控制与决策
Control and Decision

带峰值能耗约束流水线调度的协同群智能优化

王凌， 王晶晶

引用本文:
王凌, 王晶晶. 带峰值能耗约束流水线调度的协同群智能优化[J]. 控制与决策, 2021, 36(10): 2350−2358.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2020.0429

带峰值能耗约束流水线调度的协同群智能优化

王 凌†, 王晶晶

(清华大学 自动化系，北京 100084)

摘 要: 当今社会环境问题日益严重,能源成本日益提高,峰值能耗在生产制造中备受关注. 针对带峰值能耗约束的流水线调度问题,即生产过程中各时间节点机器总功耗不得超过给定阈值,以最小化最大完工时间为目标,提出一种协同群智能算法. 首先,协同多种解码方法产生多样化的可行调度,融合启发式方法与随机方法以初始化种群;其次,设计两类基于问题特性的搜索操作,分别调整工件序列和加工速度;接着,根据目标空间中个体的分布,设计多种搜索操作的协同机制,对不同区域的个体执行不同的搜索操作,并对精英个体进行局部增强搜索以进一步改善性能;最后,采用大量算例开展数值实验,验证了所设计协同机制的有效性,并通过与数学求解器和现有算法的对比结果表明所提出算法能够更有效求解带峰值能耗约束的流水线调度问题.

关键词: 流水线调度； 节能； 峰值能耗约束； 协同群智能

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2020.0429 **开放科学(资源服务)标识码(OSID):**

引用格式: 王凌,王晶晶. 带峰值能耗约束流水线调度的协同群智能优化[J]. 控制与决策, 2021, 36(10): 2350-2358.

Cooperative memetic optimization for flowshop scheduling with peak power consumption constraint

WANG Ling†, WANG Jing-jing

(Department of Automation，Tsinghua University，Beijing 100084，China)

Abstract: With the increasing energy costs and the serious environmental issue, peak power consumption attracts much attention by manufacturing industries. It requires that the real-time power consumption cannot exceed a given peak power at any time during manufacturing process. Aiming at the permutation flowshop scheduling problem with peak power consumption constraint (PFSPP), a cooperative memetic algorithm is proposed. First, multiple decoding methods are collaborated to generate diverse and feasible schedules, and a heuristic and random method are fused to initialize population. Second, two problem-specific search operators are designed according to the characteristics of the problem for adjusting the job sequence and speed selection, respectively. Furthermore, according to distribution of individuals in the objective space, a cooperation scheme is designed. Different search operators are used for the individuals in different regions. In addition, a local intensification is performed on the elite individual to further improve the performance. Numerical tests are conducted by using extensive instances. Finally, the effectiveness of the designed cooperation mechanisms is demonstrated. Moreover, the comparisons with the math solver and the existing algorithms show that the proposed algorithm is more effective in solving the PFSPP.

Keywords: flowshop scheduling; energy saving; peak power consumption constraint; cooperative memetic optimization

0 引 言

面临全球严重的环境问题,绿色制造已成为现代化生产日趋关注的制造模式. 在过去60年,工业界的能耗几乎翻倍,约占世界总能耗的一半[1]. 随着不可再生能源的快速消耗,能源价格不断提高,能源的高效利用受到广泛关注. 绿色调度是绿色制造的关键环节,高效的优化调度方法能够协同优化企业的经济指标和绿色指标,实现增效、节能、减排、降耗、降成本[2-3].

早期关于绿色调度的研究集中于能源管理策略的设计,即设备的处理速度可调. 一般认为,随着处理速度的增加,处理时间越快功耗越大[4]. 文献[5]考虑连续可变的处理器速度,功耗为速度的指数函数. 针对单机调度问题,Mouzon等[6]提出了一种节能调度

收稿日期: 2020-04-16； **修回日期:** 2020-06-17.

基金项目: 国家重点研发计划项目(2016YFB0901900)； 国家自然科学基金项目(61873328)； 国家杰出青年科学基金项目(61525304).

†通讯作者. E-mail: wangling@tsinghua.edu.cn.

框架. 针对多产品批量调度问题, Capón-García 等[7]建立了非线性混合整数规划模型联合优化环境和经济指标. 针对分布式流水线调度问题, 文献[8]提出了一种基于知识的协同优化算法最小化最大完工时间和总能耗. 鉴于工业界普遍的分时电价(time-of-use, TOU)政策, 一些学者提出了优化电费的调度方法[9]. 针对混合流水线调度问题, Luo 等[10]提出了一种蚁群算法提高生产效率并优化电费. 针对流水线调度问题, Zhang 等[11]提出了一种时间相关的整数规划模型优化电费和碳足迹. 此外, Zhang 等分别针对单工厂[12]和多工厂两类流水线[13]提出了节能调度方法.

大多数制造企业的主要能源来源为电能, 生产过程中所有机器的最大总功耗称为峰值能耗. 如果峰值能耗超过既定阈值, 则电价将大大高于额定电价[14]. 因此, 将峰值能耗控制在合理范围内, 能够有效节省费用. 目前, 峰值能耗约束的车间调度研究还不多, 多数工作位于模型和启发式方法层面. 考虑机器速度离散可调和连续可调的置换流水线调度, Fang 等[15]分别提出了几种有效的混合整数规划模型. Nagasawa 等[16]提出了一种鲁棒生产调度模型减少峰值能耗. 针对给定操作排序, Módos 等[17]提出了一种多项式算法, 寻找满足在特定时间区间内能耗约束的最优鲁棒调度. 上述研究通过调整参考调度方案的时间表, 使给定时间区间内的总能耗满足约束, 利用智能优化处理带峰值能耗约束的车间调度的研究有待发展.

流水线是应用最为广泛的调度问题[18-19], 已有大量有效算法, 包括启发式方法[20-21]和迭代搜索算法, 如模拟退火[22]、贪婪迭代算法[23]、差分进化[24]等. 本文研究带峰值能耗约束的置换流水线调度问题, 在决策工件排序的同时还需考虑各操作的加工速度, 以实时满足功耗约束, 并优化最大完工时间. 由于问题的约束与决策更为复杂, 将群体智能与问题知识融合有望取得更好的效果[25]. 同时, 源于自然界和社会系统广泛的协同现象, 协同进化利用多个对象通过一定的机制和策略开展协同搜索, 能够有效处理复杂问题[26]. 目前, 协同群智能优化已在诸多问题上取得了良好的效果[8,26-27]. 基于此动机, 本文针对PFSPP提出一种协同群智能算法(cooperative memetic algorithm, CMA), 通过多种解码方法的协同、多种搜索操作的协同、随机调整操作与局部增强操作的协同, 获取优良的调度方案, 在满足峰值能耗约束的条件下调度指标.

