混合交叉熵算法求解复杂零等待流水线调度问题

张梓琪[1,2], 钱 斌[2†], 胡 蓉[2]

(1. 昆明理工大学 机电工程学院, 云南 昆明 650500;

2. 昆明理工大学 信息工程与自动化学院, 云南 昆明 650500)

摘要: 针对制造行业中广泛存在的一类复杂零等待流水线调度问题, 即带序相关设置时间和释放时间的零等待流水线调度问题(NFSSP_SDSTs_RTs), 建立问题的排序模型并提出一种混合交叉熵算法(HCEA)进行求解, 优化目标为最小化总提前和延迟时间. 首先, 设计了一种基于问题性质的快速评价方法, 有效降低评价解的计算复杂度. 其次, 采用交叉熵算法学习并积累优质解的结构特征, 建立概率模型对优质解的工件块分布进行有效地估计. 通过合理的采样和更新方法, 实现对解空间中优质区域的全局搜索. 然后, 为提高算法搜索效率, 设计带两种搜索策略的快速局部搜索方法, 对全局搜索发现的优质区域进行细致且深入的搜索. 最后, 仿真实验与算法对比验证了HCEA可有效求解NFSSP_SDSTs_RTs.

关键词: 零等待; 流水线调度; 序相关设置时间; 释放时间; 局部搜索; 交叉熵

引用格式: 张梓琪, 钱斌, 胡蓉. 混合交叉熵算法求解复杂零等待流水线调度问题. 控制理论与应用, 2021, 38(12): 1919 – 1934

DOI: 10.7641/CTA.2021.00755

Hybrid cross-entropy algorithm for solving complex no-wait flow-shop scheduling problem

ZHANG Zi-qi[1,2], QIAN Bin[2†], HU Rong[2]

(1.Faculty of Mechanical and Electronic Engineering, Kunming University of Science and Technology, Kunming Yunnan 650500, China;
2. Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming Yunnan 650500, China)

Abstract: This paper proposes a hybrid cross-entropy algorithm (HCEA) and formulates a sequence-based model for solving a type of complex no-wait flow-shop scheduling problem with sequence dependent setup times and release times (NFSSP_SDSTs_RTs), which widely exists in the manufacturing industry. The criterion of the NFSSP SDSTs RTs is to minimize the total earliness and tardiness. Firstly, a speed-up evaluation method based on problem property is devised, which can effectively reduce the computational complexity of solution evaluation. Secondly, a cross entropy algorithm is used to learn and accumulate the structural characteristics of high-quality solutions, and a probability model is established to effectively estimate the distribution of job blocks in superior solutions. Then, the global search for promising regions in solution space is performed by using the reasonable sampling and updating methods. Thirdly, in order to enhance the search efficiency of HCEA, a fast local search with two search strategies is developed to execute detailed and in-depth exploitation in these promising regions found by the global exploration. Finally, simulation experiments and comparison results demonstrate that the proposed HCEA can effectively solve the NFSSP_SDSTs_RTs.

Key words: no-wait; flow shop scheduling; sequence dependent setup times; release times; local search; cross-entropy

Citation: ZHANG Ziqi, QIAN Bin, HU Rong. Hybrid cross-entropy algorithm for solving complex no-wait flow-shop scheduling problem. *Control Theory & Applications*, 2021, 38(12): 1919 – 1934

1 引言

生产调度在现代化生产制造系统中起着十分重要的作用, 尤其是在德国实施工业4.0战略、美国加速工业互联网建设和中国推进智能制造2025的国际化大背景下, 生产调度在智能制造的关键环节上显得至关重要. 流水线调度问题(flow shop scheduling problem,

收稿日期: 2020−10−30; 录用日期: 2021−02−17.

†通信作者. E-mail: bin.qian@vip.163.com; Tel.: +86 13312529481.

本文责任编委: 王凌.

国家自然科学基金项目(51665025, 61963022, 62173169)资助.

Supported by the National Natural Science Foundation of China (51665025, 61963022, 62173169).

FSSP)是制造系统中广泛存在的一类典型生产调度问题. 数学上, FSSP属于NP-hard问题. 近年来, FSSP在计算机科学和运筹优化领域被广泛关注与研究[1–3]. 零等待流水线调度问题(no-wait flow-shop scheduling problem, NFSSP)是FSSP的特例. NFSSP作为一类具有实际工程应用背景的重要生产调度问题[4–5], 要求加工过程必须连续进行, 不能出现中断或等待. NFSSP通常假设工件加工时间包含设置时间且工件无释放时间, 但实际生产制造中难以避免会频繁出现切换生产工况、更换工具和调整设备参数等情况, 这将导致存在除加工时间以外的设置时间[6–7]. 一般而言, 设置时间可以划分为两种类型[6], 即序无关的设置时间(sequence independent setup times, SISTs)和序相关的设置时间(sequence dependent setup times, SDSTs). 绝大多数情况下, 设置时间不仅由当前机器上待加工的工件决定, 还依赖于该机器上前一个加工的工件[8]. 因此, 考虑带有序相关的设置时间的情形更加具有现实意义. 此外, 由于实际生产过程中待加工的工件在进入生产车间之前存在运输过程, 即工件的开始加工时间不仅取决于当前可供加工使用的机器是否空闲, 还必须等到待加工工件的释放时间满足条件后才能开始加工, 故机器的设置时间和工件的释放时间是实际生产中不可回避的两个重要约束. 因此, 研究带有序相关的设置时间和释放时间的零等待流水线调度问题 (no-wait flow-shop scheduling problem with sequence dependent setup times and release times, NFSSP_SDSTs_RTs)具有重要的理论价值和实际意义. 为了满足实际生产过程中准时制(just in time, JIT)的生产要求[9], 本文综合考虑生产过程中的提前与延迟, 以最小化提前和延迟总代价(total earliness and tardiness, TET)为优化目标. 根据调度问题的三元组描述法[10], 优化目标为TET的NFSSP_SDSTs_RTs可归类为Fm/no-wait, $ST_{\rm sd}, r_j/\sum(E_j+T_j)$. 在计算复杂度上, 优化目标为TET的一类单机调度问题1//$\sum(E_j+T_j)$已经被证明为NP-hard问题[11], 显然, 这一类单机调度问题1//$\sum(E_j+T_j)$又可以约归为Fm/no-wait, $ST_{\rm sd}, r_j/\sum(E_j+T_j)$, 故本文研究的问题属于NP-hard. 同时, NFSSP_SDSTs_RTs也是NFSSP中较为复杂的一类问题[4–5], 许多NFSSP及其扩展问题都可以归约为 NFSSP_SDSTs_RTs. 因此, 研究优化目标为TET的NFSSP_SDSTs_RTs问题具有重要的学术意义和工程应用价值.

近年来, 针对复杂NFSSP得到了较多的关注与研究[12–23]. 在问题求解层面, NFSSP_SDSTs_RTs具有非线性和强耦合等诸多特性. 在问题的建模方面, 一般可以建立两类模型, 即数学规划模型和排序模型. Bianco等[12]首次建立了NFSSP_SDSTs_RTs的数学规划模型, 并从数学上推导得出该类问题可以等效为带准备时间的非对称旅行商问题(asymmetric travelling salesman problem with ready time, ATSP_RT). 针对数学规划模型的求解目前主要有分支定界(branch and bound, B&B)、列生成 (column generation) 和 Benders 分解(benders decomposition)等方法. 传统运筹学方法可通过商用求解器如GUROBI或CPLEX等实现问题解空间的全部遍历或部分遍历, 一般能在较短时间内给出小规模问题的最优解, 但对于大规模复杂优化问题, 这类方法在可接受时间内获取满意解的难度较大. 对于排序模型, 通常是由一系列用于确定工件在机器上开始加工时间的计算式组成, 问题的约束条件隐式包含在排序编码中, 通过解码来实现对约束的处理. 排序模型一般采用近似算法、启发式算法和智能优化算法进行求解. 近似算法如动态规划算法, 需要构造Behrman状态递推方程, 并严格证明问题最优解包含在每次迭代的状态空间中方可使用. 动态规划算法虽然理论上能保证搜索到问题的最优解, 但随着问题规模的递增, 其状态空间往往呈指数级增长, 导致动态规划算法难以实际应用. 启发式算法通常基于特定的调度规则或者问题的特性构造可行解, 譬如, Johnson算法、Palmer算法、Nawaz-Enscore-Ham算法、Gupta算法和Campbell-Dudek-Smith算法等[10]. 这类启发式算法可以在较短时间内获得可行且质量相对较高的问题解, 但并不能保证全局意义上的最优性. 智能优化算法基于计算智能机制, 结合问题的特性并通过模拟各类自然机理来实现对问题解空间的有效搜索, 往往在较短时间内就能获得复杂生产调度问题的最优解或满意解. He等[13]指出, 智能优化算法的有效性是由调度问题排序模型的解空间特性和解的特点, 以及智能优化算法自身机制共同决定. 由于离散优化问题排序模型的解空间具有“极为扁平”性, 解与解之间具有“无梯度”性, 使得智能优化算法在较短时间内仅需搜索排序模型解空间中很小的区域, 就能够在目标函数上获得明显多于传统运筹学方法的较优解, 从而能够有效地求解各类复杂调度问题.

目前, 应用智能优化算法求解复杂NFSSP已有大量的研究成果[4]. 根据所研究的问题可以划分为3种类型, 分别是带SISTs的NFSSP (NFSSP_SISTs)[14]、带 SDSTs 的 NFSSP(NFSSP_SDSTs)[15, 22]和 NFSSP_SDSTs_RTs[12, 23]. 在NFSSP_SISTs的研究方面, Allahverdi等[6–7]针对NFSSP_SISTs进行了全面且系统的综述. Ding等[14]针对优化目标为最小化最大完工时间的NFSSP_SISTs, 提出一种带禁忌机制的改进迭代贪婪(tabu mechanism improved iterated greedy, TMIIG)算法进行求解. TMIIG算法在IG算法的基础上加入禁忌重构机制, 利用启发式算法生成高质量的初始解, 同时设计基于多种邻域结构的局部搜索, 进一步提高算法的性能. 实验结果表明 TMIIG 是求解 NFSSP_

SISTs的有效算法. 在NFSSP_SDSTs的研究方面, 优化指标通常为最小化最大完工时间. Ruiz和Allahverdi[15]设计了几种启发式算法并提出一种基于Insert邻域结构的迭代局部搜索算法进行求解. Araujoa和Nagano[16]分析问题结构性质并提出一种有效的启发式算法进行求解. Jolai等[17]分别设计了基于种群的模拟退火算法、适应性帝国主义竞争算法以及二者混合的优化算法对柔性 NFSSP 进行求解. Allahverdi和Aydilek[18]提出了遗传算法、差分进化算法和模拟退火算法等一系列有效算法进行求解, 计算结果表明在相同的计算时间内所提改进模拟退火算法(ISA_2)的性能显著优于其他算法. Nagano等[19]提出一种基于进化聚类机制的混合优化算法进行求解. Samarghandi和ElMekkawy[20]设计一种矩阵编码方式对解序列进行编码, 同时提出一种基于该编码方式的的粒子群算法进行求解. Samarghandi[21]提出一种基于遗传算法(GA)的优化算法进行求解, 有效地解决了服务器端约束对最大完工时间的影响, 该方法改进了标准测试集中部分已知的最优解. 此外, 除了考虑最小化最大完工时间指标外, Arabameri和Salmasi[22]还进一步针对优化目标为最小化加权的提前和延迟时间的NFSSP_SDSTs, 设计一种基于禁忌搜索和粒子群优化的混合算法进行求解. 在NFSSP_SDSTs_RTs的研究方面, 优化指标主要考虑最小化最大完工时间. Bianco等[12]分别提出一种最优添加启发式(best adding heuristic, BAH)算法和一种最优插入启发式(best insertion heuristic, BIH)算法进行求解, 计算结果表明BIH算法的性能优于BAH算法. Franca等[23]设计一种混合遗传算法(HGA)进行求解, 并在HGA中嵌入递归局部搜索方法进一步增强算法的性能. 综上所述, 目前针对NFSSP_SDSTs_RTs的研究仍十分有限, 故对其开展建模与算法研究具有较大的理论和实际意义.

