基于新型帝国竞争算法的高维多目标柔性作业车间调度

李　明, 雷德明†

(武汉理工大学 自动化学院, 湖北 武汉 430070)

摘要: 针对高维多目标柔性作业车间调度问题(MaOFJSP), 提出了一种新型帝国竞争算法(ICA)以同时最小化最大完成时间、最大拖期、最大机器负荷和总能耗, 该算法采用新方法构建初始帝国使得大多数殖民国家分配数量相近的殖民地, 引入殖民国家的同化, 并应用新的革命策略和帝国竞争方法以获得高质量解. 最后通过大量实验测试ICA新策略对其性能的影响并将ICA与其他算法对比, 实验结果表明新型ICA在求解MaOFJSP方面具有较强的优势.

关键词: 高维多目标优化; 柔性作业车间调度; 帝国竞争算法; 低碳调度

引用格式: 李明, 雷德明. 基于新型帝国竞争算法的高维多目标柔性作业车间调度. 控制理论与应用, 2019, 36(6): 893 – 901

DOI: 10.7641/CTA.2018.80105

Novel imperialist competitive algorithm for many-objective flexible job shop scheduling

LI Ming, LEI De-ming†

(School of Automation, Wuhan University of Technology, Wuhan Hubei 430070, China)

Abstract: In this study many-objective flexible job shop scheduling problem (MaOFJSP) is investigated, the goal of which is to minimize makespan, maximum tardiness, maximum workload of machine and total energy consumption simultaneously. A novel imperialist competitive algorithm (ICA) is proposed, in which initial empires are newly constructed so that most of imperialists are assigned close number of colonies. Assimilation of imperialist is introduced, and revolution and imperialist competition are implemented in a new way, respectively, to produce high quality solutions. Extensive experiments are conducted to test the impact of strategies of ICA on its performance and compare ICA with other algorithms finally. The experimental results validate that ICA has a strong advantage for MaOFJSP.

Key words: many-objective optimization; flexible job shop scheduling problem; imperialist competitive algorithm; energy-efficient scheduling

Citation: LI Ming, LEI Deming. Novel imperialist competitive algorithm for many-objective flexible job shop scheduling. *Control Theory & Applications*, 2019, 36(6): 893 – 901

1　引言

柔性作业车间调度问题(flexible job shop scheduling problem, FJSP)是作业车间调度问题(job shop scheduling problem, JSP)的扩展, 其每一道工序可以在给定机器集中的任一机器上加工, 突破了资源唯一性. FJSP是复杂的NP–hard问题, 广泛应用于汽车装配、纺织、化工材料加工和半导体制造等[1–2]. 自Bruker和Schlie[3]的开创性工作以来, FJSP备受关注, 出现了大量研究成果[1–21].

目前, 智能算法已广泛应用于多目标柔性作业车间调度问题(multi-objective flexible job shop scheduling problem, MOFJSP)的求解. 早期的工作为Kacem等[4]提出的基于遗传算法 (genetic algorithm, GA) 和局部方法的混合算法, 后来, 研究者大量应用GA[5]、粒子群算法[2,6]、人工蜂群算法[7]、禁忌搜索算法[8]、变邻域搜索算法[9]、蛙跳算法[10]以及分布估计算法[11]等解决MOFJSP.

低碳调度是一类综合考虑能源效率和制造活动目的且子问题较多的复杂优化问题, 该问题及其优化方法是实施低碳制造的重要途径, 近年来, 低碳调度问题引起了学术界和工业界的广泛关注[12]. 关于低碳FJSP, Jiang等[13]提出了基于血缘变异的改进NSGA–

收稿日期: 2018−02−06; 录用日期: 2018−07−02.

†通信作者. E-mail: deminglei11@163.com; Tel.: +86 15327311013.

本文责任编委: 胡跃明.

国家自然科学基金项目(61573264, 71471151)资助.

Supported by the National Natural Science Foundation of China (61573264, 71471151).

II. He 等[14]提出了一种节能优化算法, 它通过机床选择减少机器加工能耗和操作序列调整减少机器闲置时的能源浪费. Yin等[15]设计了一种多目标遗传算法(multi-objective genetic algorithm, MOGA)求解以生产率、能源效率和噪音减少为目标的FJSP. Lei等[16–19]分别应用蛙跳算法、教学优化算法和帝国竞争算法(imperialist competitive algorithm, ICA)解决了具有不同目标的低碳FJSP. Piroozfard等[20]提出一种MOGA以同时最小化碳排放量和总拖期. Mokhtari等[21]给出基于进化算法和模拟退火算法的混合算法以同时优化总完成时间、总能耗和总利用率.

MOFJSP根据目标个数可以分成: 1) 常规MOFJSP, 通常只优化2个或者3个目标, 包括最大完成时间和总拖期等; 2) 高维多目标柔性作业车间调度问题(many-objective flexible job shop scheduling problem, MaOFJSP), 其目标个数大于等于4. 现有研究以常规MOFJSP为主, 很少考虑MaOFJSP, 实际上, MaOFJSP广泛存在于实际生产系统中, 例如, 随着总能耗或碳排放目标的引入, 文献[1, 7]所研究的三目标FJSP将转化为MaOFJSP. 和常规MOFJSP相比, MaOFJSP的计算复杂度和搜索难度更大, 求解该问题的智能算法所获得的非劣解在种群中所占比例显著上升, 因此, 有必要根据MaOFJSP的特点探讨有效的解决方法.

帝国竞争算法 (imperialist competitive algorithm, ICA[22])是一种模拟社会政治行为的新型智能算法, 其搜索始于由一组国家组成的初始种群, 其中一些成本值最小的国家被选作殖民国家, 其他国家为殖民地. ICA主要由殖民地同化、殖民地革命和帝国竞争等步骤组成, 它已成功应用于设备布局[23]、调度[19,24]和装配线平衡[25]等, 只是很少用来解决MOFJSP[19]和MaOFJSP. ICA既具有较强的邻域搜索能力, 又是有效的全局优化方法且结构灵活[26], 这些特点和ICA的成功应用表明, 应该大力探讨ICA在MaOFJSP求解方面的优势.

