控制与决策
Control and Decision

考虑学习效应的单人作业车间调度算法

胡金昌， 吴颖颖， 王艳艳， 吴耀华

引用本文:
胡金昌, 吴颖颖, 王艳艳, 等. 考虑学习效应的单人作业车间调度算法[J]. 控制与决策, 2022, 37(1): 37−46.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2020.0938

考虑学习效应的单人作业车间调度算法

胡金昌[1]，吴颖颖[1]，王艳艳[2]，吴耀华[1†]

(1. 山东大学 控制科学与工程学院，济南 250061；2. 山东大学 深圳研究院，广东 深圳 518052)

摘 要: 单人负责多台机器的单一工序作业车间场景中,工人由于重复操作机器而产生学习效应. 针对考虑依赖工件位置学习效应的单人单工序作业车间最小化最大完工时间的调度问题,建立一种混合整数规划模型. 为解决该问题,设计一个考虑学习效应的贪婪算子,利用该算子构造两种贪婪算法,并提出一种基于贪婪的模拟退火算法. 为衡量混合整数规划模型、贪婪算法和基于贪婪的模拟退火算法的性能,设计两种规模问题的数据实验. 通过实验得出:现代混合整数规划模型求解器可以解决机器数量和工件总数量乘积小于75的小规模问题;基于贪婪的模拟退火算法求解此问题具有有效性,适用于各种规模的问题;间隔插入贪婪算法解决此问题速度较快,效果良好,可以应用于需要快速求解的场景.

关键词: 作业车间调度；学习效应；混合整数规划；贪婪算法；模拟退火算法；最大完工时间

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2020.0938 **开放科学(资源服务)标识码(OSID):**

引用格式: 胡金昌,吴颖颖,王艳艳,等. 考虑学习效应的单人作业车间调度算法[J]. 控制与决策, 2022, 37(1): 37-46.

Worker job shop scheduling algorithm considering learning effect

HU Jin-chang[1], *WU Ying-ying*[1], *WANG Yan-yan*[2], *WU Yao-hua*[1†]

(1. College of Control Science and Engineering，Shandong University，Ji'nan 250061，China；2. Shenzhen Research Institute，Shandong University，Shenzhen 518052，China)

Abstract: In some job shops where one worker needs to operate more than one machine, there is learning effect because the worker operates the machines repeatedly. For this one worker and one process production job shop scheduling problem minimizing makespan considering position-based learning effect, a mixed integer programming model is proposed. In order to solve this problem, a greedy operator considering learning effect is designed, and two kinds of greedy algorithms with greedy operator are presented. Then the simulated annealing algorithm based on greedy is proposed. To evaluate the performance of the mixed integer programming model, greedy algorithms and the simulated annealing algorithm based on greedy numerical experiments of the small-sized and large-sized problems are designed. Numerical experiments show that the modern mixed integer programming solver can solve some small size problems that the product of the number of machines and total jobs number is less than 75, simulated annealing algorithm based on greedy is effective and adapt to all kinds of size problems, and the interval insertion greedy algorithm can also obtain satisfactory results rapidly and suit for scenarios that need to be solved quickly.

Keywords: job shop scheduling；learning effect；mixed integer programming；greedy algorithm；simulated annealing；makespan

0 引 言

在人力成本不断增加的背景下,很多车间引入了自动化程度较高的加工机器,并且实行一人负责多台机器的管理模式. 由于单一工人不能同时操作每个闲置的机器,该种模式容易造成机器的空闲率提高. 并且,当多台机器需要同时操作时,工人一般选择较近的机器,这样必然会影响全部工件的加工完成时间. 因此,有必要在工人选择机器时引入调度算法,以提高设备利用率和生产效率. 另外,工人操作机器的频率提高,操作越来越熟练,使得操作的时间相应减少,即产生学习效应[1]. Heizer等[2]指出,不同的产品有不同的学习曲线,学习率取决于生产流程的潜力;同时还指出不同产品学习率在70%～90%之间,这意味着第2个生产单位的操作时间是第1个的

收稿日期: 2020-07-10；**修回日期:** 2020-09-09.

基金项目: 国家自然科学基金项目(61703241)；深圳市科技创新委员会面上基金项目(JCYJ20190807094803721).

责任编委: 王凌.

†通讯作者. E-mail: mike.wu@263.net.

70%～90%,在不同位置生产的同种工件的操作时间存在明显差异,是否考虑学习效应会对调度的效果产生较大影响. 因此,应该将学习效应引入调度算法中,以提高调度的准确性.

学习效应对加工制造的影响[3]最早是由Wright于1936年提出的. 1999年,Biskup[4]将学习效应应用于排序问题中,提出了依赖工件位置的学习效应模型. 随后学习效应被大量应用于单机、平行机、流水车间等调度问题中. 其中:一部分研究没有考虑工件的安装时间,工件加工的全部时间都受学习效应的影响,如文献[5-9];另一部分考虑了工件的安装时间,这类问题工件的生产耗时被分成安装时间和加工时间两部分,这两部分受学习效应的影响不同,问题也会不同. Sun等[10]、闫杨等[11]和马卫民等[12]研究了成组加工单机调度问题,工件加工时间受依赖工件位置的学习效应的影响,各个工件组的安装时间取定值与组排序相关或者与开始时间相关. Pei等[13-14]考虑了工件串行批次加工的单机调度问题,批安装时间是开始安装时间的线性函数,工件加工时间受学习效应的影响. Soleimani等[15]和Pei等[14]研究了平行机问题的调度,安装时间与工件排序相关,工件加工时间受学习效应的影响. Mousavi等[16]分析了安装时间和加工时间都受学习效应影响的流水车间调度模型. Tayebi Araghi等[17]在作业车间调度问题的研究中考虑了受学习效应影响的安装时间和受恶化效应影响的工件加工时间. 然而,这些研究所涉及的生产环境中学习效应不能在不同机器之间传递. 本文的单人作业模式则使得工人的操作经验可以跨机器累积,从而使学习效应可以连续作用于不同的机器的安装环节,这与之前单机和多机调度问题的研究都有所不同. 考虑到加工阶段由机器自动完成,所以本文设定加工时间为定值.

