法, 表示该测试问题中包含 n_1 个工件、n_2 台机器和 n_3 个单元.

表 1 生成算例属性值

Table 1 Attributes for generating test problems

算例产生参数	取值范围
工件数	U[5, 450]
机器数	U[6, 120]
单元数 (小车数)	U(3, 15]
每个单元内机器数	U[2, 6]
小车容量	U[2, 10]
工件权重	U(0, 1]
工序加工时间	U[1, 30]
单元间转移时间	U[6, 50]

为了验证算法的有效性, 实验分别在不同问题规模下获取多个测试用例的目标函数 TWT 的平均值, 通过 DABH 方法与其他方法之间的 Gap 值进行对比分析.

3.2 参数分析

本文参数设计采用全因子设计方法[22], 使用 ANOVA (Analysis of variance) 方法从单因子效应和双因子交互作用两个方面进行分析, 观测多个因素对算法优化目标 TWT 的影响, 从而确定算法最优参数的设置.

由于 DABH 采用 ACO 算法搜索规则, 因此信息素挥发因子 ρ、信息素更新量影响因子 Q 和信息素浓度最大值 $\tau_{\max}$ 等参数会对算法的优化性能有所影响. 由于信息素更新量与 Q 直接相关, 在若干次迭代更新后优质解路径上的信息素浓度将达到 $\tau_{\max}$, 因此将 $Q/\tau_{\max}$ 作为实验参数, 其中 $\tau_{\max}$ 取值为 5, ρ 和 $Q/\tau_{\max}$ 的参数取值分 4 个级别, 如表 2 所示.

表 2　DABH 参数

Table 2　Parameters in DABH

参数	范围
ρ	(0.01, 0.05, 0.2, 0.8)
$Q/\tau_{\max}$	(0.01, 0.05, 0.2, 0.8)

本文以最小化 TWT 为优化目标值, 图 4 展示了单因子主效应. 由于影响算法性能包含两个因子, 因此还需要分析因子之间的交互作用, 图 5 展示了双因子的交互影响. 从图 4 和图 5 可看出, 当 $\rho = 0.01$, $Q/\tau_{\max} = 0.05$ 时, DABH 都具有最优解性能.

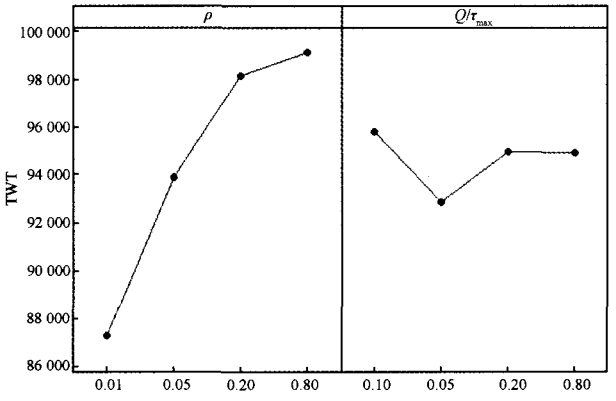

图 4　最小化 TWT 目标下的单因子主效应图

Fig. 4　Influence of each factor with respect to minimizing TWT

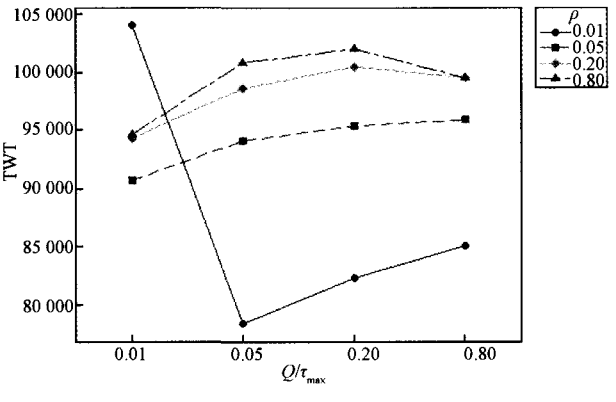

图 5　最小化 TWT 目标下的双因子交互作用图

Fig. 5　Influence of 2-factor interaction with respect to minimizing TWT

在决策块大小上限的参数实验中, 工件、机器和小车决策块数量的上限 D'、D''、D''' 分别根据三种实体数量的百分比 $\{10\%, 20\%, \cdots, 100\%\}$ 进行取值, 测试 10 种不同规格的参数设置, 最终效果最好的上限参数为 100%, 因此 D'、D''、D''' 分别被设为三种实体的数量.

