基于分解策略混合算法的DAJSP求解

仵政源[1] 赵诗奎[1,2*] 解瑞建[3] 于丰顺[1] 李 彤[1] 赵 林[1]
1.济南大学机械工程学院,济南,250022
2.山东省金属关键构件表面处理与智能装备重点实验室,济南,250022
3.济南二机床集团有限公司,济南,250022

摘要:针对分布式装配作业车间调度问题(DAJSP),以最小化最大完工时间为目标,建立了混合整数线性规划(MILP)模型,提出了一种融合贪心策略的遗传-禁忌搜索算法(GTSAIGS)。将DAJSP分解为工件加工与产品装配的两阶段问题,在加工阶段采用启发式算法优化,针对装配阶段的调度问题设计了一种高效的贪心算法进行快速求解。40个算例的测试结果表明,相比于整体优化方法,GTSAIGS分解策略减少了计算耗时,提高了求解质量,进而验证了所提算法的有效性和优越性。

关键词:分布式装配作业车间调度问题(DAJSP);最大完工时间;混合算法;贪心算法
中图分类号:TP182
DOI:10.3969/j.issn.1004-132X.2026.07.017 **开放科学(资源服务)标识码(OSID)**:

A Hybrid Algorithm Based on Decomposition Strategy for DAJSP Solution

WU Zhengyuan[1] ZHAO Shikui[1,2*] XIE Ruijian[3] YU Fengshun[1] LI Tong[1] ZHAO Lin[1]
1.School of Mechanical Engineering, University of Jinan, Jinan, 250022
2.Shandong Provincial Key Laboratory of Surface Treatment and Intelligent Equipment for Key Metal Components, University of Jinan, Jinan, 250022
3.JIER Machine-Tool Group Co., Ltd, Jinan, 250022

Abstract: This paper addresses the distributed assembly job-shop scheduling problem (DAJSP) by establishing a mixed-integer linear programming (MILP) model to minimize the makespan, and proposes a hybrid genetic-tabu search algorithm with a greedy strategy. The DAJSP is decomposed into two subproblems: job processing and product assembly. For the processing stage, a heuristic algorithm is applied; for the assembly stage, an efficient greedy algorithm is designed to provide a fast and effective scheduling solution. Experimental results on 40 instances show that, compared with a monolithic optimization method, the proposed decomposition strategy reduces computational time and improves solution quality, thereby validating its effectiveness and superiority.

Key words: distributed assembly job-shop scheduling; makespan; hybrid genetic-tabu search; greedy algorithm

0 引言

当前,企业间的合作生产日益普及,分布式制造已成为常见的生产模式。在汽车制造、航空航天和工程机械等行业,零部件常在不同地区的分布式工厂中生产,最后再运至总装厂进行最终装配。这种模式在充分利用不同地区生产成本和技术优势的同时也带来了跨地区工厂生产与装配协同调度的挑战。因此,研究分布式装配作业车间调度问题(distributed assembly job-shop scheduling problem,DAJSP)具有理论意义和应用价值。

DAJSP由工件分布式加工和产品装配两阶段组合而成。与作业车间调度问题(job-shop scheduling problem,JSP)相比,装配作业车间调度问题(assembly job-shop scheduling problem,AJSP)中的工件有多道工序,且工件须装配成最终产品。这一特性使得AJSP的优化需同时协调加工与装配阶段的时序约束。针对AJSP,已有研究提出了多种求解方法。对于柔性AJSP,ZHANG等[1]提出的分布式蚁群优化算法可同时优化完工时间、总延迟时间和总工作量。SHI等[2]提出一种将改进的扩展转移瓶颈程序和遗传算法相结合的混合算法来求解AJSP。李仲华等[3]针对带外协的柔性AJSP,建立了以最小化最大完工时间为目标的模型,并采用改进鲸鱼动态

收稿日期:2025-04-10
基金项目:国家自然科学基金(52275490);山东省自然科学基金(ZR2025MS766);山东省科技型中小企业创新能力提升工程(2025TSGCCZZB0845);中央引导地方科技发展资金(YDZX2024127)

调度算法求解。包含装配车间调度的产品综合调度优化问题中，基于工序约束链编码的遗传算法[4]能准确体现产品各个工序之间的装配约束关系，且不会增加新的约束，保证了初始解空间的可行性和完备性，其有效性已被实验证明。