对于考虑跨机器影响的依赖工件位置学习效应的单人单工序作业车间调度问题,目前尚未有合适的算法直接求解. 元启发式算法,如遗传算法[18-21]、模拟退火算法[7,9,20]、粒子群算法[20,22]等解决调度问题的传统算法,可以用于本问题,但是,这些传统算法在邻域搜索时,大多采用两随机序列交叉、排序内两随机工件互换或随机提取工件插入的完全随机的方式,搜索效率较低,尤其是本文考虑了学习效应和运输时间,增加了目标值的计算时间,导致这些传统算法在求解每个个体的目标值时耗时较长. 另外一类启发式算法是定向的邻域搜索,如迭代贪婪算法[23-27],其主要思想是在工件排序中随机提取工件后定向插入. 对于本文问题,前后工件的机器位置会直接影响工人的行走距离,而频繁地随机提取工件,容易破坏之前形成的局部结构,存在影响搜索效果的可能性. 在定向插入方面,以往的研究采用贪婪的方式,即插入令目标值增加量最小的位置,而本文考虑受学习效应的影响,每个工件的安装时间随排序的不同而变化,所以目标值的计算复杂度增加,之前文献计算目标值增量的算法不能有效应用于本文问题. 因此,之前的迭代搜索算法无论在工件提取方面还是在插入方面的处理都不能有效适应于本文问题.

为改善上述算法存在的不足,本文提出一种考虑学习效应的贪婪算子,用以确定工件的插入位置,该贪婪算子改进了经典贪婪算法中计算目标值增量的方法. 应用考虑学习效应的贪婪算子,本文完成了以下工作:首先,设计两种贪婪算法,其中,间隔插入贪婪算法可以避免传统贪婪算法中选取插入工件的方法带来的工人空闲的问题;其次,设计多工件变化顺序和全部工件变化顺序两种产生新解的方式,相对于传统的元启发算法的随机搜索和迭代贪婪算法的随机提取工件的方式,可以提高对有效解的搜索能力;再次,因模拟退火算法简单高效,能有效避免陷入局部最优,故本文选择模拟退火算法作为框架,融合基于贪婪算子的贪婪算法和产生新解的方式;最后,通过数据实验表明所提出的间隔插入贪婪算法和基于贪婪的模拟退火算法的有效性.

1 数学模型

设某一作业车间共有M台机器,分别需要加工$n_1, n_2, \ldots, n_M$个工件. 所有机器的操作由一个工人负责,包括取下已经完成加工的工件、安装加工下一个工件的原材料、清理机器等准备操作(下文简称操作). 每台机器只能加工一种工件,不同机器标准操作工时不同,分别为$s_1, s_2, \ldots, s_M$. 工人操作完成之后,机器自动加工,不需要人员看守,工人可以到达其他机器旁操作,各个机器上每个工件自动加工的时间分别为$p_1, p_2, \ldots, p_M$. 已知两两机器之间距离,模型中用时间代替该距离,t_{ij}表示机器i到机器j之间的转换时间,$t_{ii} = 0, t_{ij} = t_{ji}$. 各台机器同一时间只能加工一个工件,所有工件在加工过程中不能中断,并且只能在某一工件完成之后开始加工下一工件. 为避免歧义,本文机器名和工件名相同,即工件i表示由机器i加工的工件.

工人操作时间受依赖工件位置的学习效应的影响,定义第k个工件的实际操作时间为s'_k. 根据依赖工件位置的学习效应模型[2],工件j实际的工作时间

$p_{(r,j)} = t_j r^{\alpha}$. 其中: t_j 表示工件 j 的标准工时, r 表示工件 j 被安排加工的位置. $\alpha \leqslant 0$, 代表学习因子, α 的计算可参考文献[4], 例如, 当工件的实际加工时间分别为前一个工件的90%、80%和70%时, $\alpha = -0.15, -0.32, -0.51$.

当第 k 个工件在机器 i 上加工时, 第 k 个工件的实际操作时间 $s'_k = s_i k^{\alpha}$. 其他符号及含义如下:

N: 工件总数;

k: 工件序号, $k \leqslant N$;

$C_{\max}$: 目标值, 最大完工时间;

z_{ki}: 决策变量, $z_{ki} = 1$ 代表第 k 个工件是在机器 i 上加工的, 否则 $z_{ki} = 0$;

x_{ki}: 准备操作第 k 个工件时机器 i 可以操作的最早时间.

构建如下数学规划模型:

$$\min C_{\max}. \tag{1}$$

s.t.

$$\sum_{i=1}^{M} z_{ki} = 1,\ k = 1, 2, \ldots, N; \tag{2}$$

$$\sum_{k=1}^{N} z_{ki} = n_i,\ i = 1, 2, \ldots, M; \tag{3}$$

$$x_{ki} \geqslant 0,\ k = 1, 2, \ldots, N,\ i = 1, 2, \ldots, M; \tag{4}$$

$$x_{ki} \geqslant x_{(k-1)i} + z_{(k-1)i}(p_i + s'_{k-1}),$$
$$k = 2, \ldots, N,\ i = 1, 2, \ldots, M; \tag{5}$$

$$x_{ki} \geqslant x_{(k-1)l} + z_{(k-1)l} s'_{k-1} + z_{ki} z_{(k-1)l} t_{il} -$$
$$V(1 - z_{ki} z_{(k-1)l}),\ k = 2, \ldots, N,$$
$$i = 1, 2, \ldots, M,\ l = 1, 2, \ldots, M,\ i \neq l; \tag{6}$$

$$C_{\max} \geqslant x_{ki} + z_{ki}(p_i + s'_k);$$
$$k = 1, 2, \ldots, N,\ i = 1, 2, \ldots, M. \tag{7}$$

其中: 式(2)表示每个工件只能被加工一次; 式(3)表示每个机器只能加工相应数量的工件; 式(4)表示每个工件只能在零时刻之后被操作; 式(5)表示第 k 个工件的加工机器允许操作的时间(如果第 k 个工件的加工机器与第 $k-1$ 个工件的相同, 则需要第 $k-1$ 个工件的操作完成和机器加工之后才允许操作第 i 个工件; 如果第 k 个工件的加工机器与第 $k-1$ 个工件的不同, 则第 k 个工件加工机器允许操作的时刻等于第 $k-1$ 个工件加工机器允许操作的时刻); 式(6)表示第 k 个工件的加工机器只能在工人操作完成上一个工件到达该机器之后被操作, 即 x_{ki} 大于等于工人完成第 $k-1$ 个工件操作后到达第 k 个工件加工机器的时间, V 为足够大的数, 如 $V = N \times \max\{s_1, s_2, \ldots, s_M\} + (N-1) \times \max\{t_{11}, \ldots, t_{1M}, \ldots, t_{MM}\}$, $z_{ki} z_{(k-1)l} = 0$ 时, $x_{ki} \geqslant x_{(k-1)l} + z_{(k-1)l} s'_{k-1} + z_{ki} z_{(k-1)l} t_{il} - V$ 恒成立, 该条件将无效; 式(7)表示最大完工时间应在所有工件加工完成之后.