1　问题描述

1.1　PFSPP描述

PFSPP描述如下: n个工件$(j = 1, 2, \ldots, n)$需要按流水线方式依次通过m台机器$(i = 1, 2, \ldots, m)$进行加工, 操作$O_{j,i}$表示工件j在机器i上的加工操作. 每台机器i均有s档加工速度可调, 操作$O_{j,i}$的标准加工时间为$t_{j,i}$. 若工件j在机器i上的加工速度为$v_k(k = 1, 2, \ldots, s)$, 则操作$O_{j,i}$的实际加工时间为$t_{j,i}/v_k$. 随着机器速度v_k的增加, 实际加工时间减小而功耗增加, 但峰值能耗不能超过给定阈值$Q_{\max}$. 为保证可行调度的存在性, 假设$Q_{\max}$不小于机器的最小功耗. 调度目标为最小化最大完工$C_{\max}$, 即机器m上最后一个工件的完工时间.

类似于诸多文献, 约定: 所有工件的标准加工时间已知; 所有工件相互独立且在零时刻即可加工; 机器连续可用, 不考虑机器故障; 同一时刻, 一台机器只能加工一个工件, 且一个工件不会被多个机器同时加工; 机器在加工某一工件过程中, 速度不可调; 机器间缓存区无限大; 工件运输时间和机器设置时间已包含在加工时间内. PFSPP的求解比传统流水线调度更为复杂, 需同时决策工件序列和各操作的加工速度, 在满足峰值能耗约束的条件下优化最大完工时间. 其数学模型如下:

$$C_{\max} = C_{\pi_n, m}; \tag{1}$$

$$C_{\pi_1,1} \geqslant t_{\pi_1,1}/v_{\pi_1,1}; \tag{2}$$

$$C_{\pi_j,1} \geqslant C_{\pi_{j-1},1} + t_{\pi_j,1}/v_{\pi_j,1},\ j = 2, 3, \ldots, n; \tag{3}$$

$$C_{\pi_1,i} \geqslant C_{\pi_1,i-1} + t_{\pi_1,i}/v_{\pi_1,i},\ i = 2, 3, \ldots, m; \tag{4}$$

$$C_{\pi_j,i} \geqslant \max(C_{\pi_{j-1},i}, C_{\pi_j,i-1}) + t_{\pi_j,i}/v_{\pi_j,i},$$
$$j = 2, 3, \ldots, n,\ i = 2, 3, \ldots, m; \tag{5}$$

$$\sum_{i=1}^{m} q_{j,i} \leqslant Q_{\max},\ j \in J_t,\ \forall t \in [0, C_{\max}]. \tag{6}$$

其中: 式(1)表明调度目标为最大完工时间, 式(2)～(5)表明流水线工艺约束, 式(6)表明峰值能耗约束, $C_{j,i}$、$t_{j,i}$、$v_{j,i}$和$q_{j,i}$分别为操作$O_{j,i}$的完工时间、标准加工时间、加工速度和功耗, $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$为工件加工序列, J_t为时刻t处于加工状态的工件集合.

1.2　问题特点

由于PFSPP中工件j在机器i上操作$O_{j,i}$的加工速度v_k在其加工过程中不变, 则相应的机器功耗也保持不变. 对于在时刻t处于加工状态的工件集合J_t, 用$C_t^L(S_t^L)$表示在时刻t之前完成(开始)加工的最后一个工件的完成(开始)加工时间, 用$C_t^R(S_t^R)$表

示在时刻t之后完成(开始)加工的第1个工件的完成(开始)加工时间. 令$t_1 = \max\{C_t^L, S_t^L\}, t_2 = \min\{C_t^R, S_t^R\}$,则对于任意时刻$t' \in (t_1, t_2], J_{t'} = J_t$,即区间$(t_1, t_2]$的能耗为定值.

通过甘特图对上述特点给予解释,如图1所示. 在时刻t,有两个工件处于加工状态,即$J_t = \{j_2, j_3\}$. 可见:$t_1 = S_t^L$是工件j_2在机器M_1上的开始加工时间,$t_2 = S_t^R = C_t^R$是工件j_5在机器M_3上的开始时间和工件j_3在机器M_2上的完成时间. 因此,时间区间$(t_1, t_2]$内的功耗为定值,为工件j_2在机器M_1和工件j_3在机器M_2上的功耗之和.

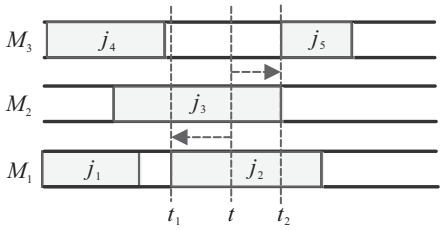

图1 PFSPP特性示意图

根据PFSPP的特性,可将加工过程中随时间变化的功耗按工件的开始和完成加工时间节点离散化. 要求各时间节点的能耗不超过给定阈值,使调度方案满足峰值能耗约束. 图2给出了三工件三机器问题的可行调度甘特图和随时间变化的能耗曲线. 由图2可见,每个工件的开始和完成加工时间节点的功耗均不超过给定阈值,故为可行调度.

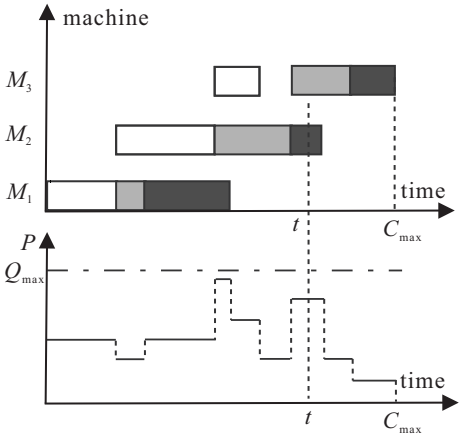

图2 PFSPP甘特图及其功耗曲线

2 协同群智能算法

2.1 编码和解码

对于PFSPP,每个解包含两个部分:工件的加工序列和各操作的加工速度. 因此,采用三元组(π, V, flag)表示一个个体,$\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$为工件的加工顺序,$V = (v_{j,i})_{n\times m}$为速度矩阵,$v_{j,i}$为操作$O_{j,i}$的加工速度,flag为个体所采用的解码方法.

