控制与决策
Control and Decision

混合迭代贪婪算法求解准时生产分布式流水线调度问题

钱斌, 刘获飞, 胡蓉, 张梓琪

引用本文:
钱斌,刘获飞,胡蓉,张梓琪. 混合迭代贪婪算法求解准时生产分布式流水线调度问题[J]. 控制与决策, 2022, 37(11): 3042–3051.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2021.0426

混合迭代贪婪算法求解准时生产分布式流水线调度问题

钱 斌†, 刘获飞, 胡 蓉, 张梓琪

(昆明理工大学 信息与自动化学院，昆明 650500)

摘 要: 针对以最小化总延迟时间为优化目标的分布式置换流水线问题(distributed permutation flowshop scheduling problem, DPFSP), 建立问题排序模型, 并提出混合迭代贪婪算法(hybrid iterated greedy, HIG)进行求解. 基于问题特点提出最小工期差值(smallest due date difference value, SDV)规则及3种工厂分配规则, 同时结合问题性质提出两种工件插入各工厂内部时问题目标值的下界估计方法. 首先, 通过实验确定使用分配规则1将工件向各工厂进行分配, 同时结合下界估计方法的NEH作为改进启发式算法以生成较高质量初始解; 其次, 为了增加解的多样性, 提出一种关键工厂的移除策略和适用于问题的模拟退火机制; 然后, 设计基于4种有效邻域操作的两阶段变邻域下降搜索策略, 用于在HIG每代中对问题解空间的不同区域进行较深入和细致的搜索; 最后, 通过仿真实验和算法比较验证了采用HIG求解所提出问题的有效性.

关键词: 分布式流水线调度； 总延迟时间； 混合迭代贪婪算法； 下界

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2021.0426

引用格式: 钱斌, 刘获飞, 胡蓉, 等. 混合迭代贪婪算法求解准时生产分布式流水线调度问题[J]. 控制与决策, 2022, 37(11): 3042-2051.

Hybrid iterated greedy algorithm for just in time distributed permutation flowshop scheduling problem

QIAN Bin†, LIU Di-fei, HU Rong, ZHANG Zi-qi

(School of Information Engineering and Automation，Kunming University of Science and Technology，Kunming 650500，China)

Abstract: In this paper, for the distributed permutation flowshop scheduling problem, which is optimized to minimize the total tardiness, a problem sorting model is established, and a hybrid iterated greedy (HIG) algorithm is proposed to solve the problem. Based on the characteristics of the problem, this paper proposes the minimum due date difference value (SDV) rule and three kinds of factory assignment rules. At the same time, combining with the nature of the problem, this paper proposes two methods to estimate the lower bound of the target value of the problem when the job is assigned to each factor. At the same time, combining with the nature of the problem, two methods for estimating the lower bound of the target value of the problem are proposed. Firstly, through experimental analysis, it is determined that the SDV is used as a encoding rule, and the NEH combined with the lower bound estimation method is used as an improved heuristic algorithm to generate a higher quality initial solution. Secondly, in order to increase the diversity of solutions, a key factory removal strategy and simulated annealing mechanism are proposed. Then, the design based on four two-stage variable neighborhood descent search strategy for effective neighborhood operation is used to search different regions of the problem solution space in depth and detail in each generation of the HIG. Finally, simulation experiments and algorithm comparison verify the effectiveness of HIG in solving this problem.

Keywords: distributed permutation flowshop scheduling； total tardiness； hybrid iterated greedy； low bound

0 引 言

近年来, 生产制造开始从集中式向分布式进行转变, 多工厂生产环境下的生产调度问题, 即分布式调度问题受到越来越多的关注[1]. 分布式置换流水线调度问题(distributed permutation flowshop scheduling problem, DPFSP)是对传统置换流水线调度问题(permutation flowshop scheduling problem, PFSP)的延展, 即每个工厂都是一个置换流水车间. DPFSP常

收稿日期: 2021-03-15； **录用日期:** 2021-08-09.

基金项目: 国家自然科学基金项目(62173169, 61963022, 51665025)； 云南省基础研究重点项目(202101AS070097).

责任编委: 唐加福.

†通讯作者. E-mail: bin.qian@vip.163.com.

以最小化最大完工时间(makespan)和最小化总流经时间(total flowtime time, TF)为优化目标,两种优化目标既涉及机器的平衡使用,又涉及工件的快速处理,但不符合准时生产要求. 在实际生产过程中,客户要求交付时间已确定,生产企业需在规定时间内完成客户需求,但生产企业因某些因素延迟交货时常发生,导致产生成本提高,客户满意度下降. 因此总延迟时间(total tardiness, TT)最小化逐步成为一个重要的评价指标[2]. 它既涉及工件的快速加工处理,又考虑工件工期按时完工. 针对以最小化总延迟时间为优化目标的PFSP已有部分学者进行了研究. Karabulut[3]设计了一种具有随机选择邻域搜索策略的迭代贪婪算法(K iterated greedy, KIG). Fernandez-Viagas[4]通过对问题分析发现, $\mathrm{NEH_{edd}}$启发式算法迭代过程对于解的质量有很大限制,提出了8种具有关联打破机制的改进启发式算法以提高初始解的质量. Framinan等[5]设计了一种基于工期松弛特性的启发式算法与变邻域搜索局部搜索策略的混合算法(hybrid algorithm, HA). Cura[6]为了避免种群收敛速度过快,设计了一种改进进化算法(evolutionary algorithm, EA). Vallada等[7]设计了一种针对有效路径重连技术的遗传算法(genetic algorithms, GA)进行求解. 随着经济全球化的加速,越来越多的企业意识到快速响应客户需求的重要性,中心化生产已很难满足社会需求. 因此,研究以最小化总延迟时间为优化目标的DPFSP更具有现实的经济价值. 数学上,总延迟时间问题已被证明属于NP-hard问题[8], DPFSP问题也被证明属于NP-hard问题[9],故本文研究问题亦属于NP-hard问题. 在问题求解层面,该问题具有大规模、强耦合、不确定等复杂性. 因此,对其展开研究具有重要的学术意义和工程应用价值.

DPFSP因其应用的重要性和理论的复杂性受到众多学者的广泛关注和研究. 对于DPFSP,通常以最小化最大完工时间和最小化总流经时间为优化目标. 对于以最小化最大完工时间为优化目标, Naderi等[9]提出了6种混合整数线性规划模型和2种分配规则,进而基于规则提出了14种启发式算法和2种变邻域下降算法(variable neighborhood descent, VND). Gao等[10]针对关键工厂设计一种禁忌搜索算法(tabu search, TS)对完工时间最长与最短的工厂执行交换操作. Wang等[11]采用置换编码,提出了基于最短完工时间工厂分配规则分布估计算法(estimation of distribution algorithm, EDA). Fernandez等[12]由插入工件到某一工厂后完工时间的下界,设计有界搜索迭代贪婪算法(bounded search iterated greedy algorithm, BSIG)以提高搜索效率. Naderi等[13]设计了一种在生成解阶段由完整解和分配工厂子集两个集合共同作用以增加解的多样性的离散搜索算法(scatter search, SS). Bargaoui等[14]设计了一种化学反应优化算法(chemical reaction optimization, CRO). Ruiz等[15]设计了一种针对关键工厂和全部工厂执行插入操作的两阶段嵌套IG算法(IG2S)进行求解. 对于以最小化总流经时间为优化目标, Fernandez-Viagas等[16]提出了6种工厂分配规则和18种启发式算法以获得优质初始解,并设计EA算法进一步求解. Pan等[17]在Fernandez提出的分配规则的基础上提出了3种构造启发式算法,同时设计4种元启发式算法进行求解. 综上所述,现今对于分布式置换流水线模型各类优化目标的研究非常广泛,但以总延迟时间为优化目标的DPFSP研究还十分有限.