本文研究MaOFJSP并提出了一种新型ICA以同时最小化最大完成时间、最大拖期、最大机器负荷和总能耗, 该算法通过应用初始帝国构建的新策略、引入殖民国家的同化, 并结合新的革命策略和帝国竞争方法以逼近问题的Pareto最优前端. 最后通过大量实验测试ICA的新策略对其性能的影响并将ICA与其他算法对比, 实验结果表明新型ICA在求解MaOFJSP方面具有较强的优势.

2 问题描述

MaOFJSP描述如下: 存在工件集$J=\{J_1, J_2, \cdots, J_n\}$和机器集$M=\{M_1, M_2, \cdots, M_m\}$. 工件$J_i$具有$h_i$道工序, 工序$o_{ij}$为工件$J_i$的第$j$道工序, 该工序可由相容机器集$S_{ij}$中的任何一台机器加工, $S_{ij}\subset M$; 机器M_k具有2种模式: 加工模式和空闲模式; E_k, SE_k分别表示机器M_k在加工模式和空闲模式时单位时间的能耗.

存在一些与机器和工件相关的约束, 包括同一时刻一台机器最多只能加工一道工序; 同一时刻一个工件最多只能在一台机器上加工; 机器加工不能中断; 准备时间和清理时间包含在加工时间内等.

MaOFJSP包括调度子问题和机器分配子问题, 前者确定每台机器上各个工序的加工顺序, 后者为每道工序选择合适的机器. 通常, 调度子问题比机器分配子问题更复杂, 求解难度也更大.

MaOFJSP的目的是在所有约束得到满足的条件下同时最小化如下4个目标函数:

$$f_1 = C_{\max}, \tag{1}$$

$$f_2 = \max_{1\leqslant i\leqslant n}\{C_i - D_i, 0\}, \tag{2}$$

$$f_3 = \sum_{k=1}^{m}\int_0^{C_{\max}}(\sum_{i=1}^{n}\sum_{j=1}^{h_i}E_k y_{ijk}(t)+SE_k z_k(t))\mathrm{d}t, \tag{3}$$

$$f_4 = \max_{1\leqslant k\leqslant m} W_k, \tag{4}$$

其中: $y_{ijk}(t)$为二进制量, 如果在时刻t机器$M_k\in S_{ij}$处于加工模式, 则$y_{ijk}(t)=1$; 否则$y_{ijk}(t)=0$; 如果机器M_k在时刻t处于空闲, 则$z_k(t)=1$; 否则$z_k(t)=0$. $C_{\max}$表示最大完成时间. C_i, D_i分别表示工件J_i的完成时间和交货期. W_k表示机器M_k的负荷. f_1, f_2, f_3和f_4分别表示最大完成时间、最大拖期、总能耗和最大机器负荷. 4个目标中, f_1反映车间的生产效率, 优化f_2可以提高客户满意度, 总能耗f_3为机器在加工模式和空闲模式下能耗之和, 对其优化能够有效降低能耗, 而减小f_4有利于提高机器利用率.

现有研究表明, 最大完成时间与最大机器负荷之间存在冲突关系[1,7], 最大完成时间、最大拖期和总能耗之间也存在冲突关系[16–21], 另外, 最大机器负荷、最大拖期和总能耗也彼此冲突[27–28], 由此可以得出, 本文的4个目标间不可避免地存在冲突关系.

机器分配子问题的优化将直接决定f_4和所有机器加工模式下的能耗总和, 即f_3的第1部分的积分, 同时也会改变f_1和f_2, 而f_3的第2部分的积分, 即空闲模式下的总能耗直接取决于调度子问题的求解质量, 由f_4的定义可知, f_4的大小不受调度子问题的影响, 和机器分配相比, 调度子问题的解的质量对前2个目标的影响更大; 再结合调度子问题具有更高的复杂程度, 需要对调度子问题分配更多计算资源以提高算法的搜索速度.

对于多目标优化问题, 假设其目标总数为G且都是最小化目标, 其最优解不再只是一个解, 而是一组

解, 且需要通过比较所有解才能确定最优解. 通常, 要用到如下几个概念: 对于解x和y, 如果对于$\forall i \in \{1, 2, \cdots, G\}$, $f_i(x) \leqslant f_i(y)$且$\exists i \in \{1, 2, \cdots, G\}$, $f_i(x) < f_i(y)$, 则解x支配y. 对于集合Φ和解$x \in \Phi$, 如果x不受集合Φ的其他任何解支配, 则x关于集合Φ是非劣的. 如果不受问题解空间内的任何其他解支配, 则该解为Pareto最优解.

3　帝国竞争算法描述

ICA是一种基于种群的智能算法. 种群中的每一个体表示一个国家, 在种群初始化时, 一些最优的国家被选作殖民国家. 每个帝国是由一个殖民国家和若干殖民地组成, 通过殖民地的同化和革命、殖民国家更新以及帝国竞争不断逼近最优解.

ICA的主要步骤如下:

步骤 1　初始化帝国. 随机生成N个初始国家, 将成本最小的$N_{\rm im}$个国家定义为殖民国家, 然后根据其自身势力大小为每个殖民国家分配相应数量的殖民地, 构建初始帝国.

步骤 2　同化和革命. 在每个帝国内对殖民地执行同化和革命.

步骤 3　殖民国家更新. 比较帝国内的殖民地和殖民国家成本值, 成本最小的国家成为新的殖民国家.

步骤 4　帝国竞争. 根据帝国的总成本, 将最弱帝国的最弱殖民地重新分配到新的帝国.

步骤 5　帝国删除. 如果一个帝国内不存在殖民地, 则直接删除该帝国.

步骤 6　如果满足终止条件, 则搜索结束; 否则, 转至步骤2.

根据目标函数计算每个国家的成本值. 解的质量越好, 则其成本值越小. 成本最小的$N_{\rm im}$个解选作殖民国家, 其余国家为殖民地. 殖民地的数量为$N_{\rm c} = N - N_{\rm im}$.

为了构建初始帝国, 需要将殖民地分配给各个殖民国家, 每个殖民国家分配殖民地的数量与其势力成正比, 其势力计算如下:

$$p_k = \left|\frac{Y_k}{\sum\limits_{v=1}^{N_{\rm im}} Y_v}\right|, \tag{5}$$

其中: p_k为殖民国家k的势力, $Y_v = \max\limits_k \{c_k\} - c_v$表示归一化成本, c_k为殖民国家k的成本值.