3.3　与基于静态决策块的方法比较

为了检验 DABH 算法中动态决策块划分策略的有效性, 本节对比实验基于采用 DABH 的算法框架, 将动态决策块划分策略与三种不同的静态决策块划分策略进行比较. 为了保证实验的公平性, 三组对比实验采取相同的运行时间作为终止条件.

1) 每个决策块的范围是调度中的一个实体

在调度中每个决策块包含一个实体, 将这种方法记为 DABH-ONE.

实验按照式 (27) 计算 DABH 与 DABH-ONE 之间的 Gap 值, 记为 Gap_{ONE}. 由于计算方法类似, 在后续的对比实验中, DABH 与其它算法之间的 Gap 值计算方法均参考式 (27).

$$\text{Gap}_{\text{ONE}} = \frac{\text{score}_{\text{ONE}} - \text{score}_{\text{DABH}}}{\text{score}_{\text{DABH}}} \tag{27}$$

其中, $\text{score}_{\text{ONE}}$ 和 $\text{score}_{\text{DABH}}$ 分别代表决策块大小均为 1 的方法和 DABH 的目标函数值.

实验结果如表 3 所示, 在不同规模的测试问题下, DABH 均优于 DABH-ONE 的优化性能, Gap 平均值为 11.5%. 这组实验表明, 随着问题规模增大, 实体的数量增多, DABH-ONE 的解编码长度也增大, 因此搜索空间变大. 而 DABH 根据问题规模将所有实体划分为若干个决策块, 每个决策块最少包含一个实体, 决策块的数量小于或等于实体的数量, 相应的解编码长度减少, 从而适当地缩小了搜索空间. 由此可见, DABH-ONE 虽然能关注每个实体的状态, 但由于搜索空间过大, 使得在相同条件下前者的计算能力下降, 所以综合求解能力与 DABH 相比较差.

2) 每个决策块的范围是调度中的一类实体

在调度中每个决策块包含一类实体, 将这种方法记为 DABH-ALL. 本文的问题针对工件、机器和小车三类实体. DABH 与 DABH-ALL 之间的 Gap 值记为 Gap_{ALL}. 如表 3 所示, DABH 相比于 DABH-ALL 在平均性能上提升了 26.8%. 实验表明, DABH-ALL 过度缩小了搜索空间, 使解的多样性受到限制, 从而影响解的质量. 而 DABH 通过动态寻找合理的决策块划分方案, 既保留了解的多样性, 又使得计算时间处于合理的范围.

3) 每个决策块的范围是调度中的多个实体

为了避免决策块过大或过小, 本节针对 1) 和 2) 提出的决策块划分方案, 将决策块的大小设置为决策块候选取值的中间值, 记为 DABH-AVG. DABH 与 DABH-AVG 之间的 Gap 值记为 Gap_{AVG}. 如表 3 所示, DABH 相比于 DABH-AVG 在平均性能上提升了 14.2%. 为了直观地进行对比, 将 DABH 与三种策略对比的 Gap 值绘制成折线图, 如图 6 所示.

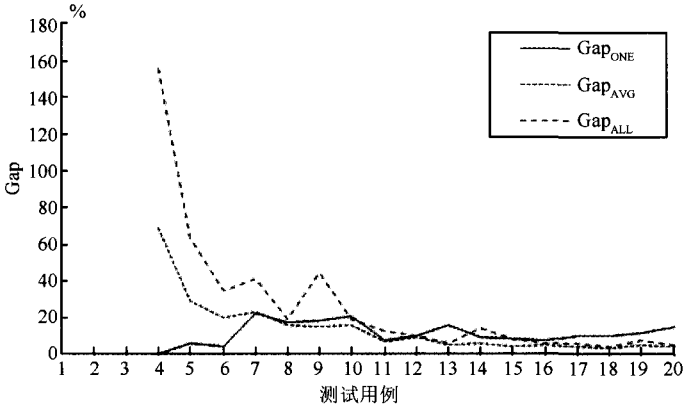

图 6 DABH 与不同决策块划分策略之间的 Gap 比较

Fig. 6 Gap values between DABH and different decision block strategies

结合表 3 与图 6 分析发现, 从平均性能来看, DABH-AVG 处于中间的位置, 这主要与 DABH-AVG 的决策块大小适中有关. 从图 6 可以看出, 当问题规模较小时, DABH-ALL 的相对性能最差, 当问题规模较大时, DABH-ONE 的性能最差, 而 DABH-AVG 的表现相对平稳.