IG算法是由Jacobs等[18]提出的一种结构简单、参数少、快速有效的智能优化算法. 在每一次迭代过程中, IG算法从当前解出发,通过迭代贪婪操作对当前解进行破坏和重新构造,进而产生一系列新解,若新解中有更好的解则对最好解进行更新,同时采用类似模拟退火的接受准则更新当前解. IG算法针对不同的优化目标和不同的流水线调度问题都展现出其优异的性能[17,19]. 最近, IG算法被用于求解各种优化目标DPFSP. Fernandez等[12]和Ruiz等[15]针对以最小化完工时间为优化目标的DPFSP,分别设计了BSIG和IG2S算法进行求解. Lin等[20]针对以最小化最大完工时间为优化目标的DPFSP,提出一种改进IG算法(modifiied iterated greedy, MIG)以增加解的质量. Pan等[17]针对以最小化总流经时间为优化目标的DPFSP,设计一种简单而有效的插入邻域搜索作为IG算法局部搜索策略进行求解.

VND算法是一种简单有效的元启发式算法,由Hansen等[21]最先提出. VND对优质解空间各邻域进行交替地、循环地、系统地搜索,广泛应用于发现高质量的解或对解空间中的潜在区域进行更深入地搜索. VND已成功应用于多种组合优化问题[22-25],其一般用于执行迭代局部搜索阶段,通常以确定性的方式探索多个邻域结构(通常从小邻域到大邻域)[26]. 在生产调度问题中, VND算法在生产调度领域得到了越来越多的研究和应用, Gao等[27]针对以最小化最大完工时间为优化目标的柔性作业车间问题,设计VND和GA混合算法,将VND代替原有的局部操作以加快对解的评价速度并提高解的质量. Tasgetiren

等[28]针对以最小化最大完工时间为优化目标满足阻塞条件的流水线问题,设计结合IG和具有概率选择局部搜索操作功能的VND混合算法进行求解. Peng等[29]考虑到实际生产中的机器故障,提出一种混合解码策略和3种邻域结构的VND算法进行求解.

本文主要研究DPFSP的建模与求解. 在建模方面,建立以最小化总延迟时间为优化目标的DPFSP数学模型. 在求解方面,考虑到DPFSP这类问题解空间巨大且复杂,常规智能算法难以在短时间内实现有效搜索,故设计一种混合迭代贪婪算法(hybrid iterated greedy, HIG)进行求解. 首先,基于问题性质提出两种具有不同复杂度的插入工件下界估计定理,以加快评价生成解的适应度的效率. 在生成初始解阶段,考虑到本文优化目标与工件工期有非常强的关联性,提出最小工期差规则(smallest due date difference value, SDV),同时提出3种工厂分配规则用于工件到工厂的分配. 采用与下界估计定理相结合的改进NEH[30]算法快速生成高质量的初始解. 在移除和重构阶段设计一种针对关键工厂工件的移除策略和基于改进NEH算法的重构机制构建新解. 首先,移除策略与改进模拟退火接受机制共同作用,以增加解的多样性,对解空间中不同的区域进行搜索,从而避免算法过早收敛;然后,在局部搜索阶段,设计一种具有下界定理判断的两阶段VND迭代搜索策略,同时提出基于插入和交换邻域算子的加速机制,以实现对问题解空间中的优质问题解区域进行细致且高效的搜索;最后,通过仿真实验和算法比较验证所提出算法的有效性.

1 问题描述及性质分析

1.1 问题模型

DPFSP_TT可描述如下:将N个工件$J=\{J_1,J_2,\dots,J_N\}$分配给f个工厂进行加工,每个工厂具有相同的M台机器,$M=\{M_1,M_2,\dots,M_m\}$. 每个工件J_i可在任意一个工厂依次完成m道加工操作$O_{j,m}=\{O_{1,m},O_{2,m},\dots,O_{j,m}\}$. 每个工件$J_j$的工期为$d_j$. 假设所有工件相互独立且在零时刻均可加工;机器连续可用,即不考虑机器故障等因素;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;每个工件可以分配到任意工厂,且工厂分配一旦确定便不能改变;同一工厂的各个机器上加工的工件序列相同,每个工件加工的工序也相同;工件在机器上移动的时间和机器的设置时间忽略不计(或包含在加工时间内). 图1给出了上述问题的一个具体实例($N=5, M=3, F=2$).

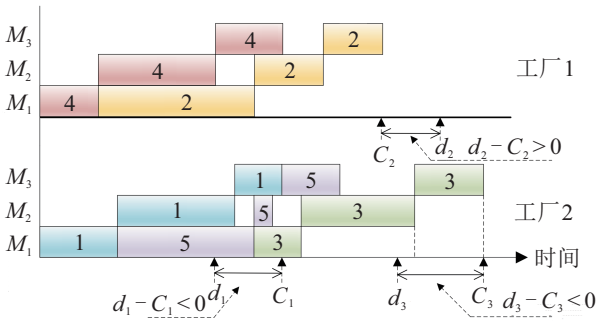


图1 DPFSP_TT示例图

根据上述定义,建立如下模型,优化目标为在所有产品排序的集合Π中找到一个最优排序π^*,使得$\mathrm{TT}(\pi)$最小:

$$\mathrm{TT}(\pi^*)=\min_{\pi\in\Pi}\mathrm{TT}(\pi); \tag{1}$$

$$C_{j,m}=\max(C_{j-1,m},C_{j,M-1})+P_{j,m},\quad j=1,2,\dots,N,\ m=1,2,\dots,M; \tag{2}$$

$$C_j=C_{j,m},\ m=M; \tag{3}$$

$$\mathrm{TT}_f(\pi^f)=\sum_{j=1}^{N}T_j=\sum_{j=1}^{N}\max\{C_j(\pi^f)-d_j,0\},\quad j=1,2,\dots,N,\ f=1,2,\dots,F; \tag{4}$$

$$\mathrm{TT}(\pi)=\sum_{f=1}^{F}\mathrm{TT}_f(\pi^f),\ f=1,2,\dots,F. \tag{5}$$

式(2)表示工件J_i在机器m的完工时间;式(3)表示工件J_i的完工时间;式(4)表示工厂f的总延迟时间;式(5)表示所有工厂的总延迟时间.

1.2 问题分析

本文求解以最小化总延迟时间为优化目标的DPFSP,与常见的两种优化目标不同,本文优化目标与工期有非常密切的联系. 根据工期和工件完工时间的关系可分为两种情况:1)松工期;2)紧工期.