式(6)中 $z_{ki} z_{(k-1)l}$ 可进行线性转换, 如转换为 $(z_{ki} + z_{(k-1)l} - 1 + |z_{ki} + z_{(k-1)l} - 1|)/2$, 所以最后的模型为混合整数规划模型, 可以使用求解器求解, 本文采用Gurobi进行求解. 另外, 模型中 z_{ki} 和 x_{ki} 的个数均为 $N \times M$, 此数量会直接影响该模型解决问题的效率, 将在第4节展开讨论.

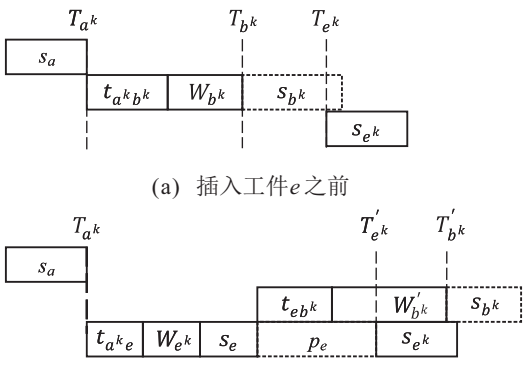
2　贪婪算法

贪婪算法是寻找插入调度列表后加工时间增加量最小的位置插入选定工件, 直到插入所有工件. 为减少每次插入时加工时间增加量的计算难度, 本文首先提出了贪婪算子.

2.1　贪婪算子

假设某一时刻, 需要将工件 e 插入到既定的调度列表中, 调度列表中存在一组相邻的工件 a^k、b^k, 分别是第 k 和 $k+1$ 个工件, b^k 工件之后第一个 e 工件用 e^k 表示, $k = 0, 1, \ldots, N$. 当 $k = 0$ 时, 表示工件 e 插入原调度列表的首位, a^k 视为虚拟工件; 当 $k = N$ 时, 工件 e 插入原调度列表的末位, b^k 视为虚拟工件. T_{a^k}、T_{b^k}、T_{e^k} 分别表示 a^k 完成操作的时间、b^k 开始操作的时间、e^k 开始操作的时间. W_i 表示加工 i 工件之前需要等待的时间, $i \in a^k, b^k, e^k$. 上述变量在顺序确定时, 可视为已知量.

当 a^k 与 b^k 之间插入 e 工件时, 工件 b^k 的开始操作时间 (T_{b^k}) 有可能延后, 并且 e 的插入有可能推迟 e^k 工件开始操作的时间 (T_{e^k}). b^k 和 e^k 的后续工件提前加工还是延后加工的情况分别与 b^k 和 e^k 的相同, 并且提前或延后的幅度不大于 b^k 和 e^k, 所以只需分析 b^k 和 e^k 的影响. 插入 e 工件后 T_{b^k} 和 T_{e^k} 变化为 T'_{b^k} 和 T'_{e^k}, 如图1所示.



(a) 插入工件 e 之前

(b) 插入工件 e 之后

图1　插入工件 e 前后的甘特图

令N'表示当前调度列表的工件数量,计算公式如下:

1) 当$0 < k < N', T'_{b^k} = \max\{T_{a^k} + t_{a^k e} + W_{e^k} + s_e + t_{eb^k} + W'_{b^k}, T_{b^k}\}$. 如果$b^k$与$e$相同,则$W'_{b^k} = p_e$;否则$W'_{b^k} = \max\{T_{b^k} - (T_{a^k} + t_{a^k e} + W_{e^k} + s_e + t_{eb^k}), 0\}$, $T'_{e^k} = \max\{T_{a^k} + t_{a^k e} + W_{e^k} + s_e + p_e, T_{e^k}\}$.

2) 当$k = 0$时,$T'_{b^k} = \max\{s_e + t_{eb^k} + W'_{b^k}, 0\}$. 如果$b^k$与$e$相同,则$W'_{b^k} = p_e$;否则$W'_{b^k} = 0, T'_{e^k} = \max\{s_e + p_e, T_{e^k}\}$.

3) 当$k = N'$时,令$C'_{\max}$表示当前列表的最大完成时间,$T'_{b^k} = \max\{T_{a^k} + t_{a^k e} + W_{e^k} + s_e + p_e - C'_{\max}, 0\}$,设定$T_{b^k} = 0, T_{e^k} = T'_{e^k} = 0$.

在既定加工顺序下,通过比较b^k之前最后一个e工件的完成时间和插入的e工件最早开始操作时间可以得到W_{e^k},其他变量也都可以视为已知量.所以,第k和$k+1$个工件之间插入e工件最大延后时间为$\Delta(k) = \max\{T'_{b^k} - T_{b^k}, T'_{e^k} - T_{e^k}\}$. 因为学习效应,工件的操作时间与工件位置有关,所以插入并令e之后所有工件的操作时间总和减少,减少量用$\varphi(k)$表示,e插入前后该值分别用S^k_{inter}和S'^k_{inter}表示. 令函数$s(x)$代表原始的调度列表中第x工件的标准操作时间,有以下计算公式:

1) 当$k < N'$时,有

$$S^k_{\text{inter}} = \sum_{x=k+1}^{N'} s(x) \times x^a,$$

$$S'^k_{\text{inter}} = \sum_{x=k+1}^{N'} s(x) \times (x+1)^a,$$

$$\varphi(k) = S^k_{\text{inter}} - S'^k_{\text{inter}} = \sum_{x=k+1}^{N'} s(x)(x^a - (x+1)^a);$$

2) 当$k = N'$时,$\varphi(k) = 0$.

考虑了学习效应对操作时间的影响之后,设定第k、$k+1$工件之间插入e的最大延后时间为$\psi(k)$,$\psi(k) = \Delta(k) - \varphi(k)$,所以,当需要插入某工件时,只需要计算已生成列表中每两个相邻工件之间的ψ,选取ψ最小的相邻工件插入即可实现每次插入工件时加工时间增加量最小. 定义ψ为贪婪算子.

本文利用贪婪算子,根据选取插入工件的不同规则构建两种贪婪算法,分别是随机插入贪婪算法和间隔插入贪婪算法.

2.2 随机插入贪婪算法(RIG)

RIG (random insertion greedy)算法是随机选取工件插入列表,具体方法如下.

step 1: 初始化调度列表. 随机选择某一工件最先放入调度列表,生成初始列表.

step 2: 随机选择某一工件,根据2.1节计算调度列表中每两个相邻工件之间的贪婪算子ψ,选取ψ最小的位置插入此工件,重复此步骤直到所有工件都被插入调度列表.