分布式作业车间调度问题（distributed job-shop scheduling problem，DJSP）是DAJSP的基础，主要研究工件在加工工厂间的分配以及各工厂内的加工顺序，以实现调度指标的最优。CHAOUCH等[5]以最小化最大完工时间为目标，设计3个仿生算法来求解工件的工厂分配问题，并通过实验对比了三者的性能。在以最小化最大完工时间为目标的DJSP中，ŞAHMAN[6]将机器负载排序机制和贪婪的启发式算法与离散斑点鬣狗算法相结合进行求解，取得了良好的结果。JIANG等[7]提出一种基于分解的改进型多目标进化算法，并设计一种动态调整局部搜索算子利用率的选择策略，以最小化最大完工时间和总能耗为目标，对DJSP进行求解。WANG等[8]提出一种解决DJSP的改进遗传算法，算法在工件分配工厂子问题中考虑个体间的相似性，在交叉过程中有效地平衡了解空间的探索和较优解基因保留的问题。

DJSP主要关注将一系列独立的工件分配到不同的工厂，并优化其在工厂内部的加工顺序，目标通常是最小化最大完工时间。DAJSP在此基础上增加了装配阶段，这意味着多个工件不再独立，而是构成最终产品的零部件，必须在各自完成加工后汇集到装配中心进行最终组装。在以最大完工时间为目标的研究中，DJSP需要平衡并缩短每个加工工厂的最大完工时间，以得到更优解。DAJSP的优化目标是包含装配工序在内的最大完工时间，单个工厂的最大完工时间影响其所关联的装配工序开工时间，进而影响最大完工时间。DAJSP的主要挑战在于引入了装配工序的齐套性约束，即一个产品的装配工序必须在其所有零部件都加工完成之后才能开始。这种约束使得分布在不同工厂、分属同一产品的各工件的调度彼此关联，形成耦合关系，增加了问题的复杂性。

目前，对DAJSP的研究尚处于起步阶段，研究也较少。TIAN等[9]建立了以最小化最大完工时间为目标的DAJSP混合整数线性规划（mixed integer linear programming，MILP）模型，提出一种基于可变邻域搜索的遗传算法，采用三层编码表示DAJSP的3个决策过程，即将工件分配到工厂、确定每个工厂内的工序、确定产品的装配顺序。YANG等[10]针对有转移、异构和柔性的DAJSP，以最小化最大完工时间和总能耗为目标，提出一种基于Q学习的改进型多目标遗传算法，并设计了2种交叉算子、4种变异算子和6种面向目标的邻域搜索算子，以增强算法的探索和开发能力。针对广泛存在带有车辆配送的DAJSP，杨绍文等[11]以最小化运输和延迟惩罚总成本为优化目标，提出一种混合三维分布估计算法进行求解。

分析已有研究发现，现有DAJSP研究多将“工件分配—加工排序—装配排序”整体耦合优化，导致搜索空间大、装配阶段重复迭代耗时的问题。工件分配与加工调度确定后，装配可转化为带释放时间约束的单机排序问题，适合用规则快速求解。因此提出分解式混合算法，即在加工阶段采用“遗传算法全局搜索+禁忌搜索局部强化”提升解质量，在装配阶段采用贪心装配规则减少不必要迭代，从而在保证解质量的同时提高求解效率。

1 问题模型

1.1 问题描述

DAJSP是包含加工阶段和装配阶段的调度问题。在加工阶段，工件J_1、J_2、…、J_n需在分布的多个工厂加工。每个工厂配置相同的机器M_1、M_2、…、M_m，且具有加工所有工件的能力。工件J_i需按给定加工工艺路线在任一工厂的机器上依次完成若干工序。在装配阶段，一个装配工厂（仅有一台装配机器M_A）将工件装配为N_p个不同的产品。研究假设工件在同一工厂内不同机器间的转移时间与跨工厂的运输时间忽略不计。

研究目标是确定工件的加工工厂分配、工序加工顺序和产品的装配顺序，以最小化最大完工时间C_{max}。此外，DAJSP有以下约束：所有机器从0时间开始可用；工件的加工应满足工艺路线约束；一台机器在同一时间最多完成一道工序；一道工序一旦在给定的机器上开始，则在完成之前不能中断；某一工件的所有工序应在一个工厂内完成。