交叉熵算法(cross entropy algorithm, CEA)是一种估计复杂网络中稀有事件的有效方法[24]. CEA通过将优化问题转化为该问题对应解空间上的概率分布估计问题, 利用小概率事件信息更新概率模型参数, 采样概率模型获得新解, 实现对问题解空间的有效搜索, 引导搜索方向逐渐逼近全局最优. 近年来, CEA及其改进算法已被广泛应用于求解各类组合优化问题. Caserta等[25]针对带设置成本约束的一类背包问题, 提出了一种混合CEA进行求解, 仿真实验验证了所提算法的性能优于分支定界法. Santosa等[26]针对优化目标为最小化最大完工时间的零等待作业车间调度问题, 设计了一种混合交叉熵遗传算法(HCEGA)进行求解. HCEGA利用交叉熵算法引导全局搜索方向, 同时采用遗传算法的交叉和变异操作来丰富算法的搜索行为, 以实现对解空间不同区域的有效搜索. Wang等[27]针对加工时间不确定的炼钢连铸生产调度问题, 提出了一种串级CEA进行求解. Wang等[28]针对可再生能源发电不确定性的电力系统优化调度问题, 提出了一种多目标CEA进行求解. 因此, 本文选用CEA为基本框架, 设计求解NFSSP_SDSTs_RTs的有效算法. 本文研究的NF–SSP_SDSTs_RTs具有诸多复杂性, 不仅要考虑零等待的工艺约束和序相关设置时间与释放时间的加工约束, 同时还需要满足准时制的生产要求. 问题求解的关键难点在于可行解空间十分庞大且复杂, 在这样庞杂的解空间中发现优质解是一种极小概率的事件. 因此, 非常适合采用CEA进行求解. 目前, 尚无应用CEA求解NFSSP_SDSTs_RTs的相关研究.

本文研究NFSSP_SDSTs_RTs的建模与求解. 首先, 在问题建模方面, 建立以最小化总提前和延迟为优化目标的NFSSP_SDSTs_RTs的排序模型; 其次, 在问题求解方面, 分析问题零等待性质, 设计一种快速评价方法, 有效降低评价解的计算复杂度; 然后, 在算法设计方面, 采用交叉熵算法学习并积累优质解的结构特征, 建立概率模型对优质解中工件块分布进行合理且有效估计, 通过采样概率模型生成新种群, 实现对解空间中优质区域的全局搜索. 同时, 利用带两种搜索策略的快速局部搜索对全局发现的优质区域进行更为细致且深入的搜索. 最后, 通过大量仿真实验和算法对比分析, 验证了所提算法的有效性和鲁棒性.

2　问题描述

2.1　符号定义

本文所涉及的数学符号及定义如表1所示.

表 1　符号表

Table 1　Notations

符号	说明
π	工件加工序列$\pi=[j_1,j_2,\cdots,j_n]$;
$p_{j_i,l}$	工件j_i在机器l上的加工时间 ($p_{j_0,l}=0,\ l=1,\cdots,m,\ i=1,\cdots,n$);
sp_{j_i}	工件j_i在所有机器上的加工时间总和;
$ML_{j_i,l}$	工件j_{i-1}和工件j_i在机器l上完工时间之间的最小延迟;
C_{j_i}	工件j_i的完工时间;
L_{j_{i-1},j_i}	工件j_{i-1}和工件j_i在第1台机器上开始加工时间之间的最小延迟;
$s_{j_{i-1},j_i,l}$	工件j_{i-1}和工件j_i在机器l上的序相关设置时间;
r_{j_i}	工件j_i的释放时间($r_{j_0}=0,\ i=1,\cdots,n$);
St_{j_i}	工件j_i在第1台机器上的开始加工时间;
d_{j_i}	工件j_i的交货时间;
E_{j_i}	工件j_i在机器m上的提前时间;
T_{j_i}	工件j_i在机器m上的延迟时间;
TET(π)	工件加工序列π的总提前和延迟时间.

2.2 问题模型

NFSSP_SDSTs_RTs描述为: n个工件在m台机器上进行加工, 每个工件的加工时间、序相关设置时间和释放时间是确定的. 每台机器在同一时刻只能加工一个工件, 并且每个工件在同一时刻只能由一台机器进行加工. 为了满足零等待工艺约束, 各工件在机器上必须连续不断地进行加工, 任意时刻都不允许中断. 所有工件按照相同工序在每台机器上进行加工且无任何等待. 每台机器上当前加工工件的完工时间到下一个工件的开始加工时间, 这期间内存在依赖于工件加工顺序的序相关设置时间. 此外, 如果生产线上第一台机器已经准备就绪但待加工工件还未到达, 此时该机器只能处于空闲等待状态直到工件的释放时间满足要求为止. NFSSP_SDSTs_RTs的优化目标为最小化总提前和延迟时间. 基于以上描述, 建立如下模型:

$$ML_{j_i,l}=\begin{cases}\max\{s_{j_{i-1},j_i,1}+p_{j_i,1}-p_{j_{i-1},2},\\ \qquad s_{j_{i-1},j_i,2}\}+p_{j_i,2},\ l=2,\\ \max\{ML_{j_i,l-1}-p_{j_{i-1},l},\\ \qquad s_{j_{i-1},j_i,l}\}+p_{j_i,l},\ l=3,\cdots,m.\end{cases} \tag{1}$$

$$L_{j_{i-1},j_i}=ML_{j_i,m}+sp_{j_{i-1}}-sp_{j_i}, \tag{2}$$

$$St_{j_i}=\begin{cases}\max\{ML_{j_i,m}-sp_{j_i},r_{j_i}\},\ i=1,\\ St_{j_{i-1}}+\max\{L_{j_{i-1},j_i},r_{j_i}-St_{j_{i-1}}\},\\ \qquad\qquad i=2,\cdots,n,\end{cases} \tag{3}$$

$$C_{j_i}=St_{j_i}+sp_{j_i},\ i=1,\cdots,n, \tag{4}$$

$$E_{j_i}=\max(d_{j_i}-C_{j_i},0),\ i=1,\cdots,n, \tag{5}$$

$$T_{j_i}=\max(C_{j_i}-d_{j_i},0),\ i=1,\cdots,n, \tag{6}$$

$$\mathrm{TET}(\pi)=\sum_{i=1}^{n}(E_{j_i}+T_{j_i}). \tag{7}$$

调度目标为在所有可行调度方案集合Π中找到总提前和延迟时间最小的最优调度方案π^*, 即

$$\mathrm{TET}(\pi^*)=\min_{\pi\in\Pi}\mathrm{TET}(\pi), \tag{8}$$

其中: 式(1)为机器l上工件j_{i-1}与工件j_i完工时间之间最小延迟的计算公式, 式(2)为第1台机器上工件j_{i-1}与工件j_i开工时间之间最小延迟的计算公式, 式(3)为工件j_i在第1台机器上开工时间的计算公式, 式(4)至式(6)分别为工件j_i在机器m上的完工时间、提前时间和延迟时间的计算公式, 式(7)为优化目标, 式(8)表示在所有方案集合Π中找到使得TET最小的最优排序π^*. 图1为$n=3$和$m=3$的示例甘特图.

2.3 快速评价方法

根据NFSSP_SDSTs_RTs排序模型的零等待特征可知, 式(2)计算j_{i-1}与工件j_i在第1台机器上开工时间之间的最小延迟L_{j_{i-1},j_i}仅由工件j_{i-1}与工件j_i的加工顺序所决定, 与工件j_{i-1}与工件j_i的释放时间无关[12]. 本节基于该性质提出计算TET(π)的快速评价方法, 以有效减少计算式(7)的复杂度. 具体描述如下:

步骤 1 利用工件j_k ($k=1,2,\cdots,n$)的加工时间计算并保存对应的sp_{j_k}, 其中 $sp_{j_k}=\sum_{l=1}^{m}p_{j_k,l}$;

步骤 2 利用工件j_u与工件j_v在各机器上的加工时间和对应的序相关设置时间, 由式(1)和式(2)计算并保存工件j_u与工件j_v在第1台机器上开工时间之间的最小延迟L_{j_u,j_v}, 其中$j_u,j_v\in\{1,2,\cdots,n\}$, $j_u\neq j_v$;

步骤 3 对于给定的解序列$\pi=[j_1,j_2,\cdots,j_n]$, 计算TET(π)时, 式(3)和式(4)中涉及到的sp_{j_i}和L_{j_{i-1},j_i}直接调用步骤1和步骤2中提前计算并保存的数据.

根据上述步骤可知, 采用快速评价方法能够较大程度上降低评价解的计算复杂度, 使第2.2节式(7)中计算TET(π)的复杂度从O(nm)下降到O(n). 基于问题特性的快速评价方法有利于提高算法的搜索效率, 使得算法在相同时间内评价解的次数增加, 从而能较快探索到更多优质解所在区域.

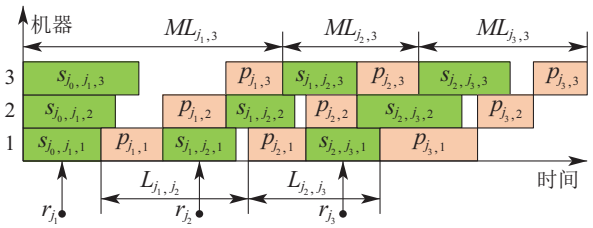


图 1 示例的甘特图

Fig. 1 The Gantt chart for the example

3 混合交叉熵算法

3.1 解的编码、工件块与相似块

智能优化算法求解基于排序模型的零等待流水线调度问题, 通常采用基于工件加工顺序的编码方式来表征问题的解[14,23]. 种群中的每个个体都对应问题解空间中的一个解, 即一个长度为n的工件序列π. 例如, $\pi=$[2, 6, 4, 8, 3, 9, 1, 5, 7]表示优先加工工件序列π首位置上的工件2, 其次加工位置2上的工件6, 最后再加工位置8上的工件7.

工件块定义为解序列中相邻的两个工件. 譬如, 对于序列$\pi=$[2, 6, 4, 8, 3, 9, 1, 5, 7], 位置1到位置8上的工件块分别为 [2, 6], [6, 4], [4, 8], [8, 3], [3, 9], [9, 1], [1, 5]和[5, 7]. 在此基础上, 相似块(similar block)定义为不同解序列中共同存在的工件块. 为便于理解, 随机生成一个$n=9$的NFSSP_SDSTs_RTs实例, 其解空间规模为9! = 362880. 通过穷举所有解, 可得最优解序列的TET目标值为1520. 图2给出部分解序列和对应的TET目标值, 其中π_1为该实例的最优解, 相应的工件块[2, 6], [8, 3], [1, 5] 和[9, 7]为相似块.

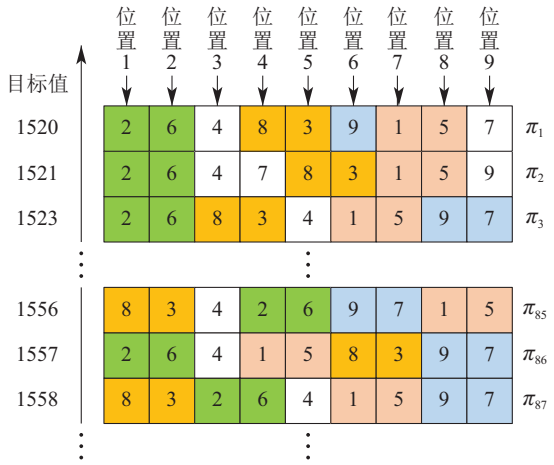


图 2　工件块示意图

Fig. 2　The job blocks for the example

3.2　交叉熵算法

交叉熵算法通过建立概率模型对解空间中优质解的结构特征进行合理估计[24]. 对于本文研究的问题, 应用交叉熵算法的关键在于概率模型的确定. 根据图2可以看出, 不少相似块在各解序列中分布的位置不同. 以相似块$[8,3]$为例, 其在π_1~π_3和π_{85}~π_{87}中出现的位置不同. 如果采用已有文献中常用的$n\times n$维概率模型, 只能积累优质解中相似块的序信息(即相邻工件前后顺序信息), 但无法准确学习并积累相似块的位置信息. 例如, π_1~π_3和π_{85}~π_{87}中相似块$[8,3]$的"出现频率"或"出现概率"只能存贮在$n\times n$维概率模型下标为$(8,3)$的元素中, 其中下标本身使得8前3后的序信息可以保留, 但无法准确区分相似块$[8,3]$在各解序列中的具体位置信息, 不能够合理地学习优质解的结构特征[29]. 对于$n\times n$维的概率模型采样生成新解或个体, 通常难以引导算法在优质解附近区域执行有效搜索. 因此, 本文设计了一种$n\times(n\times n)$维概率模型$\mathcal{P}$, 用于学习并积累优质解序列中的优良模式信息. 令$\mathcal{P}_i=(p_{i,1},p_{i,2},\cdots,p_{i,n\times n})$为概率模型$\mathcal{P}$中的第$i$行, 且$p_{i,n(u-1)+u}=0\ (u=1,\cdots,n)$, 同时$\mathcal{P}_i$满足归一化条件, 即$\sum\limits_{l=1}^{n\times n}p_{i,l}=1$. 此时对于任意个体$\pi$第$i$个位置上工件块$[j_i=u,j_{i+1}=v]$的"出现概率"可以存储于元素$p_{i,n(u-1)+v}$中. 例如, 对于图2中优质个体$\pi_1$和$\pi_2$中的相似块$[8,3]$, 则可以分别存贮在$\mathcal{P}_4$的$p_{4,66}$元素中和$\mathcal{P}_5$的$p_{5,66}$元素中, 从而能够同时保留工件块或相似块的序信息和位置信息. 当概率模型$\mathcal{P}$ 的结构确定以后, 可以利用如下的交叉熵方法以及选取的优质个体来确定算法每一代概率模型$\mathcal{P}$的取值.

