引用:何轩,狄士璐,刘欢,等. 一种有效的数学启发式算法优化分布式置换流水车间调度问题[J]. 聊城大学学报(自然科学版),2025,38(4):475-484.

Citation: HE Xuan, DI Shilu, LIU Huan, DU Songlin, GUO Hengwei, SHAO Zhongshi. An effective matheuristic algorithm for distributed permutation flowshop problem[J]. Journal of Liaocheng University (Natural Science Edition),2025,38(4):475-484.

文章编号 1672-6634(2025)04-0475-10 **DOI** 10.19728/j.issn1672-6634.2024070003

一种有效的数学启发式算法优化分布式置换流水车间调度问题

何轩[1],狄士璐[2],刘欢[3],杜松霖[2],郭恒伟[2],邵仲世[4]

(1. 上海海事大学 物流工程学院,上海 201306;2. 上海大学 机电工程与自动化学院,上海 200093;
3. 上海理工大学 光电信息与计算机工程学院,上海 200072;4. 陕西师范大学 计算机科学学院,西安 710062)

摘要 近年来,企业的制造模式从传统的单工厂集中式生产转变为分布在不同地理位置的多工厂分布式协同生产。分布式置换流水车间调度问题应运而生。尽管学术界对该问题已经有了大量的研究,但研究其高效的求解方法仍然是一个开放性的话题。针对总流经时间优化目标,比较了现有的两个混合整数规划模型的优劣,并在小规模问题实例上获得了最优解。对于大规模问题实例,提出了一个有效的数学启发式算法。该算法在已有的 DLR-DNEH 启发式算法的基础上,设计了一个集合覆盖模型,用于收集插入邻域解中蕴含的有效搜索模式。最后,在标准测试集上的大量实验结果显示了提出算法的有效性。

关键词 分布式置换流水车间调度问题;总流经时间;数学启发式算法;集合覆盖模型

中图分类号 TH186;TP18 **文献标志码** A **开放科学(资源服务)标识码(OSID)**

An effective matheuristic algorithm for distributed permutation flowshop problem

HE Xuan[1], DI Shilu[2], LIU Huan[3], DU Songlin[2],
GUO Hengwei[2], SHAO Zhongshi

(1. School of Logistics Engineering, Shanghai Maritime University, Shanghai 201306, China; 2. School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China; 3. School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China;
4. School of Computer Science, Shaanxi Normal University, Xi'an; 710119, China)

Abstract In recent years, the manufacturing model of enterprises has changed from traditional single-factory centralized production to distributed collaborative production of multiple factories distributed in different geographical locations. Distributed permutation flowshop scheduling problem has emerged. Although a lot of research has been done on this problem in academia, studying its efficient solution method is still an open topic. This paper compares the advantages and disadvantages of two existing mixed integer pro-

收稿日期:2024-07-13

基金项目:中国博士后科学基金项目(2023M732166);国家自然科学基金项目(62406196)资助

通信作者:刘欢,男,汉族,博士,讲师,研究方向:智能优化与调度,演化计算,博弈对抗,E-mail:liuh@usst.edu.cn。

gramming models for the total flow time optimization objective, and obtains the optimal solution on a small-scale problem instance. For large-scale problem instances, this paper proposes an effective matheuristic algorithm. Based on the existing DLR-DNEH, the proposed matheuristic algorithm designs a set covering model to collect effective search patterns contained in insertion neighborhood structure. Finally, a large number of experimental results on the standard test set show the effectiveness of the proposed algorithm.

Keywords distributed permutation flowshop scheduling problem; total flow time; matheuristic algorithm; set covering model

0 引言

如今,传统的集中式制造缺乏灵活性,难以满足不断变化的市场需求,随着经济全球化趋势的不断增强,制造企业选择在不同地区建立工厂或生产中心,以提高生产效率并满足市场需求[1],以小批量生产和满足客户定制化需求的分布式制造正成为制造业的主流趋势[2]。从单工厂模式到多工厂模式需要考虑更复杂的生产调度问题。其中包括将工件分配到不同的工厂,并优化每个工厂内的生产流程,以实现最佳生产目标。2010 年 Naderi 和 Ruiz 首次提出分布式置换流水车间调度问题(DPFSP)问题[3],目前 DPFSP 已成为运筹学领域的一类热门话题[4]。

迭代贪心算法是一种简单且有效的解决组合优化问题的方法,已广泛应用于各种优化调度问题中[5-8]。迭代贪心算法包含两个参数:扰动因子和温度系数。这两个参数具有高敏感性[9],为了增强搜索过程中算法参数的适应性,文献[10-12]分别提出了三种不同的参数自适应迭代贪心算法。该算法可以根据其搜索阶段和搜索空间的变化自适应的调整扰动因子,从而提高迭代搜索效率。Lin[13]提出了一种改进的迭代贪心算法,包含了破坏阶段和重组阶段两个阶段,在破坏阶段采用了自适应的工件移除策略。2020 年,Huang 等人[6]提出一种新型迭代贪心算法来解决 DPFSP 问题,摒弃了传统迭代贪婪算法中常用的模拟退火接受准则,提出了一种具有六个不同运算符的重启方案,以维持解的多样性。并在破坏和重组阶段引入算法控制参数,平衡全局和局部搜索能力,进一步提升了算法的探索性能。迭代贪心算法是一类基于单个解的迭代优化算法,收敛速度快,但最终的优化结果通常受限于初始解的选择,容易陷入局部最优。

