# 具有总能耗约束的柔性作业车间调度问题研究

雷德明 [1, 2] 杨冬婧 [1]

**摘 要** 针对具有总能耗约束的柔性作业车间调度问题 (Flexible job shop scheduling problem, FJSP), 提出一种基于帝国竞争算法 (Imperialist competitive algorithm, ICA) 和变邻域搜索 (Variable neighborhood search, VNS) 的双阶段算法, 该算法在总能耗不超过给定阈值的条件下最小化 Makespan 和总延迟时间. 由于能耗约束不是总能满足且阈值往往难以事先给定, 为此, 第一阶段, 首先, 将原问题转化为具有 Makespan、总延迟时间和总能耗的三目标 FJSP, 然后, 利用初始帝国构建和帝国竞争的新策略设计一种 ICA 对问题求解, 并根据 ICA 的结果确定总能耗阈值; 第二阶段, 应用解的比较新策略、非劣解集更新方法和当前解周期性更新, 构建 VNS 对原问题求解. 计算实验和结果分析表明, 两阶段算法对于所研究的问题搜索能力强.

**关键词** 双阶段算法, 总能耗约束, 柔性作业车间, 调度问题, 帝国竞争算法, 变邻域搜索

**引用格式** 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

**DOI** 10.16383/j.aas.2018.c170345

## Research on Flexible Job Shop Scheduling Problem With Total Energy Consumption Constraint

LEI De-Ming[1, 2] YANG Dong-Jing[1]

**Abstract** A two-phase algorithm based on imperialist competitive algorithm (ICA) and variable neighborhood search (VNS) is proposed for the flexible job shop scheduling problem (FJSP) with total energy consumption constraint. The goal is to minimize the makespan and total tardiness under the condition that total energy consumption does not exceed a given threshold. Energy consumption constraint is not always met and a threshold is difficult to be decided in advance, so in the first phase, the original problem is converted into FJSP with makespan, total tardiness and total energy consumption, and an ICA is used to solve the converted problem based on some new strategies for initial empires and imperialist competition. An energy consumption threshold is obtained in terms of the results of ICA. In the second phase, a VNS is presented for the original FJSP by using new methods for comparing solutions and updating the non-dominated set and renewing current solution periodically. Computational experiments are conducted and the result shows that the two-phase algorithm has strong search ability for the considered FJSP.

**Key words** Two-phase algorithm, total energy consumption constraint, flexible job shop, scheduling problem, imperialist competitive algorithm (ICA), variable neighborhood search (VNS)

**Citation** Lei De-Ming, Yang Dong-Jing. Research on flexible job shop scheduling problem with total energy consumption constraint. *Acta Automatica Sinica*, 2018, **44**(11): 2083−2091

柔性作业车间调度问题 (Flexible job shop scheduling problem, FJSP) 广泛存在于汽车装配、纺织、化工生产和半导体制造等制造过程和行业中, 自 Bruker 等在 1990 年的开创性工作[1] 之后, FJSP 的研究受到广泛关注, 取得了很大进展[1−27].

由于 FJSP 的高度复杂性, 精确算法难以在合理的时间内对其求解, 使得智能算法成为解决 FJSP 的主要手段, 智能算法已成功应用于各类 FJSP 的求解, 其中关于多目标 FJSP, 早期的工作为 Kacem 等[2] 提出的基于遗传算法和局部方法的混合算法, 在此之后, 研究者大量应用 GA (Genetic algorithm)[2−9]、粒子群算法[10−11]、和声搜索算法[12]、人工蜂群算法[13]、禁忌搜索[14−15]、变邻域搜索[16]、蛙跳算法[17] 和分布估计算法[18] 等求解 FJSP.

近些年, 低碳制造和低碳生产调度受到工业界和学术界的广泛关注, 其中关于低碳 FJSP, Tang

收稿日期 2017-06-22 录用日期 2017-09-07
Manuscript received June 22, 2017; accepted September 7, 2017
国家自然科学基金 (61573264, 71471151), 数字制造装备与技术国家重点实验室开放课题 (DMETKF2017015) 资助
Supported by National Natural Science Foundation of China (61573264, 71471151), Open project of State Key Laboratory of Digital Manufacturing Equipment and Technology (DMETKF2017015)
本文责任编委 鲁仁全
Recommended by Associate Editor LU Ren-Quan
1. 武汉理工大学自动化学院 武汉 430070 2. 数字制造装备与技术国家重点实验室 武汉 430074
1. School of Automation, Wuhan University of Technology, Wuhan 430070 2. State Key Laboratory of Digital Manufacturing Equipment and Technology, Wuhan 430074

等[19] 考虑了 FJSP 的能耗并运用遗传模拟退火算法求解. Liu 等[20] 运用 GA 求解双目标低碳 FJSP. He 等[21] 提出一种节能优化算法, 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. Zhang 等[22] 设计结合改进局部搜索的多目标 GA 以解决低碳 FJSP. 蒋增强等[23] 提出了基于血缘变异的改进非劣排序遗传算法 (Non-dominated sorting genetic algorithm-II, NSGA-II[28]). 唐立力[24] 给出一种改进型候鸟优化算法. Yin 等[25] 运用多目标 GA 解决了具有产能、能源效率和噪声减少等目标的 FJSP. 文献 [26−27] 分别利用一种新型蛙跳算法和一种新的教学优化算法求解低碳 FJSP.

尽管低碳 FJSP 的研究取得了较大进展, 但其研究仍不够深入, 有些问题的研究还未引起研究者的重视, 例如, 具有总能耗约束的 FJSP, 在该问题中, 总能耗不是目标函数, 无需最优化, 总能耗只需不超过给定的阈值即可, 能耗约束的加入, 产生了如下几个新特征: 1) 总能耗约束不是总能得到满足, 即智能算法的解所具有的总能耗经常超过给定的阈值. 2) 总能耗与机器分配和工件加工顺序密切相关, 其阈值难以事先确定. 3) 由于以上两个特点, 需要利用新原理和策略比较问题的解并更新非劣解集.

如上所述, 多种智能算法在 FJSP 和低碳 FJSP 方面都具有成功的应用, 不过, 作为一种模拟社会政治行为的帝国竞争算法 (Imperialist competitive algorithm, ICA)[29], 尽管已在设备布局[30−31]、调度[32−37]、装配线平衡[38−40]、旅行商问题[41] 等方面具有一些成果, 但 ICA 较少用于 FJSP 和低碳 FJSP 的求解[36−37]. 根据文献 [29], 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, 这些特点表明 ICA 在求解 FJSP 和低碳 FJSP 方面具有不同其他算法如 GA 的优势, 如不必像 GA 那样要加强局部搜索才能实现全局搜索与局部搜索的协调, 因此, 有必要研究 ICA 在 FJSP 和低碳 FJSP 方面的应用.