2.3 间隔插入贪婪算法(IIG)

IIG (interval insertion greedy)算法是首先生成插入列表,插入列表为插入工件的排序,该列表需要尽量保证每两个相同机器的工件间隔最大,具体方法如下.

step 1: 生成插入列表,方法如下.

step 1.1: 计算机器i的工件间隔$\text{Inter}_i = N/n_i$;

step 1.2: 计算机器i第k个加工的工件的位置权重$w^i_k = k \cdot \text{Inter}_i$;

step 1.3: 将w^i_k按从小到大排序,w^i_k所对应的i的排序即为插入列表中工件的排序.

step 2: 初始化调度列表.从插入列表中取出第1个工件放入调度列表,生成初始列表.

step 3: 按由前到后的顺序依次提取出插入列表中的工件,根据2.1节计算调度列表中每两个相邻工件之间的贪婪算子ψ,选取ψ最小的位置插入此工件,重复此步骤直到所有工件都被插入调度列表.

3 基于贪婪的模拟退火算法

模拟退火算法是解决组合优化问题最常用的元启发式算法之一[7,20],并且广泛应用于车间调度问题,其特点是简单高效,避免模型过快收敛陷入局部最优,因此,成为常见的启发式算法框架. 如文献[23-27]的迭代搜索算法即使用了模拟退火算法的新解接受准则. 另外,在本文设计算法之前对比了经典的模拟退火算法、遗传算法、粒子群算法解决本文问题的性能,结果表明,模拟退火算法明显优于另外两种算法(这3种经典算法的求解结果展现在第4节实验部分图3和图5中),所以本文以模拟退火算法作为基本框架. 基于贪婪的模拟退火算法(simulated annealing algorithm based on greedy, SAG)是在传统模拟退火算法中引入贪婪算子,改进了初始解,构造了两种基于贪婪算子的产生新解的方式,这些都是在之前算法的基础上做出的改进.

1)初始解.

文献[23-27]均采用调度问题中经典的选择插入工件的方式,即按照工件在所有机器上的加工时间总和由大到小进行排序(longest processing time, LPT),被称为NEH启发式算法.

本文实验部分对比了NEH启发式算法、RIG以及IIG,结果显示IIG性能最优,因此,没有特殊说明时,SAG使用IIG作为初始解.

2)多工件变更顺序(MCP).

之前文献中也将MCP(multi-job change positions)这种产生新解的方式称为加工序列的解构和重构,分别指的是本文中提到的加工列表中工件的提取和重新插入. 如文献[23-27]中采用的方法为:随机提取多个工件,然后依次按一定规则插入,采用固定的工件变更数量.

本文所研究的问题是,加工序列中工件的开始时间不但受机器和工人是否空闲的影响,还会受前一工件机器位置的影响,如果频繁提取其中单个工件,则容易破坏之前形成的局部结构. 所以,本文设计了连续提取多个工件重新插入的方式,并且提取工件的数量随执行时间变化而减少,这样既实现了算法执行初期对当前解的强烈扰动,又实现了后期对解的微调.

定义变更系数μ代表每次迭代提取工件的个数占总工件数的比例,变更工件的数量为μN. 随机生成变更起始位置$\sigma = \mathrm{rand}(N - \mu N + 1), \mathrm{rand}(x)$表示生成一个不大于$x$的随机整数,变更终止位置$\rho = \sigma + \mu N$,变更区间$[\sigma, \rho]$. μ随算法执行时间变化,假设算法已执行时间为NowTime, μ_0表示初始比例,则$\mu = \mu_0 \dfrac{\mathrm{NowTime}}{\mathrm{TotalTime}}$. 具体操作为:从原调度列表中提取出区间$[\sigma, \rho]$的工件,根据2.1节计算调度列表中剩余工件中每两个相邻工件之间的贪婪算子ψ,选取ψ最小的位置插入此工件,生成新解.

3)全工件变更顺序(ACP).

为更充分利用贪婪算子,参考文献[23, 26-27]的邻域搜索方法,本节提出了基于贪婪算子的ACP(all job change positions),实现所有工件的定向调整. 具体算法如下.

step 1: 为原列表中所有工件标号,在原列表中随机选择某一工件(选择的工件标号不能重复),执行step 2,直到所有工件重新调整位置为止.

step 2: 从原列表中提取出所选择的工件,根据2.1节计算调度列表中剩余工件中每两个相邻工件之间的贪婪算子ψ,选取ψ最小的位置插入此工件,生成新解. 计算新解的目标值,如果该值小于当前目标值,则新解取代原调度列表;否则,原调度列表不变. 由于ACP产生新解耗时较长,设置概率参数ε,令算法以一定概率使用该方法产生新解.

4)算法流程.

step 1: 初始化. 设置基础参数初始温度T_0、迭代次数L、终止温度T_{end}、温度衰减系数α;设置改进型算法中使用的参数μ_0、ε和TotalTime;生成初始解X_0;令迭代次数$i = 0$,最优解$X_b = X_0$,当前解$X_c = X_0$,当前温度$T = T_0$.

step 2: 若满足终止条件,则终止算法,X_b为最终解;否则,执行step 3.

step 3: 若$i \leqslant L$,则执行step 4;否则,令$i = 0, T = \alpha T$,返回step 2;

step 4: 更新μ,在X_c的基础上利用MCP产生新解X_{new}, $i = i + 1$,利用step 6检验X_{new},得到X_c和X_b,以概率ε进入step 5. 若未进入step 5,则返回step 3.

step 5: 在X_c基础上执行ACP,返回step 3.

step 6 :检验新解X_{new},返回X_c和X_b.

计算目标函数增量$\Delta E = C_{\max}(X_{\mathrm{new}}) - C_{\max}(X_c)$. 若$\Delta E < 0$,则$X_c = X_{\mathrm{new}}$;否则,以概率$\exp(-\Delta E/T)$更新$X_c$,并令$X_c = X_{\mathrm{new}}$. 如果$C_{\max}(X_{\mathrm{new}}) < C_{\max}(X_b)$,则$X_b = X_{\mathrm{new}}$;否则$X_b$不变.

5)终止条件.

模拟退火算法终止条件一般设置为温度阈值,而传统IG算法一般设置为时间阈值,如文献[23-27]设置了与问题规模相关的运行时间上限作为终止条件. 本文综合这两方面因素,设置两种终止迭代的条件:一是温度衰减到终止温度以下,即$T < T_{\mathrm{end}}$;另外一种是到达时间阈值TotalTime.