殖民国家k分配殖民地的数量NC_k为round $\{P_k \times N_{\rm c}\}$, round为四舍五入取整函数.

同化过程中, 帝国中的殖民地沿着朝向殖民国家的方向移动ε. 移动距离ε是区间$[0, s \times e]$中的一个随机数, 其中$s \in (1, 2)$; e是殖民地与殖民国家间的距离. 设置$s > 1$使得殖民地可以从两侧朝着殖民国家移动. 为了扩大搜索范围, 加入一个随机偏移量θ, 它服从$[-\varphi, \varphi]$之间的均匀分布, 其中φ是一个参数.

革命与GA的变异操作相似, 它能够增强ICA的搜索能力, 防止早期收敛到局部最优.

在同化与革命结束后, 把每个殖民地的成本与殖民国家的成本进行比较, 如果殖民地的成本低于殖民国家, 则将两者进行互换.

4　基于新型ICA的MaOFJSP

由于MaOFJSP具有4个目标, 初始种群中的非劣解的数量相对较大, 导致大多数殖民国家甚至所有殖民国家彼此非劣, 应为这些殖民国家分配相近数量的殖民地以同等对待它们. 除了殖民地同化, 还引入殖民国家的同化, 有助于不断地提高殖民国家的质量, 而随机选择殖民地参加革命, 很可能导致计算资源的浪费. 另外, 帝国竞争时, 至少一个帝国的势力为0, 导致这些帝国在竞争中难以取胜. 根据MaOFJSP和ICA的特点, 提出了一种新型ICA以解决该问题.

4.1　初始帝国

初始帝国构建是ICA的关键步骤. 为了构建初始帝国, 首先产生初始种群, 其中每个解都用2个串表示. 针对具有n个工件和m台机器的MaOFJSP, 通常采用调度串$[(\theta_1, r_1)\ \ (\theta_2, r_2)\ \ \cdots\ \ (\theta_i, r_i)\ \ \cdots\ \ (\theta_h, r_h)]$和机器分配串$[q_{11}\ \ q_{12}\ \ \cdots\ \ q_{1h_1}\ \ \cdots\ \ q_{nh_n}]$来表示问题的一个解, 其中$h = \sum\limits_{i=1}^{n} h_i$表示工序总数, 调度串中, $\theta_i \in \{1, 2, \cdots, n\}$, $1 \leqslant r_i \leqslant h_{\theta_i}$. 二元组$(\theta_i, r_i)$对应工序$o_{\theta_i r_i}$, 这样整个串对应一个有序工序表$[o_{\theta_1 r_1}\ \ o_{\theta_2 r_2}\ \ \cdots\ \ o_{\theta_i r_i}\ \ \cdots\ \ o_{\theta_h r_h}]$. 机器分配串中, 基因$q_{ij} \in S_{ij}$表示用于加工工序$o_{ij}$的一台机器.

初始帝国构建过程如算法1所示. 随机产生W个初始种群后, 容易找到有η个非劣解的种群, η与$N_{\rm im}$接近甚至大于$N_{\rm im}$. 与殖民地随机分配给殖民国家不同, 本文在所有殖民地按rank值升序排序后, 依次将殖民地分配给相应的殖民国家, 这样有助于使大多数帝国具有相近的势力, 从而使帝国竞争可以充分进行.

算法 1　初始帝国构建.

随机产生W个初始种群.

选择非劣解数量η最大的种群作为初始种群P.

对种群P非劣排序.

if $\eta < N_{\rm im}$, then η个非劣解以及$N_{\rm im} - \eta$个最优的受支配解作为殖民国家

else 随机选择$N_{\rm im}$个非劣解作为殖民国家

令$A = \lfloor N/N_{\rm im} \rfloor$

for $k = 1$ to $N_{\rm im}$

　if 殖民国家k是非劣解, then

$NC_k = A + a,\ a \in \{0, 1\}$

else $NC_k = A + a,\ a \in \{-1, -2\}$

if $k = N_{\rm im}$, then $NC_{N_{\rm im}} = N_{\rm c} - \sum_{k=1}^{N_{\rm im}-1} NC_k$

end for

对所有殖民地按rank值升序排序, 令$s = 1$

for $k = 1$ to $N_{\rm im}$

if 殖民国家k未分配NC_k个殖民地, then

殖民地s分配给殖民国家k, $s = s + 1$

end for

4.2 同化

同化过程使得殖民地与殖民国家更加相似, 通常通过殖民地和殖民国家之间的交叉[24]来实现, 只是现有研究很少考虑殖民国家的同化. 给出了新的同化过程, 如算法2所示.

算法 2 同化.

for $k = 1$ to $N_{\rm im}$

if Ω中存在与殖民国家k不同的解$x \in \Omega$, then

if $s < \alpha$, then

对解$x \in \Omega$与殖民国家k执行第1种交叉操作

else 对两个解执行第2种交叉操作

得到新解z并和殖民国家k进行比较

if 解z不被殖民国家k支配, then

用解z替换殖民国家k并更新Ω

end if

end for

for $g = 1$ to NC_k

if $s < \alpha$, then

对殖民地g与殖民国家k执行第1种交叉操作

else 对两个解执行第2种交叉操作

得到新解z并和殖民地g比较

if 解z不被殖民地g支配, then

用解z替换殖民地g并更新Ω

end if

end for

针对MaOFJSP的2个子问题, 各选择一种交叉操作. 第1种交叉作用于调度串, 具体描述如下: 对于解x和y的调度串, 从第1个位置开始, 随机产生一个随机数s, 如果$s < \delta$, 则选择解x 的第1个二元组, 否则选择y的第1个二元组; 假设选择二元组(θ_i, r_i), 将该二元组直接添加到子代, 然后从x和y中删除, 如果(θ_i, r_i)位于y的调度串的第g位, 则将位于第$g + 1$位和最后一位之间的所有二元组左移一位, 重复上述步骤直到解x和y的调度串为空; 最后产生一个新的二元组调度串, 其中δ是一个概率.

第2种交叉作用于机器分配串, 其具体过程如下: 对于解x和y, 随机确定两个位置$g_1, g_2 \in [1, h]$, 然后将解x的机器分配串中位于两个位置间的机器用解y相同位置上的机器取代.