基于种群的演化算法通过维护多个候选解[14],能更好地探索全局最优解。2017 年,Deng 等人[15]提出一种竞争模因算法解决多目标分布式置换流水车间调度问题,以最小化最大完工时间和总延迟为目标,该算法利用两个种群分别优化两个目标,并为每个种群设计了适配的算子和种群交互机制。2022 年,Pan 等人[16]通过探索 DPFSP 问题的特定知识,提出了一种具有新型协作模型和种群初始化的协同进化算法,该算法基于反直觉悖论和两套加速方法大幅度节约了计算开销,取得了卓越的搜索性能。2023 年,Guo 等人[17]提出了一种离散果蝇优化算法来解决 DPFSP 问题,分别设计了移动算子、交换算子、混合算子和组合更新机制四种气味搜索策略,并提出了一种改进的局部搜索方法来提高果蝇的局部探索能力。2023 年,Yu 等人[18]提出了一种基于加速机制的人工蜂群优化算法,设计了三种构造性的种群初始方法,利用加速机制来评估邻域候选解,基于序列相关的问题特性改进了雇用蜂阶段、观察蜂阶段和侦察蜂阶段的搜索策略。2023 年,Guo 等人[19]提出了一种基于差分飞行策略的离散果蝇优化算法来解决 DPFSP 问题,在嗅觉探索阶段,作者设计了四种邻域扰动算子,为了避免算法陷入局部最优,摒弃了所有果蝇向最佳个体飞行的模式,提出了差分飞行策略,以更好地利用不同个体的信息。2024 年,Bai 等人[20]提出了一种新型的离散人工蜂群算法来解决 DPFSP 问题,针对问题特征利用分支定界算法,对候选解进行修剪并定义了下界和上界以减少解空间,并设计了可用性规则和放弃准则来提高算法搜索效率。相比于单个解的迭代优化算法,基于种群的演化算法可以更全面地搜索解空间[21],但计算开销更大,算法参数较多,调整和优化更复杂。

超启发式算法是一类高级的优化方法,旨在通过结合多种启发式或元启发式算法来解决复杂的优化问题[22]。2017 年 Lin 等人[23]提出了一种回溯搜索超启发式算法来解决 DPFSP 问题,设计了十个简单有效的启发式规则来构建一组低级启发式策略,并采用回溯搜索算法作为高级策略来操纵十个启发式规则在解空

间上进行搜索,算法的全局搜索能力得到验证。2023 年,Yu 等人[24]提出了一种基于 Q 学习的改进元启发式算法求解 DPFSP 问题,设计了四种元启发式算法,包括人工蜂群、粒子群优化、遗传算法和 Jaya 算法。开发了三种初始化策略来生成高质量的初始候选解。设计了四个局部搜索操作来提高算法的局部探索能力。利用 Q 学习在迭代过程中自适应的选择优质的局部搜索策略。超启发式算法通常具备自适应机制,能够根据问题的特性和搜索进程动态调整搜索策略,提高优化效率和效果,但算法的参数较多,设计和实现的复杂度较高。

在构造型启发式算法领域,大多数研究都是基于 NEH 算法和各种分配规则来构造候选解。2010 年 Naderi 和 Ruiz[3]以最小化最大完工时间为评估准则提出了六种不同的混合整数线性规划模型,十四种依赖于两个工厂分配规则和一个变量邻域下降方法的启发式方法,基于 DPFSP 问题的特点,进一步提出了 VND(a)和 VND(b)两种变邻域搜索方法。2011 年 Gao 和 Chen [25]通过改进 NEH2 提出了 NEH-df 方法,并提出了一种新的按组分配原则,即一次向工厂中插入一组工件。同年,Gao 和 Chen[26]提出了一种基于增强局部搜索的混合遗传算法,GA_LS 的算法性能明显优于 VND(a)。Wang 等人[27]提出了一种分布估计算法,Xu 等人[28]提出了一种混合免疫算法,但这两种算法在实验中只与 VND(a)进行了比较,并且都比 VND(a)消耗更长的 CPU 时间。2013 年,Gao 等人[29]提出了基于局部搜索的禁忌搜索算法,通过在工厂间交换子序列来生成邻域解,增强了禁忌搜索算法的局部探索能力。2020 年,Alper Hamzaday 等人[30]提出了一个有效的 Benders 分解算法来解决 DPFSP 问题,作者改进了 DPFSP 问题的整数规划模型,进一步限制了决策变量的作用范围,基于构建的数学模型提出了一种新的基于启发式规则和 Benders 分解的调度求解算法,更新了 18 个测试用例的最优解。2018 年 Fernandez-Viagas 等人[31]首次提出了以总流经时间为目标的 DPFSP 问题,提出了 18 种构造性启发式方法来创建高质量的初始解,利用两种加速机制节省计算开销。随后,Pan 等人[32]提出了 DLR、DNEH 和 DR-DNEH(x)三种启发式方法,以及离散人工蜂群、分散搜索、迭代局部搜索和 IG 四种元启发式方法。基于构建性的启发式算法通过逐步构建解的方式,能够在每一步提供部分解,从而简化问题的复杂性,可以灵活地适应不同的约束和目标,具有构建逻辑简单、容易实现、求解效率高等优势。

综上所述,本文的研究动机有两点:(1) 现有文献较少关注最小化 DPFSP 的总流经时间目标,未提出该问题的混合整数规划模型并对其求解效率进行定量化的评估。(2) 研究高效的启发式算法找到大规模实例的最优解或者更高质量的近似最优解,目前仍然是一个开放性的话题。鉴于此,本文在已有的算法的基础上,提出一个高效的数学启发式算法,该算法是 DLR-DNEH(x)和集合覆盖模型的集成。其中,集合覆盖模型用于收集插入邻域结构产生的大量候选解中的有效的搜索模式,从而构造一个完整解,达到优化解的质量的效果。本文的创新点总结如下:(1) 针对小规模问题,提出了两个混合整数规划模型,并使用 Cplex 求解器获得了最优解。同时,分析比较了两个模型的优劣。(2) 针对大规模问题,提出了一个基于 DLR-DNEH(x)和集合覆盖模型的构造型数学启发式算法,其中集合覆盖模型用于插入邻域结构中有效搜索模式。