4　测试实验

本文设计了对比实验以衡量混合整数规划模型、IIG和SAG的性能. 参考文献[28-29],将混合整数规划模型在可接受时间范围内能解决的问题规模视为小规模问题,其对应的实验为小规模问题实验,目的是分析混合整数规划模型的能力;其他规模的问题视为大规模问题. 大规模问题实验设计时主要考虑3个因素:一是该规模可以发现随着规模的增加,IIG和SAG优化效果的变化规律;二是该规模可以得出

各个参数及学习效应对算法的影响;三是假设工人持续作业的时间为1 h,该时间能保持学习效应的稳定. 所以设置问题规模时,问题的$C_{\max}$应尽量小于1 h. 所有实验在Intel (R) Core (TM) i7-8550M CPU @ 1.80 GHz双核处理器、8.00 GB的计算机上进行,混合整数规划模型使用Gurobi 7.0求解器求解,其他算法利用Matlab 2014A完成.

实验中各机器工件数量n_i、标准操作时间p_i、机器加工标准时间s_i、计算t_{ij}的机器坐标值均满足一定区间内的均匀分布,下文用$[L,U]$形式表示该区间.

4.1 小规模问题实验

小规模问题是指机器数量和工件总数量较少的问题,可以用混合整数规划算法求解. 设计该类实验有以下目的: 1)衡量混合整数规划算法的性能,得到该算法可以解决问题的最大规模; 2)评估贪婪算法和SAG对解决小规模问题的有效性. 实验中机器数量M分别取值2、3、4、5、6,n_i的取值区间分别为$[1,5]$、$[1,10]$、$[1,15]$、$[1,20]$、$[1,25]$、$[1,30]$,p_i的取值区间为$[2,6]$,s_i的取值区间为$[5,20]$,t_{ij}由坐标取值区间为$[1,10]$的点计算得到,学习因子$a=-0.15$. 共30种实验场景,每种实验场景实验11次,得到表1～表3数据. 设置Gurobi求解MIP问题时间上限为1 h. SAG的参数取值为$T_0=0.4, L=200, T_{\text{end}}=0.01, \alpha=0.995, \mu_0=0.3, \varepsilon=0.1, T_0=0.4$.

通过表1可知,工件数量越多、机器数量越多时,MIP可以取得相对最优解的概率越小. MIP对于2、3、4、5、6台机器有50 %以上概率取得相对最优解的问题的规模分别为每台机器工件数量上限为30、25、15、5、5个工件的问题. 因为不同机器数量和工件数量决定了决策变量的数量,即$N\times M$. 决策变量的数量影响了MIP解决问题的速度,表2总结了不同决策变量数量区间取得最优解的情况,显然,决策变量数量在100以内时MIP能够取得相对最优解的概率较大. 决策变量数量最多在51～75内时可以在1 h内取得最优解,平均时间为378.74 s. 3种算法中的最优解称为“相对最优解”,表3展示了MIP、IIG、SAG在不同决策变量数量区间可以取得相对最优解的比例. 决策变量数量在75以内时,MIP可以取得问题最优解,因此,MIP可以解决机器数量和工件总数量乘积小于75的小规模问题. SAG在各个决策变量区间取得相对最优解比例均为最大,所以SAG求解本文小规模问题时具有有效性. IIG解决小规模问题时效果良好,约93 %的概率可以得到最优解.

表1 不同机器和工件数量时MIP取得相对最优解的概率

机器数量	n_i的取值区间上限(下限为1)					
	5	10	15	20	25	30
2	1	1	1	1	1	1
3	1	1	1	0.818	0.636	0.364
4	1	0.546	0.546	0.182	0	0.091
5	0.909	0.364	0	0.091	0	0
6	0.546	0	0	0	0	0

表2 各决策变量数量区间MIP的执行结果

决策变量数量区间	决策变量数量均值	相对最优概率	1 h内最优概率	1 h内最优解时间均值
1～25	15.98	1	1	0.272
26～50	37.53	1	1	15.831
51～75	63.97	1	0.838	378.74
76～100	88.27	0.707	0.195	590.631
101～125	113.27	0.409	0.091	1 465.391
126～150	132.4	0.2	0.067	2 158.107

表3 MIP、IIG、SAG取得相对最优解的概率和执行时间

决策变量数量区间	取得相对最优解的比例			执行时间/s		
	MIP	SAG	IIG	MIP	SAG	IIG
1～25	1	1	1	0.272	1.202	0.002
26～50	1	1	1	15.831	1.953	0.002
51～75	0.730	1	0.983	901.110	2.232	0.005
76～100	0.488	1	0.962	3 012.819	2.435	0.008
101～125	0.227	0.992	0.913	3 405.980	3.247	0.012
126～150	0.200	0.980	0.867	3 503.901	4.595	0.014
151～175	0.133	0.965	0.833	3 600	5.263	0.024

另外,表3给出了不同决策变量数量区间时MIP、IIG、SAG解决问题的时间. 当决策变量在25个以内时,MIP用时较少; 150个以内时MIP可以在1 h内得到最优解. 除决策变量数量小于25个之外,对于其他决策变量的区间的问题,SAG和IIG用时均远远小于MIP. 因此,小规模问题实验不能完全体现出SAG和IIG的性能,所以本文设计了大规模实验来分析不同算法参数、学习效应及问题规模对算法的影响,并验证SAG和IIG的有效性.

4.2 大规模问题实验

大规模问题实验的设计思路如下: 首先,对比算法可能的初始解,即多种贪婪算法效果对比实验; 其次,分析学习效应对问题的影响,从而说明考虑学习效应的必要性,同时分析学习效应对SAG的影响; 最后,对比SAG与其他多种算法的求解效果以表明SAG的有效性,并通过对比得到初始解和两种产生新解的方式对SAG的作用.

实验中M取值为10、15、20,n_i的取值区间分别为$[1,20]$、$[11,30]$、$[21,40]$,p_i为$[1,15]$,s_i为$[5,25]$,t_{ij}由坐标取值区间为$[1,10]$的点计算得到,共9个实验场景,每个场景列举5个案例,学习因子$a=-0.15$. 不同算法的比较均以相对实践仿真的优化比例作为参考,实践仿真是工人选择下一个加工机器时,如果存在工人到达后即可操作的机器,则选择距离最近的机器;否则,选择加工时间最长的机器,此仿真简写为MD-LPT (minimum distance-longest processing time). 实验中SAG_IIG、SAG_RIG和SAG_MD-LPT分别代表以IIG、RIG和MD-LPT为初始解的SAG. 经过实验验证,SAG在$L=50$、$T_{\text{end}}=0.01$、$\alpha=0.01$、$\mu_0=0.3$、$\varepsilon=0.3$、$T_0=0.4$时效果较优,下文以此作为算法参数.