基于交叉熵方法, 第2.2节式(7)中的优化目标可以等价表示为

$$S(X^*)=\gamma^*=\min_{X\in\Omega}S(X),\tag{9}$$

其中: X^*对应最优个体π^*, γ^*是X^*在目标约束S下的最优值. 假设X服从参数为μ的概率分布$f(X;\mu)$, 式(9)中对于X^*的求解则可以转化为确定最优分布$f(X;\mu)$的一个概率估计问题

$$\xi(\gamma)=P_\mu(S(X)\leqslant\gamma)=E_\mu I_{\{S(X)\leqslant\gamma\}},\tag{10}$$

其中: P_μ和E_μ为概率度量和期望, $E_\mu I_{\{S(X)\leqslant\gamma\}}$表示样本中目标优于$\gamma$的期望. 式(10)中当$\gamma$越接近最优值, 则概率$\xi(\gamma)$越小, 此时$S(X)\leqslant\gamma$为小概率事件. 如果要估计式(10)的期望, 必须进行大量的抽样. 为了减少抽样, 利用重要性抽样方法对Ω上的分布g采样得到N个样本$\{X_1,\cdots,X_N\}$, 对ξ进行估计

$$\hat{\xi}(\gamma)=\frac{1}{N}\sum_{i=1}^{N}I_{\{S(X_i)\leqslant\gamma\}}[f(X_i;\mu)/g(X_i)],\tag{11}$$

$$g_{\text{opt}}(X)=[I_{\{S(X)\leqslant\gamma\}}f(X;\mu)]/\xi.\tag{12}$$

为了用式(11)中的$\hat{\xi}$估计ξ, 要选择合适概率分布参数μ使式(12)中g的最优分布g_{opt}与分布f的差距最小. 通常采用Kullback-Leibler (K–L)距离来衡量两个概率分布之间的相似度, 最小化K–L距离等价于最小化交叉熵[22]. 令概率分布$F_{\text{p}}(X)$对$F_{\text{q}}(X)$分别是对应随机变量X的两个分布, 则分布F_{p}对分布F_{q}的交叉熵定义为F_{q}分布的自信息对F_{p}分布的期望

$$H(F_{\text{p}},F_{\text{q}})=E_{x\sim F_{\text{p}}}[-\ln F_{\text{q}}(X)]=-\int F_{\text{p}}(X)\ln F_{\text{q}}(X)\mathrm{d}x.\tag{13}$$

极小化概率分布g_{opt}与$f(X;\mu)$之间的距离可转化为确定参数μ, 使这两个分布的交叉熵最小化

$$\mu*=\arg\min_{\mu}[-\int g_{\text{opt}}(X)\ln f(X;\mu)\mathrm{d}x]=\arg\max_{\mu}E_\mu I_{\{S(X)\leqslant\gamma\}}\ln f(X;\mu).\tag{14}$$

由于样本X可以通过概率向量$\mathcal{P}_i$采样生成, 因此概率分布$f(X;\mu)$可等价表示为

$$f(X;\mathcal{P})=\prod_{i=1}^{n}\prod_{l=1}^{n\times n}p_{i,l}I_{\{x_i=l\}},\tag{15}$$

其中$p_{i,l}I_{\{x_i=l\}}$表示解空间中个体的第i个位置上选择第l个工件块的概率. 由于$\mathcal{P}_i$满足$\sum\limits_{l=1}^{n\times n}p_{i,l}=1$, 对式(14)采用拉格朗日乘子法求解

$$\max_{\mathcal{P}}\{\min_{\lambda_1,\cdots,\lambda_n}[E_{\mathcal{P}}I_{\{S(X)\leqslant\gamma\}}\ln f(X;\mathcal{P})+\sum_{i=1}^{n}\lambda_i(\sum_{l=1}^{n\times n}p_{i,l}-1)]\},\tag{16}$$

对于$l=1,2,\cdots,n\times n$, 由式(16)可得

$$E_{\mathcal{P}}I_{\{S(X)\leqslant\gamma\}}I_{\{x_i=l\}}/p_{i,l}+\lambda_i=0.\tag{17}$$

令$-\lambda_i=E_{\mathcal{P}}I_{\{S(X)\leqslant\gamma\}}$, 对于$N$个有效样本, 学习并积累工件块信息的最优概率分布$p_{i,l}$可计算如下:

$$p_{i,l}=\frac{E_{\mathcal{P}}I_{\{S(X)\leqslant\gamma\}}I_{\{x_i=l\}}}{E_{\mathcal{P}}I_{\{S(X)\leqslant\gamma\}}}=$$

$$\frac{\sum\limits_{\upsilon=1}^{N} I_{\{S(X_\upsilon)\leqslant\gamma\}} I_{\{x_{\upsilon i}=l\}}}{\sum\limits_{\upsilon=1}^{N} I_{\{S(X_\upsilon)\leqslant\gamma\}}}, \tag{18}$$

其中: $\sum\limits_{\upsilon=1}^{N} I_{\{S(X_\upsilon)\leqslant\gamma\}}$ 表示N个样本中目标值优于γ的样本个数, 示性函数$I_{\{x_{\upsilon i}=l\}}$ 表示样本υ中的第i个位置$(i=1,\cdots,n-1)$是否选择了第l个工件块. 定义$\mathcal{P}(\text{gen}-1)$为第gen$(\text{gen}=1,\cdots,\text{MaxGen})$次迭代时的概率分布. 通过式(18)所确定的概率分布, 对其采样可生成新解或个体, 具体如下:

步骤 1 选择一组随机样本$\{X_1,\cdots,X_N\}$, 即初始化种群, 并通过式(18)对概率分布$\mathcal{P}(0)$初始化;

步骤 2 通过轮盘赌采样方式, 对第gen次迭代的$\mathcal{P}(\text{gen}-1)$进行采样, 产生一定数量的可行解, 构成本次迭代的一组随机样本$\{X_1,\cdots,X_N\}$. 式(11)中样本数N即对应问题解空间中的种群规模popsize;

步骤 3 针对步骤2中所产生的N个随机样本$\{X_1,\cdots,X_N\}$的$\{S(X_1),\cdots,S(X_N)\}$进行排序, 计算样本的φ分位数, 使得$P(S(X)\leqslant\gamma_{\text{gen}})=\varphi$. 再按式(18)确定本次迭代的概率分布$\mathcal{P}(\text{gen})$;

步骤 4 为了逐代学习并积累优质解的工件块信息, 对于第$\text{gen}+1$次迭代的概率分布$\mathcal{P}(\text{gen})$, 通过学习速率r调控更新过程如下:

$$\mathcal{P}(\text{gen})=r\mathcal{P}(\text{gen})+(1-r)\mathcal{P}(\text{gen}-1). \tag{19}$$

3.3 局部搜索

为了增强算法的搜索能力, 需要对CEA全局搜索发现的优质解的邻近区域再进行精细的局部搜索. 通常, 在常用邻域搜索操作中, 交换(Interchange)和插入(Insert)操作产生的新解能最大程度地保持原解的工件块特征, 有助于进一步细致的局部搜索. 因此, 本文对算法全局获取的优质解进行基于这两种操作的局部搜索. 具体来说, 先对全局搜索得到的优质解进行Interchange操作, 扰动产生一个中间解序列; 随后再对扰动后的中间解序列进行基于Insert邻域结构的局部搜索.

3.3.1 邻域结构

定义Interchange(π,i,l)为对解序列π执行一次交换操作, 即将解序列π中位置i上的工件j_i和位置l上的工件j_l进行交换. 定义Insert(π,i,l)为对解序列π执行一次插入操作, 即当$i>l$时将位置i上的工件j_i插入到位置l上的工件j_l之前, 如果$i<l$时则将位置i上的工件j_i插入到位置l上的工件j_l之后. 由于$\pi^{n,i-1,i}=\pi^{n,i,i-1}$故$l\neq i,i-1$其中$i,l=1,\cdots,n$. 同理, 定义Insert$(\pi)$为对应解序列$\pi$的Insert邻域结构, 即满足$N_{\text{Insert}}(\pi)=\{\pi^{n,i,l}=\text{Insert}(\pi,i,l)\}, i,l=1,\cdots,n$. 令$N_{\text{Insert}}(\pi,i)$为$N_{\text{Insert}}(\pi)$的第$i$个子邻域, 则$N_{\text{Insert}}(\pi)$也可以表示为一系列子邻域的并集, 即$N_{\text{Insert}}(\pi)=\bigcup\limits_{i=1}^{n} N_{\text{Insert}}(\pi,i)$. 显然, 当且仅当$i=1$时$N_{\text{Insert}}(\pi,i)$的规模为$n-1$, 若$i>1$时则为$n-2$, 故$N_{\text{Insert}}(\pi)$邻域规模为$(n-1)^2$.

3.3.2 快速邻域搜索方法

为了提高搜索效率, 针对问题排序模型的特性, 有必要设计快速邻域搜索方法以加速局部搜索过程. 首先, 定义FindBest$N_{\text{Insert}}(\pi)$为在Insert邻域结构中搜索到最优解π_{best}的操作. 由第2.3.1节可知, $\pi^{n,i,l}$为通过Insert(π,i,l)操作后获得的$\pi=[j_1,\cdots,j_s,\cdots,j_n]$邻近区域的解. 其次, 定义$[j'_1,j'_2,\cdots,j'_s,\cdots,j'_n]$为对应解序列$\pi^{n,i,l}$的一个工件排序, 其中$s=\min(i,l)$.

当$s=2,\cdots,n-1$且$k=1,\cdots,s-1$时, 显然存在$j_k=j'_k$, 此时$E_{j'_k}+T_{j'_k}=E_{j_k}+T_{j_k}$. 因此, 由式(7)可得$\sum\limits_{k=1}^{s-1}(E_{j'_k}+T_{j'_k})=\sum\limits_{k=1}^{s-1}(E_{j_k}+T_{j_k})$, 进而可推导出以下性质:

$$\begin{aligned}&\text{TET}(\pi^{n,i,l})=\\&\sum_{k=1}^{s-1}(E_{j'_k}+T_{j'_k})+\sum_{k=s}^{n}(E_{j'_k}+T_{j'_k})=\\&\sum_{k=1}^{s-1}(E_{j_k}+T_{j_k})+\sum_{k=s}^{n}(E_{j'_k}+T_{j'_k}),\\&i,l=1,\cdots,n;\ l\neq i,i-1;\ s=\min\{i,l\}.\end{aligned} \tag{20}$$

由式(20)可知, 当执行FindBest$N_{\text{Insert}}(\pi)$操作时, 原始解π的$St_{j_{s-1}}$和$\sum\limits_{k=1}^{s-1}(E_{j_k}+T_{j_k})$ $(s-1=1,\cdots,n-2)$可直接调用第2.3节步骤1 和步骤2中提前已保存的sp_{j_i}和L_{j_{i-1},j_i}进行计算. 某种程度上能有效减少执行FindBest$N_{\text{Insert}}(\pi)$操作的计算复杂度. 换言之, 当$s>1$时, 评价原始解π的邻域解$\pi^{n,i,l}$所涉及到的$St_{j'_{s-1}}$和$\sum\limits_{k=1}^{s-1}(E_{j'_k}+T_{j'_k})$不需要再重复计算, 直接使用$\pi$的$St_{j_{s-1}}$和$\sum\limits_{k=1}^{s-1}(E_{j_k}+T_{j_k})$替换即可. 邻域解$\pi^{n,i,l}$的$E_{j'_s}+T_{j'_s}$可以通过式(3)至式(10) 由原始解$\pi$的$St_{j_{s-1}}$直接计算得到. 图3为对解序列$\pi$执行$\pi^{n,i,l}$操作的示意图, 其中, $n=10, i=4, l=7, s=\min\{i,l\}=4$. 根据文献[30]中给出的NFSSP快速邻域搜索方法, 定义SP_FindBest$N_{\text{Insert}}(\pi)$为采用快速邻域搜索方法的FindBest$N_{\text{Insert}}(\pi)$操作, 其中SP_FindBest$N_{\text{Insert}}(\pi,i)$为利用快速邻域搜索方法在子邻域$N_{\text{Insert}}(\pi,i)$中寻找到最优的邻域解$\pi^{n,i}_{\text{local_best}}$的搜索操作. 具体步骤如下:

步骤 1 计算并保存算法在全局搜索过程中找到的全局最优解π的$St_{j_{s-1}}$和$\sum\limits_{k=1}^{s-1}(E_{j_k}+T_{j_k})$, 其中$s-1=1,\cdots,n-2$;

步骤 2　令$i=1$且$\pi_{\text{best}}=\pi$, 执行快速邻域搜索$\pi^{n,i}_{\text{local_best}}=SP_\text{FindBest}N_{\text{Insert}}(\pi,i)$;

步骤 3　若目标TET$(\pi^{n,i}_{\text{local_best}})<\text{TET}(\pi_{\text{best}})$, 则令$\pi_{\text{best}}=\pi^{n,i}_{\text{local_best}}$;

步骤 4　若$i<n$, 则令$i=i+1$并跳转到步骤3;

步骤 5　输出π_{best}.