1.2.1 松 工 期

松工期,即每一个工件工期都大于或等于最大完工时间($d_j\geqslant\max C_j$). 本文优化目标为$\min\mathrm{TT}(\pi)$,由式(4)可知,对于任意工厂,总延迟时间为

$$\mathrm{TT}_f(\pi^f)=\sum_{j=1}^{N}\max\{C_j(\pi^f)-d_j,0\}=\sum_{j=1}^{N}\max\{-D,0\}=0,$$

D为非负数. 故$\mathrm{TT}(\pi)=0$,即所得任意一个解均为可行解.

1.2.2 紧 工 期

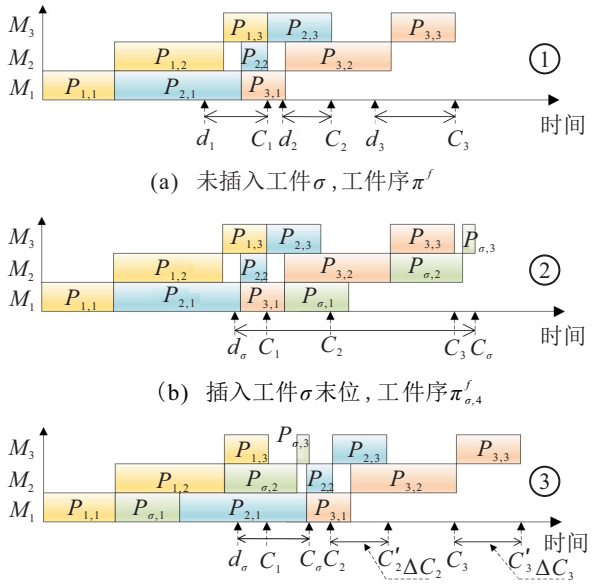
紧工期,即每一个工件的工期都小于或等于完工时间($C_j\geqslant d_j$). 本文的优化目标为$\min\mathrm{TT}(\pi)$,对应

于每个工厂 $\min \mathrm{TT}_f(\pi^f)$. 由于工期 d_j 已经被确定, 由下式可知:

$$\mathrm{TT}_f(\pi^f)=\sum_{j=1}^{N}T_j=\sum_{j=1}^{N}\max\{C_j(\pi^f)-d_j,0\}=\sum_{j=1}^{N}(C_j-d_j)=\sum_{j=1}^{N}C_j-\sum_{j=1}^{N}d_j=\sum_{j=1}^{N}C_j-\mathrm{const},\tag{6}$$

问题可以转化为求解最小化总流经时间 $\min \mathrm{TF}(\pi)$.

如图 2 所示, 在紧工期条件下, 将工件 J_σ 分配给已有部分工件序的工厂 f 的位置 k. 图 2(a) 表示未放入工件 J_σ 的工件序 π^f 与每个工件 d_j; 图 2(b) 表示工件 J_σ 插入 π^f 末位, 得到新工件序 $\bar{\pi}^f_{\sigma,k}$ 和工件 J_σ 的 d_σ; 图 2(c) 表示工件 J_σ 插入 π^f 非末位, 得到新工件序 $\bar{\pi}^f_{\sigma,k}$ 和工件 J_σ 的 d_σ, 以及在工件 J_σ 之后工件新的完工时间与完工时间差 ΔC_j.



(a) 未插入工件 σ, 工件序 π^f

(b) 插入工件 σ 末位, 工件序 $\pi^f_{\sigma,4}$

(c) 插入工件 σ 非末位, 工件序 $\pi^f_{\sigma,k}$, $k=1,2,3$

图 2　紧工期下工件插入工厂位置甘特图

引理 1　当工件 J_σ 插入位置 k 时, 得到新的序列 $\pi^f_{\sigma,k}$, 位于位置 k 之后的每个工件 J_j 的完工时间至少增加 $\min\limits_{1\leqslant m\leqslant M}P_{\sigma,m}$, 推导得到

$$C_j(\pi^f_{\sigma,k})\geqslant C_{j-1}(\pi^f)+\min_{1\leqslant m\leqslant M}P_{\sigma,m}.$$

由引理 1 可以推出, 当工件 J_σ 插入某一工厂 f 某一位置 k 生成新的序列时, 位于 k 之后工件完工时间会改变. 因为 $d_j\leqslant C_j$, 完工时间改变导致工件 T_j 发生变换, 工件 T_j 增加, 增加量为 ΔC_j. 由此本文提出两种不同的工件序变化后的 TT_f 下界定理, 两种定理具有不同的时间复杂度.

定理 1　工件 J_σ 插入工厂 f 位置 k 时, 得到新工件序 $\bar{\pi}^f_{\sigma,k}(1\leqslant k\leqslant N_f+1)$, 则 $\mathrm{TT}_f^{\sigma,k}$ 下界 $\mathrm{LB}_1^{\mathrm{TT}_f^{\sigma,k}}$ 为

$$\mathrm{LB}_1^{\mathrm{TF}_f^{\sigma,k}}=\sum_{j=1}^{N}C_j+\sum_{m=1}^{M}P_{\sigma,m}+\min\left\{\min_{1\leqslant m\leqslant M}P_{\sigma,m};\sum_{j=1}^{N}P_{j,1}\right\},\tag{7}$$

$$\mathrm{LB}_1^{\mathrm{TT}_f^{\sigma,k}}=\mathrm{LB}_1^{\mathrm{TF}_f^{\sigma,k}}-\sum_{j=1}^{N+1}d_1=\mathrm{TT}(\pi_f)+\sum_{m=1}^{M}P_{\sigma,m}+\min\left\{\min_{1\leqslant m\leqslant M}P_{\sigma,m};\sum_{j=1}^{N}P_{j,1}\right\}-d_\sigma.\tag{8}$$

证明　$\sum\limits_{m=1}^{M}P_{\sigma,m}$ 为工件 J_σ 插入第 1 个位置的完工时间, 在每台机器上连续加工, 没有空闲时间. 所以当它插入工厂任意位置时, 它是工件 J_σ 加工时间的下界. 另外, 当工件 J_σ 插入工厂 f 位置 k 时, 位于 k 之后工件 T_j 增加量下界为 $\min\limits_{1\leqslant m\leqslant M}P_{\sigma,m}$.

情形 1: $k=N_f+1, \pi^f_{\sigma,N_f+1}=\{J_1,\ldots,J_N,J_\sigma\}$, 将工件 J_σ 插入末位, 有

$$\mathrm{TT}_f^{\sigma,k}\geqslant\sum_{j=1}^{N}C_1+\sum_{j=1}^{N}P_{j,1}+\sum_{i=1}^{m}C_{\sigma,i}-\sum_{j=1}^{N+1}d_1=\mathrm{TT}(\pi_f)+\sum_{j=1}^{N}P_{j,1}+\sum_{i=1}^{m}C_{\sigma,i}-d_\sigma=\mathrm{TT}(\pi_f)+T_\sigma.$$

情形 2: $k<N_f+1$, 将工件 J_σ 插入末位, $\pi^f_{\sigma,N_f+1}=\{J_1,\ldots,J_{k-1},J_\sigma\,J_k,\ldots,J_{N_f}\}$, 有

$$\mathrm{TT}_f^{\sigma,k}\geqslant\sum_{j=1}^{N}C_1+\sum_{j=1}^{k-1}P_{j,1}+\sum_{i=1}^{m}P_{\sigma,1}+(N_f+1-k)\min_{1\leqslant m\leqslant M}P_{\sigma,m}-\sum_{j=1}^{N+1}d_j=\mathrm{TT}(\pi_f)+\sum_{j=1}^{k-1}P_{j,1}+\sum_{i=1}^{m}P_{\sigma,i}-d_\sigma+(N_f+1-k)\min_{1\leqslant m\leqslant M}P_{\sigma,m}\geqslant\mathrm{TT}(\pi_f)+T_\sigma+\min_{1\leqslant m\leqslant M}P_{\sigma,m}.$$

□

在迭代过程中, 定理 1 的计算复杂度为 $O(1)$, 因为 $\min\limits_{1\leqslant m\leqslant M}P_{\sigma,m}$ 和 $\sum\limits_{m=1}^{M}P_{\sigma,m}$ 可以在运行程序之前根据数据计算得到, d_σ 已知, TT_f 和 $\sum\limits_{j=1}^{N}P_{j,1}$ 的值可由之前对工序 π^f 的计算求得.