1)多种贪婪算法效果对比.

表4给出了所有案例分别使用IIG、RIG和NEH启发式算法求解结果的平均值,其中RIG计算10次得到平均值和标准差. 对比可知IIG和RIG优于NEH启发式算法,分别提高3.1%和1.78%. 由RIG结果的标准差可知RIG求解效果不稳定. IIG较优的主要原因是可以尽量保证同一个机器的加工工件间隔加工,以避免工人的等待时间. 因此,如果直接求解该问题,则IIG效果更好并且稳定.

表4　不同实验场景中各种贪婪算法的性能

M	n_i	IIG	NEH	RIG	
				平均值	标准差
10	1～20	883.800	916.675	891.918	2.788
	11～30	1 328.009	1 374.563	1 360.973	14.390
	21～40	1 887.640	2 011.236	1 931.624	18.733
15	1～20	1 038.283	1 067.387	1 056.014	8.341
	11～30	2 222.445	2 273.644	2 249.957	8.011
	21～40	3 107.754	3 152.403	3 105.926	9.102
20	1～20	1 144.975	1 168.362	1 166.162	8.819
	11～30	2 466.663	2 511.513	2 502.799	4.168
	21～40	3 552.900	3 734.353	3 573.763	15.541

2)学习因子对问题和算法的影响.

为了对比不同学习因子对问题和算法的影响,本文参考文献[30],分别在$\alpha=0$ (无学习效应)、-0.15、-0.32和-0.51时求解所有场景的案例,并统计每个案例某学习因子时得到的SAG的解应用于该案例受其他学习效应影响时的结果. 定义α_2表示实际的学习因子,即最终解计算目标值时学习因子;α_1表示SAG计算时使用的学习因子;将$\alpha=\alpha_1$时SAG计算的解在$\alpha=\alpha_2$时的目标值记为$C_{\max}(\alpha_1,\alpha_2)$. 定义误差比例

$$E_{\alpha_1,\alpha_2}=\frac{C_{\max}(\alpha_1,\alpha_2)-C_{\max}^{\text{best}}}{C_{\max}^{\text{best}}}.$$

其中:$C_{\max}^{\text{best}}=\min C_{\max}(\alpha_i,\alpha_2)$,$\alpha_i$取值$-0.15$、$-0.32$、$-0.51$、0. 由表5中$\alpha_1=\alpha_2$时$E_{\alpha_1,\alpha_2}=0$和$\alpha_1\neq\alpha_2$时所有$E_{\alpha_1,\alpha_2}>0$可以得出,当实际中存在学习效应时,只有在模型中应用准确学习因子才能得到最佳效果. 例如表5中,实际的$\alpha_2=-0.32$,若SAG计算时不考虑学习效应,则最大完工时间平均增加24.83%. 学习因子不准确也会影响调度结果,计算使用的学习因子与实际学习因子误差越小,结果差别越小.

表5　学习因子对问题和SAG的影响(E_{α_1,α_2})　%

SGA计算时使用的α_1	实际(计算目标值)的α_2			
	0	-0.15	-0.32	-0.51
0	0	7.02	24.83	43.64
-0.15	1.22	0	6.44	17.32
-0.32	2.03	1.00	0	5.31
-0.51	2.35	1.62	0.48	0

图2展示了不同学习因子时,SA得到的解的目标值比SAG高出的平均比例,即SAG相较于SA的优化比例. 图中:横坐标n_i取值1、11、21,分别指n_i的取值区间为$[1,20]$、$[11,30]$、$[21,40]$,M为机器数量. 可以得出,问题学习因子绝对值越大,问题规模越大,两个算法的优化效果差别越大.

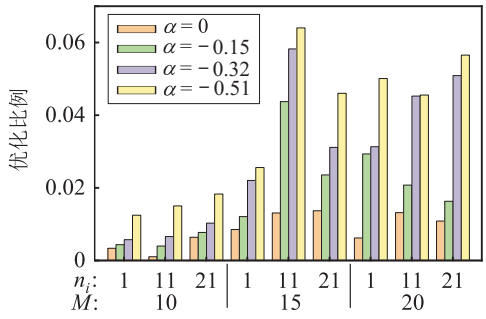


图2　不同α时SAG相对于SA的优化比例

3)不同算法效果的比较.

为了衡量IIG和SAG的性能,本文从以下3个方面设计对比实验:首先,与传统算法对比,验证IIG和SAG_IIG的有效性;其次,变化SAG的初始解,即构造SAG_RIG和SAG_MD-LPT,分析初始解对SAG的影响;最后,比较相同初始解的SA、IG与SAG,即构造SA_IIG、IG_IIG、SA_MD-LPT、IG_MD-LPT,分析

MCP和ACP对SAG的影响.

作为对比的传统算法有:模拟退火算法SA[9]、遗传算法GA[21]、粒子群算法PSO[22]和迭代贪婪算法IG[23]. 本文在选取PSO、GA和SA的参数值时,尽量保证充分利用算法的能力(比如种群数量和迭代次数选取较大数值),以提高解的质量. PSO、GA利用MD-LPT和随机解作为初始群体,SA利用MD-LPT生成初始解. 产生新解、新粒子或新染色体的方式是传统的随机交叉或两随机工件位置互换. 具体参数如下:PSO种群数量为1 000,迭代次数为500,惯性系数为0.96,认知系数为0.7,社会学习系数为0.3,初始解中MD-LPT占比0.75;SA初始温度为0.1,迭代次数为2 000,终止温度为2×10^{-6},温度衰减系数为0.98;GA种群数量为1 000,迭代次数为1 000,初始解中MD-LPT占比0.5,每代交叉染色体数为$2P_g$,变异数为P_g;IG的时间阈值与SAG相同,其他参数参见文献[23]的结论. 变更顺序的多工件数量取4,初始温度取0.4,初始解采用NEH启发式算法.

图3对比了SAG_IIG与传统算法SA、GA、PSO和IG相对于MD-LPT的优化比例. 结果显示:PSO和GA只能提高约18 %和19 %,低于其他算法,所以传统的PSO和GA在解决本问题时没有优势;SAG_IIG比SA的优化效果提高约4 %,这说明对比于传统的初始解和两工件随机交叉产生新解的方式,SAG_IIG的初始解和产生新解的方式更有优势;IG算法运用文献[23-27]的NEH启发式算法和固定数量随机选取多工件插入调度列表的方式生成新解,SAG_IIG优化效果比IG提高了2.75 %,说明SAG_IIG的改进具有有效性.