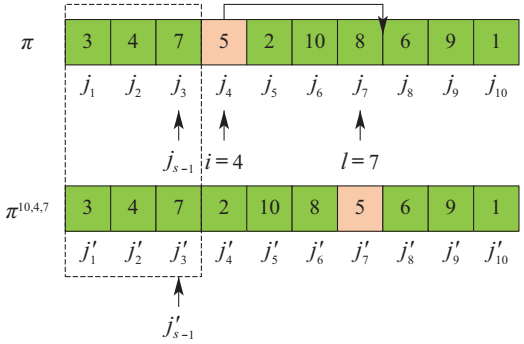


图 3　$\pi^{n,i,l}$插入操作示意图

Fig. 3　Insert operation for $\pi^{n,i,l}$

3.3.3　邻域搜索策略

为了在有限时间内探索到解空间中更多区域, 针对$SP_\text{FindBest}N_{\text{Insert}}(\pi)$设计了两种邻域搜索策略. 第 1 种策略为首次改进跳出策略, 即在子邻域$N_{\text{Insert}}(\pi,i)$搜索过程中一旦发现能够改进解π_{best}的邻域解, 则直接终止第2.3.2节搜索步骤2中的程序$SP_\text{FindBest}N_{\text{Insert}}(\pi,i)$, 输出当前最优邻域解$\pi^{n,i}_{\text{local_best}}$. 第2种策略为首次改进后变邻域搜索策略, 该策略就是将第 2.3.2 节搜索步骤 2 中使用的快速邻域搜索$SP_\text{FindBest}N_{\text{Insert}}(\pi,i)$中的$\pi$直接替换为当前搜索到的能够改进$\pi_{\text{best}}$的邻域解$\pi^{n,i}_{\text{local_best}}$, 之后再继续执行后续的邻域搜索操作. 为了方便描述, 定义$SP_\text{FindBestFisrtSkip}N_{\text{Insert}}(\pi,i)$为采用首次改进跳出策略的快速邻域搜索$SP_\text{FindBest}N_{\text{Insert}}(\pi,i)$操作, 令$\pi_{\text{temp}}=[j_1^{\text{temp}},j_2^{\text{temp}},\cdots,j_s^{\text{temp}},\cdots,j_n^{\text{temp}}]$为该方法输出的一个工件排序, 记$[j_1^{\text{best}},j_2^{\text{best}},\cdots,j_s^{\text{best}},\cdots,j_n^{\text{best}}]$为解$\pi_{\text{best}}$对应的一个工件排序, 定义$TS_SP_\text{FindBestFisrtSkip}N_{\text{Insert}}(\pi,KK)$为带有两种邻域搜索策略的快速邻域搜索方法, 具体如下:

步骤 1　计算并保存算法在全局搜索过程中找到的全局最优解π的$St_{j_{s-1}}$和$\sum_{k=1}^{s-1}(E_{j_k}+T_{j_k})$, 其中$s-1=1,\cdots,n-2$;

步 骤 2　对$s-1=1,\cdots,n-2$, 令$i=KK$, $\pi_{\text{best}}=\pi$, 满足$St_{j^{\text{best}}_{s-1}}=St_{j_{s-1}}$且$\sum_{k=1}^{s-1}(E_{j_k^{\text{best}}}+T_{j_k^{\text{best}}})=\sum_{k=1}^{s-1}(E_{j_k}+T_{j_k})$;

步骤 3　对π_{best}执行带两种邻域策略的快速邻域搜索操作, 即$\pi_{\text{temp}}=TS_SP_\text{FindBestFirstSkip}N_{\text{Insert}}(\pi_{\text{best}},i)$;

步骤 4　若目标值 TET$(\pi_{\text{temp}})<\text{TET}(\pi_{\text{best}})$, 则$\pi_{\text{best}}=\pi_{\text{temp}}$. 对于$s-1=1,\cdots,n-2$, 解序列中均存在$St_{j^{\text{best}}_{s-1}}=St_{j^{\text{temp}}_{s-1}}$和$\sum_{k=1}^{s-1}(E_{j_k^{\text{best}}}+T_{j_k^{\text{best}}})=\sum_{k=1}^{s-1}(E_{j_k^{\text{temp}}}+T_{j_k^{\text{temp}}})$;

步骤 5　若$i<n$, 则令$i=i+1$并跳转到步骤3, 否则执行步骤6;

步骤 6　输出π_{best}.

步骤2中的KK为控制邻域$N_{\text{Insert}}(\pi)$实际搜索范围的参数, 其取值为$KK\in\{1,\cdots,n\}$. 若KK设置过大, 则邻域$N_{\text{Insert}}(\pi)$的搜索范围将缩小. 此外, 当前最优解π_{best}的$St_{j^{\text{best}}_{s-1}}$和$\sum_{k=1}^{s-1}(E_{j_k^{\text{best}}}+T_{j_k^{\text{best}}})$仅在步骤 2 和步骤4中被更新, 在步骤3中执行带两种邻域策略的快速邻域搜索时可直接作为常数使用, 这样能够有效减少局部搜索过程中评价解的计算复杂度.

3.3.4　局部搜索流程

局部搜索流程如图4所示.

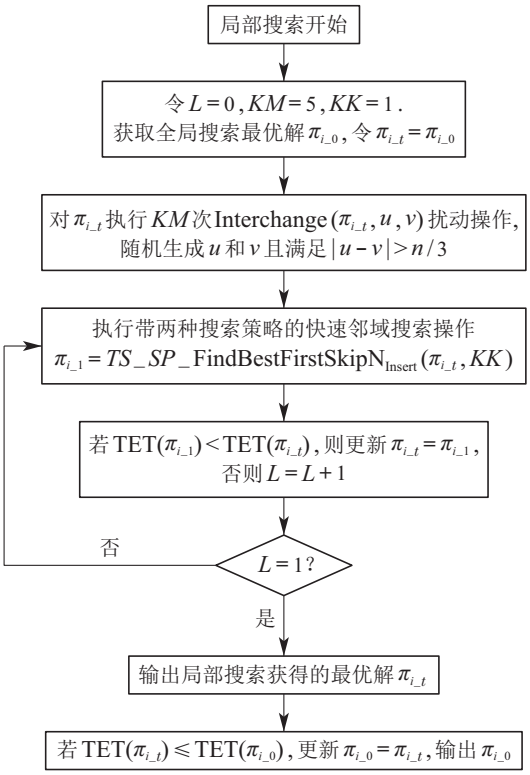


图 4　局部搜索的流程图

Fig. 4　Flow chart of local search

具体步骤如下:

步骤 1　令$L=0$, 设置参数$KM=5$, $KK=1$. π_{i_0}为全局搜索得到的最优解, 令$\pi_{i_t}=\pi_{i_0}$. 随机生成u和v且满足$|u-v|>n/3$, 对π_{i_t}执行KM次基于Interchange(π_{i_t},u,v)的扰动操作;

步骤 2　若$L\neq 1$, 则对解π_{i_t}执行邻域搜索操作

得到解π_{i_1}, 即$\pi_{i_1}=TS_SP_\text{FindBestFisrtSkip}N_{\text{Insert}}(\pi_{i_t}, KK)$; 若$L=1$则跳转至步骤4;

步骤 3 若$\text{TET}(\pi_{i_1})<\text{TET}(\pi_{i_t})$, 则$\pi_{i_t}=\pi_{i_1}$; 否则$L=L+1$. 跳转至步骤2;

步骤 4 若$\text{TET}(\pi_{i_t})\leqslant\text{TET}(\pi_{i_0})$, 则$\pi_{i_0}=\pi_{i_t}$;

步骤 5 输出更新后的解序列π_{i_0}.

3.4 算法流程

根据上述描述, HCEA的具体步骤如下:

步骤 1 由式(1)和式(2)计算并保存L_{j_{i-1},j_i}和sp_{j_i};

步骤 2 初始化种群并初始化工件块概率分布$\boldsymbol{P}(0)$, 同时设置算法的关键参数popsize, φ, r;

步骤 3 对种群进行评价并得到π_{best};

步骤 4 通过交叉熵算法对解空间执行全局搜索, 采样概率分布产生popsize个个体, 更新π_{best};

步骤 5 选择popsize $\times\ \varphi$个优质个体组成优质子种群, 更新概率分布. 对优质子种群执行第3.3.4节中的局部搜索并更新π_{best};

步骤 6 判断是否满足终止条件, 若不满足则转到步骤4, 否则终止循环.

由以上算法步骤可知, 步骤4建立概率分布学习并积累优质解的结构特征, 用于在解空间中进行全局搜索, 发现优质解所在区域. 步骤5针对全局搜索找到的优质区域进行较为细致的局部搜索, 进一步提升算法的性能, 同时使算法在全局和局部搜索上达到合理平衡. HCEA流程如图5所示.

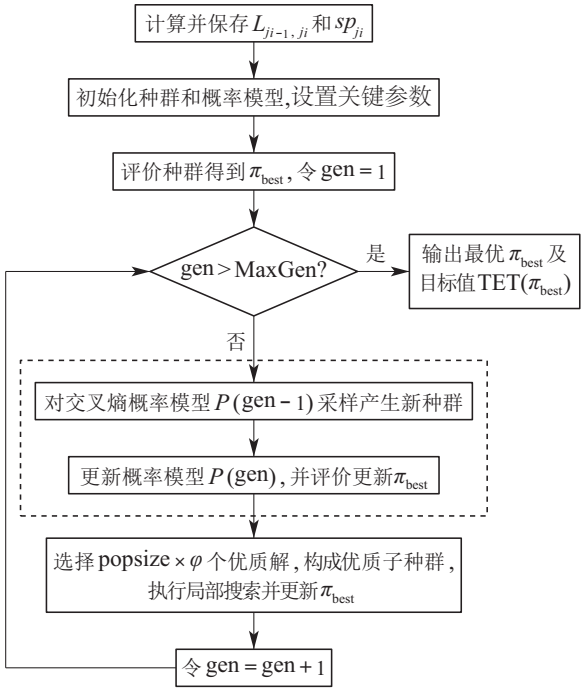


图 5 HCEA流程图

Fig. 5 Flow chart of HCEA

3.5 算法复杂度分析

HCEA关键操作的计算复杂度分析如下:

1) 快速评价方法(见第2.3节)计算L_{j_{i-1},j_i}和sp_{j_i}的复杂度分别为O[$n^2\times m$]和O[$n\times m$]. 基于快速评价方法(见第2.3节)计算TET(π)的复杂度为O[n], 相应地对种群进行快速评价的复杂度为O[popsize $\times\ n$].

2) HCEA全局搜索阶段, 对种群进行快速排序的复杂度为O[popsize $\times$ log popsize], 初始化工件块概率分布的复杂度为O[$n\times n(n-1)$], 采样概率分布产生新种群的复杂度为O[popsize $\times\ n(n-1)$], 更新概率分布的复杂度为O[$n\times n(n-1)\times$ popsize $\times\ \varphi$].

3) HCEA局部搜索阶段, 每执行一次快速邻域搜索$TS_SP_\text{FindBestFirstSkip}N_{\text{Insert}}(\pi, i)$(见第3.3.3节中步骤3)的计算复杂度为O[$n\times\log n$]. 针对邻域$N_{\text{Insert}}(\pi)$进行快速邻域搜索(见第3.3.3节中步骤2到步骤5)的复杂度为O[$(n-KK+1)\times n\times\log n$].

综上, 并考虑通常$n>m$, 第3.4节中HCEA的整体计算复杂度T_{cc}为

$$\begin{aligned}T_{\text{cc}}=&\text{O}[\text{MaxGen}\times\{n^2m+\text{popsize}\times n+\\&nm+\text{popsize}\times\log\text{popsize}+n^2(n-1)+\\&\text{popsize}\times n(n-1)+n^2(n-1)\times\text{popsize}\times\\&\varphi+n\log n+(n-KK+1)\times n\log n\}]=\\&\text{O}[\text{MaxGen}\times\{\text{popsize}\times\log\text{popsize}+\\&n^3\times\text{popsize}\times\varphi\}].\end{aligned}\tag{21}$$

根据以上分析可知, 本文第2.3节提出的快速评价方法和第3.3.2节提出的快速邻域搜索方法可以有效降低计算复杂度. 同时, 由式(21)可看出HCEA的整体计算复杂度取决于工件数n, 且最高次仅为3. 除了问题规模以外, 影响算法计算效率的因素主要还有种群规模popsize和优势个体占比φ, 实验部分将对这两个因素进行详细探讨.