表1所示为具有2个分布式加工工厂的DAJSP案例，案例包含产品P_1、P_2、P_3，其中，产品P_1由工件J_2、J_5、J_6组成，产品P_2由工件J_1、J_3、J_8组成，产品P_3由工件J_4、J_7组成。2个分布式加工工厂均配置机器M_1和M_2。表1给定了每个工件每道工序的加工机器和加工用时，以及每个产品的

装配用时，表中空白表示该工序不在对应机器上加工。

表1　具有3产品8工件2工厂的DAJSP案例

Tab.1　DAJSP example with 3 products, 8 jobs and 2 factories

产品	工件	工序	各加工机器上的加工时间				装配用时
			工厂1		工厂2		
			M_1	M_2	M_1	M_2	
P_1	J_2	$O_{2,1}$		4		4	5
		$O_{2,2}$	5		5		
	J_5	$O_{5,1}$	5		5		
		$O_{5,2}$		6		6	
	J_6	$O_{6,1}$		4		4	
		$O_{6,2}$	4		4		
P_2	J_1	$O_{1,1}$	6		6		4
		$O_{1,2}$		2		2	
	J_3	$O_{3,1}$	3		3		
		$O_{3,2}$		4		4	
	J_8	$O_{8,1}$	2		2		
		$O_{8,2}$		3		3	
P_3	J_4	$O_{4,1}$	6		6		5
		$O_{4,2}$		3		3	
	J_7	$O_{7,1}$	6		6		
		$O_{7,2}$		5		5	

1.2　数学建模

根据问题描述，建立MILP模型，其中，优化目标为

$$\min C_{\max}$$

约束条件为

$$C_{\max} \geqslant C_p \quad \forall p \tag{1}$$

$$\sum_{f\in F} X_{i,f} = 1 \quad \forall i \in I \tag{2}$$

$$X_{i,f} = X_{i,j,f} \quad \forall i \in I, j \in O_i, f \in F \tag{3}$$

$$S_{i,j+1} \geqslant S_{i,j} + T_{i,j} \quad \forall i \in I, \forall j, j+1 \in O_i \tag{4}$$

$$S_{i,j} + T_{i,j} \leqslant S_{i',j'} + L(1 - Y_{i,j,i',j'}) + L(2 - X_{i,f} - X_{i',f})$$
$$\forall i, i' \in I, i \neq i';\ \forall j, j' \in O_i, k_{i,j} = k_{i',j};\ \forall f \in F \tag{5}$$

$$S_{i',j'} + T_{i',j} \leqslant S_{i,j} + LY_{i,j,i',j'} + L(2 - X_{i,f} - X_{i',f})$$
$$\forall i, i' \in I, i \neq i';\ \forall j, j' \in O_i, k_{i,j} = k_{i',j};\ \forall f \in F \tag{6}$$

$$C_i \geqslant S_{i,j} + T_{i,j} \quad \forall i \in I, \forall j \in O_i \tag{7}$$

$$A_p \geqslant C_i \quad \forall i \in S_p, \forall p \in P \tag{8}$$

$$A_p + T_p \leqslant A_{p'} + L(1 - Z_{p,p'}) \quad \forall p, p' \in P, p < p' \tag{9}$$

$$A_{p'} + T_{p'} \leqslant A_p + LZ_{p,p'} \quad \forall p, p' \in P, p < p' \tag{10}$$

$$C_p = A_p + T_p \quad \forall p \in P \tag{11}$$

其中，式(1)为最大完工时间上界约束；式(2)为工件分配约束，确保每个工件必须且只能分配到一个工厂；式(3)为工序分配约束，确保同一工件的所有工序分配到同一工厂；式(4)为工件工艺路线约束，确保某工序必须在其紧前工序完成后才能开始加工，工件的首道工序无此约束；式(5)、式(6)为加工机器资源约束，确保同一机器上不同工序的加工时间不会重叠；式(7)为工件完工时间约束，确保一个工件的完工时间必须大于或等于其所有工序的完工时间；式(8)为装配齐套性约束，确保一个产品的装配必须在其所有工件完工后才能开始；式(9)和式(10)为装配机器资源约束，确保任意两个产品在装配机器上的装配时间不会重叠；式(11)计算产品完工时间。