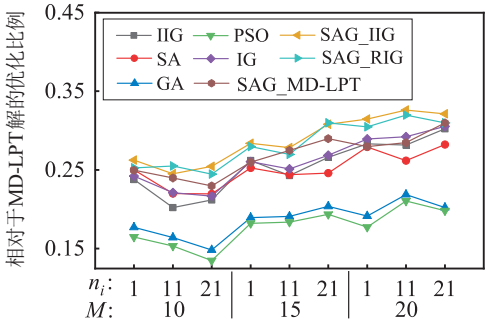


图3 多种算法优化效果对比

通过图3中SAG_IIG、SAG_RIG和SAG_MD-LPT的比较,还可以进一步分析初始解对SAG的影响. 显然,SAG_IIG是最优的,不过SAG_IIG相对于SAG_RIG优势不明显,因为IIG和RIG本身差别不大(从表4数据可以得出). 这两者都明显优于SAG_MD-LPT,从而说明贪婪算法作为初始解对SAG的作用较为明显,可以有效提高SAG性能. 另外,通过图3可以发现:IIG取得了较好的优化效果,平均可以提高25 %,明显高于PSO和GA;并且在较大规模时,IIG效果优于SA. 所以IIG可以取得较为满意的优化结果.

为了更明显地体现SAG在初始解的基础上的优化能力,本文对比了SAG与SA、IG的收敛曲线,见图4. 使用的案例为$M=20$,n_i的取值区间为[21, 40],学习因子$a=-0.15$. 3个算法设置了相同的初始解,即IIG和MD-LPT. 通过图4中曲线可以看出:MD-LPT作为初始解时,SA在运行前期收敛迅速,但是容易进入局部最优解且很难跳出局部最优;SAG收敛相对缓慢,但是具备更强的持续寻优能力,所以随机交换工件位置产生新解的方式在优化后期容易进入瓶颈,而SAG中MCP和ACP在产生新解时依次尝试可以提高解质量的工件位置,所以更加高效;IG相对于SAG的收敛能力差一点,主要是传统IG求解目标较为耗时,迭代次数也会减少. 当初始解为IIG时,SA可以进一步优化的空间不大,IG的效果居中,SAG的优化效果最好. 通过曲线图的形状也可以看出,SAG使解大幅提高的机会更大,这就是MCP和ACP发挥的作用. 另外,SAG优化效果好的主要原因是IIG作为初始解,使算法有较好的搜索起点.

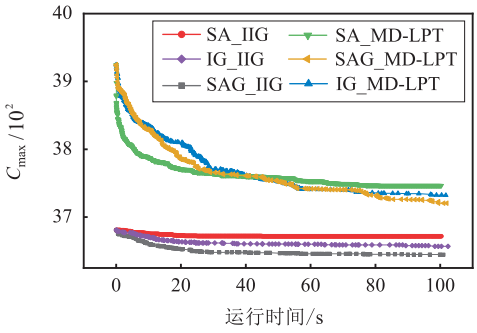


图4 不同初始解时SAG、SA和IG的收敛曲线对比

图5记录了各种算法的平均执行时间,时间阈值为$0.2N$的算法有:IG、SAG_IIG、SAG_RIG、SAG_MD-LPT. 由图5可知,群智能优化PSO和GA耗时较长,并且随着问题规模增加执行时间明显增多,每一次迭代求解种群中所有个体的目标值耗时较长. 因为单一个体逐步优化的方式在计算目标函数时节约了时间,所以SA、SAG和IG耗时较短. SAG与SA的执行时间相差不大,节约大约了2 %,比GA、PSO节约了76 %、62 %. 值得注意的是,IIG的执行时间远远小于其他算法,虽然执行时间随着问题规模增大而增长的趋势明显,但是案例中最大执行时间仍不足SAG

的10 %,所以IIG是最具时间优势的算法.

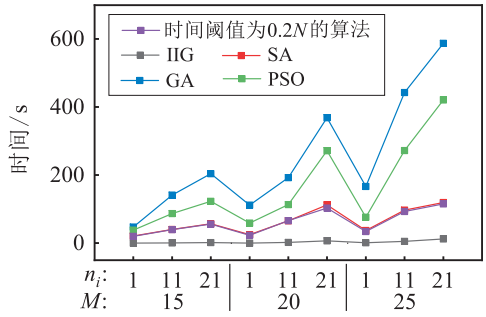


图 5　多种算法的平均求解时间

综上,SAG可以取得良好效果的原因主要有3个: 1) IIG作为初始解,使算法的搜索起点较好,为后期的邻域搜索奠定了基础; 2) ACP和MCP都是有方向性地搜索解的邻域,效果比随机的交叉、变异等方式强,提高解质量的效果明显; 3) 利用贪婪算子求解目标,减少了运算时间,增加了迭代次数. 因此,SAG可以在较短时间内获得较优解.

5　结　论

本文研究了考虑依赖工件位置的学习效应的单人作业车间调度问题,建立了混合整数规划模型. 为求解大规模问题,本文首先设计了考虑学习效应的贪婪算子,提出了随机插入和间隔插入贪婪算法,并基于贪婪算法和贪婪算子改进了模拟退火算法. 对比实验表明:混合整数规划模型适合需要精确解的小规模问题;间隔插入贪婪算法在求解时间上具有优势,效果良好,可以应用于有求解时间要求的问题;基于贪婪的模拟退火算法对间隔插入贪婪算法作为初始解和生成新解的改进较为有效,解决问题的能力有所提高,该方法适合各种规模的问题. 然而,实际生产作业中还存在单人多工序、多人单工序和多人多工序等更为复杂的车间调度问题,优化调度这些问题可以作为进一步研究的方向.

参考文献(References)

[1] Badiru A B. Computational survey of univariate and multivariate learning curve models[J]. IEEE Transactions on Engineering Management, 1992, 39(2): 176-188.

[2] Heizer J, Render B, Munson C. Operations management: sustainability and supply chain management[M]. The 12th edition. New York: Pearson Education, 2016: 777-778.

[3] Wright T P. Factors affecting the cost of airplanes[J]. Journal of the Aeronautical Sciences, 1936, 3(4): 122-128.

[4] Biskup D. Single-machine scheduling with learning considerations[J]. European Journal of Operational Research, 1999, 115(1): 173-178.