4 实验设计与分析

为了验证HCEA求解NFSSP_SDSTs_RTs的有效性, 进行大量数值实验与算法对比分析. 所有测试实验均通过Embarcadero RAD Studio 10.4编程实现, 实验平台Windows 10, Inter (R) Core (TM) i7-8700 CPU 3.20 GHz处理器16 GB内存的PC机.

4.1 实验设置

由于目前还没有适合NFSSP_SDSTs_RTs的标准测试集, 根据文献[18]中解决流水线调度问题的方法, 针对工件n与机器m, 随机生成不同规模的测试问题, $n\times m$ 范围为 {20, 30, 50, 70, 100} × {5, 10, 20}. 工件加工时间$p_{j_i,l}$和序相关设置时间$s_{j_{i-1},j_i,l}$分别服从区间[1, 100]和[0, 100]的离散均匀分布, 工件释放时间r_{j_i}为区间$[0, 150n\alpha]$内随机生成的一个整数, 其中参数α控制工件的到达速率. 此外, 参数α的选择直接

影响问题解空间的特性与问题求解难度. 为了能够充分地测试算法的性能, 分别选取7种不同的α值: 0, 0.2, 0.4, 0.6, 0.8, 1.0和1.5, 一共生成105个算例进行测试. 工件的交货时间由以下步骤产生:

步骤 1　对测试问题p随机产生一个工件排序;

步骤 2　计算步骤1产生的排序中各工件j_i的完工时间$C_{p,j_i}, i = 1, 2, \cdots, n$;

步骤 3　工件j_i的交货时间d_{p,j_i}按式(22)产生:

$$d_{p,j_i} = C_{p,j_i} + \text{random}[-C_{p,j_i}, 0]. \tag{22}$$

式(22)中random$[-C_{p,j_i}, 0]$表示$[-C_{p,j_i}, 0]$区间内的随机数. 为了公平测试算法性能, 对不同规模问题, 所有算法的运行时间设置为$\rho \times n \times (m/2)$毫秒, 其中控制参数$\rho$的取值分别为2, 4和6. 为了保证测试结果的无偏性, 各算法独立重复30次实验, 因此, 每个测试算法均可得到$105 \times 3 \times 30 = 9450$个结果.

4.2　性能指标

定义$\pi(\alpha)$为在参数α水平下的一个工件排序, $\pi'(\alpha)$为各工件按照其释放时间的升序得到的排序, 即满足先到先加工原则. TET$(\pi(\alpha))$为$\pi(\alpha)$的总提前与延迟时间. $\text{TET}_{\text{avg}}(\pi(\alpha))$为算法运行30次对应TET$(\pi(\alpha))$的平均值. 为了评价各算法对所有测试问题的求解能力, 定义两个统计指标, 分别为

$$\text{ARI}(\alpha) = \left(\frac{\text{TET}(\pi'(\alpha)) - \text{TET}_{\text{avg}}(\pi(\alpha))}{\text{TET}(\pi'(\alpha))}\right) \times 100\%, \tag{23}$$

$$\text{SD}(\alpha) = \sqrt{\frac{1}{29}\sum_{i=1}^{30}\left[\text{TET}_i(\pi(\alpha)) - \text{TET}_{\text{avg}}(\pi(\alpha))\right]^2}, \tag{24}$$

其中: 式(23)中的ARI(α)为TET$(\pi'(\alpha))$的平均百分提高率, 式(24)中的SD(α)为TET$(\pi(\alpha))$的标准偏差. 令S_α为参数α 取值集合, $|S_\alpha|$为S_α中元素个数. 为评价算法的整体性能, 定义两个性能指标, 分别为:

$$\text{ARI} = \sum_{\alpha \in S\alpha} \text{ARI}(\alpha)/|S_\alpha|, \tag{25}$$

$$\text{SD} = \sum_{\alpha \in S\alpha} \text{SD}(\alpha)/|S_\alpha|. \tag{26}$$

显然, 性能指标ARI越大越好, SD越小越好.

4.3　参数设置

HCEA的3个关键参数分别为种群规模popsize, 优势个体占比φ和学习速率r. 本节选取中等规模问题$n \times m = 50 \times 10$, 其中$\alpha$取第3.1节中7种不同值, 采用实验设计方法(design of experiment, DOE)[31]探讨各参数对算法性能的影响. 算法参数选取5个水平值, 如表2所示. 选择规模为$L_{25}(5^3)$的正交实验进行参数整定. 算法在每种参数组合下独立重复运行30次, 终止条件为$4 \times 50 \times 10$毫秒. 算法在不同参数组合下各算例得到的ARI的均值作为平均响应值(average response value, ARV). 显然, ARV的值越大, 其所对应参数组合下算法的求解性能就越好.

表 2　主要参数与水平表

Table 2　Main parameters and level table

参数	水平				
	1	2	3	4	5
popsize	30	60	100	120	150
φ	0.1	0.2	0.3	0.4	0.5
r	0.05	0.1	0.2	0.3	0.4

正交表和参数响应值如表3所示, 根据正交表统计得到各参数的平均响应值如表4所示, 各参数对算法性能影响的响应趋势如图6所示. 由表4及图6可以发现, 不同的参数组合会对算法性能产生较大的影响. 根据以上实验结果可知, HCEA的最佳参数设置为popsize $= 120$, $\varphi = 0.2$, $r = 0.2$, 该参数也将应用于后续的算法的性能测试与比较中.

表 3　参数设置的正交表

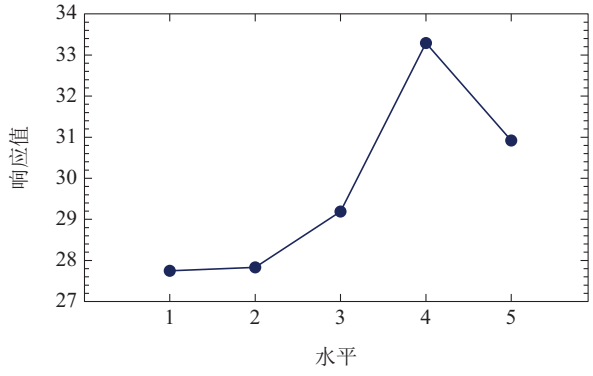
Table 3　Orthogonal table of parameter settings

参数	参数水平			ARV	SD
	popsize	φ	r		
1	1	1	1	29.0134	2.2234
2	1	2	2	31.5123	2.1034
3	1	3	3	25.3485	2.2352
4	1	4	4	23.2318	2.6237
5	1	5	5	29.6342	2.2385
6	2	1	2	28.9345	2.2334
7	2	2	3	28.7128	2.1982
8	2	3	4	25.8734	2.2251
9	2	4	5	29.8531	2.2375
10	2	5	1	25.7848	2.3215
11	3	1	3	29.7572	1.9493
12	3	2	4	31.8126	2.1945
13	3	3	5	29.8934	2.3219
14	3	4	1	25.7432	2.2485
15	3	5	2	28.7549	2.2892
16	4	1	4	34.7635	2.2384
17	4	2	5	32.7432	2.0679
18	4	3	1	36.1892	2.2846
19	4	4	2	32.7451	2.2023
20	4	5	3	30.0242	2.1541
21	5	1	5	27.6738	2.2743
22	5	2	1	33.1782	2.1385
23	5	3	2	34.9458	2.0679
24	5	4	3	26.8452	2.2135
25	5	5	4	31.9547	2.2892

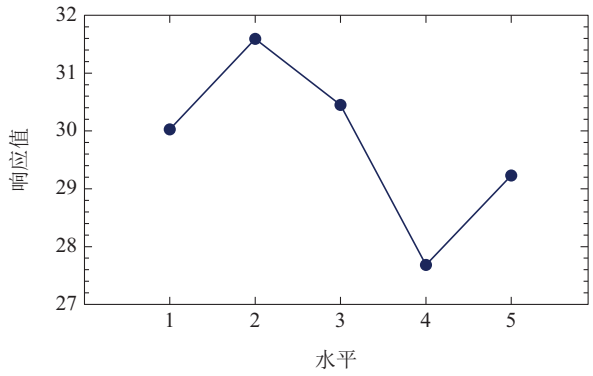
表 4 各参数不同水平下的平均响应值和等级表

Table 4 Average response value and level table at different levels of each parameter

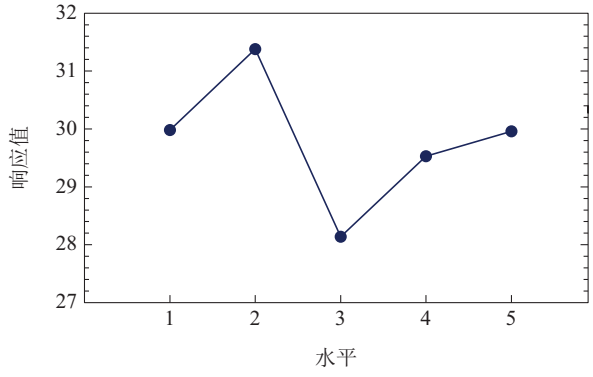
水平	popsize	φ	r
1	27.7480	30.0285	29.9818
2	27.8317	31.5918	31.3785
3	29.1923	30.4501	28.1376
4	33.2930	27.6837	29.5272
5	30.9195	29.2306	29.9595
极差	5.5450	3.9081	3.2409
等级	1	2	3



(a) 种群规模popsize对算法性能的影响



(b) 优势个体占比φ对算法性能的影响



(c) 学习速率r对算法性能的影响

图 6 参数响应图

Fig. 6 Response diagram of parameters

4.4 仿真结果与比较

为了验证第3.3节所设计的局部搜索方法的有效性, 本节首先针对第3.3.2节提出的快速邻域搜索方法和第3.3.3节提出的邻域搜索策略进行探究. 对HCEA及其6种不同的局部搜索变体进行比较, 具体如下:

1) HCEA$_{\rm V1}$: 将HCEA的带两种策略的快速搜索方法$TS_SP_{\rm FindBestFirstSkip}N_{\rm Insert}(\pi, KK)$ ($KK=1$)中的快速邻域搜索方法(见第3.3.2节)、首次改进跳出策略和改进后变邻域搜索策略(见第3.3.3节)移除, 局部搜索中仅保留快速评价方法(见第2.3节).

2) HCEA$_{\rm V2}$: 将HCEA的带两种策略的快速搜索方法$TS_SP_{\rm FindBestFirstSkip}N_{\rm Insert}(\pi, KK)$ ($KK=1$)中的快速评价方法(见第2.3节)、首次改进跳出策略和改进后变邻域搜索策略(见第3.3.3节)移除, 局部搜索中仅保留快速邻域搜索方法(见第3.3.2节).

3) HCEA$_{\rm V3}$: 将HCEA的带两种策略的快速搜索方法$TS_SP_{\rm FindBestFirstSkip}N_{\rm Insert}(\pi, KK)$ ($KK=1$)中的首次改进跳出策略和改进后变邻域搜索策略(见第3.3.3节)移除, 局部搜索中保留快速评价方法(见第2.3节)和快速邻域搜索方法(见第3.3.2节).

4) HCEA$_{\rm V4}$: 将HCEA的带两种策略的快速搜索方法$TS_SP_{\rm FindBestFirstSkip}N_{\rm Insert}(\pi, KK)$ ($KK=1$)中的改进后变邻域搜索策略(见第3.3.3节)移除, 局部搜索中保留快速评价方法(见第2.3节), 首次改进跳出策略(见第3.3.3节)和快速邻域搜索方法(见第3.3.2节).

5) HCEA$_{\rm V5}$: 将HCEA的带两种策略的快速搜索方法$TS_SP_{\rm FindBestFirstSkip}N_{\rm Insert}(\pi, KK)$中$KK$设置为$\lceil 0.6\times n\rceil$, 其中$\lceil\delta\rceil$表示对$\delta$向上取整.

6) HCEA$_{\rm V6}$: 将HCEA的带两种策略的快速搜索方法$TS_SP_{\rm FindBestFirstSkip}N_{\rm Insert}(\pi, KK)$中$KK$设置为$\lceil 0.3\times n\rceil$, 其中$\lceil\delta\rceil$表示对$\delta$向上取整.

对于不同规模问题, HCEA与6种变体算法在运行时间为$4\times n\times m$毫秒下的统计结果如表5所示. 图7为HCEA与6种变体分别在不同工件数n下的柱状统计图. 根据第4.2节可知, ARI指标的值越大越好, 故首先从图7中可明显看出HCEA$_{\rm V3}$得到的ARI值均比HCEA$_{\rm V1}$和HCEA$_{\rm V2}$的好, 验证了本文所设计的快速评价方法与快速邻域搜索方法能够有效降低计算复杂度, 使得算法在相同时间内能够评价更多的解. 其次, HCEA$_{\rm V4}$得到的ARI值比HCEA$_{\rm V3}$的好, 说明首次改进跳出策略能够对子邻域$N_{\rm Insert}(\pi, i)$进行有效地搜索, 在相对有限的时间内能搜索到更多区域. 此外, HCEA$_{\rm V6}$得到的ARI值比HCEA$_{\rm V5}$的好, 表明参数KK直接影响局部搜索的性能, 若其值设置过大, 则邻域$N_{\rm Insert}(\pi)$的实际搜索范围将缩小, 进而无法对子邻域$N_{\rm Insert}(\pi, i)$进行全面且深入的搜索.