模型相关符号如表2所示。

表2　符号说明

Tab.2　Explanation of symbols

符号	意义
i, i'	工件索引，$i, i' \in I$
n	工件总数
I	工件索引集，$I=\{1,\cdots,n\}$
j, j'	加工工序的索引，$j, j' \in O_i$
O_i	工件i的工序索引集
$O_{i,j}$	工件i的第j道工序
f	分布式加工工厂索引，$f \in F$
N_f	分布式加工工厂总数
F	分布式加工工厂索引集，$F=\{1,\cdots,N_f\}$
$T_{i,j}$	工件i的第j道工序的加工时间
T_p	产品p的装配时间
p, p'	产品索引，$p, p' \in P$
S_p	构成产品p的工件索引集，$S_p \subseteq I$
N_p	产品数量
P	产品索引集，$P=\{1,\cdots,N_p\}$
L	一个极大的正数
m	每个工厂的机器数
$k_{i,j}$	工件i的第j道工序的指定加工机器索引，$k_{i,j} \in \{1,2,\cdots,m\}$
$X_{i,f}$	如果工件i被分配到工厂f，则为1，否则为0
$X_{i,j,f}$	如果工件i的第j道工序被分配到工厂f，则为1，否则为0
$Z_{p,p'}$	如果产品p在产品p'之前进行装配，则为1，否则为0
$S_{i,j}$	工件i的工序j的开工时间，$S_{i,j} \geqslant 0$
C_i	工件i的完工时间，$C_i \geqslant 0$
A_p	产品p的装配开始时间，$A_p \geqslant 0$
C_p	产品p的完工时间，$C_p \geqslant 0$
$Y_{i,j,i',j'}$	如果在一台机器上工件i的工序j在工件i'的工序j'之前加工，则为1，否则为0
$C_{\max}$	整个调度方案的最大完工时间，$C_{\max} \geqslant 0$

2　混合算法设计

本文提出一种融合贪心策略的遗传-禁忌搜索算法(genetic-tabu search algorithm incorporating greedy strategy，GTSAIGS)。GTSAIGS的核心思想是分解DAJSP，并为子问题匹配最合适的求解策略。加工阶段采用遗传算法与禁忌搜索相结合的方式进行求解。遗传算法在全局搜索中表现出色，能有效探索解空间，但局部搜索能力较弱，为提高解的质量，需结合问题特征进行局部优化。禁忌搜索算法专注于局部搜索，能有效与

DAJSP的特征信息和邻域结构结合，引导搜索过程，减少搜索的盲目性，在局部范围内找到更优解。因此，将遗传算法与禁忌搜索算法相结合能很好平衡全局搜索与局部搜索，提高求得高质量解的能力。在装配阶段，基于贪心算法快速生成装配工序调度，并据此得到完整调度方案的最大完工时间C_{max}。通过3种算法的优势互补，混合算法在较短时间能求得DAJSP的高质量解。

2.1 遗传算法

2.1.1 编码和解码

遗传算法编码是将问题解的描述转化为算法可以处理的染色体的过程，现有DAJSP研究中的编码通常由工件的加工工厂分配、工序完成顺序和产品装配顺序三部分组成。研究发现，当工件的工厂分配和加工顺序确定后，装配阶段可依据加工结果通过贪心规则快速生成调度方案，因此本文在编码中不再显式表示装配顺序，且装配工序调度无需参与后续的交叉、变异等操作。因此，本文采用两层编码描述DAJSP解。第一层为工厂选择链，用于表示工件到工厂的分配；第二层为工序顺序链，用于表示工序完成顺序。在装配阶段，引入贪心算法来快速生成装配工序调度，简化编码且减少遗传操作与后续禁忌搜索中针对装配顺序的搜索开销。

图1所示为表1案例一个可行解的编码方案，其中，工厂选择链内的元素由1～N_f的整数构成，位置i的整数f表示将工件J_i分配到加工工厂f。图1中的工厂选择链表示工件J_2、J_3、J_5、J_6、J_8由工厂1中的机器加工，其余的工件由工厂2加工。工序顺序链采用基于工序的编码方式，用工件编号的重复出现次序隐式表示其各道工序的先后完成关系。链内元素由1～n的整数构成，链中第c个位置上的编号i表示工件J_i，若从左到右扫描至位置c时，编号i已出现j次，则该位置对应工件J_i的第j道工序。结合工厂选择链并从左到右扫描工序顺序链，可确定各工序在对应加工工厂中的加工先后次序。例如，图1中的工序顺序链2→3→1→6→…表示加工顺序$O_{2,1}$→$O_{3,1}$→$O_{1,1}$→$O_{6,1}$→…。对于包含n个工件，每个分布式加工工厂具有m台机器的DAJSP，工序顺序链由$n\times m$个整数组成，每个工件号i只能在染色体中出现m次。