1　问题描述和数学模型

DPFSP 正式描述如下:n 个工件 $N=\{1,2,\cdots,n\}$,在 f 个工厂中加工 $F=\{1,2,\cdots,f\}$。所有工厂是相同的,每个工厂由 m 台机器 $M=\{1,2,\cdots,m\}$ 构成一个流水车间。每个工件必须被分配到一个工厂中加工。在每个工厂中,所有的工件必须以相同的路径加工,即首先在机器 1 上加工,然后在机器 2 上加工,依次类推,直到在机器 m 上加工完成。工件 $j \in N$ 在机器 $i \in M$ 上的工序被记作 $O_{j,i}$,加工时间被记作 $p_{j,i}$。工序 $O_{j,i}$ 一旦开始加工,在 $p_{j,i}$ 时间段内不允许被中断。$p_{j,i}$ 在任何一个工厂中是相同的。所有工件和所有机器在 0 时刻均是可用的。在任何时刻一个工件只能在一台机器上加工,一台机器最多只能加工一个工件。需要优化的问题是同时确定工件分配到哪个工厂加工以及工厂内工件的排序,使得所有工件的完工时间之和最小。为此,本文提供两个混合整数规划模型,分别是基于位置的模型和基于工件序关系的模型。

1.1　基于位置的模型

为了便于描述基于位置的模型(记作模型 1),首先就涉及的符号进行说明:$i=1,2,\cdots,m$ 表示机器索

引,其中 m 为机器数;$j,j'=0,1,2,\cdots,n$ 为工件索引,其中 n 为工件数,0 表示虚拟工件;$l=1,2,\cdots,f$ 为工厂索引,其中 f 为工厂数。$p_{j,i}$ 表示工序 $O_{j,i}$ 的加工时间;h 表示一个足够大的整数;$X_{l,k,j}$ 为决策变量,如果工厂 l 中位置 k 加工工件 j,则 $X_{l,k,j}=1$,否则 $X_{l,k,j}=0$;$C_{l,k,j}$ 表示工厂 l 中工件 j 在第 k 个位置加工的完工时间;d_j 表示工件 j 的完工时间;TFT 表示优化目标。

模型 1:

$$\min \mathrm{TFT} \geqslant \sum_{j=1}^{n} d_j, \tag{1}$$

$$s.t.$$

$$\sum_{f=1}^{F}\sum_{k=1}^{n} X_{f,k,j}=1, j=1,\cdots,n, \tag{2}$$

$$\sum_{f=1}^{F}\sum_{j=1}^{n} X_{f,k,j}=1, k=1,\cdots,n, \tag{3}$$

$$C_{l,1,1} \geqslant \sum_{j=1}^{n} X_{l,1,j} \cdot p_{j,1}, l=1,\cdots,f, \tag{4}$$

$$C_{l,k,i} \geqslant C_{l,k,i-1} + \sum_{j=1}^{n} X_{l,k,j} \cdot p_{j,i}, l=1,\cdots,f, k=1,\cdots,n, i=2,\cdots,m, \tag{5}$$

$$C_{l,k,i} \geqslant C_{l,k-1,i} + \sum_{j=1}^{n} X_{l,k,j} \cdot p_{j,i}, l=1,\cdots,f, k=2,\cdots,n, i=1,\cdots,m, \tag{6}$$

$$d_j \geqslant C_{l,k,m} + (X_{l,k,j}-1)\times h, l=1,\cdots,f, k=1,\cdots,n, j=1,\cdots,n。 \tag{7}$$

公式(1)定义了目标函数。约束(2)表示每个工件只能被分配到一个工厂的一个位置进行加工。约束(3)表示一个工厂的每一个位置只能由一个工件所占据。约束(4)表示每个工厂第一个位置上加工的工件在首台机器上的完工时间。约束(5)表示每个位置的工序只有在其前一个工序完成后才能被加工。约束(6)表示相邻位置的工件在同一台机器上加工,只有前一个位置的工件在该机器上加工完后才能加工后一个位置上的工件。约束(7)表示每个工件的完工时间。

1.2 基于工件序关系的模型

为了描述基于工件序关系的模型(记作模型 2),本小节额外定义了两个决策变量:$x_{j',j}$,如果工件 j' 是工件 j 的直接前驱,$x_{j',j}=1$,否则等于 $x_{j',j}=0$;$c_{j,i}$ 表示工序 $O_{j,i}$ 的完工时间。