与常规交叉操作不同, 以上两种交叉操作仅产生一个新解且解y保持不变. 如前所述, 调度子问题比机器分配子问题更复杂, 求解难度更大, 对前3个目标的影响也更大, 通常设置$\alpha > 0.5$以分配更多的计算资源给调度子问题, 通过实验确定$\alpha = 0.6$.

外部档案Ω的更新过程如下: 将新解加入到档案Ω之后, 对档案内的所有解进行Pareto比较, 保留所有非劣解, 同时剔除所有受支配解.

通常, 殖民国家仅仅根据同化和革命的计算结果进行更新. 如果一个帝国中的殖民国家在很多代内无法得到改善, 由于帝国中的所有殖民地都试图变得与殖民国家更相似, 这样殖民地的相似性将大大增加, 导致帝国或者整个种群的多样性显著减少, ICA的搜索将停滞不前; 而引入殖民国家同化, 提供了产生新殖民国家的新途径, 从而提高了殖民国家的更新频率; 由于殖民国家是优秀的个体, 使用优秀解更容易产生质量较好的解, 因此, 殖民国家同化的引入是合理的.

4.3 革命

殖民地革命是ICA产生新解的另一种方式. 通常, 帝国中的每个殖民地都有相同的概率被选中作为革命者. 由于革命概率$U_{\rm R}$通常小于0.4, Afruzi 等[29] 和Bayareh等[30]分别将其设为0.35和0.3, 故革命是稀缺资源. 如果选择一些较弱殖民地进行革命, 将很难产生较好的解并且可能浪费计算资源; 相反, 如果仅让部分较好的殖民地参加革命, 并将所有用于革命的计算资源分配给它们, 则这些殖民地可以得到较充分进化, 容易产生更好的殖民地.

提出了新的革命策略, 其具体过程如算法3所示, 其中R是一个整数. 算法3中所有殖民地依据Pareto支配进行排序, 最优的殖民地为受最少的殖民地支配的解.

算法 3 革命.

for $k = 1$ to $N_{\rm im}$

根据$U_{\rm R}$确定参加革命的殖民地数量α

if $\alpha > 1$, then

从帝国k中未被选择的殖民地选择最好的殖民地λ, 令$g \leftarrow 1$

for $l = 1$ to R

if $g = 1$, then 对殖民地λ执行insert

else 对殖民地λ执行change

得到新解z并和殖民地λ比较

if 解z不被殖民地λ支配, then

用解z替换殖民地λ并更新Ω

else $g \leftarrow g+1$且若$g=3$, 则令$g \leftarrow 1$

end for

end if

end for

运用2种邻域结构insert和change产生新解, 其中insert将从调度串中随机选择的元素(θ_i, r_i)插入到新位置$k \neq i$并随后根据θ_i在调度串中出现的次序为r_i重新赋值. change的实现过程如下: 首先构建集合

$$\Theta=\{q_{ij}||S_{ij}|>1,\ i=1,2\cdots,n,\ j=1,2\cdots,h_i\};$$

然后从该集合中随机选择一些元素为它们重新赋值, 假设q_{ij}被选中, 则从S_{ij}中随机确定一台机器替代当前的q_{ij}.

同化和革命后更新殖民国家. 每个殖民地都与其殖民国家进行比较, 如果殖民地支配殖民国家或者与殖民国家彼此非劣, 则殖民地变为新的殖民国家, 而原殖民国家变为殖民地.

4.4　帝国竞争

ICA中, 通常至少存在一个帝国其势力为0, 这些势力为0的帝国难以在帝国竞争中成为胜利者, 这样所有国家将很快聚集在一个帝国中, 帝国竞争无法充分进行, 因此, 需要保证每个帝国的势力都大于0.

为了实现帝国竞争, 首先定义每个国家的成本值. 对于解i其成本值c_i计算如下:

$$c_i = \mathrm{rank}_i + \frac{\mathrm{dist}_3^i}{(\varepsilon + \sum\limits_{l\in H_{\mathrm{rank}_i}} \mathrm{dist}_3^l)}, \tag{6}$$

其中: rank_i是解i通过非劣排序确定的rank值, dist_3^i是解i与其最近3个解之间的平均距离, H_l表示rank值为l的集合, ε是一个接近0的正数, 目的是为了防止分母等于0.

总成本值TC_k为

$$\mathrm{TC}_k = c_k + \zeta \times \mathrm{mean}\left\{\mathrm{Cost}(\text{colonies of empire } k)\right\}.$$

通常, 归一化总成本为$\mathrm{NTC}_k = \max\limits_h\{\mathrm{TC}_h\} - \mathrm{TC}_k$; 这样帝国$k$的势力$\mathrm{EP}_k$可能为0.

$$\mathrm{EP}_k = \left|\frac{\mathrm{NTC}_k}{\sum\limits_{v=1}^{N_{\mathrm{im}}} \mathrm{NTC}_v}\right|. \tag{7}$$

为了确保所有帝国的势力都大于0, 重新定义归一化总成本:

$$\mathrm{NTC}_k = 2 \times \max_h\{\mathrm{TC}_h\} - \mathrm{TC}_k. \tag{8}$$

由式(8)可知, 帝国间势力值差别将大大减少, 这样每个帝国都可能成为竞争中的胜利者, 从而使得帝国竞争充分.

4.5　算法描述

新型ICA具体过程与第3节的过程相同, 只是采用新方式实现了初始帝国的构建、同化、革命和帝国竞争, 终止条件为最大目标向量估计次数$\max_it$. 新型ICA的流程图如图1所示. 该算法的时间复杂度为$\mathrm{O}(N \times G \times \bar{N}^2 \times L)$, 其中种群规模为$N$, 外部档案的大小是$\bar{N}$, L表示算法循环部分的循环次数.