本文研究具有总能耗约束的 FJSP, 由于该问题的能耗约束经常无法得到满足, 为了有效地解决问题, 首先将问题转化所有约束都可满足的三目标 FJSP, 以简化对能耗约束的处理, 同时扩大搜索区域, 然后直接优化原问题, 以进一步改进第一阶段的搜索结果, 为此, 设计基于 ICA 和 VNS (Variable neighborhood search) 的双阶段算法, 先应用 ICA 产生一定数量的可行解, 然后运用 VNS 对可行解改进, 其中, ICA 不直接求解原问题, 而是优化具有总能耗、总延迟时间和 Makespan 的三目标 FJSP; 而 VNS 则从第一阶段获得的解出发, 对原问题求解, 并通过周期性更新当前解增强 VNS 的探索能力和它本身较强的局部搜索能力, 不断提高解的质量. 通过大量仿真实验, 验证所得阈值的合理性和算法的搜索性能.

## 1 问题描述

具有能耗约束的 FJSP 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 $M_k$ 具有两种模式: 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, 工序加工不能中断准备时间和清理时间包含在加工时间内等.

另外, 还考虑总能耗约束 $TEC\le Q_{EC}$, 式中,

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

其中, $TEC$ 表示总能耗, $y_{ijk}(t)$ 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, $z_k(t)=0$. $Q_{EC}$ 为总能耗阈值. $C_{\max}$ 表示最大完成时间.

问题包括调度子问题和机器分配子问题. 问题的目的在于在所有约束满足的条件下同时最小化如下两个目标函数.

$$f_1=C_{\max} \tag{2}$$

$$f_2=\sum_{i=1}^{n}\max\{C_i-D_i,0\} \tag{3}$$

其中, $C_i, D_i$ 表示工件 $J_i$ 的完成时间和交货期, 目标函数 $f_1, f_2$ 分别为 Makespan 和总延迟时间.

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, 其最优解不再只是单独一个解, 而是一组解的集合, 且需要通过比较所有解才能确定最优解集. 通常, 要用到如下几个概念: 对于解 $x, y$, 如果对于 $\forall i\in\{1,2,\cdots,G\}$, $f_i(x)\le f_i(y)$; 且 $\exists i\in\{1,2,\cdots,G\}$, $f_i(x)<f_i(y)$, 则解 $x$Pareto 支配 $y$. 对于集合 $\Phi$ 和解 $x\in\Phi$, 如果 $x$ 不受集合 $\Phi$ 的其他任何解支配, 则 $x$ 关于集合 $\Phi$ 是非劣的. 如

果一个解不受问题解空间内的任何其他解支配, 则该解为 Pareto 最优解.

## 2 求解具有总能耗约束的 FJSP 的双阶段算法

对于所研究的 FJSP, 需要确定合适的能耗阈值, 同时在所有约束条件满足的情况下, 最优化两个目标函数, 为了完成这两个任务, 提出一种基于 ICA 和 VNS 的双阶段算法.

### 2.1 编码

针对具有 $n$ 个工件和 $m$ 台机器的 FJSP, 通常利用调度串 $[(\theta_1, r_1),\ (\theta_2, r_2),\ \cdots,\ (\theta_i, r_i),\ \cdots,\ (\theta_h, r_h)]$ 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, $h = \sum_{i=1}^{n} h_i$ 表示工序总数.

在第一个串中, $\theta_i \in \{1, 2, \cdots, n\}$, $1 \le r_i \le h_{\theta_i}$, 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 $[o_{\theta_1 r_1}, o_{\theta_2 r_2}, \cdots, o_{\theta_i r_i}, \cdots, o_{\theta_h r_h}]$. 第二个串中, 基因 $q_{ij} \in S_{ij}$ 表示用于加工工序 $o_{ij}$ 的相容机器. 上述编码方法[27, 41] 能保证除总能耗约束 $TEC \le Q_{EC}$ 之外的其他约束总是成立, 但无法保证能耗约束总是成立. 对于新加入的约束 $TEC \le Q_{EC}$, 需要确定阈值 $Q_{EC}$ 并处理不可行解.

### 2.2 第一阶段: ICA

本阶段首先将能耗约束从原问题中去掉, 而增加第三个目标 $f_3 = TEC$, 这样问题变成具有 $f_1, f_2, f_3$ 的 FJSP, 通过问题的转化, 将原问题转化为所有约束都可以得到满足的问题; 然后, 设计一种新的 ICA 对三目标 FJSP 进行求解; 最后, 根据 ICA 的结果确定合适的阈值. 由于原问题的不可行解是转化后问题的可行解, 这样扩大了 ICA 的搜索区域, 有助于获得高质量的解.

ICA 的主要步骤包括构建初始帝国、殖民地同化、殖民地革命、交换和帝国竞争, 本文根据转化后的 FJSP 的特点, 采用一些新策略实现 ICA 的以上步骤, 构建新的 ICA.

由于只增加一个目标, 只在第一阶段优化三目标 FJSP, ICA 的非劣排序复杂性将有所增加, 但非劣排序不是每代都做, 故问题转化对算法复杂度的影响不大.

#### 2.2.1 初始帝国构建

由于所研究 FJSP 的多目标特性, 导致种群中的优秀个体经常彼此非劣, 当这些个体被选作殖民国家时, 应该分配相近数量的殖民地以同等对待它们. 基于该思想, 提出了一种初始帝国构造新方法. 假设初始帝国总数为 $N_{im}$, 种群规模为 $N$, 显然, 殖民地总数 $N_c = N - N_{im}$, 其具体过程如下:

**步骤 1.** 对初始种群 $P$ 中的解按文献 [28] 的方法进行非劣排序, 得到每个解的 rank 值.

**步骤 2.** 确定 rank 值为 1 的 $\eta$ 个解.

**步骤 3.** 如果 $\eta < N_{im}$, 则种群 $P$ 中 $\eta$ 个非劣解分别为帝国 $k = 1, 2, \cdots, \eta$ 的殖民国家, 剩余帝国的殖民国家为种群中 rank 值最小的受支配解; 否则, 随机选择 $N_{im}$ 个非劣解, 使它们成为帝国的殖民国家.

**步骤 4.** 对于每个帝国 $k = 1, 2, \cdots, N_{im}$, 如果 $k < N_{im}$, 则 $F_k = A + u$ 或 $B + u$, $u \in \{0, 1\}$; 否则 $F_{N_{im}} = N - \sum_{k=1}^{N_{im}-1} F_k$; 确定 $F_k$ 之后, 为帝国随机分配 $F_k - 1$ 个国家作为殖民地. 其中, $F_k$ 表示帝国内的国家总数, $A, B$ 如式 (4) ∼ (5) 所示, 假设 $N_{im} = 6$.

$$A = \begin{cases} \text{round}\,(0.3N), & \eta = 1 \\ \text{round}\,(0.2N), & \eta = 2 \\ \text{round}\,(0.2N), & \eta = 3 \\ \text{round}\,(0.18N), & \eta = 4 \\ \text{round}\,(0.18N), & \eta = 5 \\ \text{round}\,(N/6), & \eta \ge 6 \end{cases} \tag{4}$$

$$B = \begin{cases} \text{round}\left(\dfrac{N - \eta A}{N_{im} - \eta}\right), & \text{若 } \eta < N_{im} \\ 0, & \text{其他} \end{cases} \tag{5}$$

其中, round $(z)$ 为最接近 $z$ 的整数.