在完成染色体编码设计后，初始化种群是遗传算法的起始步骤，随机编码可以最大限度地确保种群在算法初期就具备高度的多样性，实现对解空间的广泛覆盖，因此采用随机方式初始化种群。

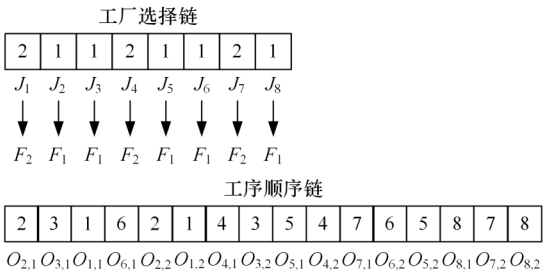


图1 编码示意图

Fig.1 Encoding scheme

解码是将染色体携带的编码信息转化为实际调度方案的过程。常见的解码方式有半主动解码和主动解码，半主动解码仅能保证工序按顺序尽可能早地开始，但无法改变工序的先后次序。主动解码通过搜索机器加工序列中间的空闲时段，在满足工艺约束的前提下，允许后续工序插空加工，在满足约束条件下的提前调度工序，压缩了机器的无效空闲时间。因此，本文采用主动解码方式，具体步骤如下：首先，根据工厂选择链将加工阶段划分为多个相互独立的工厂内JSP子问题。其次，依次读取工序顺序链，对于读取的某道工序，搜索其分配工厂内对应机器上的空闲时间窗，将该工序插入对应机器上满足加工耗时且不早于紧前工序完工时间的最早可行空闲时段内，从而确定所有工件的完工时间。最后，由贪心算法生成完整的调度方案，得到最大完工时间C_{max}。图2为图1示例编码解码得到的甘特图。

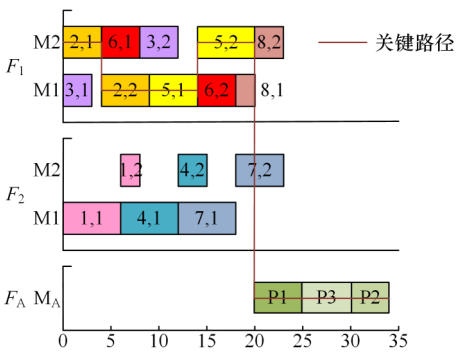


图2 可行解对应的甘特图

Fig.2 Gantt chart corresponding to feasible solution

2.1.2 选择方法

遗传算法中，选择操作的作用是模拟自然界优胜劣汰的机制，引导种群向全局最优的方向搜索。为平衡算法的收敛速度与种群多样性，选择操作由两部分组成：首先采用精英保留策略直接复制部分最优个体；随后通过锦标赛选择补齐其余个体。首先选取父代种群中最优的一部分个体无条件复制到子代。然后从父代种群内随机抽取的部分个体中选出适应度最优的个体，并将其放

入子代种群。重复此过程，直至子代种群的规模与父代种群相等。

2.1.3　交叉方法

交叉是决定遗传算法全局搜索能力的重要操作之一。某个个体具有较高适应度时，其基因序列中的某些片段通常表现良好。若这些优良片段在交叉过程中得到有效继承并与其他片段重新组合，就有可能产生适应度更高的后代，从而增强算法的全局搜索能力。

交叉算子的设计必须与染色体编码方式相匹配。由于本文采用的染色体编码由工厂选择链和工序顺序链两部分组成，为保证交叉后生成解的可行性，需针对两层编码的结构特点分别设计交叉方法[12]。工厂选择链逻辑简单，易确保生成解的可行性，因此工厂选择链采用图3a所示的均匀交叉，即2个父代染色体元素以一定的概率进行交换。考虑工序顺序链的特点，采用JSP中常用的POX交叉法以继承父代的优良特征。如图3b所示，POX交叉首先将工件集随机划分为2个非空的子集Set_1和Set_2，随后将Parent1和Parent2中属于Set_1的工件分别复制到Children1和Children2中，并保留其在父代中的位置；再将Parent2和Parent1中属于Set_2的工件按照原有顺序分别填充至Children1和Children2的剩余空位中[13]。这种交叉方式在保证解可行性的同时，实现了父代优良片段信息的传递与重组。