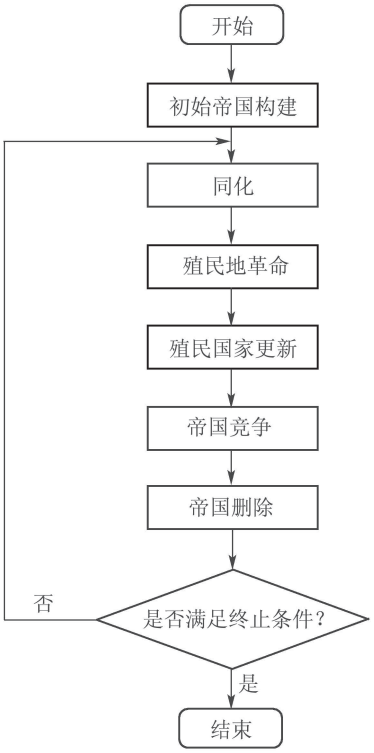

图 1　新型ICA流程图

Fig. 1　Flow chart of the novel ICA

新算法具有如下特点:

1) 未利用成本值确定殖民地的数量, 并采用顺序方式给大多数殖民国家分配数量相近的殖民地.

2) 殖民国家和殖民地都执行同化; 不过, 这两类国家的同化过程各不相同.

3) 为了充分利用计算资源,选择最优的殖民地进行革命, 帝国竞争采用新的执行方式, 以实现帝国间的充分竞争.

4) ICA中既存在邻域搜索也存在全局搜索, 这样有利于维持探索和利用间的良好平衡.

5　计算实验

为了测试ICA在求解MaOFJSP方面的性能, 进行了大量实验. 所有的实验均由Microsoft Visual C++ 2015编程实现, 并运行于4.0 G RAM 2.00 GHz CPU PC.

5.1　测试实例、评价指标和比较算法

选择MK01–15[31]和DP1–18[32]作为测试实例, 为此在实例中引入能耗信息$E_k \in [2,4]$, $SE_k = 1$, 其

交货期计算公式如下:

$$D_i = \delta \sum_{j=1}^{h_i} \max_{k=1,2,\cdots,m} \{p_{ijk}\}, \tag{9}$$

其中δ为某一区间的随机实数, 如表1所示.

表 1 δ的设置

Table 1 Setting on δ

δ	实例	δ	实例
[0.5, 0.7]	MK01, 03–05	[1, 1.5]	DP1–12
[0.3, 0.5]	MK02, 06, 08–10	[1.2, 1.6]	MK11–15
[1.2, 1.7]	DP13–18	[0.7, 0.9]	MK07

多目标优化算法结果评价指标较多, 本文选择$DI_{\rm R}$, ρ_l和nd_l3个指标评估算法性能.

指标$DI_{\rm R}$[33]用于评价算法的收敛性能, 它为非劣解集Ω_l与参考集Ω^*之间的距离.

$$DI_{\rm R}(\Omega_l) = \frac{1}{|\Omega^*|} \sum_{y\in\Omega^*} \min\{\sigma_{xy} \,|x \in \Omega_l\}, \tag{10}$$

其中: σ_{xy}表示解x与参考解y在归一化目标空间中的距离. 参考集Ω^*由并集$\cup_l\Omega_l$中的非劣解组成. $DI_{\rm R}$ (Ω_l) 越小, 则Ω_l的解越好.

指标ρ_l[34]等于集合$\{x \in \Omega_l \,|x \in \Omega^*\}$的大小与$|\Omega^*|$的比值.

nd_l为非劣解数量, 定义为$|\{x \in \Omega_l | x \in \Omega^*\}|$, nd_l表示由每一种算法产生的为参考集Ω^*提供的非劣解数量.

选择VNS[9]和多目标遗传算法(MOGA[20])作为比较算法. Bagheri和Zandieh[9]运用VNS求解双目标FJSP, 加入外部档案更新策略和当前解被新解代替的条件后, VNS可用于求解MaOFJSP. Piroozfard等[20]提出了MOGA求解双目标低碳FJSP, 该算法能够直接应用于MaOFJSP的求解.

ICA具有4个主要参数: 种群规模N, 初始帝国总数$N_{\rm im}$, 目标向量估计次数max_it和邻域搜索次数R. 对这4个参数的3种设置即N的60, 80, 100, $N_{\rm im}$的5, 6, 7, R 的6, 8, 10 以及max_it的0.8×10^5, 10^5, 1.2×10^5进行81组对比实验, 最后确定使ICA性能更好的一组参数为$N=80$, $N_{\rm im}=6$, $\max\_{\rm it}=10^5$, $R=8$.

MOGA的参数设置来自文献[20]: 种群规模为100, 交叉概率为0.7, 变异概率为0.4. VNS的终止条件为$\max_{\rm it}=10^5$, 其他参数取自文献[9]. MOGA和VNS的主要参数虽来自相应的文献, 但实验表明, 利用这些参数设置2种算法都获得了最好的计算结果. 3种算法当最大目标向量估计次数达到10^5时, 停止搜索.

5.2 ICA新策略的影响

首先构建新型ICA的3种变形: ICA1, ICA2和ICA3. 在ICA1中, 使用常规方法计算帝国的归一化总成本, 通过比较ICA和ICA1, 可确定新定义的归一化总成本对ICA性能的影响. 在ICA2中, 不存在殖民国家同化. ICA3中, 随机选择殖民地参加革命. 表2–3列出了4种算法的计算结果, 参考集Ω^*由4种算法外部档案的并集$\cup_l\Omega_l$中的非劣解组成.

如表2–3所示, ICA关于17个实例获得了比ICA1更小的$DI_{\rm R}$值, 关于11个实例, ICA提供了参考集中的所有成员, 这表明避免帝国势力为0是必要的, 可以使得帝国间竞争更加充分.