为得到可行且有效的调度方案,文献[28]提出了5种基于最早加工规则的解码方法. 根据各操作的开始和完成加工时间节点,待加工操作在保证满足能耗约束的条件下选择最早可加工时间节点进行加工. 根据流水线特点,按照以下5种操作顺序给出5种解码规则,即工件为先(job-precedence, JP)、机器为先(machine-precedence, MP)、最大总剩余负荷(largest total remaining load, LTRL)、最早完工时间(earliest completion time, ECT)和工件-机器平衡(balanced job-machine, BJM)规则.

由于上述规则存在互不支配的关系,没有一种规则占据绝对优势. 为了获得性能更好的调度方案,本文采用基于概率选择的方式将5种规则联合使用. 记$p_d^i(g)(i = 1, 2, 3, 4, 5)$为在第$g$代中对第$i$种解码方法的选择概率,依据精英解集(elite set, ES)的历史信息,按下式对其进行更新:

$$p_d^i(g) = (1-\alpha)p_d^i(g-1) + \alpha\frac{\sum_{r=1}^{|\text{ES}|} I^r}{|\text{ES}|}; \tag{7}$$

$$I^r = \begin{cases} 1, \ \text{flag}(r) = i; \\ 0, \ \text{otherwise}. \end{cases} \tag{8}$$

其中:ES为当前种群中最优的10%个体,ES为精英解的数量,学习速率α设为0.1.

2.2 种群初始化

为保证初始种群具有一定的质量和多样性,在此提出一种启发式初始化方法,并将其与随机方法协同使用.

Nawaz-Enscore-Ham(NEH)是求解置换流水线调度最有效的启发式方法之一[20]. 文献[21]给出了一种改进NEH,即NEHFF. 考虑到机器速度可调的情况,在此提出面向PFSPP的NEHFF规则,具体步骤如下.

step 1: 随机初始化速度矩阵V,将所有工件按其实际总加工时间和降序排列得到$\pi_0 = \{\pi_0(1), \pi_0(2), \ldots, \pi_0(n)\}$,令$\pi_1 = \{\pi_0(1)\}, k = 2$.

step 2: 令$j = \pi_0(k)$,将j分配至当前工件序列π_{k-1}的各个位置,记分配至第u个位置后的最大完工时间为$C_{\max}(\pi_k^u, V)$.

step 3: 找到所有使得$C_{\max}(\pi_k^u, V)$最小的位置,记为$\{u_1, u_2, \ldots, u_h\}$,其中$h$为位置个数.

step 4: $\forall i \in \{1, 2, \ldots, h\}$,计算$(\pi_k^{u_i}, V)$的总机器空闲时间$T(\pi_k^{u_i}, V)$,保留使$T(\pi_k^{u_i}, V)$最小的$u_i$为$u_{i^*}$.

step 5: 令$\pi_k = \pi_k^{u_{i^*}}, k = k+1$. 若$k > n$,则输出解$\pi_k$,否则返回step 2.

由于初始化时速度矩阵是随机产生的,多次重复以上步骤可获得多个不同的初始个体. 为了保证种群的多样性,初始种群中P_i%个体按由上述NEHFF产生,其余个体完全随机产生.

2.3　搜索操作协同

2.3.1　搜索操作设计

PFSPP需同时决策工件序列和各操作的加工速度,因此设计两种搜索操作分别调整工件的加工序列和速度矩阵.

1) 调整工件序列的搜索操作.

插入操作: 从置换编码序列中随机选取工件j_1,将其插入另一随机位置.

交换操作: 从置换编码序列中随机交换两个工件的位置.

2) 调整速度矩阵的搜索操作.

为了有效优化最大完工时间,基于关键操作设计调整速度的搜索操作. 若操作$O_{\pi(j),i}$满足$C_{\pi(j),i} = S_{\pi(j+1),i}$或$C_{\pi(j),i} = S_{\pi(j),i+1}$,则定义为关键操作,其中$S_{j,i}$和$C_{j,i}$分别表示操作$O_{j,i}$的开始加工时间和完工时间. 图3给出了一个调度方案的关键操作,操作$\{O_{2,1}, O_{3,1}, O_{2,2}, O_{3,2}, O_{1,2}, O_{3,3}, O_{1,3}\}$是关键操作,其余为非关键操作.

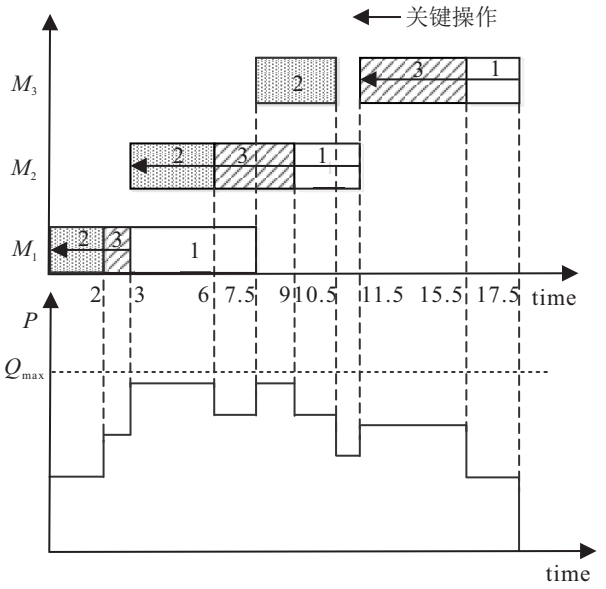

图3　调度方案中关键操作示意图

加速操作: 随机选择一个关键操作$O_{j,i}$,其速度满足$v_{j,i} = a < s$,则增加$v_{j,i}$到b $(a < b \leqslant s)$.

减速操作: 随机选择一个非关键操作$O_{j,i}$,其速度满足$v_{j,i} = a > 1$,则降低$v_{j,i}$到b $(1 \leqslant b < a)$.

对于图3的调度方案,通过加速操作可得到性能更优的调度如图4所示,即选择关键操作$O_{3,3}$加速,操作$O_{1,3}$左移. 通过减速操作也可得到更优的调度如图5所示,即选择非关键操作$O_{2,3}$进行减速,操作$O_{2,3}$、$O_{3,3}$和$O_{1,3}$左移且满足峰值能耗约束.