模型 2:

$$\min \mathrm{TFT} = \sum_{j=1}^{n} c_{j,m}, \tag{8}$$

$$s.t.$$

$$\sum_{j'=0,j'\neq j}^{n} x_{j',j}=1, j=1,\cdots,n, \tag{9}$$

$$\sum_{j=0,j\neq j'}^{n} x_{j',j}=1, j'=1,\cdots,n, \tag{10}$$

$$\sum_{j=1}^{n} x_{0,j}=f, \tag{11}$$

$$\sum_{j'=1}^{n} x_{j',0}=f, \tag{12}$$

$$x_{j',j}+x_{j,j'} \leqslant 1, j'=1,\cdots,n, j=1,\cdots,n, j'\neq j, \tag{13}$$

$$c_{j,1} \geqslant p_{j,1} + (x_{0,j}-1)\times h, j=1,\cdots,n, \tag{14}$$

$$c_{j,i} \geqslant c_{j,i-1} + p_{j,i} + (x_{0,j}-1)\times h, j=1,\cdots,n, i=2,\cdots,m, \tag{15}$$

$$c_{j,i} \geqslant c_{j',i} + p_{j,i} + (x_{j',j}-1)\times h, j'=1,\cdots,n, j=1,\cdots,n, j'\neq j, i=1,\cdots,m, \tag{16}$$

$$c_{j,i} \geqslant c_{j,i-1} + p_{j,i}, j=1,\cdots,n, i=2,\cdots,m。 \tag{17}$$

公式(8)定义了目标函数。约束(9)表示每个工件有且只有一个直接前驱。约束(10)表示每个工件有且

只有一个直接后继。约束(11)将虚拟工件 0 放置在调度序列的开头,表示调度序列的开始。约束(12)将虚拟工件 0 放置在调度序列的结尾,表示调度序列的结束。约束(13)表示消除非法子环约束。约束(14)表示首个加工工件在第一台机器上的完工时间。约束(15)表示首个加工工件在非首台机器上的完工时间。约束(16)表示相邻工件在同一台机器上加工,只有前一个工件在该机器上加工完后才能加工后一个工件。约束(17)表示每个工序只有在其前一个工序完成后才能被加工。

1.3　两个模型的比较

1.3.1　定性分析。求解整数规划模型可以获得小规模问题的最优解,但模型并不能降低问题的复杂性。不同的建模方式或者思路只是对问题的描述角度不同。然而,求解器 Cplex 对同一问题的不同模型的求解效率是不同的,这归因于变量和约束的不同。模型 1 和模型 2 的定性比较结果如表 1 所示,可以看出尽管模型 1 的约束数量少于模型 2,但模型 1 的变量数多于模型 2。此外,在模型 1 被求解之前,工件在工厂中的位置以及工件之间的邻接关系是不确定的。因此,模型 1 无法引入工件之间序相关的机器设置时间。相比之下,模型 2 的决策变量反映了工件之间的邻接关系,因此可以引入工件之间序相关的机器设置时间,从而更精确地模拟实际生产过程中的动态变化。此外,模型 2 在求解过程中更好地利用了问题结构的特点,提高了求解效率。然而,模型 1 或许有利于进行 Dantzig-Wolfe 分解,因为模型 1 被分解后的主问题是不相关的并行机调度问题,其模型是基于工件在工厂中的位置建立的。总之,虽然两种模型适用于不同的场景,但模型 2 在反映工件之间序相关的机器设置时间方面具有明显优势。

表 1　两个模型的定性比较

Tab. 1　Qualitative comparison of the two models

	Number of constraints	Number of decision variables
Model 1	$fn^2+2fnm-f(n+m)+2n+f+1$	$fn^2+2nm+n+1$
Model 2	$(m+1)n^2+nm+n+4$	$(n+1)^2+nm+1$

1.3.2　定量分析。为了定量测试两个模型,本文构造了一个小规模的问题实例。设 $f=2$, $n=8$, $m=3$,工件加工时间如表 2 所示。最优调度解的甘特图如图 1 所示,工厂 1 中的工件的加工序列为 5-1-4-7,工厂 2 中工件的加工序列为 6-2-8-3,该实例的目标值都为 108。由此看出,两个模型对该实例求解的最优解一致,互相印证了两个模型的正确性。此外,Cplex 求解器在几个不同规模实例上(当 $m=2$ 时,加工时间取表 2 中 M_1 和 M_2 上的加工时间)求解的两个模型的结果如表 3 所示,从模型对该实例的求解时间上来看,模型 2 所需时间显著少于模型 1 所需时间。

表 2　工件在机器上的加工时间

Tab. 2　The processing times of the jobs on the machines

Job	M_1	M_2	M_3
J_1	3	4	3
J_2	4	2	3
J_3	2	8	2
J_4	4	3	4
J_5	1	3	4
J_6	2	4	2
J_7	3	4	4
J_8	4	2	2

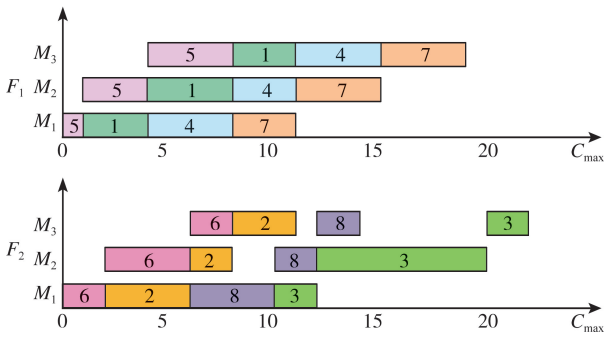

图 1　一个调度解的甘特图

Fig. 1　Gantt chart of optimal solution

2　数学启发式算法

提出的数学启发式算法(记作 DLR-DNEH-CS)由三部分组成:DLR-DNEH、插入邻域搜索算子(计算复杂度是 $O(n(n-1+f))$)和集合覆盖模型。本文首先使用 DLR-DNEH 产生一个调度解 π,然后插入邻域搜索算子将每个工件从 π 中取出,重新插入到具有最小目标值的位置处,从而产生 n 个候选解。最后集合