定理 2　工件 J_σ 插入工厂 f 位置 k 时, 得到新工

件序$\bar{\pi}^f_{\sigma,k}(1\leqslant k\leqslant N_f+1)$,则$\mathrm{TT}^{\sigma,k}_f$下界$\mathrm{LB}_2^{\mathrm{TT}^{\sigma,k}_f}$为

$$\mathrm{LB}_2^{\mathrm{TF}^{\sigma,k}_f}=$$

$$\sum_{j=1}^{N}C_j+\min\Big\{\sum_{i=1}^{M}P_{\sigma,i}+N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m};$$

$$\min_{2\leqslant k\leqslant N_f+1}\Big\{\max\Big\{\sum_{l=1}^{k-1}P_{l,1}+\sum_{i=1}^{M}P_{\sigma,i};$$

$$C_{k-1}+P_{\sigma,M}\Big\}+(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m}\Big\}\Big\},\quad(9)$$

$$\mathrm{LB}_2^{\mathrm{TT}^{\sigma,k}_f}=\mathrm{LB}_2^{\mathrm{TF}^{\sigma,k}_f}-\sum_{j=1}^{N+1}d_j=$$

$$\mathrm{TT}(\pi_f)+\min\Big\{\sum_{i=1}^{M}p_{\sigma,i}+N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m};$$

$$\min_{2\leqslant k\leqslant N_f+1}\Big\{\max\Big\{\sum_{l=1}^{k-1}P_{l,1}+\sum_{i=1}^{M}P_{\sigma,i};C_{k-1}+P_{\sigma,M}\Big\}+$$

$$(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m}\Big\}\Big\}-d_\sigma.\quad(10)$$

证明 因为工件J_σ未插入位置k与插入工件σ在位置k之前工件的延迟时间相同,即$\mathrm{TT}^l_f(\pi^f)=\mathrm{TT}^l_f(\pi^f_{\sigma,k})$,所以有:

情形1: $k>1$,将工件J_σ插入非首位,$\pi^f_{\sigma,N_f+1}=\{J_1,\ldots,J_{k-1},J_\sigma\,J_k,\ldots,J_{N_f}\}$,有

$$\mathrm{TT}^{\sigma,k}_f\geqslant$$

$$\sum_{j=1}^{N}C_1+\max\Big\{\sum_{l=1}^{k-1}P_{l,1}+\sum_{i=1}^{M}P_{\sigma,i};C_{k-1}+P_{\sigma,M}\Big\}+$$

$$(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m}-\sum_{j=1}^{N+1}d_j\geqslant$$

$$\mathrm{TT}(\pi_f)+T_\sigma+(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m}.$$

情形2: $k=1$,将工件J_σ插入首位,$\pi^f_{\sigma,N_f+1}=\{J_\sigma,J_1,\ldots,J_{N_f}\}$,有

$$\mathrm{TT}^{\sigma,k}_f\geqslant$$

$$\sum_{j=1}^{N}C_1+\sum_{i=1}^{M}P_{\sigma,i}+N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m}-\min_{1\leqslant m\leqslant M}P_{\sigma,m}=$$

$$\mathrm{TT}(\pi_f)+T_\sigma+N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m}.\qquad\square$$

注1 在迭代过程中,定理2计算复杂度为$O(n)$,因为要测试每一个位置的下界定理,虽然定理2复杂度更高,但是定理2下界要比定理1更紧($\mathrm{LB}_1\leqslant\mathrm{LB}_2$).

注1证明如下:

证明 有

$$\mathrm{LB}_1^{\mathrm{TT}^{\sigma}_f,k}\leqslant\mathrm{TT}(\pi_f)+\sum_{m=1}^{M}P_{\sigma,m}+\min_{1\leqslant m\leqslant M}P_{\sigma,m}-d_\sigma,$$

$$\mathrm{LB}_2^{\mathrm{TT}^{\sigma,k}_f}\geqslant$$

$$\mathrm{TT}(\pi_f)+\sum_{m=1}^{M}P_{\sigma,m}+\min\Big\{N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m};$$

$$\min_{2\leqslant k\leqslant N_f+1}\Big\{\sum_{l=1}^{M}P_{l,1}+(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m}\Big\}\Big\}-d_\sigma.$$

令

$$A=\mathrm{TT}(\pi_f)+\sum_{m=1}^{M}P_{\sigma,m}+\min_{1\leqslant m\leqslant M}P_{\sigma,m}-d_\sigma,$$

$$B=$$

$$\mathrm{TT}(\pi_f)+\sum_{m=1}^{M}P_{\sigma,m}+\min\Big\{N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m};$$

$$\min_{2\leqslant k\leqslant N_f+1}\Big\{\sum_{l=1}^{M}P_{l,1}+(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m}\Big\}\Big\}-d_\sigma.$$

由于$\mathrm{LB}_1^{\mathrm{TT}^{\sigma}_f,k}\leqslant A\leqslant B\leqslant\mathrm{LB}_2^{\mathrm{TT}^{\sigma}_f,k}\Rightarrow A\leqslant B$,有

$$\min_{1\leqslant m\leqslant M}P_{\sigma,m}\leqslant N_f\min_{1\leqslant m\leqslant M}P_{\sigma,m}\leqslant$$

$$\sum_{l=1}^{M}P_{l,1}+(N_f-k+1)\min_{1\leqslant m\leqslant M}P_{\sigma,m},$$

$$2\leqslant k\leqslant N_f+1,\ N_f>0.\qquad\square$$

2 HIG算法

HIG算法将IG与VND相结合. IG算法包含移除阶段、重构阶段、局部搜索和接受解4个阶段. 移除阶段和接受解阶段一起增加解的多样性,使算法对解空间中不同的区域进行搜索. 在局部搜索阶段,设计一种两阶段VND迭代搜索策略代替单一的邻域搜索. 利用预定的邻域结构,系统地对优质解的排序模型解空间进行细致地搜索,对优质解进行挖掘的过程中得到不同局部的最优解.