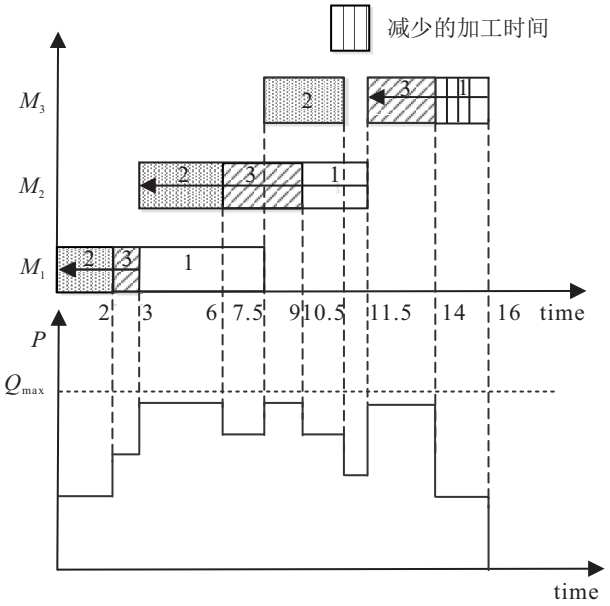

图4　通过加速操作优化的调度方案

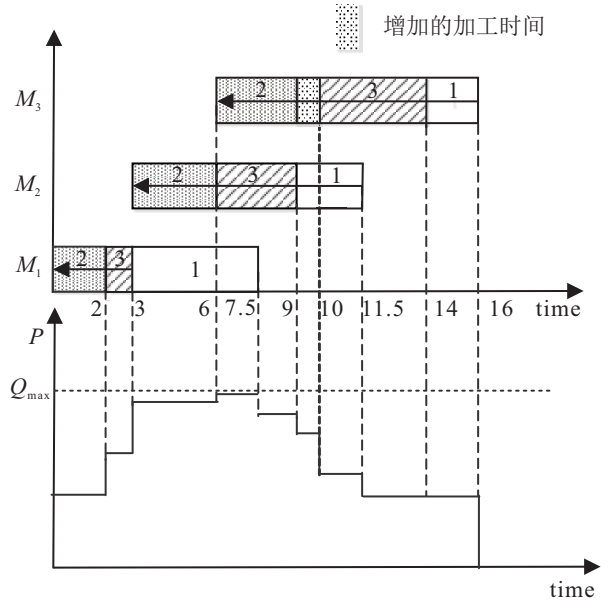

图5　通过减速操作优化的调度方案

2.3.2　搜索操作设计

为提高搜索效率,设计一种操作协同机制,对不同特点的个体执行不同的搜索操作.

对于每个个体,总能耗(total energy consumption, TEC)为

$$\mathrm{TEC} = \sum_{i=1}^{m}\sum_{j=1}^{n} q_{j,i} \times p_{j,i} = \sum_{i=1}^{m}\sum_{j=1}^{n} \epsilon \times t_{j,i} \times v_{j,i}, \tag{9}$$

其中$q_{j,i}$、$p_{j,i}$、$v_{j,i}$、$t_{j,i}$分别为操作$O_{j,i}$的功耗、实际加工时间、加工速度和标准加工时间. 通常,能耗与速度的关系为$q_{j,i} = \epsilon \times v_{j,i}^2$,故TEC与机器的加工速度呈

正相关[8].

在优化最大完工时间时,增加TEC作为个体的参考评价指标,借鉴文献[27]的分区思想,根据个体在目标空间的分布按如下流程确定所要执行的搜索操作.

首先,将个体X_l的$C_{\max}$和TEC按照$g_p(X_l) = (f_p(X_l)-f_p^{\min})/(f_p^{\max}-f_p^{\min})$归一化. 其中:$p=0$表示$C_{\max}$指标,$p=1$表示TEC指标,$f_p(X_l)$和$g_p(X_l)$分别表示第$p$个指标值及其归一化后的值,$f_p^{\max}$和$f_p^{\min}$分别表示第$p$个指标在当前种群中的最大值和最小值. 归一化后的目标空间如图6所示.

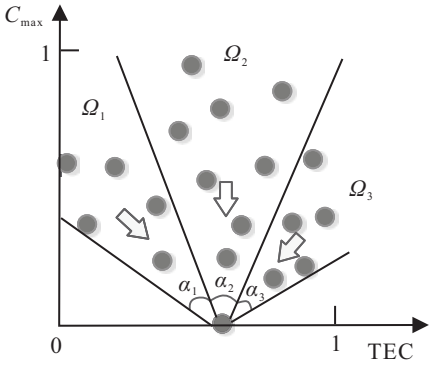

图6 归一化的目标空间

其次,以$C_{\max}$最小的个体为参考点,将其分为3个区域Ω_1、Ω_2和Ω_3. 显然,Ω_1中个体的TEC值较小,Ω_3中个体的TEC值较大. 为满足峰值能耗约束,适宜的机器速度能够有效优化$C_{\max}$. 进而,对于Ω_1中的个体执行加速操作;Ω_3中的个体执行减速操作;Ω_2中的个体依次实行插入和交换操作以调整工件序列.

每一代进化后,各区域的大小依据所包含个体的改善程度进行自适应调整. 按下式计算各区域的得分:

$$\mathrm{Sr}(\Omega_k)=\frac{\max\{0,C_{\max}(X_l^{(k)})-C_{\max}(X_l^{(k)'})\}}{C_{\max}(X_l^{(k)})}. \tag{10}$$

其中:$X_l^{(k)}$为区域Ω_k内的个体,$k=1,2,3$;$X_l^{(k)'}$为$X_l^{(k)}$执行区域Ω_k操作后的个体. 因此,第g代区域Ω_k的面积占比为

$$\alpha_k=\frac{\mathrm{Sr}(\Omega_k)/\mathrm{Nu}(\Omega_k)}{\sum\limits_k(\mathrm{Sr}(\Omega_k)/\mathrm{Nu}(\Omega_k))}+\beta. \tag{11}$$

其中:$\mathrm{Nu}(\Omega_k)$为区域Ω_k内的个体数量,$\beta=0.1$以确保$\alpha_k\neq 0$.

2.4 局部增强搜索

为了增强算法对特定问题的优化能力,在此将问题相关的局部增强搜索融入群体搜索流程. 同时,为了节省计算量,局部增强操作仅作用于种群中的精英个体. 局部操作包括插入操作、交换操作和如下速度调整操作:即随机选择一个操作$O_{j,i}$,若为关键操作,则提高其加工速度$v_{j,i}$从a到b $(a<b\leqslant s)$,否则降低其加工速度$v_{j,i}$从a到c $(1\leqslant c<a)$.