表 2 ICA, ICA1, ICA2和ICA3关于指标$DI_{\rm R}$的计算结果

Table 2 Computational results of four ICAs on metric $DI_{\rm R}$

实例	ICA	ICA1	ICA2	ICA3
MK01	5.357	6.442	8.192	11.11
MK02	18.99	12.17	23.50	4.164
MK03	11.67	11.13	10.97	19.89
MK04	5.401	3.386	8.944	7.917
MK05	5.504	5.721	8.032	4.377
MK06	8.064	8.359	7.289	6.484
MK07	4.713	3.540	8.549	8.870
MK08	8.582	10.73	5.624	13.76
MK09	17.21	8.874	5.481	19.93
MK10	3.648	17.85	24.07	5.790
MK11	5.265	5.118	7.986	3.105
MK12	7.761	3.809	9.824	7.083
MK13	4.248	4.658	18.56	7.796
MK14	4.202	6.834	15.23	9.526
MK15	3.186	7.763	26.40	12.36
DP1	3.419	17.39	15.73	12.06
DP2	6.753	3.632	6.754	5.018
DP3	15.31	14.40	17.68	17.04
DP4	18.02	0.000	23.16	25.98
DP5	22.59	1.135	31.64	12.22
DP6	2.594	8.317	7.392	9.183
DP7	17.19	4.704	8.613	9.564
DP8	5.020	7.805	13.07	5.223
DP9	7.334	5.233	12.94	6.201
DP10	5.087	3.410	11.03	8.165
DP11	5.921	8.279	12.50	6.089
DP12	6.799	8.971	9.545	5.880
DP13	6.878	7.951	14.42	4.112
DP14	7.627	11.49	14.49	8.996
DP15	7.140	2.680	12.26	8.043
DP16	8.365	6.502	16.61	12.98
DP17	3.926	9.796	19.18	8.909
DP18	7.960	8.484	9.287	8.585

表 3 ICA, ICA1, ICA2和ICA3关于指标nd_l和ρ_l的计算结果

Table 3 Computational results of four ICAs on metrics nd_l and ρ_l

实例	ICA	ICA1	ICA2	ICA3
MK01	18, 0.400	17, 0.378	9, 0.200	1, 0.022
MK02	2, 0.154	3, 0.231	4, 0.308	4, 0.308
MK03	13, 0.277	20, 0.426	13, 0.277	1, 0.021
MK04	23, 0.250	42, 0.457	10, 0.109	17, 0.185
MK05	12, 0.250	13, 0.271	3, 0.063	20, 0.417
MK06	27, 0.231	24, 0.205	26, 0.222	40, 0.342
MK07	30, 0.380	41, 0.519	7, 0.089	1, 0.127
MK08	33, 0.355	10, 0.108	35, 0.376	15, 0.161
MK09	1, 0.024	16, 0.390	24, 0.585	0, 0.000
MK10	28, 0.444	0, 0.000	0, 0.000	35, 0.556
MK11	37, 0.245	33, 0.219	6, 0.040	35, 0.497
MK12	8, 0.119	43, 0.642	3, 0.045	13, 0.194
MK13	33, 0.398	29, 0.349	0, 0.000	21, 0.253
MK14	76,0.563	31, 0.230	2, 0.015	26, 0.193
MK15	53, 0.746	12, 0.169	0, 0.000	6, 0.085
DP1	17, 0.708	1, 0.042	2, 0.083	4, 0.167
DP2	5, 0.100	25, 0.500	1, 0.020	19, 0.380
DP3	5, 0.250	5, 0.250	1, 0.050	9, 0.450
DP4	0, 0.000	15, 1.000	0, 0.000	0, 0.000
DP5	0, 0.000	55, 0.932	0, 0.000	4, 0.068
DP6	32, 0.627	9, 0.176	9, 0.176	1, 0.020
DP7	1, 0.056	11, 0.611	2, 0.111	4, 0.222
DP8	47, 0.452	12, 0.115	1, 0.010	44, 0.423
DP9	18, 0.290	22, 0.355	3, 0.048	19, 0.306
DP10	57, 0.333	90, 0.526	6, 0.035	18, 0.105
DP11	29, 0.397	12, 0.164	1, 0.014	31, 0.425
DP12	18, 0.305	10, 0.169	1, 0.017	30, 0.508
DP13	40, 0.310	21, 0.163	4, 0.031	64, 0.496
DP14	16, 0.516	1, 0.032	0, 0.000	14, 0.452
DP15	30, 0.133	56, 0.747	4, 0.053	5, 0.067
DP16	42, 0.347	52, 0.430	0, 0.000	27, 0.223
DP17	55, 0.591	13, 0.140	0, 0.000	25, 0.269
DP18	14, 0.194	16, 0.222	16, 0.222	26, 0.361

从表2–3可以发现, ICA的3个指标值都优于ICA2. ICA关于27个实例所获得的DI_{R}值都小于ICA2, 且关于7个实例的2种算法的DI_{R}值差别大于10; 和ICA2相比, ICA能够为Ω^*提供更多的非劣解, 这些结果表明引入殖民国家同化操作有利于提高ICA的性能.

表2–3的结果也表明, ICA的性能优于ICA3. 关于22个实例, ICA的DI_{R}值小于ICA3, 革命者的合理选择非常必要, 事实上, 当最优的殖民地被选作革命者时, 可能产生质量更好的解, 总之, 新型ICA的3种新策略能有效地改善其性能.

5.3 ICA, VNS 和 MOGA 的比较

表4–6给出了ICA, VNS和MOGA的计算结果和运行时间, 其中参考集Ω^*由3种算法非劣解集的并集中的非劣解组成. 从中可以看出, 关于大多数实例, ICA取得了优于其他2种算法计算结果. MOGA和VNS关于31个实例的解总是远离ICA的非劣解, ICA关于10个实例产生了参考集的所有非劣解, 关于31个实例获得了参考集的大部分非劣解.

表 4 3种算法关于指标DI_{R}的计算结果

Table 4 Computational results of three algorithms on metric DI_{R}

实例	ICA	VNS	MOGA
MK01	0.943	36.49	15.39
MK02	0.000	55.81	15.55
MK03	0.000	75.93	12.87
MK04	0.545	27.04	11.36
MK05	0.150	33.84	11.43
MK06	0.700	50.80	10.52
MK07	0.169	66.00	6.612
MK08	0.000	41.78	31.60
MK09	0.000	95.51	43.08
MK10	0.000	81.25	61.77
MK11	0.090	50.74	18.27
MK12	0.000	47.41	34.99
MK13	0.000	86.67	35.97
MK14	0.000	39.95	39.41
MK15	0.000	76.60	43.87
DP1	20.75	5.588	38.93
DP2	3.417	22.92	41.09
DP3	1.713	20.08	22.96
DP4	40.10	5.776	30.49
DP5	0.962	25.30	20.69
DP6	3.327	17.38	19.02
DP7	0.000	21.23	32.27
DP8	3.226	30.49	22.02
DP9	4.365	26.39	22.19
DP10	1.911	35.96	28.35
DP11	2.582	29.26	23.38
DP12	2.269	27.69	14.46
DP13	1.096	38.68	21.29
DP14	1.490	48.28	21.05
DP15	3.541	40.15	15.34
DP16	0.952	35.66	21.21
DP17	1.820	37.52	32.26
DP18	3.571	43.20	17.47

ICA的优越性能主要来自于初始帝国的构建、同化和帝国竞争这些新策略. 初始帝国的构建使得大多数殖民国家分配数量相近的殖民地, 新帝国竞争方法使得竞争更加充分, 算法可以避免早熟. 殖民国家同化与选择最优殖民地革命可大幅度提高搜索能力

和获得高质量解的可能性, 可以得出ICA能够有效地求解MaOFJSP.