2.1 解的表示

解的表示是提高算法效率的一个关键因素,为表示对工厂的工件分配,每一个解被表示为一个二维数组$\pi=\{\pi_1,\pi_2,\ldots,\pi_f\}$,数组中共有$f$行,其中每一行由分配给工厂的工件组成$\pi_k=\{\pi_{k,1},\pi_{k,2},\ldots,\pi_{k,n_k}\}$, $k=1,2,\ldots,f$, $j=1,2,\ldots,n_k$. n_k表示分配给工厂k的工件数,每个序列中的工件都按照所选分配规则对工件进行排序.

2.2 初始解生成

当前,在关于分布式流水线的文献中,大多数学者采用最大完工时间(largest processing time, LPT)规则对工件进行排序,优先处理加工时间较长的工件. 由于本文优化目标与工件的工期关系非常紧密,LPT规则不适用于本文优化目标. 最早工期规则(earliest

due date, EDD) 作为对工期调度问题的常用规则, 本文在此基础上提出最小工期差值 (smallest due date difference value, SDV) 规则.

2.2.1 最小工期差值(SDV)规则

SDV 规则不仅考虑工件的工期, 还考虑工件的加工时间. 首先, 计算出每个工件加工时间和工期差的绝对值 $V_j = \left| \sum_{m=1}^{M} P_{j,m} - d_j \right|$; 然后, 每个工件加工时间与工期差值 V_j 按升序进行排列, 得到序列 ζ, 若有相同差值, 则工期小的优先排列.

2.2.2 分配规则

本节提出了 3 种工厂分配规则, 这些分配规则将决定工件如何分配给工厂. 分配规则可以被看作是算法插入的一部分, 也可以被认为是具有不同搜索空间不同类型的邻域.

分配规则 1　对于序列 ζ 中的待分配工件 J_i, 分别放置到每一个工厂的所有可能位置, 计算所有工厂的总延迟时间, 将其分配到加工完成此工件总延迟时间最小的工厂 k^* 的最优位置 p^*.

分配规则 2　找出未放入工件 J_i 时总延迟时间最大的工厂 $f_{\max}$, 再将工件 J_σ 分别放置到除 $f_{\max}$ 以外每一个工厂的所有可能位置, 计算当前工厂的延迟时间, 将其分配到加工完成此工件总延迟时间最小的工厂 k^* 的最优位置 p^*.

分配规则 3　对于序列 ζ 中的待分配工件 J_i, 分别放置到每一个工厂的所有可能位置, 计算当前工厂的总延迟时间, 将其分配到加工完成此工件总延迟时间最小的工厂 k^* 的最优位置 p^*.

2.2.3 启发式算法

初始解的质量会影响 HIG 的性能, 因此利用有效的启发式算法生成的一个高质量初始解进行排序操作可以提高算法搜索性能. NEH 启发式算法是求解置换流水线调度问题最有效的启发式算法之一. 文献 [6,12] 已成功地将 NEH 启发式扩展到以最小化最大完工时间和最小化总流经时间为优化目标的 DPFSP. DNEH 是利用上述提出的简单规则并结合 2.1 节下界定理在 NEH 框架下提出的, 其步骤如下.

step 1: 将所有工件按 SDV 规则要求进行排序, 然后根据排列顺序依次分配工件.

step 2: 由所选分配规则, 将当前待分配工件 J_i 分配到当前 TT_f 最小工厂的最优位置, 保留使当前 TT_f 最小的序列, 记录工件 J_i 的位置 k 与插入后 TT_f.

step 3: 利用下界定理判断当工件 J_i 插入其他工厂 f 时的下界 $\mathrm{LB}^{\mathrm{TT}_f^{\sigma,k}}$, 若 $\mathrm{LB}^{\mathrm{TT}_f^{\sigma,k}} < \mathrm{TT}_f$, 则计算工件 J_i 插入 f_n 最优位置的 TT_{f^*}, 如果 $\mathrm{TT}_{f^*} < \mathrm{TT}_f$, 则记录当前工厂和所插入工件位置, 更新 TT_f, 直到所有工厂都被尝试.

step 4: 判断剩余工件数是否小于 5, 若小于 5, 则随机选取工件 J_i 最优位置 $k-1$ 或 $k+1$ 的工件 J_n, 将工件 J_n 插入当前工厂其他位置, 保留使得当前 TT_f 最小的序列 π_f, 否则执行 step 5.

step 5: 返回 step 2 直到所有工件都被分配.

2.3 移除阶段

移除阶段与模拟退火接受解阶段一起对优质解进行扰动, 其目的是提供解的多样性. 一般 IG 算法在移除阶段, 是从现有解决方案中随机地选择工件 (随机选择工厂提取工件) 并将其移除. 提取工件形成一个序列 $\lambda_s (s = 1, 2, \ldots, n)$. 本文提出一种针对关键工厂的移除策略, 但不会完全破坏关键工厂. 将所有工厂中 $\max\{\mathrm{TT}_f\}$ 的工厂 f 定义为关键工厂, 在关键工厂中随机选择 $n/2$ 个工件移除, 剩下的 $n/2$ 个工件随机从关键工厂以外的工厂中移除. 将移除工件存入 λ_s, 未移除的工件存入 π_D.

2.4 重构阶段

在重构阶段, 对 λ_s 中工件按工期进行升序排序, 再利用 DNEH 算法机制, 将 λ_s 中的工件利用下界定理加速插入机制, 依次放入到所有工厂的所有可能位置, 最终将工件 J_{λ_s} 插入工厂的最佳位置, 即总延迟时间增幅最小的位置 k. 直到所有被移除工件全部插入完毕, 重构新解.

2.5 局部搜索

对于组合优化问题, 每个最优解都是所有邻域结构下的局部最优解, 接近最优解的高质量解往往是多个邻域结构下的局部最优解. 采用单邻域操作的重复贪婪搜索过程容易达到并陷入相应邻域结构的局部最优解, 解的质量往往不够好. 即许多高质量的局部最优解分散在其解空间的非常深的区域. 因此, 为了增强局部搜索能力, 针对问题特点结合 4 种有效的邻域搜索结构, 设计两阶段变邻域下降搜索策略, 对有潜力区域或优质解空间进行深度挖掘.

2.5.1 工厂间局部搜索

1) F_{insert}: 从当前最优解 π 中提取工件 J_n, 重新插入到所有工厂 π^f 所有可能位置. 若所生成的邻域解中的最优解好于当前解 $(\mathrm{TT}(\pi^*) < \mathrm{TT}(\pi))$, 则将当前解替换为最佳邻域解. 重复这一过程, 直到所有工件都被考虑, 没有更优解生成.

2) F_{swap}: 从当前最优解 π 中提取工件 J_n, 与所有工厂 π^f 的所有工件 J_s^f 交换. 若所生成的邻域解中的

最优解好于当前解($\mathrm{TT}(\pi^*) < \mathrm{TT}(\pi)$),则将当前解替换为最佳邻域解. 重复这一过程,直到所有工件都被考虑,没有更优解生成.

在工厂间局部搜索时,工件J_n从工件序λ中选取,λ由当前最优解π^*按工厂顺序连接而成,$\lambda = \{\pi^*_{1,1},\dots,\pi^*_{1,N_1},\pi^*_{2,1},\dots,\pi^*_{f,1},\dots,\pi^*_{f,N_f}\}$. 采用下界定理对执行$F_{\mathrm{insert}}$操作时工件是否插入该工厂进行加速判断.