综合实验 1) ~ 3) 进行分析, 决策块大小过小或过大都会影响算法的计算效率和优化性能, 即使 DABH-AVG 的决策块大小适中, 但由于决策块大小确定不变, 因此性能也会受到影响. 而 DABH 在每次迭代中根据信息素引导更新决策块大小, 然后为每个决策块选择合适有效的启发式规则. 这样不但在一定程度上减少了搜索空间, 而且寻优能力也得到了保障. 通过本节的实验可看出, 动态决策块策略相比与静态决策块策略体现出多方面的优势.

3.4 与基于动态决策块的方法比较

在现有的研究中, Vázquez-Rodríguez 等[16] 采用基于动态决策块和 GA 的超启发式方法 (Dynamic decision block GA-based hyper-heuristic), 以解决作业车间调度问题, 本文将其称作 DGBH. DGBH 针对调度中的决策, 将所有决策划分为若干个大小不等的决策块. 而 DABH 是将所有实体划分为多个决策块. 本节将进行 DABH 与 DGBH 的对比实验.

表 3 DABH 与静态决策块划分策略的性能比较

Table 3 Comparison between DABH and static decision block strategies

测试用例	TWT				运行时间	Gap_{ONE}	Gap_{ALL}	Gap_{AVG}
	DABH	DABH-ONE	DABH-ALL	DABH-AVG	(s)	(%)	(%)	(%)
$p5m6c3$	0.0	0.0	0.0	0.0	1.8	–	–	–
$p15m8c3$	0.0	0.0	10.5	0.0	4.5	–	–	–
$p20m11c3$	0.0	0.0	15.5	0.0	5.6	–	–	–
$p40m13c5$	1 317.5	1 320.3	3 363.7	2 224.9	28.2	0.2	155.3	68.9
$p50m15c5$	4 463.4	4 717.4	7 292.8	5 750.6	38.6	5.7	63.4	28.8
$p60m16c5$	7 853.8	8 183.2	10 594.3	9 416.9	48.9	4.2	34.9	19.9
$p70m20c7$	7 966.6	9 736.6	11 216.2	9 798.4	74.5	22.2	40.8	23.0
$p80m21c7$	16 504.9	19 385.3	19 653.9	19 137.8	84.1	17.5	19.1	16.0
$p90m21c7$	8 312.6	9 853.9	11 981.9	9 554.0	98.9	18.5	44.1	14.9
$p100m25c9$	21 774.8	26 339.4	25 924.7	25 197.7	93.3	21.0	19.1	15.7
$p120m30c9$	41 868.6	45 104.2	47 292.2	44 605.0	145.2	7.7	13.0	6.5
$p140m35c11$	43 255.4	47 646.4	47 523.0	47 284.2	199.0	10.2	9.9	9.3
$p160m40c11$	53 375.9	61 828.7	56 588.8	56 262.8	232.0	15.8	6.0	5.4
$p180m45c13$	78 761.0	86 352.7	89 622.8	83 336.1	299.5	9.6	13.8	5.8
$p200m50c15$	110 299.3	119 319.6	119 545.3	115 330.3	323.3	8.2	8.4	4.6
$p250m65c15$	155 422.9	167 271.9	164 193.1	164 003.1	447.5	7.6	5.6	5.5
$p300m75c15$	232 254.5	255 936.0	247 079.6	243 052.9	514.5	10.2	6.4	4.6
$p350m90c15$	303 144.1	334 527.3	316 700.5	314 547.5	692.0	10.4	4.5	3.8
$p400m100c15$	378 489.1	421 689.9	406 151.6	397 148.0	798.5	11.4	7.3	4.9
$p450m120c15$	466 779.7	535 080.7	489 769.7	486 647.2	1 187.9	14.6	4.9	4.3
平均值						11.5	26.8	14.2

为了保证实验的公平性, 对比实验采用相同的问题模型和候选规则集, 种群数目为 100, 终止条件为 DGBH 算法运行 200 代所需要的时间.