当 $\eta < N_{im}$ 时, 存在两类初始帝国, 其中第一类 $\eta$ 个帝国的殖民国家都是种群 $P$ 中的非劣解, 帝国包含 $F_k = A + u$ 个国家; 第二类 $N_{im} - \eta$ 个帝国的殖民国家为具有最小 rank 值的受支配解, 这样的受支配解共有 $N_{im} - \eta$ 个, 相应的 $F_k = B + u$. 若 $\eta \ge N_{im}$, 则随机选择 $N_{im}$ 个非劣解作为帝国的殖民国家. 显然, 以上初始帝国建立过程不同于文献 [42−43] 中的相应过程.

#### 2.2.2 同化与革命

同化和革命是 ICA 的重要步骤, 本文提出了革命的实现新途径. 对于调度问题, 同化往往通过利用殖民地和殖民国家之间的交叉[32−37] 来实现.

本文的同化过程如下: 对于帝国内的殖民地 $\lambda$ 和殖民国家 $v$, 随机产生一个随机数 $s$, 如果 $s < \zeta$, 则对两个解的调度串应用第一种交叉; 否则, 对两个解的机器分配串应用第二种交叉, 产生新解 $z$, 并与殖民地 $\lambda$ 进行比较, 若新解支配殖民地 $\lambda$ 或者与殖民地 $\lambda$ 彼此非劣, 则利用新解替代殖民地 $\lambda$, 并

更新非劣解集 $\Omega$, 其中两种交叉的详细过程见文献 [26−27].

通常, 调度子问题比机器分配子问题更复杂, 求解难度更大, 为此, 令 $\zeta > 0.5$ 以分配更多的计算资源给调度子问题, 通过实验确定, $\zeta = 0.6$.

非劣解集 $\Omega$ 的更新过程如下: 将新解加入到集合 $\Omega$ 之后, 对集合内的所有解进行 Pareto 比较, 保留非劣解, 剔除受支配解. 以上更新过程是根据三个目标 $f_1, f_2, f_3$ 对解进行比较, 而不是原问题中的 $f_1, f_2$.

殖民地革命模拟某种非预期的改变, 例如, 改变殖民地的语言或宗教, 从而改变其特征和地位[29]. 本文采用新的策略实现殖民地革命, 其具体过程描述如下:

对于帝国 $k$

**步骤 1.** 令 $\alpha \leftarrow 1$, 对于每个 $\tau = 1, 2, \cdots, F_k$, 产生随机数 $rand$, 若 $rand < U_R$, 则 $\alpha \leftarrow \alpha + 1$.

**步骤 2.** 根据 Pareto 支配对帝国内的所有殖民地进行比较, 确定每个殖民地 $\lambda$ 的 $w_\lambda$.

**步骤 3.** 若 $\alpha > 1$, 则对于 $l = 1, 2, \cdots, \alpha$, 执行如下步骤:

确定具有最小 $w_\lambda$ 的殖民地 $\lambda$, 令 $g \leftarrow 1$

重复执行如下步骤 $R$ 次:

若 $g = 1$, 则对殖民地 $\lambda$ 执行 $insert$; 否则, 对殖民地 $\lambda$ 执行 $change$, 产生新解 $z$;

若新解支配殖民地 $\lambda$ 或者与殖民地 $\lambda$ 彼此非劣, 则利用新解替代殖民地 $\lambda$ 并更新集合 $\Omega$; 否则 $g \leftarrow g + 1$, 若 $g = 3$ 则 $g \leftarrow 1$.

其中, $R$ 为整数, $U_R$ 表示革命概率.

关于革命概率 $U_R$, 文献 [42−43] 分别将其设置为 0.35 和 0.3. 与文献 [42−43] 不同, 只有部分相对优秀的殖民地参与革命过程, 基于这一特征, 本文选择一个较小的 $U_R$, 根据实验确定 $U_R$ 为 0.1.

对于帝国内殖民地, 如果殖民地 $\lambda$ 受同一帝国的其他 $w_\lambda$ 个殖民地支配, 则该殖民地赋给值 $w_\lambda$. 只有 $w_\lambda$ 值最小的部分殖民地参与革命, 这是因为利用较优秀的殖民地更容易获得更好的殖民地, 甚至产生新的殖民国家.

由于 FJSP 具有两个子问题, 为此, 运用多种邻域结构 $insert$, $swap$ 和 $change$ 产生新解, 其中, $insert$ 将随机选择的调度串元素 $(\theta_i, r_i)$ 插入到新位置 $k \neq i$ 并随后根据 $\theta_i$ 在调度串中出现的次序为 $r_i$ 重新赋值[26]. $swap$ 则在互换调度串的两个不同元素之后为 $r_i$ 重新赋值. $change$ 的实现过程如下: 首先构建集合 $\Theta = \{q_{ij} || S_{ij}| > 1, i = 1, 2, \cdots, n, j = 1, 2, \cdots, h_i\}$, 然后从集合中随机选择一些元素, 为它们重新赋值, 例如, $q_{ij}$ 被选中, 则从 $S_{ij}$ 中随机确定一台机器替代当前的 $q_{ij}$.

#### 2.2.3 帝国竞争

帝国竞争是 ICA 的重要步骤. 对于本阶段考虑的三目标 FJSP, 给出了一种新方式来计算一个解的成本 (cost), 成本值主要根据解的 rank 值和拥挤距离来确定, 为此, 首先通过非劣排序[28] 将种群分解成 $rank_{\max}$ 个 $H_l$, 其中, $H_l$ 内元素的 rank 值均为 $l$; 然后, 对每个集合 $H_l$ 的非边界解, 按文献 [28] 的方法计算拥挤距离, 得到这些距离的最大值 $\psi$, 边界解的拥挤距离为 $\psi + \alpha$, 其中, $\alpha \geq 1$, 利用 $\alpha$ 是为了区分边界解与其他解.

帝国竞争的新过程描述如下: 如果 $N_{im} \geq 2$

**步骤 1.** 确定种群 $P$ 内所有解的 $rank$ 值和拥挤距离.

**步骤 2.** 对于集合 $H_l$, $l = 1, 2, \cdots, rank_{\max}$, 将该集合内解的成本定义为 $l + dist_i \Big/ \sum_{f \in H_l} dist_f$, 其中, $dist_f$ 表示个体 $f \in H_l$ 的拥挤距离.

**步骤 3.** 按照基本 ICA[29] 的过程实现帝国竞争的剩余步骤.

计算帝国总成本时的参数 $\zeta$ 仍取 0.1.