2.5.2 工厂内局部搜索

1) J_{insert}: 从工厂工件序π^f中提取工件J_n,重新插入到所在工厂中的所有可能位置. 若所生成的邻域解中的最优解好于当前解($\mathrm{TT}(\pi^*) < \mathrm{TT}(\pi)$),则将当前解替换为最佳邻域解. 重复这一过程,直到所有工件都被考虑,没有发现任何改进.

2) J_{swap}: 从工厂工件序π^f中提取工件J_n,与所在工厂中的所有工件进行交换. 若生成的邻域解中最优解好于当前解($\mathrm{TT}(\pi^*) < \mathrm{TT}(\pi)$),则将当前解替换为最佳邻域解. 重复这一过程,直到所有工件都被考虑,没有发现任何改进.

两阶段变邻域下降搜索步骤如下.

step 1: 对经过移除和重构阶段的新解执行工厂间邻域F_{insert}操作. 若获得更优解,则对最优解进行更新,执行step 3;否则,执行step 2.

step 2: 对未更新解执行工厂间邻域F_{swap}操作. 若获得更优解,则对最优解进行更新,执行step 3;否则退出局部搜索.

step 3: 对新更新的解执行J_{insert}邻域搜索. 若获得更好解,则更新解,继续进行此操作;若没有获得更优解,则进行step 4.

step 4: 对当前最优解,执行J_{swap}邻域搜索. 若获得更优解,则对最优解进行更新,返回step 3;若没有获得更优解,则返回step 1.

2.6 接 受 解

接受解阶段,将决定局部搜索后生成的新解是否成为下一次迭代的起点. 本文采用恒温的模拟退火接受准则[31],若当前所得最优解小于历史最优解,则对历史最优解进行更新;否则以下式的概率公式接受当前解作为新解:

$$P = \exp\Big(\frac{\mathrm{TT}(\pi^*) - \mathrm{TT}(\pi)}{T}\Big). \tag{11}$$

对于最小化总延迟时间为优化目标的PFSP,文献[3]提出了温度T的计算公式如下:

$$T = \beta \times \Big(\sum_{j=1}^{N}(\mathrm{LB}_{C_{\max}} - d_j)\Big/(N\times 10)\Big). \tag{12}$$

$$P = \exp\Big(\frac{\mathrm{TT}(\pi^*) - \mathrm{TT}(\pi)}{T}\Big).$$

其中: β为待调整参数,$\mathrm{LB}_{C_{\max}}$为最大完工时间下界.

对于以最小化总延迟时间为优化目标的DPFSP,目前还未见对应DPFSP固定工期的恒温模拟退火公式. 本文受式(12)的启发进行了相应的改进,以适应工期条件,有

$$T = \beta \times \frac{\sum_{j=1}^{N}(C_{\max}^{\mathrm{DNEH}} - d_j)}{N\times 10}. \tag{13}$$

其中: β是待调整参数,$C_{\max}^{\mathrm{DNEH}}$为使用NEH2算法生成的初始解中工厂完工时间最大值.

2.7 加速机制

2.7.1 Insert邻域操作快速评价

基于问题的性质,提出了一种通用工件插入加速扫描方法来加速邻域搜索过程. 从$\pi^f = \{\pi_1,\dots,\pi_k,\dots,\pi_n\}$的位置$k$提出工件$J_k$,插入任意位置$l$,得到新的序列$\bar{\pi}^f = \{\bar{\pi}_1,\dots,\bar{\pi}_{l-1},\bar{\pi}_k,\bar{\pi}_l,\dots,\bar{\pi}_n\}$. 可以得到$\pi_{l-1} = \bar{\pi}_{l-1}, T_{J_{l-1}} = T_{J'_{l-1}}, \sum_{k=1}^{l-1} T_{J_k} = \sum_{k=1}^{l-1} T_{J'_k}$. 由式(4)得出

$$\mathrm{TT}(\bar{\pi}^f) = \sum_{k=1}^{l-1} T'_{J_k} + \sum_{k=l}^{n} T'_{J_k} = \sum_{k=1}^{l-1} T_{J_k} + \sum_{k=l}^{n} T'_{J_k}. \tag{14}$$

在计算插入后的新工件序时,插入位置前工件序顺序未发生变化的工件T_{l-1}可以直接利用,不需要重新计算,只需计算插入位置之后,工件序发生改变部分的工件T_i即可. 使用该加速扫描方法,工厂内工件计算复杂度可由$O(n^2)$降到$O(n\log n)$.

2.7.2 swap邻域操作快速评价

遍历交换序列π中两个不同位置的工件所得的所有序列或邻域,利用下面的定理3和定理4进行操作有效性判定,并找到遍历过程中最好的解. 只有当改变后的总延迟时间缩短,邻域操作才是有效的.

定理3 同一工厂($f = f'$). 选择工件J_k与其同一工厂内工件J_l交换,得到新序$\bar{\pi}^f$. $\mathrm{TT}(\pi^f)$为交换前工厂f的总延迟时间,$\mathrm{TT}(\bar{\pi}^f)$为交换后工厂f的总延迟时间,令$\Delta T = \mathrm{TT}(\bar{\pi}^f) - \mathrm{TT}(\pi^f)$,当且仅当$\Delta T < 0$时,swap邻域操作才是有效的. 新调度解最小总延迟时间为

$$\mathrm{TT}(\bar{\pi}^f) = \mathrm{TT}(\pi^f) + \Big(\sum_{k=l}^{n} T_{J'_k} - \sum_{k=l}^{n} T_{J_k}\Big) = \mathrm{TT}(\pi^f) + \Delta T. \tag{15}$$

定理4 不同工厂($f \neq f'$). 从工厂J_k中选择工件J_k与任意工厂f'中工件J_l进行交换,得到新序$\bar{\pi}^f$.

$\mathrm{TT}(\pi)$为交换前的总延迟时间,$\mathrm{TT}(\bar{\pi})$为交换的总延迟时间. 令$\Delta T=\mathrm{TT}(\bar{\pi})-\mathrm{TT}(\pi)$,当且仅当$\Delta T<0$时,swap邻域操作才是有效的. 新调度解最小总延迟时间为

$$\begin{aligned}&\mathrm{TT}(\bar{\pi}^f)=\\&\mathrm{TT}(\pi^f)+\Big(\sum_{k=l_1}^{n_f-1}T_{J_k}(\bar{\pi}^f)-\sum_{k=l_1}^{n_f}T_{J_k}(\pi^f)\Big)+\\&\Big(\sum_{k=l_2}^{n_f+1}T_{J_k}(\bar{\pi}^f)-\sum_{k=l_2}^{n_f}T_{J_k}(\pi^f)\Big)=\mathrm{TT}(\pi^f)+\Delta T.\end{aligned}\tag{16}$$

基于邻域操作快速评价和对其有效性的分析,一方面,可以判断邻域操作是否有效,进而避免无效的搜索操作;另一方面,在执行有效操作后,可以根据机器TT的改变量快速计算出新解的TT. 提高算法局部搜索效率,减少计算量.

2.8　算法流程

根据上述描述,整个算法流程如图3所示.