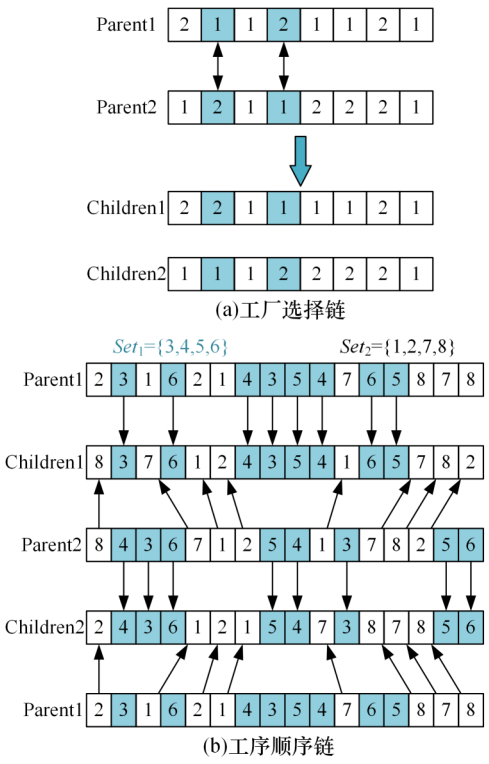


图3　交叉操作示意图

Fig.3　Illustration of crossover operations

2.1.4　变异方法

变异是对染色体进行小幅扰动的操作，旨在保持种群的多样性，避免算法过早收敛。常见的变异方法有交换、插入和逆转等。如图4a所示，对工厂选择链采用单点随机变异，随机选中工厂选择链中的某一点，将该点的元素替换为其他可选工厂编号，实现了单个工件向其他工厂的转移。如图4b所示，对工序顺序链采用两点交换变异方式，随机选择工序顺序链中的两点并交换两点的值，实现了两道工序加工顺序的改变。

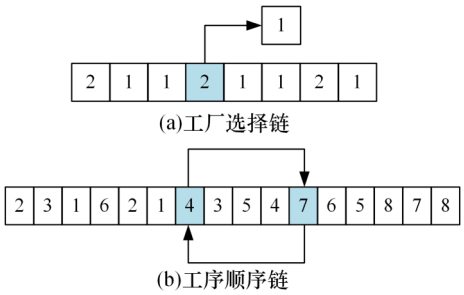


图4　变异示意图

Fig.4　Schematic diagram of mutation

2.2　禁忌搜索

禁忌搜索算法在邻域搜索的基础上，通过设置禁忌表来避免解的重复搜索，并利用特赦准则允许某些被禁忌的优良解重新进入搜索过程。邻域结构、候选解生成和禁忌长度等因素是影响禁忌搜索算法性能的关键。

2.2.1　邻域结构

作为禁忌搜索算法的核心机制，邻域结构通过对当前解的局部扰动来生成邻域解集。为提高搜索效率并尽量减少无效及不可行的移动，邻域搜索通常作用于对目标函数值影响较大的关键工序。对于以最小化最大完工时间为目标的调度问题，这些关键工序位于当前调度方案对应析取图的关键路径上，因此需要借助析取图模型对关键路径进行分析。

JSP的析取图模型包含节点、连接弧和析取弧。节点对应工序，以及开始和结束虚拟节点；连接弧连接属于同一工件且相邻的两个工序；析取弧连接属于不同工件但在同一机器上完成的两个相邻工序[14]。节点和弧可直观表示调度问题的工艺顺序约束和机器资源约束，将调度问题的求解转化为确定析取图中弧的去向。除了工艺约束和机器约束，DAJSP还需要考虑产品与工件之间的装配关系，为此，TIAN等[9]在传统JSP析取图的基础上引入装配关系弧，设计了包含三类弧的DAJSP析取图模型，为后续关键路径识别及邻域

结构设计提供图论基础。

如图5所示，析取图包含的虚拟节点O_{begin}、O_{end}表示调度的开始和结束。关键路径是从起点O_{begin}到汇点O_{end}的最长路径，以甘特图（图2）和图5所示的析取图模型为例，$O_{begin}\rightarrow O_{2,1}\rightarrow O_{2,2}\rightarrow O_{5,1}\rightarrow O_{5,2}\rightarrow P_1\rightarrow P_3\rightarrow P_2\rightarrow O_{end}$是关键路径。