#### 2.2.4 ICA 描述

ICA 的具体过程如下:

**步骤 1.** 随机产生初始种群并确定初始集合 $\Omega$.

**步骤 2.** 殖民地同化与革命.

**步骤 3.** 殖民地与殖民国家之间的交换.

**步骤 4.** 帝国竞争.

**步骤 5.** 如果第一阶段的终止条件成立, 则停止搜索; 否则转到步骤 2.

其中交换过程如下: 对于帝国内的每个殖民地, 确定殖民地能否支配殖民国家或者与殖民国家彼此非劣, 是, 则利用殖民地替代当前殖民国家, 成为帝国新的殖民国家, 而原殖民国家则变成殖民地.

第一阶段终止条件为目标函数估计次数, 由于步骤 2 每次循环产生的解个数不一样, 在步骤 2 执行过程中, 要判断终止条件是否成立, 成立则停止搜索.

总之, ICA 根据所研究 FJSP 具有多目标、双子问题和调度子问题更复杂等特点, 构建初始帝国、让优秀殖民地参加革命、同化过程偏重于调度子问题的优化以及结合 rank 值和拥挤距离进行帝国竞争, 这些策略与问题特点相适应, 有助于提高 ICA 的求解质量.

#### 2.2.5 总能耗阈值

文献 [44] 考虑了能耗阈值 $Q_{EC}$, 但并未讨论如何确定阈值大小. 本文根据第一阶段 ICA 的优化结果来确定阈值, 这样可以减少决策者的认知和决策

负担.

关于每个实例, 获得 $Q_{EC}$ 的具体过程如下:

**步骤 1.** ICA 随机运行 20 次;

**步骤 2.** 计算 $Q_i = \max\{TEC(z)\,|z \in \Omega_i\}$, $\bar{Q} = \frac{1}{20}\sum_{i=1}^{20} Q_i$;

**步骤 3.** 从所有满足 $Q_i \geq \bar{Q}$ 的 $Q_i$ 中剔除一些相近的 $Q_i$;

**步骤 4.** 运行两阶段算法并根据计算结果确定哪个 $Q_i \geq \bar{Q}$ 可以作为 $Q_{EC}$.

其中, $\Omega_i$ 表示 ICA 第 $i$ 次运行所获得的非劣解集.

### 2.3 第二阶段: VNS

本阶段利用 VNS 求解具有 $TEC \leq Q_{EC}$ 的 FJS-P, 由于总能耗约束不总是得到满足, 需要处理不可行解, 为此, 给出一种新原则来比较两个解 $x, z$:

1) $TEC(x) > Q_{EC}$ 和 $TEC(z) \leq Q_{EC}$;

2) $x, z$ 都是可行解, 且 $z$ 不受 $x$ 支配;

3) $x, z$ 都是不可行解, 且 $TEC(z) \leq TEC(x)$.

以上三个条件只要有一个满足, 则解 $x$ 被 $z$ 替代, 其中条件 (2) 根据 $f_1, f_2$ 对两个解进行 Pareto 比较.

关于集合 $\Omega$, 第一阶段它由通过比较 $f_1, f_2, f_3$ 而得到的非劣解组成, 在从第一阶段过渡到第二阶段, 该集合被直接保留下来, 这样, 导致 $\Omega$ 中存在一些不可行解.

第二阶段集合 $\Omega$ 的更新过程如下: 对于解 $x$

1) 若 $TEC(x) > Q_{EC}$

如果集合 $\Omega$ 中至少存在一个不可行解 $y$, 且满足 $TEC(x) < TEC(y)$, 则解 $y$ 被 $x$ 所替代.

2) 若 $TEC(x) \leq Q_{EC}$

如果集合 $\Omega$ 所有解都可行, 则存在两种情况.

a) 解 $x$ 与 $\Omega$ 中的一个成员具有相同的 $f_1, f_2$, 但 $x$ 的总能耗更小, 则利用 $x$ 替代该成员.

b) 情况 1 的条件不成立, 则直接将 $x$ 添加到集合 $\Omega$ 中, 对所有解根据 $f_1, f_2$ 进行 Pareto 比较, 剔除所有受支配解.

如果集合 $\Omega$ 中存在不可行解, 则直接用 $x$ 替代其中一个不可行解.

VNS 的详细步骤如下:

**步骤 1.** 将集合 $\Omega$ 中的第一个解当作当前解, 令 $w \leftarrow max\_it_1$.

**步骤 2.** 令 $g \leftarrow 1$.

**步骤 3.** 重复如下步骤直到 $g = 3$ 或 $w > max\_it$ 随机产生解 $z \in \mathcal{N}_g(x)$.

根据上述原则, 如果 $x$ 能被 $z$ 所替代, 则直接替代, 并利用新的 $x$ 更新集合 $\Omega$ 且 $g \leftarrow 1$; 否则, $g \leftarrow g + 1$.

如果 $w$ 能被整数 $\beta$ 整除, 则选择 $y \in \Omega$ 并替代 $x$.

**步骤 4.** 如果 $w \leq max\_it$, 则转到步骤 2; 否则, 停止搜索, 剔除 $\Omega$ 中的非可行解.

其中, $max\_it_1$ 为第一阶段的终止条件, $max\_it$ 为整个两阶段算法的终止条件, $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$ 分别表示邻域结构 $swap$, $insert$, $change$, $\mathcal{N}_g(x)$ 表示运用 $\mathcal{N}_g$ 所产生的 $x$ 的邻域解集.

集合 $\Omega$ 更新过程中, 每次最多只剔除一个非可行解, 是为了使 $\Omega$ 中保留足够多的解, 以便在重新选择当前解时有更多的可能性.

### 2.4 算法描述: 两阶段算法

如上所述, 运用两阶段算法可以简化对能耗约束的处理, 扩大 ICA 的搜索区域, 从而提高搜索质量.

另外, 它还具有如下特点: 它不仅能求解所考虑的 FJSP, 而且能获得合适的总能耗阈值; 它将全局搜索算法和局部搜索算法有机地结合在一起.

## 3 计算实验

为了测试两阶段算法在求解具有总能耗约束的 FJSP 方面的性能, 利用 37 个实例 MK1-15[45]、DP1-18[46]、Ka4×5、Ka10×7、Ka10×10、Ka15×10[2] 进行计算实验, 这些实验由 Microsoft Visual C++ 6.0 编程实现, 并运行于 4.0 GB RAM 1.70 GHz CPU PC.

为了用于所研究的 FJSP, 在 37 个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:

$$D_i = \delta \sum_{j=1}^{h_i} \max_{k=1,2,\cdots,m} \{p_{ijk}\} \tag{6}$$

其中, $\delta$ 的值如表 1 所示.