覆盖模型在这 n 个候选解中构造出一个完整解。

表 3 Cplex 解两个模型的结果

Tab. 3 Results of Cplex solving two models

Instances	Model 1		Model 2	
	TFT	Time/s	TFT	Time/s
$n=8$, $m=2$, $f=2$	83	1.70	83	44.80
$n=8$, $m=2$, $f=3$	83	1.61	83	44.53
$n=8$, $m=3$, $f=2$	108	5.39	108	78.16
$n=8$, $m=3$, $f=3$	94	4.02	94	334.48

2.1 DLR-DNEH

DLR-DNEH 是一个构造型启发式算法,结合了 DLR 和 DNEH 启发式算法的优点。具体来说,首先使用 DLR 生成部分工件的解决方案,然后使用 DNEH 逐个调度剩余工件。其中参数 x 用于控制 DNEH 调度的工件数。DLR 生成部分工件调度序列时,使用到的指标函数 $\mathrm{IF}_{j,n_{k*}}$ 如下所示

$$\mathrm{IF}_{j,n_{k*}}=(n/f-n_{k*}-2)\,\mathrm{IT}_{j,n_{k*}}+c_{j,m}\ , \tag{18}$$

$$\mathrm{IT}_{j,n_{k*}}=\sum_{i=2}^{m}\frac{m\times\max\{c_{j,i-1}-c_{[n_{k*}],i},0\}}{i+n_{k*}\times(m-i)/(n/f-2)}\ , \tag{19}$$

式中 n_{k*} 是工厂 $k*$ 中已经被调度的工件个数,$[n_{k*}]$ 表示工厂 $k*$ 中部分工件调度序列的最后一个工件。$\mathrm{IT}_{j,n_{k*}}$ 表示工件 j 调度在工件 $[n_{k*}]$ 后,机器产生的带权空闲时间之和。

DLR-DNEH 的运行机制被简要描述如下:首先,根据 $\mathrm{IF}_{j,0}$ 的值从大到小对所有的工件进行排序,产生一个排列 $\lambda=(\lambda_1,\lambda_2,\cdots,\lambda_n)$。将排列 λ 中的前 f 个工件依次分配到每个工厂产生部分工件调度序列 π,即 $\pi=\{\pi_1,\pi_2,\cdots,\pi_f\}$,其中 $\pi_k=(\lambda_k)$。然后将剩余的工件 $(\lambda_{f+1},\lambda_{f+2},\cdots,\lambda_n)$ 分配到工厂的过程如下:首先找到具有最小最大完工时间的工厂 k^*,然后将剩余的每个工件 j 拼接在工件 $[n_{k*}]$ 的后面,计算工件 j 的指标函数值 $\mathrm{IF}_{j,n_{k*}}$。最后将具有最小指标函数值的工件调度在工件 $[n_{k*}]$ 的后面,依次类推,直到排列 λ 中剩余 x 个工件。最后的 x 个工件将被使用 NEH 插入的方式插入在部分调度序列 π 中。根据文献[32],$x=0.3n$ 时,DLR-DNEH 的搜索效果最佳。上述 DLR-DNEH 算法的计算复杂度为 $O(\frac{1}{2}n(n+1)+nf)$。

2.2 集合覆盖模型

插入是一种有效改进解的邻域搜索算子,被广泛使用解决调度问题。这归因于插入算子具有低计算时间复杂度。然而,现有文献中使用插入算子改进当前解的质量时,在产生的 n 个候选解中仅选择一个具有最小目标值的候选解。然而,其它候选解中或许蕴含着潜在有效搜索模式未被利用。因此,本文设计一个集合覆盖模型用于收集其它候选解中的有效搜索模式,该模型被简要描述如

$$\min\sum\nolimits_{s\in H_{\mathrm{S}}}\mathrm{TFT}_s x_s\ , \tag{20}$$

$$s.t.$$

$$\sum\nolimits_{s\in H_{\mathrm{S}}}a_{j,s}x_s=1,j\in\{1,2,\cdots,n\}, \tag{21}$$

$$\sum\nolimits_{s\in H_{\mathrm{S}}}x_s=f, \tag{22}$$

$$x_s\in\{0,1\},s\in H_{\mathrm{S}}。 \tag{23}$$

式中 s 表示一个工厂中工件调度序列,TFT_s 是工件调度序列 s 的总流经时间,H_{S} 是所有候选解组成的一个集合,a 是一个二维的二进制系数矩阵。如果工件 j 在调度序列 s 中,则 $a_{j,s}=1$,否则 $a_{j,s}=0$。x_s 是一个二进制的决策变量,如果工件调度序列 s 被使用则 $x_s=1$,否则 $x_s=0$。优化目标被定义在公式(20)中。约束(21)是指每个工件必须被分配到选择的工件调度序列中。约束(22)是指需要在集合 H_{S} 中选择 f 个工件调度序列,从而组成一个完整的调度解。

3　实验结果

为了验证提出的数学启发式算法的有效性，本文在著名的分布式置换流水车间的 Taillard 标准测试集上进行仿真实验。同时，实现了五种典型的构造型启发式算法，包括 DNEH，DLR，DNEH-RZ，DNEH-SPD，DLR-DNEH-CS 与提出的算法的结果进行对比。由于构造型启发式算法控制参数是固定的，并且每次运行结果唯一，因此，不需要执行参数校准实验以及多次独立运行检验算法性能的稳定性。所有算法均采用 C++语言在 Microsoft Visual Studio 2017 编译。实验结果如表 4～6 所示，其中第一列表示算例的规模，如符号“20×5”表示该算例具有 20 个工件和 5 台机器。表中的行数据表示算法在对应实例规模上获得的总流经时间值的平均值。总流经时间值越小，对应的算法性能越好。加粗的数据表示对应算法在该实例上的结果最优。在为了方便在表格中展示数据，DNEH，DLR，DNEH-RZ，DNEH-SPD，DLR-DNEH-CS 分别被记作 A_1，A_2，A_3，A_4，A_5。