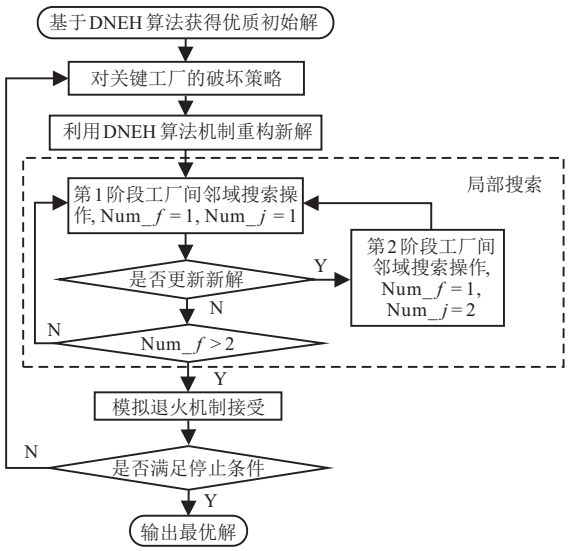


图 3　HIG算法流程

3　实验设计与分析

3.1　实验设置

所有实验均在Matlab 2018a编程实现,程序运行环境为Windows 10操作系统,2.60 GHz主频的CPU,Intel Core i7处理器,16 GB RAM的个人计算机.

为确定算法的参数和性能,实验测试问题从DPFSP标准数据集中随机选取. 工件数$n=\{20,50,100\}$,机器数$m=\{5,10,20\}$,工厂数$F=\{3,5,7\}$,共27个测试问题. 所有测试问题的确定加工时间详见http://soa.iti.es. 每个工件的工期生成如下所示:

$$d_j=[U\times(1-G-R/2),U\times(1-G+R/2)].\tag{17}$$

其中:U为使用NEH算法生成工件序每个工件完工时间,G为延迟因素,R为工期选择范围. 本文选择G, $R=0.2$.

3.2　参数设置

采用实验设计方法(design of experiment, DOE)考察参数对算法性能的影响. 对关键参数移除工件个数d,恒温退火参数β分别设置4个不同的参数. 为了避免结果出现过拟合现象,每组参数独立运行20次,以$N\times M\times 25$ ms为停止标准. 比较在不同参数下算法的性能,进而确定参数对性能的影响. 评价指标为平均相对百分偏差(average relative deviation, ARD),有

$$\mathrm{ARD}(\alpha)=\frac{\mathrm{TT}_{\mathrm{avg}}(\pi(\alpha))-\mathrm{ref}}{\mathrm{ref}}.\tag{18}$$

其中:$\mathrm{TT}_{\mathrm{avg}}$为$\mathrm{TT}(\pi(\alpha))$的平均值,ref为相应测试问题中运行$N\times M\times 25$ ms最优值. 通过实验分析,进而确定HIG的最佳关键参数组合为$\beta=2$, $n=6$.

3.3　仿真结果与统计分析

3.3.1　验证分配规则有效性

为了验证所提出SDV规则的有效性,分别将SDV与EDD两种规则和3种分配规则进行组合得到初始解,如表1所示.

表 1　各规则组合平均性能响应值

分配规则	ARD	分配规则	ARD
EDD_1	0.414	SDV_1	**0.410**
EDD_2	1.825	SDV_2	1.629
EDD_3	0.525	SDV_3	0.546

由表1可见,SDV规则的初始解质量优于EDD规则,其中SDV与分配规则1相结合得到解的质量最优. 表2为使用分配规则1的EDD和SDV两种规则结合下界定理所得的初始解,th1和th2分别为下界定理1和定理2. 可以看出,采用SDV与定理2相结合获得的初始解效果最好;只采用SDV获得的初始解次之;EDD与定理1相结合所获初始解效果最差. 通过实验验证了所提出的分配规则1、下界定理2和SDV规则相结合可以生成质量更高的初始解.

表 2　下界定理组合平均性能响应值

下界	ARD	下界	ARD
EDD	0.414	SDV	0.410
$\mathrm{EDD}_{\mathrm{th1}}$	0.456	$\mathrm{SDV}_{\mathrm{th1}}$	0.439
$\mathrm{EDD}_{\mathrm{th2}}$	0.412	$\mathrm{SDV}_{\mathrm{th2}}$	**0.403**

3.3.2　验证HIG算法的有效性

为验证HIG算法的有效性,采用IG[15]、DABC[17]、SS[13]三种在国际期刊中求解分布式流水线问题有效的算法与HIG算法进行比较,每种算法运行时间相

表3 算法对比结果

算例	HIG	DABC	SS	IG	算例	HIG	DABC	SS	IG
20 × 5 × 3	**0.033**	0.121	0.119	0.072	50 × 20 × 5	**0.034**	0.092	0.056	0.074
20 × 10 × 3	**0.023**	0.095	0.079	0.062	100 × 5 × 5	**0.065**	0.243	0.233	0.199
20 × 20 × 3	**0.010**	0.047	0.042	0.027	100 × 10 × 5	**0.042**	0.232	0.096	0.110
50 × 5 × 3	**0.064**	0.239	0.150	0.128	100 × 20 × 5	**0.032**	0.172	0.082	0.109
50 × 10 × 3	**0.158**	0.328	0.232	0.213	20 × 5 × 7	**0.012**	0.057	0.085	0.049
50 × 20 × 3	**0.053**	0.141	0.068	0.063	20 × 10 × 7	**0.005**	0.025	0.026	0.017
100 × 5 × 3	**0.057**	0.258	0.181	0.197	20 × 20 × 7	**0.006**	0.022	0.017	0.020
100 × 10 × 3	**0.114**	0.224	0.128	0.125	50 × 5 × 7	**0.032**	0.188	0.117	0.095
100 × 20 × 3	0.099	0.182	**0.090**	0.126	50 × 10 × 7	**0.049**	0.131	0.061	0.067
20 × 5 × 5	**0.013**	0.094	0.085	0.045	50 × 20 × 7	**0.027**	0.074	0.041	0.059
20 × 10 × 5	**0.008**	0.037	0.029	0.015	100 × 5 × 7	**0.067**	0.102	0.087	0.087
20 × 20 × 5	**0.004**	0.023	0.026	0.019	100 × 10 × 7	**0.016**	0.041	0.023	0.041
50 × 5 × 5	**0.056**	0.181	0.105	0.090	100 × 20 × 7	0.027	0.035	**0.019**	0.034
50 × 10 × 5	**0.075**	0.181	0.093	0.101					

同,运行停止标准均为$N \times M \times 25$ ms. 对27个测试问题均进行20次独立实验,表3为各算法的对比结果,评价指标为ARD. ARD值越小,算法所得总延迟时间性能越优. 每个问题对应最优结果用粗体表示.

由表3可见,所提出的HIG算法在ARD值方面的性能比其他3种算法要好得多. 在27个测试问题中,2个测试问题不占优,其余测试问题HIG可以实现几乎所有工厂配置的最小ARD值. IG算法和SS算法彼此接近,DABC算法在4种算法中表现最差. 图4为根据表3各算法对比结果,以算法类型作为考虑因素,采用95%置信区间Tukey's HSD检验的置信区间. 可以看出,HIG与DABC、SS和IG有显著差异.