表 1　$\delta$ 的设置

Table 1　The setting on $\delta$

| $\delta$ | Instances | $\delta$ | Instances |
|---|---|---|---|
| $[0.5, 0.7]$ | MK01, MK03-05 | $[1.0, 1.5]$ | DP1-12 |
| $[0.3, 0.5]$ | MK02, 06, 08-10 | $[1.2, 1.6]$ | MK11-15 |
| $[0, 1, 0.3]$ | Ka4×5, 10×7, 15×10 | $[0.7, 0.9]$ | MK07 |
| $[1.2, 1.7]$ | DP13-18 | $[0.05, 0.2]$ | Ka10×10 |

采用指标 $DI_R^{[47]}$ 评价算法的收敛性能, 它为非劣解集 $\Omega_l$ 与参考集 $\Omega^*$ 之间的距离, 如式 (7) 所示.

$$DI_R(\Omega_l) = \frac{1}{|\Omega^*|} \sum\nolimits_{y \in \Omega^*} \min\{\sigma_{xy}\,|x \in \Omega_l\} \tag{7}$$

其中, $\sigma_{xy}$ 表示解 $x$ 与参考解 $y$ 在归一化目标空间中的距离,

$$\sigma_{xy}=\sqrt{\left(f_1^*(x)-f_1^*(y)\right)^2+\cdots+\left(f_G^*(x)-f_G^*(y)\right)^2}$$

$f_i^*$ 为第 $i$ 个归一化目标, 参考集 $\Omega^*$ 由并集 $\bigcup_l \Omega_l$ 中的非劣解组成.

指标 $\rho_l$[48] 等于集合 $\{x\in\Omega_l\,|x\in\Omega^*\}$ 的大小与 $|\Omega^*|$ 的比值. 比值越大, 说明 $\Omega_l$ 为参考集 $\Omega^*$ 提供的非劣解越多.

选用 NSGA-II[8] 和 VNS[16] 作为对比算法, 这两种算法对 FJSP 具有较多的应用[5, 8, 16, 23], 且很容易应用于所研究的 FJSP 的求解. NSGA-II 作为求解多目标优化问题的著名算法, 其主要步骤包括非劣排序和拥挤距离的计算等. 文献 [8] 针对具有随机故障的 FJSP, 应用 NSGA-II 求解, 为了应用于具有总能耗约束的 FJSP, 对该算法做如下修改: 去掉与随机故障相关的部分, 对可行解采用算法原来的非劣排序和拥挤距离计算, 所有不可行解赋予同样的且大于可行解最大 rank 值的 rank 值, 拥挤距离设置为 $\omega_{\max}-TEC(x)$, 其中, $\omega_{\max}$ 为足够大的正数.

文献 [16] 所设计的 VNS 用来解决具有加权目标的 FJSP, 将第 2.3 节中新旧解更替原则和非劣解集更新策略引入后, 该 VNS 可直接求解本文所研究的 FJSP, 但该 VNS 与两阶段算法中的 VNS 的邻域结构和算法结构不一样.

两阶段算法的参数设置如下: $N=80$, $N_{im}=6$, $max\_it_1=20\,000$, $max\_it$ 关于 Ka4×5 为 $2.5\times10^4$, 关于其他实例为 $10^5$, $\beta$ 关于 Ka4×5 为 4 000, 关于其他实例为 5 000, $R=8$.

NSGA-II 的参数如下: 种群规模为 100, 交叉概率为 0.8, 变异概率为 0.1, 最大代数为 $max\_it/100$. 这些参数值根据计算实验而得, 是使算法获得最佳结果的一组值.

关于 VNS, $n_{\max}=350$ 由文献 [16] 给出, $max\_it$ 与两阶段算法相同.

关于总能耗阈值 $Q_{EC}$, 如果阈值过大, 大多数解都满足总能耗约束, 这样的阈值没有意义; 如果阈值过小, 则算法很难得到可行解, 导致整个算法的优化只是处理能耗约束, 目标函数难以得到充分的优化, 故需要设置一个合适的阈值.

第 2.2 节给出了确定阈值的详细过程, 以 Ka10 × 7 为例, 随机运行 ICA20 次, 得到 20 个 $Q_i$, 如表 2 所示, $\bar{Q}$ 等于 229.7; 然后, 对于大于 $\bar{Q}$ 的 $Q_i$, 剔除 230.2, 232.4, 244.4, 254.1 等相近的值, 得到剩余的 $Q_i$; 最后, 运行整个两阶段算法, 分别测试剩余的 $Q_i$, 根据上述两种指标评价计算结果, 最终确定 245.9 作为阈值.

表 2 阈值的确定

Table 2 Decision on threshold

| 所有 $Q_i$ | 大于 $\bar{Q}$ 的 $Q_i$ | 剩余的 $Q_i$ |
|---|---|---|
| 216.7, 198.9, 227.3, 218.0 | 230.2, 231.6 | 231.6, 245.9 |
| 230.2, 224.6, 231.6, 244.4 | 244.4, 237.7 | 237.7 |
| 218.9, 208.2, 237.7, 221.4 | 253.4, 245.9 | 253.4 |
| 223.1, 253.4, 245.9, 232.4 | 232.4, 254.1 | 232.4 |
| 207.6, 254.1, 284.1, 216.4 | 284.1 | 284.1 |

由于阈值是 ICA 进化的结果, 同时又大于 $\bar{Q}$, 故阈值大小合适. 表 3 给出了关于各个实例的阈值以及三种算法最终所获得的非劣解所对应的实际能耗, 其中 ATEC 和 MTEC 表示非劣解集中所有非劣解的最小能耗值和平均能耗值.

如表 3 所示, 三种算法关于所有实例所得到的 ATEC 和 MTEC 都小于 $Q_{EC}$, 而且关于大多数实例这两个指标都与 $Q_{EC}$ 接近, 这说明第二阶段总能耗的下降速度显著降低了, 从而说明第二阶段的新旧解更替原则和集合的更新策略是合理的, 第二阶段主要用于目标函数 $f_1, f_2$ 的最小化处理.

表 4 和 5 描述了三种算法的计算结果和计算时间, 可以看出, 两阶段算法关于几乎所有实例所产生的结果明显由于另外两种算法. 两阶段算法关于 19 个实例提供了参考集的所有元素, 而 NSGA-II 和 VNS 关于这 19 个实例所获得的解都受两阶段算法的非劣解支配, NSGA-II 只是关于 Ka10×7 产生了参考集的全部成员, MK06 的参考集全部都由 VNS 提供, 另外, NSGA-II 和 VNS 搜索所得的解总是距参考集较远, 总之, 两阶段算法利用和另外两种算法相似的计算时间取得了明显占优的计算结果.