本实验 DABH 与 DGBH 之间 Gap 值记为 $\mathrm{Gap}_{\mathrm{DGBH}}$. 实验结果如表 4 所示, 在不同问题规模下, DABH 在 TWT 性能方面平均提升了 5.0%. 在收敛性方面, 图 7 展示了在问题规模 $p40m25$ 下两种方法运行 200 代的收敛情况.

表 4　DABH 与 DGBH 的性能比较

Table 4　Comparison between DABH and DGBH

测试用例	TWT		运行	$\mathrm{Gap}_{\mathrm{DGBH}}$
	DABH	DGBH	时间 (s)	(%)
$p10m10$	6 865.0	7 048.0	1.9	2.7
$p20m10$	26 176.0	26 686.8	4.0	2.0
$p20m15$	37 625.0	39 070.2	6.4	3.8
$p20m20$	23 791.8	25 214.8	9.3	6.0
$p25m20$	41 136.7	44 152.3	14.3	7.3
$p25m24$	62 536.8	67 871.5	18.9	8.5
$p28m25$	118 555.2	123 994.3	18.9	4.6
$p32m25$	147 814.5	155 273.5	25.8	5.0
$p30m30$	100 600.2	105 500.0	29.4	4.9
$p40m25$	207 853.8	219 031.5	32.2	5.4
平均值				5.0

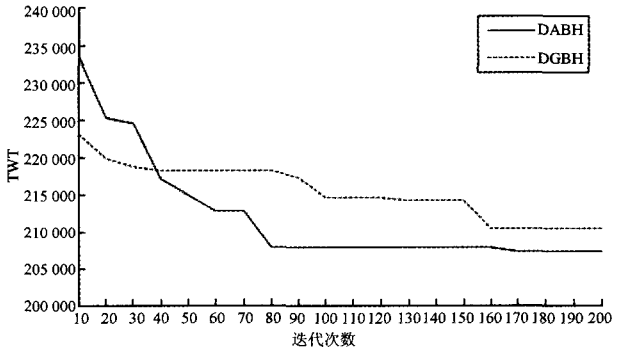

图 7　DABH 与 DGBH 的收敛过程比较

Fig. 7　Evolutionary processes of DABH and DGBH

从图 7 中可以看出, DABH 在第 80 代之后趋于稳定, DGBH 在第 150 代后趋于稳定, DABH 的收敛速度快于 DGBH 方法, 体现出 DABH 在寻优能力方面上的优势.

分析实验结果, DGBH 主要存在两方面局限性. 一方面, DGBH 的决策块编码中每位数字代表一个决策块的大小, 所有基因位上的数值总和应与决策总数相等. 然而在编码交叉变换时, DGBH 通过对两条父代染色体上随机位置上的编码进行交叉变换产生子代染色体, 这可能会产生基因位上的数值总和小于或大于决策总数的情况, 需要通过若干次加 1 或减 1 的调整获得可行解, 这个过程不但破坏了父代的优秀的遗传信息, 而且降低了计算效率. 另一方面, 在突变过程中, DGBH 的决策块编码随机选择两个元素交换, 启发式规则编码则是采取单点变换, 解的多样性受到限制. 相比而言, DABH 编码是通过信息素引导生成, 能更好地处理动态决策块大小的不确定性. 在每次迭代过程中, 所有蚂蚁根据信息素重新构建决策块及启发式规则, 增加了解的多样性, 保证了算法的寻优能力. 由此可见, ACO 算法在动态决策块划分方面具有一定优势.

4　结论

本文针对运输能力受限的跨单元调度问题的特点, 从实际应用的要求出发, 提出一种基于动态决策块的超启发式方法, 解决运输能力受限的跨单元调度问题. 在该方法中, 决策块的组成和大小通过迭代优化产生, 将传统方法中为每个实体或所有实体选择一个规则的形式转化为为每个决策块选择一个规则, 从而获得计算效率和优化能力之间的平衡. 实验结果表明, 与静态决策块相比, 动态决策块能够产生更合适的搜索空间, 既节省了计算时间, 又保留了合适的多样性. 与基于动态决策块的超启发式方法相比, ACO 比 GA 具有更好的性能. 这种差异的原因在于, 构造型算法与改进型算法相比, 能更好地处理动态决策块大小的不确定性, 而改进型算法则需要用更多的操作来处理进化过程中的不可行解. 由此可见, DABH 同时兼备计算效率和优化性能的优势, 因此非常适合在实际应用中解决规模大、复杂性高的跨单元调度问题.