为了验证HCEA求解NFSSP_SDSTs_RTs的有效性, 本节将HCEA与重要国际期刊中的3种算法进行性能比较, 即文献[32]中的有效EDA (effective EDA,

EEDA)、文献[33]中的迭代贪婪(iterated greedy, IG)算法以及文献[14]中的基于禁忌机制的增强迭代贪婪(TMIIG) 算法. EEDA[32]利用二维矩阵学习工件块信息, 建立概率模型对解空间中优质解分布进行合理估计, 通过调节学习速率更新概率模型, 采样概率模型产生新种群, 从宏观角度引导搜索方向, 目前已被广泛应用于求解各类车间调度问题. IG算法[33]融合启发式算法的高效性和模拟退火算法的优越性, 是近年来求解各类流水车间调度问题公认的有效算法, 其在求解基于排序模型的FSSP_SDSTs方面相比其他智能优化算法有明显优势. TMIIG算法[14]是对经典IG算法的改进, 采用改进NEH启发式方法产生高质量的初始种群, 并通过基于多种邻域结构的变邻域搜索方法对局部区域进行细致搜索, 是近年来求解NFSSP的有效算法. 其中, EEDA的参数设置与文献[32]保持一致, 具体取值为: 种群规模为popsize = 150, 优势个体数为$\eta = 10$, 学习速率为$\alpha = 0.1$; IG算法的参数设置与文献[33]保持一致, 具体取值为: 温度常数为$T = 0.5$, 移出工件数为$d = 4$; TMIIG算法的参数设置与文献[14]保持一致, 具体取值为: 温度常数为$T_0 = 0.6$, 移出工件数为$d = 6$, 禁忌表长度为$ML = 1$. 对于不同规模问题, 各算法在运行时间参数$\rho = 2, 4, 6$下的统计结果如表6–8所示. 为能清楚展现算法的求解性能, 表6–8中每个问题对应的最优结果用粗体表示.

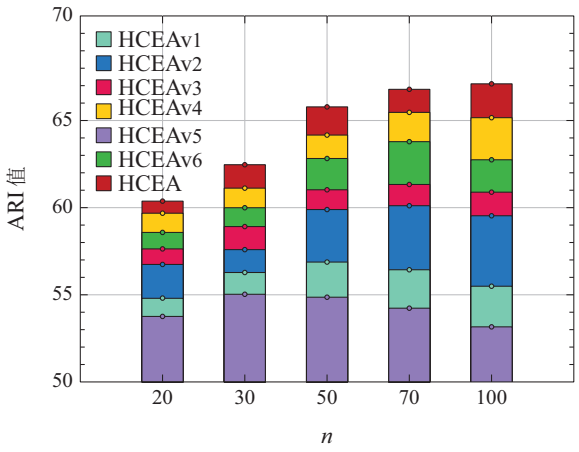


图 7 HCEA与6种变体在不同工件数n下的柱状图

Fig. 7 Histogram of HCEA and six variants under different n

表 5 HCEA与6种变体比较结果

Table 5 Comparison results of HCEA and its six variants

$n \times m$	$HCEA_{V1}$		$HCEA_{V2}$		$HCEA_{V3}$		$HCEA_{V4}$		$HCEA_{V5}$		$HCEA_{V6}$		HCEA	
	ARI	SD	ARI	SD	ARI	SD	ARI	SD	ARI	SD	ARI	SD	ARI	SD
20 × 5	59.52	1.96	61.34	1.64	62.12	1.58	63.64	1.46	58.84	1.79	62.76	0.95	**64.22**	**0.86**
20 × 10	53.47	1.73	55.35	1.53	56.23	1.46	57.81	1.31	51.68	1.58	56.83	0.86	**58.65**	**0.72**
20 × 20	50.71	1.87	52.57	1.79	53.47	1.62	56.22	1.52	50.22	1.74	54.91	0.85	**56.76**	**0.81**
30 × 5	58.53	1.85	59.22	1.65	60.13	1.52	63.76	1.45	57.11	1.68	61.64	0.92	**64.87**	**0.73**
30 × 10	54.84	1.96	56.31	1.71	57.81	1.54	58.74	1.43	53.24	1.92	58.65	0.97	**59.83**	**0.76**
30 × 20	54.57	1.93	56.15	1.77	57.54	1.58	59.28	1.55	54.03	1.84	58.27	0.84	**60.92**	**0.67**
50 × 5	57.42	1.76	60.76	1.53	61.64	1.35	65.85	1.23	55.18	1.68	63.43	0.81	**67.47**	**0.68**
50,10	56.83	1.68	59.03	1.54	60.65	1.41	63.34	1.38	54.57	1.63	62.87	0.82	**64.95**	**0.76**
50 × 20	55.39	1.77	58.46	1.61	59.22	1.56	61.29	1.49	54.15	1.82	60.32	0.76	**62.68**	**0.58**
70 × 5	57.76	1.72	60.32	1.57	61.15	1.44	65.24	1.31	55.32	1.91	63.59	0.83	**66.83**	**0.65**
70 × 10	56.88	2.19	60.89	1.92	61.57	1.83	64.33	1.61	54.52	2.27	63.07	1.04	**65.52**	**0.72**
70 × 20	53.75	1.84	57.68	1.65	59.65	1.47	64.61	1.31	52.26	2.13	62.72	0.78	**65.61**	**0.67**
100 × 10	57.46	1.75	62.73	1.52	64.92	1.48	68.62	1.24	55.54	1.85	65.83	1.19	**70.34**	**1.03**
100 × 20	59.72	2.33	64.71	2.18	65.73	1.96	69.31	1.75	57.37	2.45	66.28	1.17	**71.77**	**0.91**
100 × 40	48.52	2.69	49.81	2.29	50.46	2.23	55.38	2.18	46.12	2.61	54.31	1.26	**56.76**	**1.12**
平均值	55.69	1.94	58.36	1.73	59.49	1.60	62.49	1.48	54.01	1.93	61.03	0.94	**63.81**	**0.78**

由表6–8可知, HCEA求解不同规模问题的统计结果明显优于其他3种比较算法, 这表明加入快速邻域搜索后算法的性能得以进一步提升. 对于绝大部分问题, HCEA在$\rho = 2$时获得的结果均比EEDA, IG和TMIIG算法在$\rho = 4$和$\rho = 6$时获得的结果都要好. 对于不同规模的问题, HCEA得到的方差SD值也较小, 这说明了HCEA具有良好的求解性能.

为了进一步分析HCEA与其他算法的性能差异是否显著, 采用Tukey's HSD方法对表6–8中的4种算法分别在$\rho = 2, 4, 6$三种取值下求解各种规模的问题所得到的ARI值进行方差分析(ANOVA). 图8为4种算法在不同运行时间下的均值及95%置信度下Tukey's HSD检验的置信区间图. 由图8可知, 在不同运行时间下HCEA得到的ARI值与其他3种对比算法所得到的ARI值相比存在显著差异, 具有统计学意义. 此外, 从图8中还可看出, 随着运行时间的增加HCEA得到的

ARI均值无明显的变化, 即HCEA在3种不同运行时间下的置信区间明显存在重叠, 表明HCEA性能稳定并且能够在较短的时间内获得相对较优的结果. 图9为4种算法和3种运行时间交互作用对ARI值的交互作用图. 图中的折线无交叉, 表明交互作用不显著.

表 6 算法比较结果($\rho = 2$)

Table 6 Comparison results of algorithms ($\rho = 2$)

$n \times m$	EEDA		IG		TMIIG		HCEA	
	ARI	SD	ARI	SD	ARI	SD	ARI	SD
20 × 5	57.11	1.46	56.34	0.85	57.12	1.48	**64.34**	**0.81**
20 × 10	53.36	1.18	52.56	0.83	54.03	1.53	**58.35**	**0.79**
20 × 20	52.25	0.91	51.54	**0.84**	53.23	1.25	**56.84**	0.86
30 × 5	60.27	0.97	58.62	0.71	61.64	1.52	**64.92**	**0.68**
30 × 10	55.28	0.92	53.51	0.86	56.22	1.66	**59.95**	**0.73**
30 × 20	56.63	0.96	56.21	0.83	57.36	1.54	**60.28**	**0.76**
50 × 5	63.48	0.81	62.16	0.74	63.45	1.27	**68.12**	**0.67**
50 × 10	61.97	0.75	61.43	0.82	62.17	1.39	**65.13**	**0.76**
50 × 20	57.15	0.89	57.03	0.77	57.19	1.41	**62.91**	**0.66**
70 × 5	61.86	0.62	61.87	0.87	62.35	1.22	**67.03**	**0.71**
70 × 10	62.22	0.87	61.58	0.74	63.43	1.30	**65.41**	**0.68**
70 × 20	60.73	0.72	59.86	0.87	61.24	1.39	**65.97**	**0.74**
100 × 10	62.52	1.62	61.23	**0.93**	63.43	1.85	**70.86**	0.98
100 × 20	65.43	1.37	64.12	**0.85**	66.31	1.27	**71.92**	0.91
100 × 40	51.22	0.96	50.45	**0.94**	52.72	1.48	**58.23**	1.07
平均值	58.77	1.00	57.90	0.83	59.46	1.44	**64.02**	**0.79**

表 7 算法比较结果($\rho = 4$)

Table 7 Comparison results of algorithms ($\rho = 4$)

$n \times m$	EEDA		IG		TMIIG		HCEA	
	ARI	SD	ARI	SD	ARI	SD	ARI	SD
20 × 5	57.56	1.38	56.92	0.83	57.84	1.41	**64.64**	**0.78**
20 × 10	53.87	1.07	52.83	0.81	54.55	1.38	**58.61**	**0.77**
20 × 20	52.69	0.94	51.93	**0.82**	53.75	1.16	**57.13**	0.83
30 × 5	60.88	0.95	59.04	0.69	61.93	1.47	**65.19**	**0.64**
30 × 10	55.73	1.07	53.97	0.91	56.53	1.62	**59.93**	**0.71**
30 × 20	56.91	0.92	56.43	0.82	57.84	1.51	**61.19**	**0.73**
50 × 5	63.71	0.79	62.56	0.72	63.67	1.23	**68.23**	**0.64**
50 × 10	62.14	0.72	61.79	0.81	62.31	1.33	**65.25**	**0.75**
50 × 20	57.38	0.83	57.32	0.74	57.54	1.37	**63.06**	**0.63**
70 × 5	62.25	0.64	62.33	0.84	62.82	1.23	**67.21**	**0.72**
70 × 10	62.54	0.82	61.91	0.79	64.02	1.27	**65.63**	**0.63**
70 × 20	60.94	0.71	60.32	0.83	61.56	1.36	**66.17**	**0.73**
100 × 10	62.91	1.55	61.74	**0.92**	63.85	1.79	**71.19**	0.94
100 × 20	65.74	1.32	64.61	**0.79**	66.75	1.28	**72.13**	0.87
100 × 40	51.34	0.98	50.71	**0.91**	52.91	1.45	**58.64**	1.03
平均值	59.11	0.98	58.29	0.82	59.86	1.39	**64.28**	**0.76**

表 8 算法比较结果($\rho = 6$)

Table 8 Comparison results of algorithms ($\rho = 6$)

$n \times m$	EEDA		IG		TMIIG		HCEA	
	ARI	*SD*	ARI	SD	ARI	SD	ARI	SD
20 × 5	58.17	1.35	57.54	0.85	58.32	1.39	**64.85**	**0.79**
20 × 10	54.23	1.03	53.26	0.79	55.11	1.27	**58.87**	**0.75**
20 × 20	53.12	0.87	52.37	**0.82**	54.34	1.13	**57.32**	0.85
30 × 5	61.46	0.93	59.41	0.68	62.58	1.51	**65.32**	**0.62**
30 × 10	55.98	0.94	54.38	0.89	57.01	1.64	**60.25**	**0.72**
30 × 20	57.15	0.86	56.94	0.85	58.11	1.56	**61.32**	**0.74**
50 × 5	64.19	0.76	63.22	0.73	64.13	1.21	**68.41**	**0.61**
50 × 10	62.46	0.76	62.18	0.79	62.87	1.27	**65.34**	**0.72**
50 × 20	57.92	0.85	57.84	0.78	58.12	1.35	**63.15**	**0.64**
70 × 5	62.81	0.61	62.85	0.83	63.26	1.24	**67.32**	**0.71**
70 × 10	62.86	0.85	62.27	0.72	64.23	1.29	**65.81**	**0.61**
70 × 20	61.22	0.69	61.03	0.89	61.94	1.32	**66.36**	**0.68**
100 × 10	63.27	1.35	62.11	**0.86**	64.32	1.72	**71.38**	0.91
100 × 20	66.32	0.72	65.23	**0.83**	67.13	1.23	**72.32**	0.85
100 × 40	51.79	0.95	51.04	**0.92**	53.15	1.42	**58.87**	1.02
平均值	59.53	0.90	58.78	0.82	60.31	1.37	**64.46**	**0.75**