局部增强搜索的具体步骤如下,其中LS为控制搜索深度的参数.

step 1:从当前种群中的精英解集ES中随机选择个体$\boldsymbol{a}$,令$k=0$.

step 2:对个体$\boldsymbol{a}$依次执行插入操作、交换操作和速度调整操作,若得到更优的个体则替换原个体,否则保留原个体.

step 3:$k=k+1$,若$k>$LS,则输出新个体,否则返回step 2.

2.5 算法流程

通过上述设计,给出算法的整体流程如图7所示. 其中,种群的混合初始化包括NEHFF和随机方法,目标空间的分区以实现不同搜索操作的协同,局部增强搜索以改善精英个体的性能,解码使用概率的更新以便更好地采用合适的解码方法. 通过多方面的协同策略与自适应调整策略,所提出算法有望取得优良的性能.

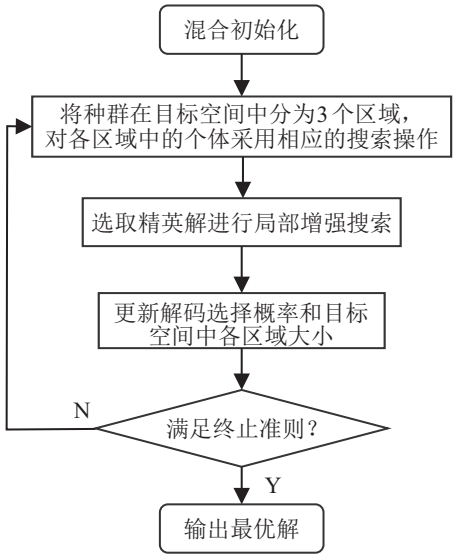

图7 算法整体流程

3 数值仿真与性能对比

3.1 实验设置

采用C++编程,运行环境为3.2 GHz主频的Intel(R) Core(TM) i7-3470 CPU. 根据文献[15, 28]的方法生成测试集,包括小规模问题集,即工件数n为$\{8,12,16,20\}$,机器数m为$\{2,4\}$;大规模问题集,即工件数n为$\{20,40,60,80,100\}$,机器数m为$\{4,8,16\}$. 对每种规模组合(n,m),随机生成10个算例. 工件

的标准加工时间 $t_{j,i}$ 为区间 [5, 50] (h) 中均匀分布随机数, 机器速度 v 设置为 $\{1, 1.3, 1.55, 1.75, 2.1\}$, 机器功耗与速度的关系为 $\mathrm{PP}_{j,v} = 4\times v^2$ (kW). 另外, 约束上下限分别为 $\overline{Q} = \max\limits_{i\in M, j\in J}\{q_{j,i,1}\}$ 和 $\underline{Q} = \sum\limits_{i\in M}\max\limits_{j\in J, s\in S}\{q_{j,i,s}\}$. 若峰值约束 $Q_{\max} < \underline{Q}$, 则不存在可行调度解; 若 $Q_{\max} > \overline{Q}$, 则所有工件能以最快速度同时加工. 将区间 $[\underline{Q}, \overline{Q}]$ 均分为 9 个子区间, 产生 10 个节点 $Q_{\mathrm{ep}}(\mathrm{ep} = 1, 2, \ldots, 10)$ 作为约束阈值 $Q_{\max}$. 因此, 考虑所有 (n, m) 组合和阈值 $Q_{\max}$, 共生成 800 个小规模算例和 1 500 个大规模算例.

算法参数设置如下: $\mathrm{PS} = 20, P_i = 0.1, \mathrm{LS} = 20$. 根据文献 [28] 的统计分析, 解码选择概率 p_d^i 的初始化因问题规模而与 $Q_{\max}$ 不同. 当 $Q_{\max} < Q_5$ 时, 小规模问题 $p_d^i(0) = \{0.1, 0.1, 0.15, 0.15, 0.5\}$, 大规模问题 $p_d^i(0) = \{0, 0, 0.1, 0.1, 0.8\}$; 当 $Q_{\max} \geqslant Q_5$ 时, 统一设置 $p_d^i(0) = \{0.1, 0.1, 0.2, 0.2, 0.4\}$. 另外, 3 个区域 Ω_k 的面积占比 α_k 初始化为 $\{0.05\times\mathrm{ep}, 0.5, 0.05\times(10 - \mathrm{ep})\}$. 采用式 (12) 相对百分偏差 (relative percentage deviation, RPD) 作为算法的评价指标, 其中 $C_{\max}^{\mathrm{best}}$ 为各算法所得最小 $C_{\max}$, $C_{\max}^{\mathrm{alg}}$ 为某一算法所得 $C_{\max}$, 有

$$\mathrm{RPD} = (C_{\max}^{\mathrm{alg}} - C_{\max}^{\mathrm{best}})/C_{\max}^{\mathrm{best}} \times 100\,\%. \qquad (12)$$

3.2　小规模算例仿真比较

对于小规模算例, 采用数学规划求解器 Gurobi (版本 7.0) 求解混合整数线性规划模型[15], 最大运行时间设为 1 800 s. 由于目前尚无智能算法求解该问题的研究, 将求解流水线调度的两种有效算法加以扩展进行对比, 包括贪婪迭代算法 (iterated greedy, IG)[23] 和离散差分进化 (discrete differential evolutionary, DDE)[24]. IG 和 DDE 的参数设置同文献 [23] 和 [24], 初始化采用 NEHFF, 局部搜索采用速度调整操作. 每种算法采用相同的最大运行时间, 即 $T_{\max} = 0.1 \times n \times m$ 秒.

对于每个算例, 3 种算法均独立运行 10 次, RPD 的最小值、平均值和方差分别记作 min、mean 和 std, 各算法的 RPD 值按 (n, m) 归类的结果如表 1 所示, 按约束 $Q_{\max}$ 归类的结果如表 2 所示, RPD 值的盒图如图 8 所示.