表 5 3种算法关于指标nd_l和ρ_l的计算结果

Table 5 Computational results of three algorithms on metrics nd_l and ρ_l

实例	ICA	VNS	MOGA
MK01	31, 0.940	0, 0.000	2, 0.061
MK02	9, 1.000	0, 0.000	0, 0.000
MK03	63, 1.000	0, 0.000	0, 0.000
MK04	45, 0.918	0, 0.000	4, 0.082
MK05	50, 0.962	0, 0.000	1, 0.038
MK06	62, 0.954	0, 0.000	3, 0.046
MK07	66, 0.943	0, 0.000	4, 0.057
MK08	75, 1.000	0, 0.000	0, 0.000
MK09	54, 1.000	0, 0.000	0, 0.000
MK10	45, 1.000	0, 0.000	0, 0.000
MK11	126, 0.992	0, 0.000	1, 0.008
MK12	76, 1.000	0, 0.000	0, 0.000
MK13	51, 1.000	0, 0.000	0, 0.000
MK14	92, 1.000	0, 0.000	0, 0.000
MK15	66, 1.000	0, 0.000	0, 0.000
DP1	19, 0.380	31,0.620	0, 0.000
DP2	78, 0.907	0, 0.000	8, 0.093
DP3	90, 0.891	4, 0.040	7, 0.069
DP4	0, 0.000	8, 0.889	1, 0.111
DP5	61, 0.938	0, 0.000	4, 0.062
DP6	43, 0.956	0, 0.000	2, 0.044
DP7	5, 1.000	0, 0.000	0, 0.000
DP8	50, 0.820	0, 0.000	11, 0.180
DP9	40, 0.741	0, 0.000	14, 0.260
DP10	111, 0.902	0, 0.000	12, 0.098
DP11	47, 0.770	0, 0.000	14, 0.230
DP12	89, 0.761	0, 0.000	28, 0.240
DP13	89, 0.908	0, 0.000	9, 0.092
DP14	109, 0.865	0, 0.000	17, 0.135
DP15	74,0.712	0,0.000	30,0.288
DP16	89, 0.908	0, 0.000	9, 0.092
DP17	73, 0.859	0, 0.000	12, 0.141
DP18	70, 0.795	0, 0.000	18, 0.205

6 结论

提出了一种新型ICA以同时最小化MaOFJSP的最大完成时间、最大拖期、最大机器负荷和总能耗, 该算法采用新方法构建初始化帝国,使得大多数殖民国家具有数量相近的殖民地, 引入殖民国家的同化, 并应用新的革命策略和帝国竞争方法以获得高质量解. 最后通过大量实验测试ICA新策略对其性能的影响并将ICA与其他算法对比, 实验结果表明新型ICA在求解MaOFJSP方面具有较强的优势.

低碳调度问题是一个很重要的问题, 普遍存在于实际制造系统, 在后续研究中将继续关注这类问题, 并应用一些新的智能算法(比如教学优化算法)求解该问题; 另一方面, 将关注与能量相关的分布式调度问题, 并尝试设计有效的调度算法; 接下来还将研究其他的高维多目标生产调度问题, 比如高维多目标混合流水车间调度问题.

表 6 ICA,VNS和MOGA的运行时间

Table 6 Computational times of ICA, VNS and MOGA

实例	运行时间/s		
	ICA	VNS	MOGA
MK01	5.753	5.034	6.131
MK02	4.316	3.939	5.975
MK03	16.43	14.63	12.51
MK04	9.068	7.720	7.687
MK05	12.63	11.38	10.74
MK06	13.98	13.89	11.17
MK07	10.55	7.054	8.788
MK08	23.05	25.52	22.10
MK09	22.21	29.08	21.76
MK10	24.26	32.38	20.08
MK11	30.56	22.74	17.64
MK12	27.67	27.98	20.12
MK13	30.55	49.24	20.14
MK14	31.83	33.97	19.13
MK15	33.69	34.28	19.16
DP1	13.68	17.27	19.18
DP2	26.94	28.02	19.99
DP3	21.74	28.56	19.76
DP4	14.94	12.30	19.92
DP5	24.98	27.16	19.51
DP6	24.82	25.40	19.50
DP7	20.70	22.22	29.38
DP8	35.40	46.79	29.88
DP9	27.11	38.26	29.90
DP10	25.59	29.20	23.96
DP11	32.22	38.22	24.11
DP12	29.39	37.61	24.07
DP13	44.33	45.83	37.77
DP14	35.50	51.81	36.40
DP15	44.57	56.06	39.81
DP16	42.71	51.60	36.17
DP17	40.06	49.17	31.05
DP18	37.07	50.26	31.26

参考文献:

[1] YUAN Y, XU H. Multi-objective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science Engineering*, 2015, 12(1): 336 – 353.

[2] MOSLEHI G, MAHNAM M. A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 2011, 129(1): 14 – 22.

[3] BRUKER R, SCHLIE R. Job-shop scheduling with multi-purpose machines. *Computing*, 1990, 45(4): 369 – 375.

[4] KACEM I, HAMMADI S, BORNE P. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary

algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 2002, 60(3/5): 245 – 276.

[5] SHEN X N, HAN Y, FU J Z. Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. *Soft Computing*, 2017, 21(12): 6531 – 655.4

[6] ZHANG Jing, WANG Wanliang, XU Xinli, et al. Hybrid particle swarm optimization for multi-objective flexible job shop scheduling problem. *Control Theory & Applications*, 2012, 29(6): 715 – 722.
(张静, 王万良, 徐新黎, 等. 混合粒子群算法求解多目标柔性作业车间调度. 控制理论与应用, 2012, 29(6): 715 – 722.)