考虑到不同的实体具有不同的属性, 在今后的工作中可以考虑根据实体之间的相关性划分决策块, 使决策块的划分过程与问题的特性相互适应, 从而提升算法的性能.

References

1 Wemmerlov U, Johnson D J. Cellular manufacturing at 46 user plants: implementation experiences and performance improvements. *International Journal of Production Research*, 1997, **35**(1): 29−49

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 Solimanpur M, Elmi A. A tabu search approach for cell scheduling problem with makespan criterion. *International Journal of Production Economics*, 2013, **141**(2): 639−645

4 Tang J, Wang X, 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

5 Tavakkoli-Moghaddam R, Javadian N, Khorrami A, Gholipour-Kanani Y. Design of a scatter search method for a novel multi-criteria group scheduling problem in a cellular

manufacturing system. *Expert Systems with Applications*, 2010, **37**(3): 2661−2669

6 Zeng C K, Tang J F, Yan C J. Job-shop cell-scheduling problem with inter-cell moves and automated guided vehicles. *Journal of Intelligent Manufacturing*, 2015, **26**(5): 845−859

7 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

8 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

9 Vepsalainen A P J, Morton T E. Priority rules for job shops with weighted tardiness costs. *Management Science*, 1987, **33**(8): 1035−1047

10 Ruiz R, Vázquez-Rodríguez J A. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 2010, **205**(1): 1−18

11 Park S C, Raman N, Shaw M J. Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach. *IEEE Transactions on Robotics and Automation*, 1997, **13**(4): 486−502

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 Huang J, Süer G A. A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels. *Computers & Industrial Engineering*, 2015, **86**: 29−42

14 Zhang R, Song S J, Wu C. A dispatching rule-based hybrid genetic algorithm focusing on non-delay schedules for the job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 2013, **67**(1−4): 5−17

15 Vázquez-Rodríguez J A, Petrovic S, Salhi A. An investigation of hyper-heuristic search spaces. In: Proceeding of the 2007 IEEE Congress on Evolutionary Computation. Singapore: IEEE, 2007. 3776−3783

16 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

17 Li D N, Li M, Meng X W, Tian Y N. A hyperheuristic approach for intercell scheduling with single processing machines and batch processing machines. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2015, **45**(2): 315−325

18 Jia Ling-Yun, Li Dong-Ni, Tian Yun-Na. An Intercell scheduling approach using shuffled frog leaping algorithm and genetic programming. *Acta Automatica Sinica*, 2015, **41**(5): 936−948
(贾凌云, 李冬妮, 田云娜. 基于混合蛙跳和遗传规划的跨单元调度方法. 自动化学报, 2015, **41**(5): 936−948)

19 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
(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)

20 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

21 Huang K L, Liao C J. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research*, 2008, **35**(4): 1030−1046

22 Montgomery D C. *Design and Analysis of Experiments (6th Edition)*. New York: John Wiley & Sons, 2005.

田云娜 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博士研究生, 延安大学数学与计算机科学学院讲师. 主要研究方向为进化计算与智能优化方法. E-mail: ydtianyunna@163.com
(**TIAN Yun-Na** Ph. D. candidate at the Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology and lecturer at the College of Mathematics and Computer Science, Yan′an University. Her research interest covers evolutionary computation and intelligent optimization approaches.)

李冬妮 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副教授. 主要研究方向为智能优化方法及其在制造业的应用. 本文通信作者.
E-mail: ldn@bit.edu.cn
(**LI Dong-Ni** Associate professor at the Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology. Her research interest covers intelligent optimization approaches and their applications to the manufacturing industry. Corresponding author of this paper.)

刘兆赫 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硕士研究生. 主要研究方向为进化计算和生产调度. E-mail: 719042341@qq.com
(**LIU Zhao-He** Master student at the Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology. His research interest covers evolutionary computation and production scheduling.)

郑 丹 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硕士研究生. 主要研究方向为进化计算和生产调度. E-mail: zhengdan04@163.com
(**ZHENG Dan** Master student at the Beijing Key Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology. Her research interest covers evolutionary computation and production scheduling.)