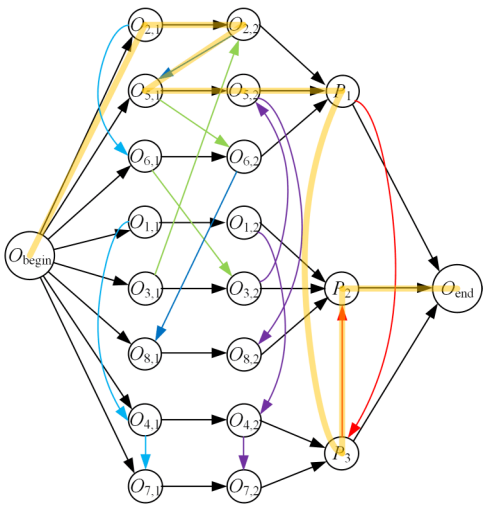


图5 DAJSP的析取图模型

Fig.5 Disjunctive graph of DAJSP

关键路径由关键工序组成。某一工序是否为关键工序根据其最早开工时间和最晚开工时间判定，若工序$O_{i,j}$满足$r_{i,j}+t_{i,j}=C_{max}$，则该工序为关键工序，可将其加入关键路径，其中，$r_{i,j}$为工序$O_{i,j}$的头长度，为起点到节点$O_{i,j}$的最长路径长度，可由染色体主动解码后得到；$t_{i,j}$为工序的尾长度，为节点$O_{i,j}$到汇点的最长路径长度与节点加工时间$T_{i,j}$之和，通过对当前调度进行逆向分析得到，具体步骤如伪代码1所示：

伪代码1：findCriticalPath(G，S)　//关键路径算法流程

Input：析取图G，调度方案S；
Set　$criticalPath=\emptyset$//初始化关键路径列表
Set　$reTopoSort$//由析取图获取工序的逆拓扑排序
Set　C_{max}//从S中获取C_{max}
For　$O_{i,j}$　in　$reTopoSort$　　do
　　If　isExist($O_{i,j+1}$)//
　　$t^{*}_{Oi,j}=t_{Oi,j+1}+T_{i,j}$
　　Else
　　$t^{*}_{Oi,j}=T_{i,j}$
　　End if
　　If　isExist($MS_{Oi,j}$)//MS表示工序的机器后续工序
　　$t^{**}_{Oi,j}=t_{MSOi,j}+T_{i,j}$
　　Else
　　$t^{**}_{Oi,j}=T_{i,j}$
　　End if
　　$t_{Oi,j}=\max(t^{*}_{Oi,j},t^{**}_{Oi,j})$
　　If　$r_{Oi,j}+t_{Oi,j}==C_{max}$//$r_{Oi,j}$对应工序$O_{i,j}$在主动解码下的$schedule$中的开工时间
　　将$O_{i,j}$加入关键路径
　　End if
End for
Output：关键路径

关键块为关键路径上多个紧密相连的关键工序组成的块，如图2所示，$O_{2,2}\rightarrow O_{5,1}$、$P_1\rightarrow P_3\rightarrow P_2$是关键块。

包含关键工序的工件为关键工件，包含关键工件的工厂为关键工厂。在工厂选择链中，针对关键工件的移动包括转移和互换两种方式。转移指将关键工件移到非关键工厂，一般移入完工时间最短的工厂，若所有工厂皆是关键工厂，则关键工件随机移到一个其他工厂。互换指选取一个关键工件，并从其他工厂中任选一个工件，将该其他工厂工件与选取的关键工件交换所属加工工厂。

工序顺序链的移动采用N7邻域结构。如图6所示，N7邻域结构的移动对象为关键块，移动操作包含关键块内的工序移动至块首之前或块尾之后，或将关键块的首工序或尾工序插入块内部。