[5] Wang J Y. Minimizing the total weighted tardiness of overlapping jobs on parallel machines with a learning effect[J]. Journal of the Operational Research Society, 2020, 71(6): 910-927.

[6] Bai D Y, Tang M Q, Zhang Z H, et al. Flow shop learning effect scheduling problem with release dates[J]. Omega, 2018, 78: 21-38.

[7] Wu C C, Wang D J, Cheng S R, et al. A two-stage three-machine assembly scheduling problem with a position-based learning effect[J]. International Journal of Production Research, 2018, 56(9): 3064-3079.

[8] Agnetis A, Mosheiov G. Scheduling with job-rejection and position-dependent processing times on proportionate flowshops[J]. Optimization Letters, 2017, 11(4): 885-892.

[9] 胡金昌, 吴耀华, 吴颖颖, 等. 考虑学习效应的最小化延误总时间的单机批次排序问题[J]. 控制与决策, 2019, 34(12): 2708-2712.
(Hu J C, Wu Y H, Wu Y Y, et al. Batch scheduling with learning effect on single-machine to minimize the total tardiness[J]. Control and Decision, 2019, 34(12): 2708-2712.)

[10] Sun L, Yu A J, Wu B. Single machine common flow allowance group scheduling with learning effect and resource allocation[J]. Computers & Industrial Engineering, 2020, 139: 106126.

[11] 闫杨, 王大志, 汪定伟, 等. 具有恶化效应和学习效应的单机成组调度问题[J]. 自动化学报, 2009, 35(10): 1290-1295.
(Yan Y, Wang D Z, Wang D W, et al. Single machine group scheduling problems with the effects of deterioration and learning[J]. Acta Automatica Sinica, 2009, 35(10): 1290-1295.)

[12] 马卫民, 孙丽, 宁磊, 等. 加工时间带恶化和指数学习效应的成组排序[J]. 系统工程理论与实践, 2017, 37(1): 205-211.
(Ma W M, Sun L, Ning L, et al. Group scheduling with deterioration and exponential learning effect processing times[J]. Systems Engineering—Theory & Practice, 2017, 37(1): 205-211.)

[13] Pei J, Liu X B, Liao B Y, et al. Single-machine scheduling with learning effect and resource-dependent processing times in the serial-batching production[J]. Applied Mathematical Modelling, 2018, 58: 245-253.

[14] Pei J, Cheng B Y, Liu X B, et al. Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup

time[J]. Annals of Operations Research, 2019, 272(1/2): 217-241.

[15] Soleimani H, Ghaderi H, Tsai P W, et al. Scheduling of unrelated parallel machines considering sequence-related setup time, start time-dependent deterioration, position-dependent learning and power consumption minimization[J]. Journal of Cleaner Production, 2020, 249: 119428.

[16] Mousavi S M, Mahdavi I, Rezaeian J, et al. An efficient bi-objective algorithm to solve re-entrant hybrid flow shop scheduling with learning effect and sep times[J]. Operational Research, 2018, 18(1): 123-158.

[17] Tayebi Araghi M E, Jolai F, Rabiee M. Incorporating learning effect and deterioration for solving an SDST flexible job-shop scheduling problem with a hybrid meta-heuristic approach[J]. International Journal of Computer Integrated Manufacturing, 2014, 27(8): 733-746.

[18] Zhang G H, Sun J H, Liu X, et al. Solving flexible job shop scheduling problems with transportation time based on improved genetic algorithm[J]. Mathematical Biosciences and Engineering, 2019, 16(3): 1334-1347.

[19] Ahmadizar F, Shahmaleki P. Group-shop scheduling with sequence-dependent set-up and transportation times[J]. Applied Mathematical Modelling, 2014, 38(21/22): 5080-5091.

[20] Zhang J, Ding G F, Zou Y, et al. Review of job shop scheduling research and its new perspectives under Industry 4.0[J]. Journal of Intelligent Manufacturing, 2019, 30(4): 1809-1830.

[21] 吴慧, 王冰.基于预防维护的单机调度问题[J]. 控制与决策, 2021, 36(2): 395-402.
(Wu H, Wang B. Single-machine scheduling problem with preventative maintenance activities[J]. Control and Decision, 2021, 36(2): 395-402.)

[22] 潘全科, 王文宏, 朱剑英, 等. 基于粒子群优化和变邻域搜索的混合调度算法[J]. 计算机集成制造系统, 2007, 13(2): 323-328.
(Pan Q K, Wang W H, Zhu J Y, et al. Hybrid heuristics based on particle swarm optimization and variable neighborhood search for job shop scheduling[J]. Computer Integrated Manufacturing Systems, 2007, 13(2): 323-328.)

[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] Ribas I, Companys R, Tort-Martorell X. Comparing three-step heuristics for the permutation flow shop problem[J]. Computers & Operations Research, 2010, 37(12): 2062-2070.

[25] Shao W S, Shao Z S, Pi D C. Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem[J]. Knowledge-Based Systems, 2020, 194: 105527.

[26] Fernandez-Viagas V, Framinan J M. A best-of-breed iterated greedy for the permutation flowshop scheduling problem with makespan objective[J]. Computers & Operations Research, 2019, 112: 104767.

[27] Li X P, Yang Z, Ruiz R, et al. An iterated greedy heuristic for no-wait flow shops with sequence dependent setup times, learning and forgetting effects[J]. Information Sciences, 2018, 453: 408-425.

[28] Öztop H, Tasgetiren M F, Eliiyi D T, et al. Metaheuristic algorithms for the hybrid flowshop scheduling problem[J]. Computers & Operations Research, 2019, 111: 177-196.

[29] Meng L L, Zhang C Y, Shao X Y, et al. MILP models for energy-aware flexible job shop scheduling problem[J]. Journal of Cleaner Production, 2019, 210: 710-723.

[30] Xu J Y, Wu C C, Yin Y Q, et al. An order scheduling problem with position-based learning effect[J]. Computers & Operations Research, 2016, 74: 175-186.

作者简介

胡金昌 (1992−), 男, 博士生, 从事生产调度、物流系统仿真和优化的研究, E-mail: 201620367@mail.sdu.edu.cn;

吴颖颖 (1985−), 女, 讲师, 博士, 从事物流系统仿真和优化等研究, E-mail: sophia.wu@sdu.edu.cn;

王艳艳 (1978−), 女, 副教授, 博士, 从事智能物流、系统优化等研究, E-mail: yanyan.wang@sdu.edu.cn;

吴耀华 (1963−), 男, 教授, 博士生导师, 从事自动拣选、仓储系统设计和优化等研究, E-mail: mike.wu@263.net.

(责任编辑：李君玲)