从表 4 到表 6 的数据可以看出，所提出的算法在几乎所有实例中均取得了优异的结果，验证了其搜索性能显著优于其他对比算法。这主要归功于基于集合覆盖模型的构造解的方法，使得母体算法 DLR-DNEH 以较低的计算时间代价，显著的改进了解的质量。

为了更进一步评估 DLR-DNEH-CS 与其他对比算法之间，是否存在搜索性能上的显著性差异，本文实施 Friedman 非参数检验对所有算法进行排名。并使用 Bonferroni-Dunn 过程作为事后程序，用于避免多重比较的谬误。表 7 总结了 Friedman 检验获得的五种算法的平均排名。可以看出，所提出的 DLR-DNEH-CS 在不同工厂数的所有实例中均排名第一。进一步地，Bonferroni-Dunn 过程中的关键显著性差异 C_D (Critical Difference，CD)计算如

$$C_D = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{23}$$

式中 k 表示所有算法的个数，N 是实例的个数，q_α 是一个参数。根据查表可知，当 $\alpha=0.1$ 时，$q_\alpha=1.2443$；当 $\alpha=0.05$ 时，$q_\alpha=1.1240$。若 DLR-DNEH-CS 的秩值加上 C_D 的值仍然小于对比算法的秩值，则表明 DLR-DNEH-CS 是严格优于对比算法的。图 2 清晰的展示了 Bonferroni-Dunn 过程的结果。可以看出，对比算法的柱状图均超过图中两条直线的位置。因此，这个结果再次严格证实提出的 DLR-DNEH-CS 获得的结果显著地优于其他对比算法的结果。

表 4　所有算法的结果($f=2$，$f=3$)

Tab. 4　Results of all algorithms ($f=2$，$f=3$)

Instances	$f=2$					$f=3$				
	A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4	A_5
20×5	9 532	9 491	9 418	9 518	**9 411**	7 847	7 911	7 790	7 839	**7 773**
20×10	15 140	15 003	14 928	15 072	**14 849**	13 111	13 160	13 000	13 096	**12 982**
20×20	26 996	26 764	26 810	26 888	**26 521**	24 387	24 360	24 156	24 347	**24 090**
50×5	41 353	39 897	40 528	41 314	**39 682**	31 178	30 323	30 684	31 163	**30 106**
50×10	58 425	58 009	57 862	58 425	**57 483**	47 224	46 884	46 764	47 108	**46 456**
50×20	91 476	90 096	90 019	91 406	**89 197**	77 349	76 764	76 431	77 240	**75 946**
100×5	142 303	134 538	138 946	142 303	**134 507**	101 575	96 932	99 762	101 549	**96 842**
100×10	182 663	177 660	179 567	182 663	**176 055**	138 319	136 645	136 514	138 316	**135 335**
100×20	256 030	254 821	254 321	256 030	**252 768**	207 041	205 631	205 393	206 988	**204 044**
200×10	614 915	591 722	607 012	614 915	**587 007**	444 605	434 204	439 340	444 605	**430 889**
200×20	781 672	774 749	774 949	781 672	**767 322**	598 008	59 3354	593 024	598 008	**588 549**
500×20	3 904 123	3 807 403	3 869 943	3 904 123	**3 787 454**	2 814 740	2 756 460	2 792 574	2 814 740	**2 744 832**

表 5 所有算法的结果($f=4,f=5$)

Tab. 5 Results of all algorithms ($f=4,f=5$)

Instances	$f=4$					$f=5$				
	A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4	A_5
20×5	**6 989**	**7 056**	**6 940**	6 985	6 947	6 503	6 584	6 470	6 489	**6 460**
20×10	12 104	12 185	**12 011**	12 091	12 024	11 465	11 587	**11 406**	11 464	11 430
20×20	22 985	22 858	22 797	22 930	**22 730**	22 158	22 173	22 029	22 147	**21 963**
50×5	26 036	25 541	25 753	25 996	**25 405**	22 953	22 661	22 713	22 903	**22 528**
50×10	41 256	41 225	40 912	41 238	**40 749**	37 791	37 590	37 487	37 745	**37 328**
50×20	70 206	69 745	69 548	70 123	**69 064**	65 907	65 370	65 287	65 798	**64 784**
100×5	81 500	78 692	80 402	81 500	**78 528**	69 454	67 363	68 612	69 439	**67 275**
100×10	116 052	114 535	114 744	116 031	**113 619**	102 731	101 510	101 681	102 689	**100 682**
100×20	181 510	179 858	180 113	181 363	**178 786**	165 408	163 888	164 089	165 283	**163 071**
200×10	358 492	350 615	354 812	358 492	**348 509**	307 322	302 075	304 763	307 322	**300 006**
200×20	504 252	499 444	500 187	504 252	**496 192**	446 555	441 601	443 522	446 555	**439 471**
500×20	2 269 055	2 226 541	2 252 345	2 269 055	**2 216 454**	1 935 410	1 898 210	1 921 419	1 935 410	**1 891 833**

表 6 所有算法的结果($f=6,f=7$)

Tab. 6 Results of all algorithms ($f=6,f=7$)