表 3 总能耗阈值和三种算法的 ATEC 和 MTEC

Table 3 Total energy consumption threshold and ATEC and MTEC of three algorithms

| Instance | $Q_{EC}$ | 两阶段算法 | | NSGA-II | | VNS | |
|---|---|---|---|---|---|---|---|
| | | ATEC | MTEC | ATEC | MTEC | ATEC | MTEC |
| Ka4×5 | 152.3 | 96.154 | 95.259 | 96.080 | 95.259 | 95.259 | 95.259 |
| Ka10×7 | 245.9 | 196.27 | 185.41 | 194.39 | 193.26 | 200.52 | 193.50 |
| Ka10×10 | 159.5 | 124.81 | 121.26 | 130.29 | 128.24 | 132.17 | 124.26 |
| Ka15×10 | 443.3 | 313.85 | 302.87 | 325.01 | 321.45 | 365.49 | 354.56 |
| MK01 | 682.4 | 653.38 | 650.79 | 666.54 | 664.84 | 655.55 | 654.41 |
| MK02 | 550.7 | 509.51 | 496.17 | 525.17 | 519.14 | 508.72 | 506.36 |
| MK03 | 3 597 | 3 148.6 | 3 128.2 | 3 311.5 | 3 291.1 | 3 278.4 | 3 235.5 |
| MK04 | 1 191 | 1 083.4 | 1 064.4 | 1 097.5 | 1 078.4 | 1 086.4 | 1 074.9 |
| MK05 | 2 748 | 2 628.1 | 2 619.8 | 2 588.0 | 2 578.7 | 2 648.4 | 2 631.6 |
| MK06 | 2 349 | 2 162.8 | 2 145.9 | 2 189.4 | 2 168.5 | 2 127.0 | 2 106.3 |
| MK07 | 1 728 | 1 504.2 | 1 486.0 | 1 567.6 | 1 561.1 | 1 524.6 | 1 517.7 |
| MK08 | 10 156 | 9 786.4 | 9 713.1 | 10 004 | 9 976.4 | 9 747.8 | 9 747.8 |
| MK09 | 9 082 | 8 627.0 | 8 604.9 | 8 996.0 | 8 961.3 | 8 622.3 | 8 572.3 |

| Instance | $Q_{EC}$ | 两阶段算法 | | NSGA-II | | VNS | |
|---|---|---|---|---|---|---|---|
| | | ATEC | MTEC | ATEC | MTEC | ATEC | MTEC |
| MK10 | 9 728 | 8 537.1 | 8 536.2 | 8 991.6 | 8 955.6 | 9 022.7 | 8 972.6 |
| MK11 | 12 890 | 12 415 | 12 276 | 12 640 | 12 577 | 12 417 | 12 353 |
| MK12 | 14 933 | 14 291 | 14 220 | 14 314 | 14 148 | 14 214 | 14 109 |
| MK13 | 14 202 | 13 131 | 13 104 | 13 845 | 13 763 | 13 382 | 13 310 |
| MK14 | 16 985 | 14 916 | 14 856 | 15 307 | 15 239 | 15 020 | 14 981 |
| MK15 | 17 000 | 13 793 | 13 656 | 13 877 | 13 786 | 13 643 | 13 612 |
| DP1 | 34 534 | 33 829 | 33 745 | 34 117 | 34 026 | 33 902 | 33 758 |
| DP2 | 40 332 | 38 574 | 38 394 | 38 874 | 38 813 | 38 654 | 38 512 |
| DP3 | 36 428 | 35 271 | 35 086 | 35 701 | 35 504 | 35 367 | 35 215 |
| DP4 | 36 880 | 36 171 | 35 656 | 36 077 | 35 742 | 36 225 | 36 089 |
| DP5 | 40 085 | 38 815 | 38 766 | 39 428 | 39 252 | 38 882 | 38 619 |
| DP6 | 37 091 | 35 882 | 35 729 | 36 445 | 36 162 | 36 185 | 35 897 |
| DP7 | 26 373 | 60 667 | 60 542 | 61 033 | 60 989 | 60 707 | 60 673 |
| DP8 | 52 422 | 49 948 | 49 870 | 50 638 | 50 627 | 50 722 | 50 527 |
| DP9 | 64 539 | 62 395 | 62 232 | 63 458 | 63 202 | 63 282 | 62 966 |
| DP10 | 60 350 | 58 580 | 58 133 | 59 242 | 59 145 | 59 277 | 59 070 |
| DP11 | 57 613 | 55 548 | 55 373 | 56 332 | 56 260 | 56 118 | 55 612 |
| DP12 | 60 711 | 58 022 | 57 981 | 59 288 | 58 947 | 58 260 | 58 220 |
| DP13 | 86 142 | 84 382 | 84 335 | 85 046 | 85 027 | 85 200 | 85 062 |
| DP14 | 83 000 | 80 940 | 80 803 | 82 332 | 82 135 | 81 889 | 81 787 |
| DP15 | 73 000 | 70 977 | 70 631 | 72 033 | 71 854 | 71 800 | 71 702 |
| DP16 | 80 468 | 78 790 | 78 584 | 79 976 | 79 878 | 79 417 | 79 266 |
| DP17 | 86 677 | 84 886 | 84 798 | 85 963 | 85 675 | 85 394 | 85 324 |
| DP18 | 86 330 | 84 114 | 83 685 | 84 951 | 84 785 | 85 045 | 84 679 |