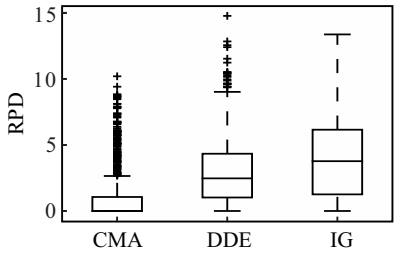

图 8　CMA、DDE 和 IG 求解小规模算例的盒图

表 1　小规模算例按 (n, m) 归类的 RPD 值对比

(n, m)	模型求解		CMA			DDE			IG		
	RPD	time / s	min	mean	std	min	mean	std	min	mean	std
(8, 2)	**0**	674.7	2.53	4.71	1.30	4.54	9.49	2.80	3.49	5.43	1.34
(8, 4)	4.82	1 394.4	**0.26**	2.34	1.49	2.92	8.16	4.18	3.30	5.02	1.01
(12, 2)	**0.63**	1 421.6	2.02	4.31	1.84	4.12	8.31	3.69	3.78	5.68	1.70
(12, 4)	12.18	1 761.9	**0.09**	2.25	2.22	2.36	6.87	5.21	4.18	5.99	1.59
(16, 2)	1.72	1 596.1	**1.70**	3.65	2.08	3.46	7.11	4.19	3.55	5.33	2.14
(16, 4)	43.74	1 838.1	**0.14**	2.13	2.64	1.68	5.74	6.22	4.56	6.22	1.94
(20, 2)	3.35	1 642.7	**1.44**	3.29	2.45	2.95	6.48	4.80	3.15	5.16	2.76
(20, 4)	49.02	1 834.5	**0.11**	2.33	3.70	1.83	5.38	6.86	4.93	6.71	2.51
ave	14.44	1 520.5	**1.04**	3.13	2.21	2.98	7.19	4.74	3.87	5.69	1.87

表 2　小规模算例按 $Q_{\max}$ 归类的 RPD 值对比

$Q_{\max}$	模型求解		CMA			DDE			IG		
	RPD	time/s	min	mean	std	min	mean	std	min	mean	std
Q_1	9.46	1 409.8	2.85	4.91	4.57	3.64	7.88	10.85	**1.58**	3.48	3.69
Q_2	20.73	1 658.7	**2.90**	6.54	4.64	6.10	12.34	8.12	3.25	6.06	3.58
Q_3	22.76	1 781.8	**1.79**	4.96	3.36	4.29	9.61	6.13	6.33	8.74	2.74
Q_4	21.74	1 715.9	**0.73**	3.01	2.02	2.87	7.23	4.09	5.28	7.22	1.82
Q_5	15.59	1 713.3	**0.70**	3.36	2.18	2.64	7.43	4.15	7.16	9.68	2.05
Q_6	16.06	1 761.3	**0.49**	2.73	1.71	2.89	6.98	3.50	6.19	8.02	1.38
Q_7	14.76	1 658.1	**0.41**	2.23	1.39	2.34	6.43	3.23	4.14	5.99	1.27
Q_8	10.98	1 550.8	**0.28**	1.94	1.19	2.51	6.72	3.36	2.73	4.19	1.05
Q_9	9.87	1 282.1	**0.22**	1.51	0.98	2.29	5.80	2.74	1.92	3.32	0.95
Q_{10}	2.41	673.2	**0.01**	0.09	0.10	0.24	1.53	1.26	0.07	0.25	0.20
ave	14.44	1 520.5	**1.04**	3.13	2.21	2.98	7.19	4.74	3.87	5.69	1.87

由表1可见,模型仅对最小规模$n = 8$和$m = 2$的算例获得最优解,但运行时间远大于智能算法. 对于大部分小规模算例,CMA的min和mean几乎全面优于模型求解以及DDE与IG,std值也优于同为群智能算法的DDE. 由表2可见,当约束最紧或最松时,模型获得次优解的时间较短,但仍远大于智能算法. CMA性能的优势较明显,尤其是约束较松的情况. 由图8直观可见,CMA的性能优于DDE和IG. 因此,CMA相较于求解器、IG和DDE能够更加高效且有效地求解小规模PFSPP.

3.3 大规模算例仿真比较

对于大规模算例,模型求解的时间难以承受,因此智能算法成为高效的求解手段. 3种算法的RPD值按(n, m)归类的结果如表3所示,按约束$Q_{\max}$归类的结果如表4所示,RPD的盒图如图9所示.

由表3可见,CMA求解大多数算例的性能最优,特别$n \geqslant 60$的算例. 就mean和std而言,CMA呈现的鲁棒性更好. 由表4可见,当约束较紧或较松时,CMA的优势更明显. 图9更是直观展现了CMA求解大规模算例的优越性能.

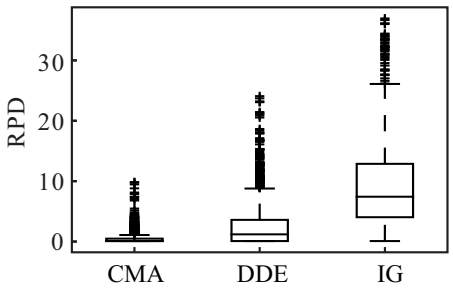

图 9 CMA, DDE和IG求解大规模算例的盒图

归纳而言,所提出算法相对模型求解器具有有效性和明显的高效性,相对其他智能算法具有有效性和鲁棒性,是求解带峰值能耗约束的大规模流水线调度问题的高效算法.

表 3 大规模算例以(n, m)归类的RPD值对比

(n, m)	CMA			DDE			IG		
	min	mean	std	min	mean	std	min	mean	std
(20, 4)	**0.42**	2.95	17.12	1.22	5.07	30.19	5.82	9.89	24.76
(20, 8)	1.13	4.11	22.75	**0.51**	5.15	38.39	12.86	19.03	42.89
(20, 16)	1.27	4.38	28.55	**1.03**	5.10	44.07	11.91	15.85	37.33
(40, 4)	0.93	2.85	25.79	**0.25**	3.90	66.86	11.03	16.27	64.78
(40, 8)	**1.00**	3.47	33.04	2.75	6.54	60.50	13.22	17.16	55.27
(40, 16)	**0.57**	2.80	37.57	2.04	5.06	53.74	5.52	8.51	49.11
(60, 4)	**0.55**	2.37	39.16	1.63	5.89	128.51	13.72	17.82	75.09
(60, 8)	**0.25**	2.43	44.46	4.46	7.47	74.06	10.21	13.52	69.59
(60, 16)	**0.11**	2.07	45.61	2.13	4.36	54.61	3.26	5.86	65.23
(80, 4)	**0.10**	2.06	63.42	4.14	8.79	180.76	13.79	17.21	81.49
(80, 8)	**0.10**	2.46	59.68	5.55	8.07	75.61	9.63	12.61	79.77
(80, 16)	**0.13**	1.94	51.24	2.12	4.07	57.65	2.82	5.52	77.86
(100, 4)	**0.06**	2.36	94.39	5.90	10.12	189.30	13.33	16.69	94.99
(100, 8)	**0.10**	1.92	58.81	4.78	6.85	70.57	7.48	10.12	91.27
(100, 16)	**0.16**	1.76	54.20	1.34	3.47	67.81	2.20	5.00	97.26
ave	**0.46**	2.66	45.05	2.66	5.99	79.51	9.12	12.74	67.11