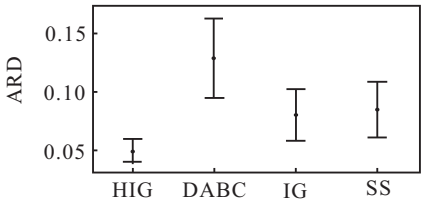


图4 对比算法均值95%置信区间

HIG使用SDV规则与下界定理2相结合可以获得较其他算法更优质的初始解. 由于问题的排序模型解空间是一个“巨大”且“极为扁平”的空间,在整个空间内遍布着大量深浅不一的山谷,即空间中大量不同的位置(对应不同的解)会具有相同的目标值. 因为解空间中解与解之间没有梯度,两个相似解之间目标值也可能差别较大,相较于其他3种算法单一的邻域操作,对优质解空间的搜索深度是有限的,HIG中使用4种邻域操作相结合的两阶段VND局部搜索,对问题解空间进行深入且细致的搜索,从而更易发现复杂解空间中的优质解. 同时,利用下界定理加速判断工件是否插入工厂和两种邻域搜索的加速机制,提高了算法效率. 综上,HIG可在上述测试问题中取得较好的结果.

4 结 论

研究以最小化总延迟时间为目标的DPFSP模型具有非常重要的理论和现实意义,本文根据问题特性推导了工件插入工厂后优化目标的下界估计公式,并对应下界定理进行复杂度分析,设计了一种混合迭代贪婪算法(hybrid iterated greedy, HIG)进行求解. 根据问题特性提出了两种邻域搜索快速评价加速机制,通过仿真实验验证了算法的有效性和高效性. 算法创新之处在于: 1) 针对问题特性提出SDV规则和分配规则,改进了启发式算法获得更优质的初始解; 2) 将下界定理与插入操作相结合并应用于算法中,以加速判断工件是否插入工厂,进而提高算法效率; 3) 相比于传统IG算法单一邻域搜索,设计了一种两阶段变邻域下降算法,并提出两种不同的邻域搜索加速机制,进一步改进了算法对于解空间搜索深度的能力和效率. 下一步工作将研究分布式装配流水线的准时生产问题,同时考虑各种约束问题,在更贴合实际生产的情况中找到更优的算法.

参考文献(References)

[1] Wang L, Deng J, Wang S Y. Survey on optimization algorithms for distributed shop scheduling[J]. Control and Decision, 2016, 31(1): 1-11.

[2] Raman N. Minimum tardiness scheduling in flow shops: Construction and evaluation of alternative solution approaches[J]. Journal of Operations Management, 1995, 12(2): 131-151.

[3] Karabulut K. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops[J]. Computers & Industrial Engineering, 2016, 98: 300-307.

[4] Fernandez-Viagas V, Framinan J M. NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness[J]. Computers & Operations Research, 2015, 60: 27-36.

[5] Framinan J M, Leisten R. Total tardiness minimization in permutation flow shops: A simple approach based on a variable greedy algorithm[J]. International Journal of

Production Research, 2008, 46(22): 6479-6498.

[6] Cura T. An evolutionary algorithm for the permutation flowshop scheduling problem with total tardiness criterion[J]. International Journal of Operational Research, 2015, 22(3): 366.

[7] Vallada E, Ruiz R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem[J]. Omega, 2010, 38(1/2): 57-67.

[8] Du J Z, Leung J Y T. Minimizing total tardiness on one machine is NP-hard[J]. Mathematics of Operations Research, 1990, 15(3): 483-495.

[9] Naderi B, Ruiz R. The distributed permutation flowshop scheduling problem[J]. Computers & Operations Research, 2010, 37(4): 754-768.

[10] Gao J, Chen R, Deng W. An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem[J]. International Journal of Production Research, 2013, 51(3): 641-651.

[11] Wang S Y, Wang L, Liu M, et al. An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem[J]. International Journal of Production Economics, 2013, 145(1): 387-396.

[12] Fernandez V V, Framinan J M. Abounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem[J]. International Journal of Production Research, 2015, 53(4): 1111-1123.

[13] Naderi B, Ruiz R. A scatter search algorithm for the distributed permutation flowshop scheduling problem[J]. European Journal of Operational Research, 2014, 239(2): 323-334.

[14] Bargaoui H, Belkahla Driss O, Ghédira K. A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion[J]. Computers & Industrial Engineering, 2017, 111: 239-250.

[15] Ruiz R, Pan Q K, Naderi B. Iterated Greedy methods for the distributed permutation flowshop scheduling problem[J]. Omega, 2019, 83: 213-222.

[16] Fernandez-Viagas V, Perez-Gonzalez P, Framinan J M. The distributed permutation flow shop to minimise the total flowtime[J]. Computers & Industrial Engineering, 2018, 118: 464-477.

[17] Pan Q K, Gao L, Wang L, et al. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem[J]. Expert Systems With Applications, 2019, 124: 309-324.

[18] Jacobs L W, Brusco M J. Note: A local-search heuristic for large set-covering problems[J]. Naval Research Logistics, 1995, 42(7): 1129-1140.

[19] 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.

[20] Lin S W, Ying K C, Huang C Y. Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm[J]. International Journal of Production Research, 2013, 51(16): 5029-5038.

[21] Hansen P, Mladenovi N. Variable neighborhood search: Principles and applications[J]. European Journal of Operational Research, 2001, 130(3): 449-467.

[22] Chen P, Huang H K, Dong X Y. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem[J]. Expert Systems With Applications, 2010, 37(2): 1620-1627.

[23] Zeng Z Z, Yu X G, He K, et al. Iterated Tabu Search and Variable Neighborhood Descent for packing unequal circles into a circular container[J]. European Journal of Operational Research, 2016, 250(2): 615-627.

[24] Zeng Z, Yu X, He K, et al. Adaptive tabu search and variable neighborhood descent for packing unequal circles into a square[J]. Application Soft Computer, 2018, 65: 196-213.

[25] Peng K K, Pan Q K, Gao L, et al. A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling[J]. Swarm and Evolutionary Computation, 2019, 45: 92-112.

[26] Li X Y, Gao L, Pan Q K, et al. An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 49(10): 1933-1945.

[27] Gao J, Sun L Y, Gen M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems[J]. Computers & Operations Research, 2008, 35(9): 2892-2907.

[28] Tasgetiren M F, Kizilay D, Pan Q K, et al. Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion[J]. Computers & Operations Research, 2017, 77: 111-126.

[29] Peng K K, Pan Q K, Gao L, et al. A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling[J]. Swarm and Evolutionary Computation, 2019, 45: 92-112.

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

[31] Stützle T. Applying iterated local search to the permutation flow shop problem[R]. AIDA-98-04, FG Intellektik, 1998.

作者简介

钱斌 (1976−), 男, 教授, 博士生导师, 从事智能优化调度、生产计划与调度等研究, E-mail: bin.qian@vip.163.com;

刘获飞 (1994−), 男, 硕士生, 从事智能优化调度的研究, E-mail: 417849507@qq.com;

胡蓉 (1974−), 女, 副教授, 博士, 从事优化方法及决策支持系统等研究, E-mail: ronghu@vip.163.com;

张梓琪 (1989−), 男, 博士生, 从事智能优化方法的研究, E-mail: Albert.ziqi@hotmail.com.

(责任编辑: 郑晓蕾)