[7] LI J Q, PAN Q K, TASGETIREN M F. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance. *Applied Mathematical Modelling*, 2014, 38(3): 1111 – 1132.

[8] LI J Q, PAN Q K, SUGANTHAN P N, et al. A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling. *International Journal of Advanced Manufacturing Technology*, 2011, 52(5): 683 – 697.

[9] BAGHERI A, ZANDIEH M. Bi-criteria flexible job-shop scheduling with sequence-dependent setup times-variable neighborhood search approach. *Journal Manufacturing Systems*, 2011, 30(1): 8 – 15.

[10] LI J Q, PAN Q K, XIE S X. An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Applied Mathematics and Computation*, 2012, 218(18): 9353 – 9371.

[11] WANG L, WANG S Y, LIU M. A pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem. *International Journal of Production Research*, 2013, 51(12): 3574 – 3592.

[12] GAHM C, DENZ F, DIRR M, et al. Energy-efficient scheduling in manufacturing companies: a review and research framework. *European Journal of Operational Research*, 2016, 248(3): 744 – 757.

[13] JIANG Z Q, ZUO L, E M C. Study on multi-objective flexible job-shop scheduling problem considering energy consumption. *Journal of Industrial Engineering and Management*, 2014, 7(3): 589 – 604.

[14] HE Y, LI Y F, WU T, et al. An energy-responsive optimization method for machine tool selection and operation sequence in flexible machining job shops. *Journal of Cleaner Production*, 2015, 87(1): 245 – 254.

[15] YIN L J, LI X Y, GAO L, et al. A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. *Sustainable Computing: informatics and Systems*, 2017, 13: 15 – 30.

[16] LEI D M, ZHENG Y L, GUO X P. A shuffled frog leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. *International Journal of Production Research*, 2017, 55(11): 3126 – 3140.

[17] AI Ziyi, LEI Deming. A novel shuffled frog leaping algorithm for low carbon flexible job shop scheduling. *Control Theory & Applications*, 2017, 34(10): 1361 – 1368.
(艾子义, 雷德明. 基于新型蛙跳算法的低碳柔性作业车间调度. 控制理论与应用, 2017, 34(10): 1361 – 1368.)

[18] LEI Deming. Novel teaching-learning-based optimization algorithm for low carbon scheduling of flexible job shop. *Control and Decision*, 2017, 32(9): 1621 – 1627.
(雷德明. 基于新型教学优化算法的低碳柔性作业车间调度. 控制与决策, 2017, 32(9): 1621 – 1627.)

[19] LEI Deming, YANG Dongjing. Research on flexible job shop scheduling problem with total energy consumption constraint. *Acta Automatica Sinica*, 2018, 44(11): 2083 – 2091.
(雷德明, 杨冬婧. 具有总能耗约束的柔性作业车间调度问题研究. 自动化学报, 2018, 44(11): 2083 – 2091.)

[20] PIROOZFARD H, WONG K Y, WONG W P. Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resources Conservation and Recycling*, 2018, 128: 267 – 283.

[21] MOKHTARI H, HASANI A. An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers and Chemical Engineering*, 2017, 104: 339 – 352.

[22] ATASHPAZ E, LUCAS C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialist competition. *IEEE Congress on Evolutionary Computation*, 2007, 21(1): 4661 – 4667.

[23] HOSSEINI S, KHALED A A, VADLAMANI S. Hybrid imperialist competitive algorithm, variable neighborhood search, simulated annealing for dynamic facility layout problem. *Neural Computing and Application*, 2014, 25(7): 1871 – 1885.

[24] KARIMI S, ARDALAN Z, NADERI B, et al. Scheduling flexible job-shops with transportation times: mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*, 2017, 41: 667 – 682.

[25] WANG B, GUAU Z, LI D, et al. Two-sided assembly line balancing problems with operator number and task constraints: a hybrid imperialist competitive algorithm. *International Journal of Advanced Manufacturing Technology*, 2014, 74(5): 791 – 805.

[26] HOSSEINI S, KHALED A A. A survey on the imperialist competitive algorithm metaheuristic: implementation in engineering domain and directions for future research. *Applied Soft Computing*, 2014, 24(C): 1078 – 1094.

[27] ZHANG Chaoyong, DONG Xing, WANG Xiaojuan, et al. Improved NSGA-II for the multi-objective flexible job-shop scheduling problem. *Journal of Mechanical Engineering*, 2010, 46(11): 156 – 164.

[28] ZHANG C Y, GU P H, JIANG P G. Low-carbon scheduling and estimating for a flexible job shop based on carbon footprint and carbon efficiency of multi-job processing. *Proceedings of the Institution of Mechanical Engineers-Part B: Journal of Engineering Manufacture*, 2014, 229(2): 328 – 342.

[29] AFRUZI E N, NAJAFI A A, ROGHANIAN E, et al. A multi-objective imperialist competitive algorithm for solving discrete time, cost and quality trade-off problem with mode-identity and resource constrained situation. *Computers and Operations Research*, 2014, 50(1): 80 – 96.

[30] BAYAREH M, MOHAMMADI M. Multi-objective optimization of a triple shaft gas compressor station using imperialist competitive algorithm. *Applied Thermal Engineering*, 2016, 109: 384 – 400.

[31] BRANDIMARTE P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 1993, 41(1): 157 – 183.

[32] DAUZERE S, PAULLI J. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 1997, 70(2): 281 – 306.

[33] KNOWLES J D, CORNE D W. On metrics for comparing non-dominated sets. *Proceedings of 2002 Congress on Evolutionary Computation*. Honolulu: IEEE, 2002: 711 – 716.

[34] LEI D M. Pareto archive particle swarm optimization for multi-objective fuzzy job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 2008, 37(1/2): 157 – 165.

作者简介:

李　明　博士研究生, 主要从事制造系统智能优化与调度的研究, E-mail: limingwhut@163.com;

雷德明　博士, 教授, 主要从事智能系统优化与控制的研究, E-mail: deminglei11@163.com.