表 4 三种算法的计算结果

Table 4 Computational results of three algorithms

| Instance | $DI_R$ | | | $\rho_l$ | | |
|---|---|---|---|---|---|---|
| | 两阶段算法 | NSGA-II | VNS | 两阶段算法 | NSGA-II | VNS |
| Ka4×5 | 1.235 | 0.890 | 1.125 | 0.307 | 0.222 | 0.370 |
| Ka10×7 | 15.10 | 0.000 | 20.65 | 0.000 | 1.000 | 0.000 |
| Ka10×10 | 0.00 | 40.34 | 0.086 | 0.969 | 0.000 | 0.031 |
| Ka15×10 | 0.256 | 9.456 | 35.88 | 0.889 | 0.111 | 0.000 |
| MK01 | 0.446 | 6 121 | 7.305 | 0.694 | 0.000 | 0.306 |
| MK02 | 0.00 | 3 731 | 9.326 | **1.000** | 0.00 | 0.00 |
| MK03 | 0.000 | 51.90 | 30.71 | **1.000** | 0.00 | 0.00 |
| MK04 | 2 763 | 2 930 | 1 652 | 0.600 | 0.150 | 0.250 |
| MK05 | 3.119 | 59.04 | 7.024 | 0.667 | 0.000 | 0.333 |
| MK06 | 18.15 | 45.79 | 0.00 | 0.00 | 0.00 | 1.000 |
| MK07 | 0.000 | 31.49 | 9.899 | **1.000** | 0.00 | 0.00 |
| MK08 | 0.000 | 25.04 | 10.86 | **1.000** | 0.00 | 0.00 |
| MK09 | 0.000 | 49.76 | 10.23 | **1.000** | 0.00 | 0.00 |
| MK10 | 0.000 | 38.24 | 18.00 | **1.000** | 0.00 | 0.00 |
| MK11 | 13.42 | 3 265 | 1 325 | 0.588 | 0.000 | 0.412 |
| MK12 | 0.00 | 3 701 | 1 542 | **1.000** | 0.00 | 0.00 |
| MK13 | 0.000 | 64.40 | 32.30 | **1.000** | 0.00 | 0.00 |
| MK14 | 0.000 | 60.04 | 23.99 | **1.000** | 0.00 | 0.00 |
| MK15 | 0.297 | 39.99 | 19.86 | 0.944 | 0.000 | 0.056 |
| DP1 | 0.00 | 2 528 | 1 324 | **1.000** | 0.00 | 0.00 |
| DP2 | 7.507 | 39.16 | 4.663 | 0.550 | 0.000 | 0.45 |
| DP3 | 1.424 | 37.62 | 5.638 | 0.915 | 0.000 | 0.085 |
| DP4 | 1 427 | 2 911 | 7 302 | 0.750 | 0.000 | 0.250 |
| DP5 | 5.861 | 50.89 | 4.452 | 0.583 | 0.000 | 0.417 |
| DP6 | 0.000 | 52.16 | 12.64 | **1.000** | 0.00 | 0.00 |
| DP7 | 0.000 | 33.53 | 19.41 | **1.000** | 0.00 | 0.00 |
| DP8 | 0.652 | 42.3 | 20.47 | 0.988 | 0.000 | 0.012 |
| DP9 | 0.000 | 66.43 | 34.33 | **1.000** | 0.00 | 0.00 |
| DP10 | 0.543 | 4 836 | 11.20 | 0.950 | 0.000 | 0.050 |
| DP11 | 0.000 | 34.21 | 14.03 | **1.000** | 0.00 | 0.00 |
| DP12 | 2.597 | 67.65 | 4.814 | 0.667 | 0.000 | 0.333 |
| DP13 | 0.00 | 3 326 | 20.31 | **1.000** | 0.00 | 0.00 |
| DP14 | 0.258 | 46.08 | 1264 | 0.981 | 0.000 | 0.019 |
| DP15 | 0.000 | 46.20 | 19.04 | **1.000** | 0.00 | 0.00 |
| DP16 | 0.000 | 67.34 | 44.41 | **1.000** | 0.00 | 0.00 |
| DP17 | 0.000 | 34.69 | 16.09 | **1.000** | 0.00 | 0.00 |
| DP18 | 0.000 | 39.64 | 21.59 | **1.000** | 0.00 | 0.00 |

表 5 三种算法的计算时间

Table 5 Comparisons on the computational times of three algorithms

| Instance | Running time (s) | | | Instance | Running time (s) | | |
|---|---|---|---|---|---|---|---|
| | 两阶段算法 | NSGA-II | VNS | | 两阶段算法 | NSGA-II | VNS |
| Ka4×5 | 0.688 | 0.666 | 0.272 | DP1 | 10.74 | 10.85 | 13.03 |
| Ka10×7 | 1.901 | 3.341 | 1.795 | DP2 | 12.91 | 11.35 | 14.52 |
| Ka10×10 | 1.879 | 3.323 | 1.984 | DP3 | 11.59 | 10.96 | 14.67 |
| Ka15×10 | 3.339 | 4.337 | 3.417 | DP4 | 11.83 | 11.49 | 11.76 |
| MK01 | 2.791 | 4.068 | 3.136 | DP5 | 11.21 | 10.94 | 11.73 |
| MK02 | 2.799 | 4.037 | 3.358 | DP6 | 10.84 | 11.18 | 11.47 |
| MK03 | 7.351 | 7.595 | 8.498 | DP7 | 18.33 | 18.09 | 19.68 |
| MK04 | 4.110 | 5.141 | 4.895 | DP8 | 18.26 | 17.46 | 19.84 |
| MK05 | 6.740 | 5.335 | 6.600 | DP9 | 17.02 | 17.80 | 18.79 |
| MK06 | 7.654 | 6.037 | 7.419 | DP10 | 18.73 | 17.84 | 20.16 |
| MK07 | 5.238 | 6.072 | 5.325 | DP11 | 17.58 | 18.02 | 19.78 |
| MK08 | 14.18 | 13.98 | 15.31 | DP12 | 17.75 | 17.92 | 19.82 |
| MK09 | 12.54 | 14.84 | 14.87 | DP13 | 31.54 | 31.94 | 28.87 |
| MK10 | 12.51 | 13.16 | 14.82 | DP14 | 30.24 | 32.14 | 27.82 |
| MK11 | 12.29 | 12.27 | 12.39 | DP15 | 29.41 | 31.70 | 28.97 |
| MK12 | 12.84 | 13.44 | 14.93 | DP16 | 31.74 | 33.56 | 27.42 |
| MK13 | 13.20 | 13.62 | 15.08 | DP17 | 31.13 | 41.84 | 28.07 |
| MK14 | 16.63 | 16.77 | 18.95 | DP18 | 29.38 | 32.96 | 28.71 |
| MK15 | 16.14 | 16.60 | 18.65 | | | | |

两阶段算法的优良性能主要来自于算法的两阶段结构以及全局搜索和局部搜索的良好协调, 而 ICA 和 VNS 的一些改进策略进一步增强了算法性能, 因此, 两阶段算法是解决具有总能耗约束的 FJSP 具有较强竞争力的方法.

## 4 结论

尽管 FJSP 已得到较充分的研究, 但具有总能耗约束的多目标 FJSP 的研究相对较少, 随着绿色制造和低碳制造的应用不断深入, 需要进一步研究节能 FJSP. 本文提出一种基于 ICA 和 VNS 的两阶段算法在总能耗不超过给定的阈值条件下最小化 Makespan 和总延迟时间. 第一阶段将原问题转化为包括总能耗的三目标 FJSP, 然后利用新策略构建 ICA 以优化转化后的 FJSP, 并根据第一阶段的结果确定总能耗阈值; 第二阶段则运用一种高效的 VNS 对原问题求解. 计算结果表明, 两阶段算法具有较强的搜索性能和优势.

未来我们将继续深入研究具有总能耗或峰值能耗约束的调度问题, 并进一步研究帝国竞争算法在低碳生产调度方面的应用; 另外, 分布式调度也是我们未来研究的主题之一.