Instances	$f=6$					$f=7$				
	A_1	A_2	A_3	A_4	A_5	A_1	A_2	A_3	A_4	A_5
20×5	**6 157**	**6 365**	**6 131**	6 157	6 136	5 931	6 161	5 897	5 931	**5 919**
20×10	11 109	11 333	**11 046**	11 105	11 066	10 785	11 355	**10 722**	10 785	10 814
20×20	21 608	21 887	21 519	21 608	**21 503**	21 188	22 240	**21 082**	21 183	21 219
50×5	20 996	20 708	20 762	20 953	**20 552**	19 469	19 411	19 312	19 454	**19 281**
50×10	35 408	35 354	35 166	35 397	**34 949**	33 650	33 552	33 356	33 629	**33 224**
50×20	62 797	62 548	62 307	62 654	**62 007**	60 693	60 316	60 201	60 659	**59 725**
100×5	61 544	59 768	60 919	61 538	**59 641**	55 529	54 316	55 026	55 520	**54 187**
100×10	93 719	92 677	92 856	93 701	**92 081**	87 169	86 377	86 471	87 072	**85 627**
100×20	154 514	153 161	153 284	154 424	**152 146**	146 780	145 333	145 783	146 752	**144 380**
200×10	273 440	267 849	270 743	273 436	**266 379**	247 794	244 224	245 992	247 794	**242 681**
200×20	406 786	401 953	403 803	406 786	**400 093**	378 336	373 963	375 500	378 274	**372 326**
500×20	1 710 533	1 682 448	1 699 318	1 710 533	**1 676 830**	1 547 636	1 521 297	1 537 968	1 547 636	**1 516 159**

表 7 所有算法的秩次

Tab. 7 Ranks of all algorithms

Algorithm	Average rank					
	$f=2$	$f=3$	$f=4$	$f=5$	$f=6$	$f=7$
DNEH	4.37	4.27	4.35	4.35	4.22	4.10
DLR	2.56	2.97	2.92	2.93	3.16	3.19
DNEH-RZ	2.55	2.35	2.36	2.37	2.31	2.24
DNEH-SPD	4.20	4.04	4.08	4.00	4.02	3.86
DLR-DNEH-CS	1.33	1.38	1.28	1.35	1.30	1.61
$\alpha=0.05$	1.244 3					
$\alpha=0.1$	1.124 0					

4　结论

本文研究了一个分布式置换流水车间调度问题，旨在优化工件的总流经时间目标。为此，提出了两个不同的混合整数规划模型，即基于位置的模型和基于工件邻接关系的模型，并采用定性分析和定量分析的方法评估模型的优劣。定性分析的结果显示不同的建模思路有不同好处。定量分析的结果显示基于工件邻接关系的模型更容易被 Cplex 求解，其运行时间明显少于基于位置模型的求解时间。对于大规模算例，本文在已有的 DLR-DNEH(x)启发式算法的基础上，提出了一个集合覆盖模型，用于收集插入邻域搜索算子产生的邻域解中蕴含的有效搜索模式。大量实验结果证实了该集合覆盖模型的有效性，它显著改善了 DLR-DNEH(x)启发式算法的结果。在未来的研究中，我们将持续关注大规模车间调度问题以及高效的数学启发式算法的研究进展。