表 4 大规模算例以$Q_{\max}$归类的RPD值对比

$Q_{\max}$	CMA			DDE			IG		
	min	mean	std	min	mean	std	min	mean	std
Q_1	**1.38**	3.85	104.95	1.77	5.95	235.78	11.51	15.60	158.95
Q_2	**0.49**	2.63	62.64	1.61	5.35	121.42	8.57	11.17	65.49
Q_3	**0.45**	2.27	40.91	0.65	3.62	83.72	6.81	9.10	47.69
Q_4	0.64	2.60	38.30	**0.40**	2.82	49.60	5.38	7.57	40.41
Q_5	0.65	2.81	38.10	**0.49**	3.28	50.03	4.87	7.41	43.69
Q_6	**0.50**	2.70	33.29	1.33	4.47	46.57	5.57	8.84	54.66
Q_7	**0.23**	2.45	31.76	2.69	5.94	47.57	7.46	11.47	59.02
Q_8	**0.13**	2.36	32.75	4.49	8.05	51.86	10.94	15.73	67.45
Q_9	**0.09**	2.55	33.75	5.88	9.46	51.80	13.59	18.61	66.65
Q_{10}	**0.03**	2.38	34.07	7.27	10.99	56.75	16.51	21.88	67.11
ave	**0.46**	2.66	45.05	2.66	5.99	79.51	9.12	12.74	67.11

3.4 算法自环节有效性验证

为验证所设计协同机制的有效性,包括初始化协同、解码协同、操作协同及局部增强搜索环节,利用大规模算例与CMA的如下变种开展性能对比. 公平起见,每种算法采用相同的最大运行时间作为终止准则,即$T_{\max}=0.1\times n\times m$秒.

1) CMA初始化环节仅采用随机初始化方法(记为CMAnN),仅采用NEHFF初始化(记为CMAnR),用于验证协同初始化的有效性.

2) CMA解码时随机选择5种解码之一执行(记为CMAnD),用于验证协同解码的有效性.

3) CMA随机选择3种搜索操作之一执行搜索(记为CMAnC),用于验证操作协同的有效性.

4) CMA不采用局部增强搜索(记为CMAnLS),用于验证局部增强搜索的有效性.

表5给出了CMA与5种变种算法所得RPD值,结果按(n,m)归类,表6按$Q_{\max}$归类给出结果,图10则给出了RPD值的盒图.

表 5　不同CMAs以(n,m)归类的RPD值对比

(n,m)	CMA	CMAnN	CMAnR	CMAnD	CMAnC	CMAnLS
(20, 4)	**0.09**	3.08	2.78	3.06	2.33	6.99
(20, 8)	**0.19**	4.59	4.50	4.28	1.15	19.38
(20, 16)	**0.47**	5.76	6.32	6.30	0.72	17.99
(40, 4)	**0.16**	2.98	3.19	2.87	1.05	11.05
(40, 8)	**0.64**	5.03	5.26	4.93	0.81	18.22
(40, 16)	**0.47**	3.68	4.30	4.34	0.56	12.22
(60, 4)	**0.18**	2.84	3.84	2.73	0.89	12.31
(60, 8)	**0.34**	4.55	5.02	4.73	0.47	15.44
(60, 16)	**0.27**	3.09	3.46	3.29	0.50	9.02
(80, 4)	**0.35**	2.87	5.91	3.22	0.55	12.98
(80, 8)	0.51	4.31	5.43	4.56	**0.35**	13.22
(80, 16)	**0.27**	2.75	2.83	2.63	0.65	6.76
(100, 4)	**0.32**	3.12	7.20	4.14	0.68	13.08
(100, 8)	0.61	3.43	5.12	3.90	**0.29**	11.18
(100, 16)	**0.23**	2.46	2.26	2.09	0.59	5.45
ave	**0.34**	3.64	4.50	3.80	0.77	12.35

表 6　不同CMAs以$Q_{\max}$归类的RPD值对比

$Q_{\max}$	CMA	CMAnN	CMAnR	CMAnD	CMAnC	CMAnLS
Q_1	0.60	5.24	7.41	5.30	**0.44**	13.99
Q_2	**0.30**	3.14	5.67	3.43	0.74	9.41
Q_3	**0.10**	2.22	3.84	2.58	1.00	6.65
Q_4	**0.15**	2.35	2.62	2.42	1.22	5.98
Q_5	**0.24**	2.60	2.52	2.91	1.47	6.91
Q_6	**0.26**	3.32	3.13	3.14	0.94	9.55
Q_7	**0.31**	3.43	3.54	3.55	0.91	12.46
Q_8	**0.33**	4.13	4.47	4.19	0.50	16.65
Q_9	0.51	4.79	5.67	5.26	**0.33**	19.47
Q_{10}	0.61	5.15	6.08	5.27	**0.18**	22.45
ave	**0.34**	3.64	4.50	3.80	0.77	12.35

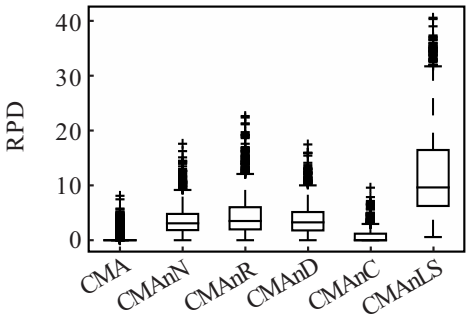

图 10　不同CMAs求解大规模算例的盒图

由图10可见,CMAnLS的性能最差,而CMA的性能最优. 由表5和表6可见,问题规模越大,启发式方法构造初始解的时间越长,CMAnN越优于CMAnR. 随机选择解码方法的CMAnD性能劣于CMA,这表明解码方法的协同使用具有优势. CMA在大多算例上的性能优于CMAnC,在峰值能耗约束最紧和最松时CMAnC略优,这表明当约束比较极端时考虑目标空间分布的操作协同机制未必能有效发挥其效用. CMAnLS在约束较紧和较松时性能更差,这表明局部增强搜索对于求解大规模问题和极端约束问题都具有重要作用. 因此,对各种变种算法的上述数值结果对比可见,通过混合初始化、协同解码、操作协同以及嵌入局部增强搜索环节,均能够有效提升算法的性能.