## References


1 Brucker R, Schlie R. Job-shop scheduling with multi-purpose machines. *Computing*, 1990, **45**(4): 369−375

2 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

3 Gao J, Gen M, Sun L Y, Zhao X H. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, 2007, **53**(1): 149−162

4 Chiang T C, Lin H J. A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. *International Journal of Production Economics*, 2013, **141**(1): 87−98

5 Yuan Y, Xu H. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transaction on Automation Science & Engineering*, 2015, **12**(1): 336−353

6 Rohaninejad M, Kheirkhah A, Fattahi P, Vahedi-Nouri B. A hybrid multi-objective genetic algorithm based on the ELECTRE method for a capacitated flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 2015, **77**(1−4): 51−66

7 Li J Y, Huang Y, Niu X W. A branch population genetic algorithm for dual-resource constrained job shop scheduling problem. *Computers & Industrial Engineering*, 2016, **102**: 113−131

8 Ahmadi E, Zandieh M, Farrokh M, Emami S M. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & Operations Research*, 2016, **73**: 56−66

9 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**(21): 6531−6554

10 Rohaninejad M, Sahraeian R, Nouri B V. Multi-objective optimization of integrated lot-sizing and scheduling problem in flexible job shops. *RAIRO-Operations Research*, 2015, **50**(3): 587−609

11 Singh M R, Singh M, Mahapatra S S, Jagadev N. Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 2016, **85**(9−12): 2353−2366

12 Gao K Z, Suganthan P N, Pan Q K, Chua T J, Cai T X, Chong C S. Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Information Sciences*, 2014, **289**: 76−90

13 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 activities. *Applied Mathematical Modelling*, 2014, **38**(3): 1111−1132

14 Li J Q, Pan Q K, Suganthan P N, Chua T J. A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 2011, **52**(5−8): 683−698

15 Jia S, Hu Z H. Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. *Computers & Operations Research*, 2014, **47**: 11−26

16 Bagheri A, Zandieh M. Bi-criteria flexible job-shop scheduling with sequence-dependent setup times-variable neighborhood search approach. *Journal of Manufacturing Systems*, 2011, **30**(1): 8−15

17 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

18 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

19 Tang D B, Dai M. Energy-efficient approach to minimizing the energy consumption in an extended job-shop scheduling problem. *Chinese Journal of Mechanical Engineering*, 2015, **28**(5): 1048−1055

20 Liu Y, Tiwari A. An investigation into minimising total energy consumption and total completion time in a flexible job shop for recycling carbon fiber reinforced polymer. *Procedia CIRP*, 2015, **29**: 722−727

21 He Y, Li Y F, Wu T, Sutherland J W. An energy-responsive optimization method for machine tool selection and operation sequence in flexible machining job shops. *Journal of Cleaner Production*, 2015, **87**: 245−254

22 Zhang R, Chiong R. Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production*, 2016, **112**: 3361−3375

23 Jiang Zeng-Qiang, Zuo Le. Multi-objective flexible job-shop scheduling based on low-carbon strategy. *Computer Integrated Manufacturing Systems*, 2015, **21**(4): 1023−1031
(蒋增强, 左乐. 低碳策略下的多目标柔性作业车间调度. 计算机集成制造系统, 2015, **21**(4): 1023−1031)

24 Tang Li-Li. Improved migrating birds optimization algorithm to solve low-carbon scheduling problem. *Computer Engineering and Applications*, 2016, **52**(17): 166−171
(唐立力. 求解低碳调度问题的改进型候鸟优化算法. 计算机工程与应用, 2016, **52**(17): 166−171)

25 Yin L J, Li X Y, Gao L, Lu C, Zhang Z. 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

26 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

27 Lei De-Ming. 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)

28 Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 2002, **6**(2): 182−197

29 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**: 1078−1094

30 Lian K L, Zhang C Y, Gao L, Shao X Y. Single row facility layout problem using an imperialist competitive algorithm. In: Proceedings of the 41st International Conference on Computers & Industrial Engineering. Los Angeles, USA: Elsevier, 2011. 578−586

31 Hosseini S, Khaled A A, Vadlamani S. Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem. *Neural Computing and Applications*, 2014, **25**(7−8): 1871−1885

32 Karimi N, Zandieh M, Najafi A A. Group scheduling in flexible flow shops: a hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism. *International Journal of Production Research*, 2011, **49**(16): 4965−4977

33 Behnamian J, Zandieh M. A discrete colonial competitive algorithm for hybrid flow shop scheduling to minimize earliness and quadratic tardiness penalties. *Expert Systems with Applications*, 2011, **38**(12): 14490−14498

34 Goldansaz S M, Jolai F, Anaraki A H Z. A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop. *Applied Mathematical Modelling*, 2013, **37**(23): 9603−9616

35 Naderi B, Yazdani M. A model and imperialist competitive algorithm for hybrid flow shops with sublots and setup times. *Journal of Manufacturing Systems*, 2014, **33**(4): 647−653

36 Matic A, Osmani V, Mayora-Ibarra O. Analysis of social interactions through mobile phones. *Mobile Networks and Applications*, 2012, **17**(6): 808−819

37 Karimi S, Ardalan Z, Naderi B, Mohammadi M. Scheduling flexible job-shops with transportation times: mathematical models and a hybrid imperialist competitive algorithm. *Applied Mathematical Modelling*, 2017, **41**: 667−682

38 Zhou W, Yan J J, Li Y, Xia C M, Zheng J R. Imperialist competitive algorithm for assembly sequence planning. *International Journal of Advanced Manufacturing Technology*, 2013, **67**(9−12): 2207−2216

39 Wang B X, Guan Z L, Li D S, Zhang C Y, Chen L. Two-sided assembly line balancing with operator number and task constraints: a hybrid imperialist competitive algorithm. *International Journal of Advanced Manufacturing Technology*, 2014, **74**(5−8): 791−805

40 Zhang Xin-Long, Zheng Xiu-Wan, Xiao Han, Li Wei. A new imperialist competitive algorithm for solving TSP problem. *Control and Decision*, 2016, **31**(4): 586−592
(张鑫龙, 郑秀万, 肖汉, 李伟. 一种求解旅行商问题的新型帝国竞争算法. 控制与决策, 2016, **31**(4): 586−592)

41 Lei D M. Simplified multi-objective genetic algorithms for stochastic job shop scheduling. *Applied Soft Computing*, 2011, **11**(8): 4991−4996

42 Afruzi E N, Najafi A A, Roghanian E, Mazinani M. A multi-objective imperialist competitive algorithm for solving discrete time, cost and quality trade-off problems with mode-identity and resource-constrained situations. *Computers & Operations Research*, 2014, **50**: 80−96

43 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

44 Kemmoé S, Lamy D, Tchernev N. A job-shop with an energy threshold issue considering operations with consumption peaks. *IFAC-PagesOnLine*, 2015, **28**(3): 788−793

45 Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 1993, **41**(3): 157−183

46 Dauzère-Pérès 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**: 281−306

47 Knowles J, Corne D. On metrics for comparing nondominated sets. In: Proceedings of the 2002 Congress on Evolutionary Computation. Honolulu, HI, USA: IEEE, 2002. 711−716

48 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



**雷德明** 武汉理工大学自动化学院教授. 主要研究方向为智能系统优化与控制. 本文通信作者.
E-mail: deminglei11@163.com
(**LEI De-Ming** Professor at the School of Automation, Wuhan University of Technology. His research interest covers intelligent system optimization and control. Corresponding author of this paper.)



**杨冬婧** 武汉理工大学自动化学院硕士研究生. 主要研究方向为制造系统智能优化与调度.
E-mail: niceydj@163.com
(**YANG Dong-Jing** Master student at the School of Automation, Wuhan University of Technology. Her research interest covers maufacturing systems intelligent optimization and scheduling.)