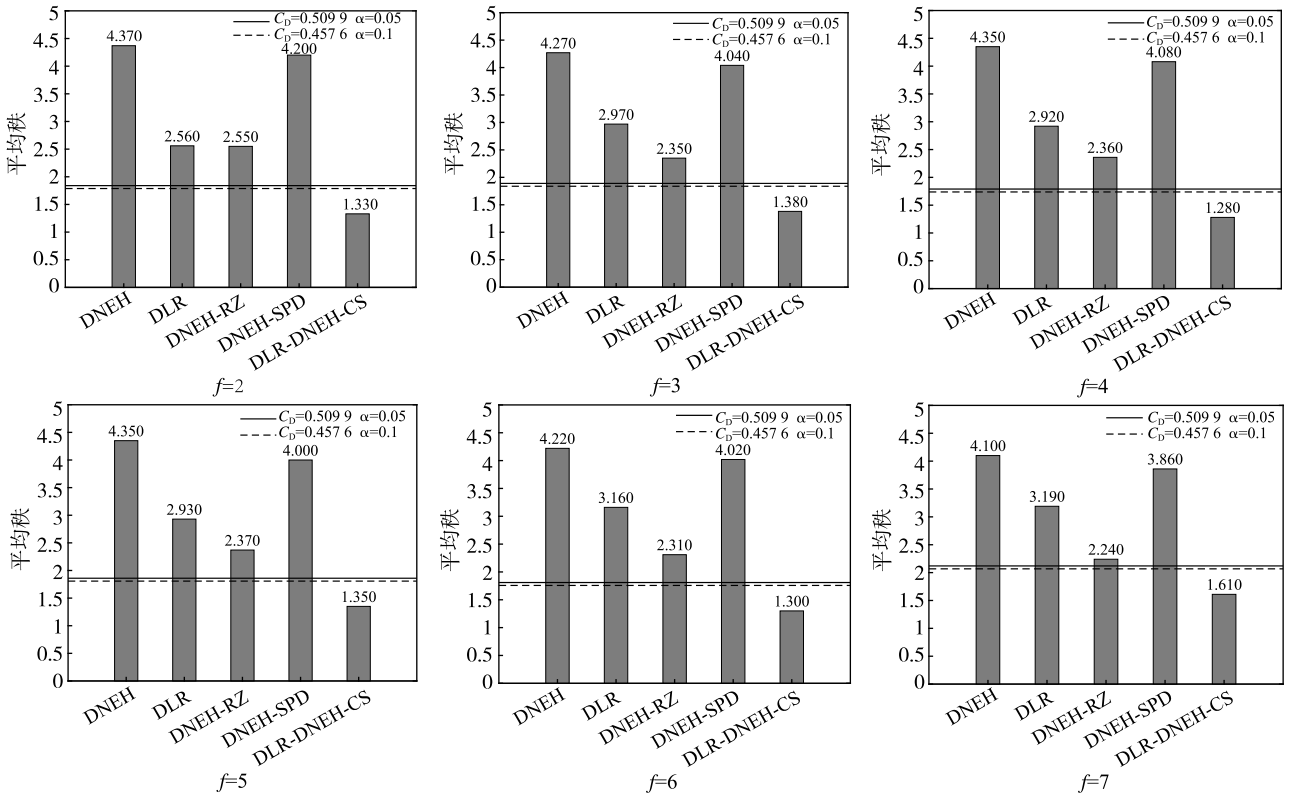

图 2　Friedman 测试和 Bonferroni-Dunn 的结果

Fig. 2　The result of Friedman's test and Bonferroni-Dunn process

参　考　文　献

[1] PAN Q K, GAO L, LI X Y, et al. Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem[J]. Applied Soft Computing, 2019, 81: 105492.

[2] OKWUDIRE C E, MADHYASTHA H V. Distributed manufacturing for and by the masses[J]. Science, 2021, 372(6540): 341-342.

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

[4] 张梓琪，钱斌，胡蓉，等. 基于多维 EDA 算法的低碳分布式装配流水车间调度[J]. 控制与决策，2022，37(5)：1367-1377.

[5] FERNANDEZ-VIAGAS V, FRAMINAN J M. A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem[J]. International Journal of Production Research, 2015, 53(4): 1111-1123.

[6] HUANG J P, PAN Q K, GAO L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Swarm and Evolutionary Computation, 2020, 59: 100742.

[7] PAN Q K, RUIZ R. An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem[J]. Omega, 2014, 44: 41-50.

[8] RUIZ R, PAN Q K, NADERI B. Iterated Greedy methods for the distributed permutation flowshop scheduling problem[J]. Omega,

2019, 83: 213-222.

[9] 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.

[10] LI Y Z, PAN Q K, LI J Q, et al. An Adaptive iterated greedy algorithm for distributed mixed no-idle permutation flowshop scheduling problems[J]. Swarm and Evolutionary Computation, 2021, 63: 100874.

[11] MAO F Y, LIU X Y, ZHAO H M. An adaptive population-based iterative greedy algorithm for optimizing the maximum completion time of hybrid flow shop [C]// In Proceedings of the 5th International Conference on Control and Computer Vision (ICCCV '22). Association for Computing Machinery, New York, NY, USA, 2022,187-192.

[12] MISSAOUI A, RUIZ R. A parameter-less iterated greedy method for the hybrid flowshop scheduling problem with setup times and due date windows[J]. European Journal of Operational Research, 2022, 303(1): 99-113.

[13] 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.

[14] 黄佳琳, 张丫丫, 顾幸生. 基于改进生物地理学优化算法的分布式装配置换流水车间调度问题[J]. 华东理工大学学报(自然科学版), 2020, 46(6): 758-769.

[15] DENG J, WANG L. A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem[J]. Swarm and Evolutionary Computation, 2017, 32: 121-131.

[16] PAN Q K, GAO L, WANG L. An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems [J]. IEEE Transactions on Cybernetics, 2022, 52(7): 5999-6012.

[17] GUO H W, SANG H Y, ZHANG X J, et al. An effective fruit fly optimization algorithm for the distributed permutation flowshop scheduling problem with total flowtime[J]. Engineering Applications of Artificial Intelligence, 2023, 123: 106347.

[18] YU Y, ZHANG F Q, HUANG J P. Acceleration-based artificial bee colony optimizer for a distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Applied Soft Computing, 2023, 135: 110029.

[19] GUO H W, SANG H Y, ZHANG B, et al. An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Knowledge-Based Systems, 2022, 242: 108328.

[20] BAI D, LIU T, ZHANG Y, et al. Scheduling a distributed permutation flowshop with uniform machines and release dates[J]. IEEE Transactions on Automation Science and Engineering, 2024: 1-13.

[21] 曾亮, 石俊洋, 胡迈, 等. 用于分布式置换流水变速车间的双种群算法[J]. 南京信息工程大学学报(自然科学版), 2024, 16(6):782-790.

[22] 连戈, 朱荣, 钱斌, 等. 超启发式人工蜂群算法求解多场景鲁棒分布式置换流水车间调度问题[J]. 控制理论与应用, 2023, 40(4): 713-723.

[23] LIN J, WANG Z J, LI X. A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem[J]. Swarm and Evolutionary Computation, 2017, 36: 124-135.

[24] YU H, GAO K Z, MA Z F, et al. Improved meta-heuristics with Q-learning for solving distributed assembly permutation flowshop scheduling problems[J]. Swarm and Evolutionary Computation, 2023, 80: 101335.

[25] GAO J, CHEN R. An NEH-based heuristic algorithm for distributed permutation flowshop scheduling problems[J]. Scientific Research and Essays, 2011, 6(14): 3094-3100.

[26] GAO J, CHEN R. A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem[J]. International Journal of Computational Intelligence Systems, 2011, 4(4): 497-508.

[27] 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.

[28] XU Y, WANG L, WANG S Y, et al. An effective hybrid immune algorithm for solving the distributed permutation flow-shop scheduling problem[J]. Engineering Optimization, 2014, 46(9): 1269-1283.

[29] 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.

[30] HAMZADAYI A. An effective benders decomposition algorithm for solving the distributed permutation flowshop scheduling problem[J]. Computers & Operations Research, 2020, 123: 105006.

[31] 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.

[32] 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.

(责任编辑:赵海军)