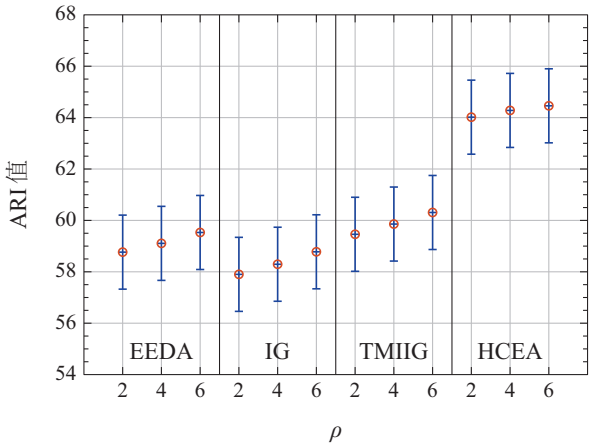


图 8 各算法在不同时间参数下的均值及95%置信度下Tukey's HSD检验的置信区间图

Fig. 8 Means plot and 95% Tukey's HSD confidence intervals for the interaction between the algorithms and the time factors

为了分析算法的收敛特性, HCEA和3种对比算法在$\rho = 2$时求解70 × 20规模问题的收敛曲线如图10所示. 从图10中可明显看出, HCEA能够快速收敛到较优解, 再次验证了第3.3节中提出的快速邻域搜索的有效性. 在比较算法中, IG和TMIIG虽然利用启发式方法产生高质量的初始解, 由于算法并未考虑排序模型解的工件块分布(见第3.1节), 在搜索过程中"移除"和"重构"操作破坏了优质解的结构特征, 使得算法仅在初始解附近反复探索, 难以搜索到全局意义上的最优解[34]. EEDA基于二维概率模型从宏观角度引导全

局搜索, 可一定程度上避免IG和TMIIG 中存在的优质解模式破坏的问题, 但二维概率模型不能准确保留工件块在解序列中的具体位置信息. 同时EEDA的局部搜索邻域数量有限且搜索方式单一, 导致算法实际搜索深度有限. HCEA利用交叉熵算法更加合理地学习并积累优质解的工件块信息, 同时采用带两种搜索策略的快速局部搜索, 有利于引导算法在多个子邻域中进行更为深入且高效地搜索, 丰富搜索行为的同时加速搜索过程, 进而能发现存在于复杂解空间中的优质解. 因此, HCEA能在上述实验中表现出更好的性能, 是一种求解NFSSP_SDSTs_RTs的有效且鲁棒的算法.

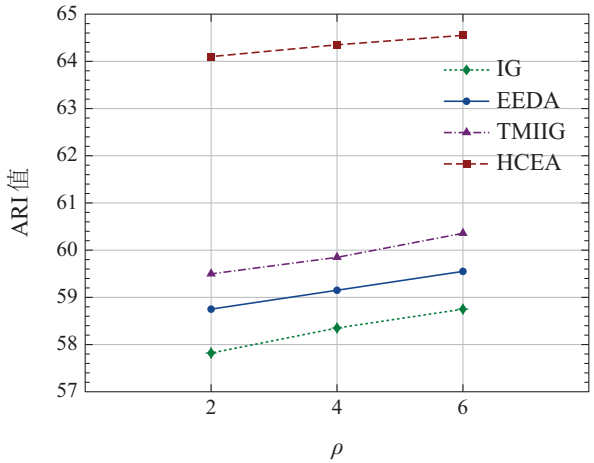


图 9　各算法在不同时间参数下的交互效应图

Fig. 9　Interaction plot between algorithms and time factors

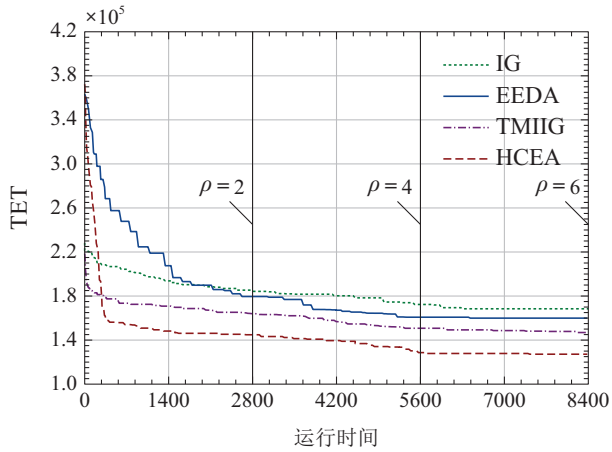


图 10　收敛曲线图

Fig. 10　Convergence curves

4.5　案例分析

钢铁制造业是发展国民经济与国防基建的支柱性产业, 也是资源高度集中且能源密集型工业. 综合考虑技术指标与工艺要求, 钢材的生产从开始加工到最终完工期间需要连续不断地进行处理[18]. 例如, 进入轧钢机压轧的连铸坯或钢坯其温度必须预先加热到特定范围. 如果由于某种原因在压轧之前出现了较长时间的等待, 则钢坯的温度将急剧下降. 一旦钢坯温度下降到压轧的临界温度以下时, 就需要再次对钢坯预热. 这样反复加热将会消耗大量能源, 造成不必要的资源浪费[4]. 钢铁厂中经过炉外精炼处理得到的钢水仅是钢材生产过程中的中间产物, 还需要再经过铸块(ingots)、脱模 (unmolding)、加热 (reheating)、初轧(rougher rolling)和精轧(finish rolling)等多道关键性工序后才能够成为合格的钢材. 铸块即液态钢水浇注到模具中, 冷却得到特定的外形和厚度的钢坯. 脱模指在传送带上将钢坯从模具内取出的操作. 高炉加热指将钢坯加热至800度以上(达到金属的相变温度), 随后快速降温冷却以提升钢坯硬度, 然后再次加热至略低于相变温度(回火操作), 以降低钢坯硬度并提高韧性, 为压轧作准备. 初轧过程指在轧机上把钢坯初步轧制成板坯、方坯或异型坯的过程. 精轧过程则是将初轧后的坯件经过辊道送入精轧机, 最终轧制成满足用户要求的型材. 图11为钢铁厂中钢坯生产过程的工艺流程.

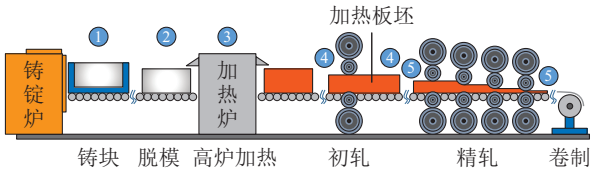


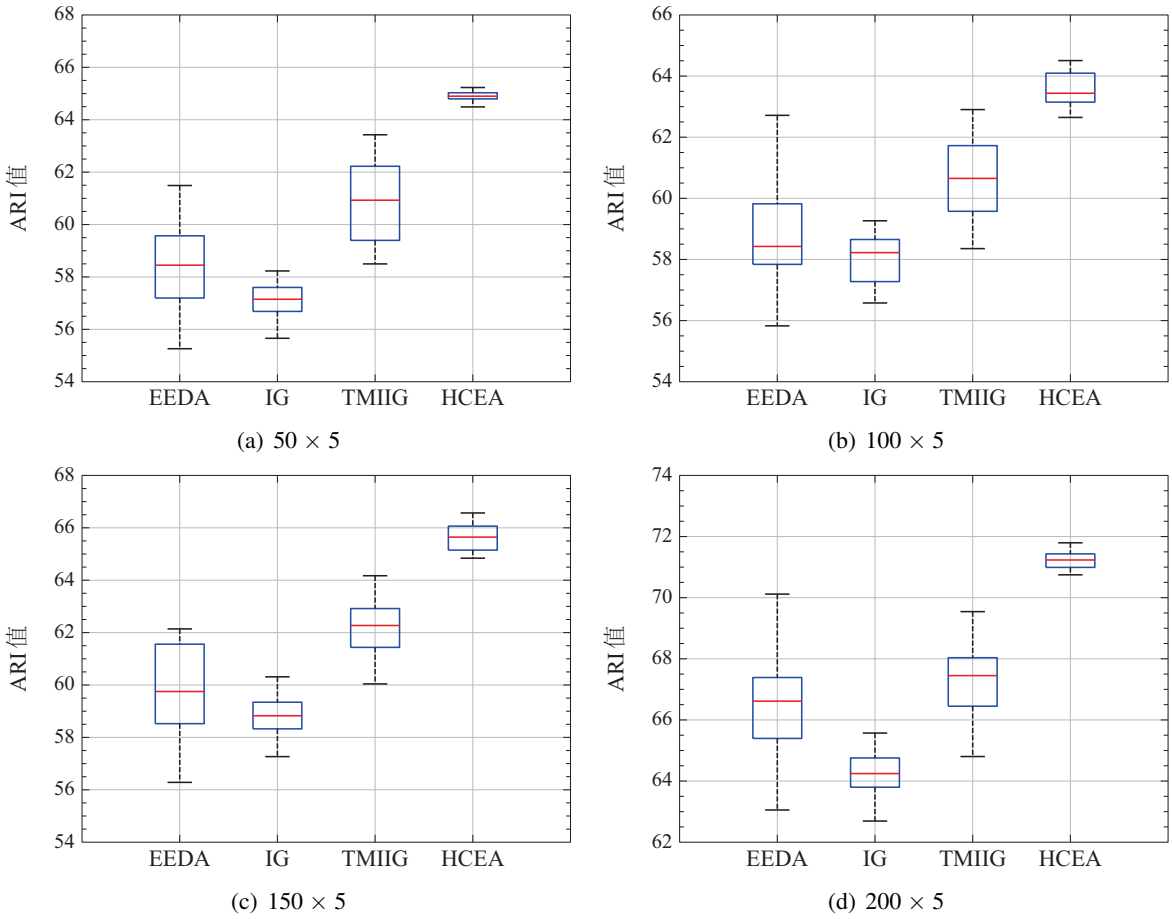
图 11　钢板加工工艺流程

Fig. 11　The process flow chart of steel slab

为了满足实际的生产要求, 需要考虑如下因素: 1) 钢坯的加工工艺路线相同且每道工序在一台加工设备上进行处理; 2) 钢坯的加工工序之间满足零等待约束[4–18]; 3) 由于加工件的更换, 加工设备需要一定的设置时间进行更换夹具或装卸工件等必要操作, 该设置时间取决于加工件的特征[6–7]; 4) 加工件通过中间传送装置运输到加工设备上需要一定的时间, 即工件存在释放时间[8]; 5) 钢铁厂根据不同客户的订单要求进行生产, 为了提供优质服务, 在生产过程中需要满足准时制生产要求[9]. 因此, 钢材加工成形的调度问题可建模为NFSSP_SDSTs_RTs.

本节在第4.1–4.4节的基础上, 采用云南某钢铁公司实际生产过程的数据, 经过预处理后进一步对HCEA的性能进行验证. 目前该厂车间内的生产调度方案是由公司里有经验的调度员按照常规的先到先加工原则进行计划排程, 即根据工件的释放时间从小到大顺序依次进行加工处理. 为了按期生产客户订单所需要的钢材, 该厂车间内现有一条生产流水线, 正常运转的设备共有5台, 即机器数$m = 5$. 通过对客户订单汇总整理, 需要加工的坯件数量n的数值分别从10到200. 针对该厂的实际生产需求, 分别使用第4.4 节中的 4 种算法: EEDA[32]、IG[33]、TMIIG[14] 和

HCEA进行求解, 各算法分别独立重复30次, 运行时间设置为$4 \times n \times m$ 毫秒, 求解结果如表9所示. 由表9可看出, 与IG相比, TMIIG和EEDA是非常具有竞争潜力的算法, HCEA在所有实际问题上的平均表现都明显优于EEDA、IG和TMIIG这3种算法, HCEA得到的平均值优于其他算法的平均值, 再次验证了HCEA具有良好的求解性能. 为了更加直观的进行比较, 4种算法的ARI值的箱线图如图12所示. 因为篇幅所限, 仅列出4个实例对应的结果. 箱线图越窄说明方差越小, 算法稳定性越高. 在绝大多数实例中, HCEA对应的箱线图的宽度最窄, 表明鲁棒性好. HCEA的ARI值中线优于其他3种算法, 说明该算法得到的解质量较好, HCEA能在相对有限的时间内获得较为满意的调度方案. 图13和图14分别给出了HCEA解决实际钢铁厂中钢材生产调度问题Case 10_5和Case 20_5的甘特图. 图13和图14中Case 10_5和Case 20_5 最优调度的最大完工时间分别为15143和23341, 订单的总提前和延迟时间分别为11501和32874.