4 结　论

本文针对带峰值能耗约束的流水线调度问题提出了一种协同群智能算法,通过大量数值实验对比表明所设计各协同策略有助于提升算法性能,所提出算法的性能优于数学求解器和其他智能算法,尤其对于大规模问题. 算法优越的性能主要归功于如下创新性设计: 多种解码的联合及其动态调整促使生成更有效的调度; 多种初始化规则的联合促使种群具有多样性; 多种搜索操作的协同以及目标区域的自适应调整促使操作使用的有效性; 局部搜索的嵌入促使精英个体的性能进一步改善. 进一步的研究考虑如下方面: 问题层面,考虑研究带能耗约束的柔性生产调度和分布式生产调度; 指标层面,考虑绿色指标的拓展以及与经济指标的综合; 算法层面,考虑问题特性的分析与知识提取以及与强化学习的结合,设计更具效用的局部搜索方式.

参考文献(References)

[1] Energy Information Administration. Annual energy outlook 2020[R]. New York: Department of Energy, 2020: 129-141.

[2] Mouzon G, Yildirim M B, Twomey J. Operational methods for minimization of energy consumption of

manufacturing equipment[J]. International Journal of Production Research, 2007, 45(18/19): 4247-4271.

[3] 王凌, 王晶晶, 吴楚格. 绿色车间调度优化研究进展[J]. 控制与决策, 2018, 33(3): 385-391.
(Wang L, Wang J J, Wu C G. Advances in green shop scheduling and optimization[J]. Control and Decision, 2018, 33(3): 385-391.)

[4] Benini L, Bogliolo A, De Micheli G. A survey of design techniques for system-level dynamic power management[J]. IEEE Transactions on Very Large Scale Integration Systems, 2000, 8(3): 299-316.

[5] Bansal N, Kimbrel T, Pruhs K. Speed scaling to manage energy and temperature[J]. Journal of the ACM, 2007, 54(1): 1-39.

[6] Mouzon G , Yildirim M B. A framework to minimise total energy consumption and total tardiness on a single machine[J]. International Journal of Sustainable Engineering, 2008, 1(2): 105-116.

[7] Capón-García E, Bojarski A D, Espuña A, et al. Multiobjective optimization of multiproduct batch plants scheduling under environmental and economic concerns[J]. AIChE Journal, 2011, 57(10): 2766-2782.

[8] Wang J J, Wang L. A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020, 50(5): 1805-1819.

[9] Babu C A, Ashok S. Peak load management in electrolytic process industries[J]. IEEE Transactions on Power Systems, 2008, 23(2): 399-405.

[10] Luo H, Du B, Huang G Q, et al. Hybrid flow shop scheduling considering machine electricity consumption cost[J]. International Journal of Production Economics, 2013, 146(2): 423-439.

[11] Zhang H, Zhao F, Fang K, et al. Energy-conscious flow shop scheduling under time-of-use electricity tariffs[J]. CIRP Annals Manufacturing Technology, 2014, 63(1): 37-40.

[12] Zhang H, Zhao F, Sutherland J W. Scheduling of a single flow shop for minimal energy cost under real-time electricity pricing[J]. Journal of Manufacturing Science and Engineering, 2017, 139(1): 014502.

[13] Zhang H, Zhao F, Sutherland J W. Energy-efficient scheduling of multiple manufacturing factories under real-time electricity pricing[J]. CIRP Annals Manufacturing Technology, 2015, 64(1): 41-44.

[14] Nghiem T, Behl M, Pappas G J. Green scheduling: Scheduling of control systems for peak power reduction[C]. International Green Computing Conference and Workshops. Orlando, 2011: 1-8.

[15] Fang K, Uhan N A, Zhao F, et al. Flow shop scheduling with peak power consumption constraints[J]. Annals of Operations Research, 2013, 206(1): 115-145.

[16] Nagasawa K, Ikeda Y, Irohara T. Robust flow shop scheduling with random processing times for reduction of peak power consumption[J]. Simulation Modelling Practice and Theory, 2015, 59(1): 102-113.

[17] Módos I, Sucha P, Hanzálek Z. Algorithms for robust production scheduling with energy consumption limits[J]. Computers & Industrial Engineering, 2017, 112(1): 391-408.

[18] Garey M R, Johnson D S, Sethi R. The complexity of flowshop and jobshop scheduling[J]. Mathematics of Operations Research, 1976, 1(2): 117-129.

[19] Xu J, Zhou X. A class of multi-objective expected value decision-making model with birandom coefficients and its application to flow shop scheduling problem[J]. Information Sciences, 2009, 179(17): 2997-3017.

[20] Nawaz M M, Enscore E, Ham Jr I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem[J]. Omega, 1983, 11(1): 91-95.

[21] Fernandez-Viagas V, Ruiz R, Framinan J M. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation[J]. European Journal of Operational Research, 2017, 257(1): 707-721.

[22] Osman I H, Potts C N. Simulated annealing for permutation flow-shop scheduling[J]. Omega, 1989, 17(6): 551-557.

[23] Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem[J]. European Journal of Operational Research, 2007, 177(3): 2033-2049.

[24] Pan Q K, Tasgetiren M F, Liang Y C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem[J]. Computers & Industrial Engineering, 2008, 55(4): 795-816.

[25] Nguyen Q H, Ong Y S, Lim M H. A probabilistic memetic framework[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(3): 604-623.

[26] 王凌, 沈婧楠, 王圣尧, 等. 协同进化算法研究进展[J]. 控制与决策, 2015, 30(2):193-202.
(Wang L, Shen J N, Wang S Y, et al. Advances in co-evolutionary algorithms[J]. Control and Decision, 2015, 30(2): 193-202.)

[27] Deng J, Wang L, Wu C G, et al. A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop[C]. International Conference on Intelligent Computing. Lanzhou, 2016: 476-488.

[28] Wang J J, Wang L. Decoding methods for the flow shop scheduling with peak power consumption constraints[J]. International Journal of Production Research, 2019, 57(10): 3200-3218.

作者简介

王凌(1972−), 男, 教授, 博士生导师, 从事智能优化调度理论与方法等研究, E-mail: wangling@tsinghua.edu.cn;

王晶晶(1995−), 女, 博士生, 从事智能优化绿色调度的研究, E-mail: wjj18@mails.tsinghua.edu.cn.

(责任编辑：郑晓蕾)