(a) 50 × 5 (b) 100 × 5

(c) 150 × 5 (d) 200 × 5

图 12 HCEA与3种对比算法的箱线图

Fig. 12 Box plot of HCEA and three compared algorithms

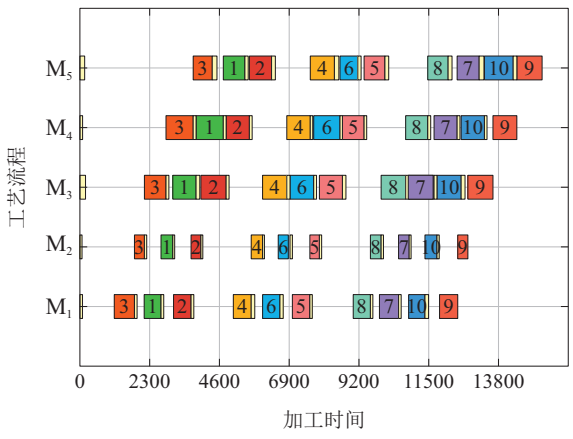


图 13 Case 10_5调度方案的甘特图

Fig. 13 The Gantt chart for Case 10_5

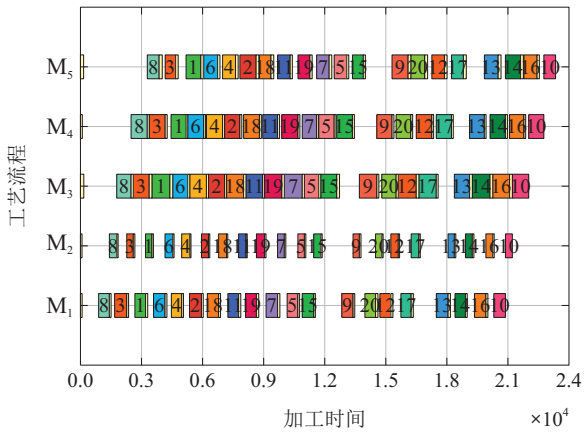


图 14 Case 20_5调度方案的甘特图

Fig. 14 The Gantt chart for Case 20_5

表 9 实例仿真结果

Table 9 Comparison results of algorithms for the real world instances

$n \times m$	EEDA		IG		TMIIG		HCEA	
	ARI	SD	ARI	SD	ARI	SD	ARI	SD
10×5	56.45	1.25	55.69	0.94	58.43	1.08	**62.33**	**0.85**
20×5	58.41	1.13	57.53	0.89	60.08	0.95	**65.49**	**0.87**
30×5	59.38	0.93	58.61	0.72	62.13	0.88	**69.13**	**0.64**
40×5	58.13	1.04	56.87	0.81	61.07	0.92	**67.38**	**0.72**
50×5	58.45	0.85	57.12	0.72	60.93	0.79	**64.89**	**0.63**
60×5	64.34	0.92	63.67	0.88	66.75	0.86	**69.35**	**0.67**
70×5	56.13	0.81	54.86	0.61	57.58	0.68	**63.74**	**0.55**
80×5	61.12	1.19	59.86	0.75	62.42	0.84	**65.69**	**0.53**
90×5	60.60	0.72	58.91	0.58	61.81	0.67	**64.26**	**0.49**
100×5	58.78	0.81	58.03	0.76	60.65	0.74	**63.53**	**0.67**
110×5	63.25	1.41	62.33	1.11	64.61	1.23	**66.88**	**0.92**
120×5	68.29	1.68	67.89	1.39	69.38	1.56	**71.39**	**1.01**
130×5	60.28	1.13	59.61	0.86	62.54	0.95	**65.14**	**0.72**
140×5	58.96	1.48	57.18	1.18	60.18	1.29	**63.55**	**1.02**
150×5	59.75	1.56	58.83	1.23	62.27	1.41	**65.64**	**0.96**
160×5	62.84	1.25	61.64	1.14	64.78	1.08	**67.67**	**0.99**
170×5	63.44	1.66	62.55	1.21	65.67	1.47	**68.31**	**1.02**
180×5	61.80	1.53	63.44	1.06	65.47	1.34	**69.38**	**0.86**
190×5	61.97	1.03	59.05	0.74	63.25	0.81	**66.96**	**0.67**
200×5	66.67	1.56	64.18	1.21	67.14	1.32	**71.23**	**1.03**
平均值	60.95	1.20	59.89	0.94	62.86	1.04	**66.60**	**0.79**

综上可知, HCEA能够建立有效的概率模型学习优质解的结构信息, 推动种群到达较多的优质解区域. 通过基于Interchange和Insert的快速局部搜索方法, 对全局搜索获得的优质区域进行较为细致的局部搜索, 使算法在全局搜索和局部搜索之间达到了较好的平衡, 从而增强了算法的整体搜索能力. 通过不同测试问题上的大量实验和比较分析, 验证了HCEA是求解NFSSP_SDSTs_RTs的有效算法.

5 结论

本文在传统零等待流水线调度问题的基础上, 进一步考虑实际生产中普遍存在的机器设置时间和工件释放时间以及准时制的生产要求, 研究工作更加具有实际意义和理论价值. 本文以同时最小化总提前和延迟时间为优化目标, 建立带有序相关设置时间和释放时间的零等待流水线调度问题(NFSSP_SDSTs_RTs)的排序模型, 并提出一种混合交叉熵算法(HCEA)进行求解. 主要贡献包括: 1) 分析问题的性质, 设计一种快速评价方法, 有效降低评价解的计算复杂度. 2) 利用交叉熵算法对优质解中工件块的分布进行有效地估计, 学习并积累优质解的结构特征信息, 通过合理的采样与更新方法, 实现对解空间中优质区域的全局搜索. 3) 提出基于Interchange和Insert操作的局部搜索方法, 对全局发现的优质区域进行较为深入的搜索. 4) 设计基于两种邻域搜索策略的快速局部搜索方法, 有效降低了搜索的计算复杂度同时也提高了搜索效率, 实现对解空间中更多优质区域的细致搜索. 大量实验和算法对比验证了本文所提HCEA是求解NFSSP_SDSTs_RTs的一种有效且鲁棒的算法.

分布式调度是生产调度领域近年来研究的热点, 节能降耗是实现低碳制造备受关注的焦点. 未来的研究工作针对问题的特性, 设计高效智能优化算法, 将HCEA扩展应用于求解更加复杂且实际的分布式低碳流水线调度问题.

参考文献:

[1] RUIZ R, MAROTO C. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 2005, 40(2): 479 – 494.

[2] YENISEY M M, YAGMAHAN B. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 2014, 45: 119 – 135.

[3] GONCHAROV Y, SEVASTYANOV S. The flow shop problem with no – idle constraints: A review and approximation. *European Journal of Operational Research*, 2009, 196(2): 450 – 456.

[4] ALLAHVERDI, A. A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, 2016, 255(3): 665 – 686.

[5] ALLAHVERDI A, AYDILEK H, AYDILEK A. No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. *European Journal of Operational Research*, 2018, 269(2): 590 – 601.

[6] ALLAHVERDI A, NG C T, CHENG T C E, et al. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 2008, 187(3): 985 – 1032.

[7] ALLAHVERDI A. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 2015, 246(2): 345 – 378.

[8] LIN S W, LEE Z J, YING K C, LU C C. Minimization of maximum lateness on parallel machines with sequence dependent setup times and job release dates. *Computers & Operations Research*, 2011, 38(5): 809 – 815.

[9] SCHALLER J, VALENTE J M S. Minimizing total earliness and tardiness in a nowait flow shop. *International Journal of Production Economics*, 2020, DOI: 10.1016/j.ijpe.2019.107542.

[10] PINEDO M. *Scheduling: Theory, Algorithms, and Systems*. Fourth Edition. Boston, MA: Springer, 2012.

[11] KELLERER H, RUSTOGI K, STRUSEVICH V A. A fast FPTAS for single machine scheduling problem of minimizing total weighted earliness and tardiness about a large common due date. *Omega*, 2020, 90: 101992.1 – 101992.5.

[12] BIANCO L, DELL'OLMO P, GIORDANI S, Flow shop no-wait scheduling with sequence dependent setup times and release dates. *Infor Information Systems & Operational Research*, 1999, 37(1): 3 – 18.

[13] HE Yujie, QIAN Bin, HU Rong. Hybrid discrete teaching-learning-based optimization algorithm for solving complex parallel machine scheduling problem. *Acta Automatica Sinica*, 2020, 46(4): 195 – 209. (何雨洁, 钱斌, 胡蓉. 混合离散教与学算法求解复杂并行机调度问题. 自动化学报, 2020, 46(4): 195 – 209.)

[14] DING J Y, SONG S, GUPTA J N D, et al. An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem. *Applied Soft Computing*, 2015, 30: 604 – 613.

[15] RUIZ R, ALLAHVERDI A. Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. *Annals of Operations Research*, 2007, 156: 143 – 171.

[16] ARAÚJOA D C, NAGANO M S. A new effective heuristic method for the no-wait flowshop with sequence dependent setup times problem. *International Journal of Industrial Engineering Computations*, 2011, 2(1): 139 – 151.

[17] JOLAI F, RABIEE M, ASEFI H. A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times. *International Journal of Production Research*, 2012, 50(22/24): 7447 – 7466.

[18] ALLAHVERDI A, AYDILEK H. Total completion time with makespan constraint in no-wait flowshops with setup times. *European Journal of Operational Research*, 2014, 238(3): 724 – 734.

[19] NAGANO M S, SILVA A A D, LORENA L A N. An evolutionary clustering search for the no-wait flow shop problem with sequence dependent setup times. *Expert Systems with Applications*, 2014, 41(8): 3628 – 3633.

[20] SAMARGHANDI H, ELMEKKAWY T Y. Solving the no-wait flowshop problem with sequence dependent set-up times. *International Journal of Computer Integrated Manufacturing*, 2014, 27(3): 213 – 228.

[21] SAMARGHANDI H. Studying the effect of server side-constraints on the makespan of the no-wait flow-shop problem with sequence dependent set-up times. *International Journal of Production Research*, 2015, 53(9/10): 2652 – 2673.

[22] ARABAMERI S, SALMASI N. Minimization of weighted earliness and tardiness for no-wait sequence dependent setup times flowshop scheduling problem. *Computers & Industrial Engineering*, 2013, 64(4): 902 – 916.

[23] FRANCA P M, G. TIN J R, BURIOL L S. Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions. *International Journal of Production Research*, 2006, 44(5): 939 – 957.

[24] RUBINSTEIN R Y. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 1997, 99(1): 89 – 112.

[25] CASERTA M, RICO E Q. A cross entropy algorithm for the Knapsack problem with setups. *Computers & Operations Research*, 2008, 35(1): 241 – 252.

[26] SANTOSA B, BUDIMAN M A, WIRATNO S E. A cross entropy-genetic algorithm for m-machines no-wait job-shop scheduling problem. *Journal of Intelligent Learning Systems and Applications*. 2011, 3(3): 171 – 180.

[27] WANG G, LI Q, WANG L. An improved cross entropy algorithm for steelmaking-continuous casting production scheduling with complicated technological routes. *Journal of Central South University*, 2015, 22(8): 2998 – 3007.

[28] WANG G, ZHA Y, WU T, et al. Cross entropy optimization based on decomposition for multi-objective economic emission dispatch considering renewable energy generation uncertainties. *Energy*, 2019, 193: 116790.

[29] ZHANG Z, QIAN B, HU R, et al. A matrix-cube-based estimation of distribution algorithm for the distributed assembly permutation flow-shop scheduling problem. *Swarm and Evolutionary Computation*, 2020: 100785.

[30] PAN Q K, TASGETIREN M F, LIANG Y C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 2008, 35(9): 2807 – 2839.

[31] MONTGOMERY DC. *Design and analysis of experiments*. New Jersey: John Wiley & Sons, 2008.

[32] WANG S, WANG L, LIU M, et al. An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem. *International Journal of Production Economics*, 2013, 145(1): 387 – 396.

[33] RUIZ R, STÜTZLE T. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 2008, 187(3): 1143 – 1159.

[34] YANG Haijun, LI Jianwu, LI Minqiang. *Evolutionary Algorithms: Schema, Emergence and Hardness*. Beijing: Science Press, 2012.
(杨海军, 李建武, 李敏强. 进化算法的模式、涌现与困难性研究. 北京: 科学出版社, 2012.)

作者简介:

张梓琪 博士研究生, 目前研究方向为优化调度理论与方法、智能优化方法, E-mail: Albert.ziqi@hotmail.com;

钱 斌 教授, 博士生导师, 目前研究方向为优化调度理论与方法、智能优化方法, E-mail: bin.qian@vip.163.com;

胡 蓉 副教授, 硕士生导师, 目前研究方向为智能优化调度、物流优化, E-mail: ronghu@vip.163.com.