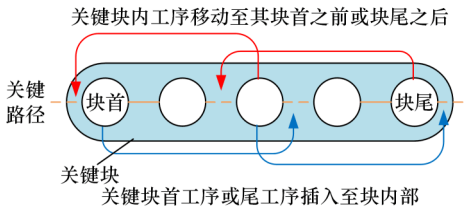


图6 N7邻域结构示意图

Fig.6 Schematic diagram of the N7 neighborhood structure

2.2.2 禁忌搜索算法流程

结合析取图模型、工厂选择链、工序顺序链邻域结构，设计禁忌搜索算法，算法流程如伪代码2所示：

伪代码2：TS(x_0)//禁忌搜索算法流程

Input：初始解x_0；
1：　Set　禁忌表$=\emptyset$//初始化禁忌表
2：　Set　$counter=0$//当前最优解未改善代数计数器
3：　Set　$x_{best}=x_0$//初始化最优解
4：　While　当前最优解未改善代数小于终止要求　　do
5：　　基于N7邻域结构和工件在工厂间的移动生成x_0的邻域解集合$children$；
6：　　使用贪心算法快速生成$children$的完整调度方案，计算适应度值，挑选出未被禁忌的最优解或满足特赦准则的解x'
7：　　$x_0=x'$
8：　　更新禁忌表
9：　　If　$C_{max}(x')<C_{max}(x_{best})$　then
10：　　$x_{best}=x'$
11：　　重置当前最优解未改善代数$=0$
12：　　Else
13：　　$counter++$
14：　　End if
15：　End while
Output：最优解x_{best}

2.3 贪心算法

2.3.1 基本贪心算法

贪心算法在问题的每一步都做当前状态下最

优的选择，并期望最终结果是最优。具有效率高、易于实现等优点，能在较短时间内获得令人满意的可行解。

2.3.2　基于贪心算法的装配工序调度

基于DAJSP的特性，当加工阶段的工件工厂分配与工序排序方案由启发式算法确定后，装配调度方案可由贪心策略快速求解。该方法无需对装配阶段进行编码，降低了编码复杂性，且装配阶段调度不需参与遗传算法和禁忌搜索的优化，缩短了算法的运行时间。

依据加工工序的调度结果，采用贪心算法生成装配工序调度方案的具体步骤如下：

1）初始化当前时间 *nowTime* 为0，将所有装配工序加入待调度产品列表 *waitlist*，初始化装配工序调度序列 *assemblySchedule* 为空。根据加工工序的调度结果，将所有装配工序最早可开工时间设置为构成对应产品的所有工件的最大完工时间，并按最早可开工时间在 *waitlist* 中升序排列。

2）从 *waitlist* 中选择当前时间可开工的装配工序。若当前时间没有可开工的装配工序，则将 *nowTime* 更新为按最早可开工时间升序排列后的 *waitlist* 首元素的最早可开工时间。

3）在可选的待装配工序中选择装配时间最短的工序进行处理，并将所选工序加入 *assemblySchedule* 的末尾。装配完成后，更新当前时间为所选工序的完工时间，随后将该工序从 *waitlist* 中移除。

重复步骤2）和3），直至 *waitlist* 为空，最终求得完整的装配工序调度序列 *assemblySchedule*。

表3所示为10个产品装配任务示例，为简便表示，直接给出每个产品的可开工时间和装配时间，采用贪心法生成装配工序调度。

表4所示为使用贪心算法对10个产品的装配工序进行调度的示例，“选择工序”这行内容对应本轮所选装配任务。

表3　具有10个产品的装配任务示例

Tab.3　Example of assembly tasks for 10 products

产品	可开工时间	装配用时
P1	76	13
P2	162	35
P3	132	14
P4	183	28
P5	142	33
P6	132	19
P7	201	32
P8	97	4
P9	113	16
P10	186	22

表4　使用贪心算法进行装配工序调度的示例

Tab.4　Example of using a greedy algorithm for assembly process scheduling

序号	当前时间	待调度产品列表	选择工序	装配用时
1	0	P1,P8,P9,P3,P6,P5,P2,P4,P10,P7	P1	13
2	89	P8,P9,P3,P6,P5,P2,P4,P10,P7	P8	4
3	101	P9,P3,P6,P5,P2,P4,P10,P7	P9	16
4	129	P3,P6,P5,P2,P4,P10,P7	P3	14
5	146	P6,P5,P2,P4,P10,P7	P6	19
6	165	P5,P2,P4,P10,P7	P5	33
7	198	P2,P4,P10,P7	P10	22
8	220	P2,P4,P7	P4	28
9	248	P2,P7	P7	32
10	280	P2	P2	35

首先初始化数据，将当前时间 *nowTime* 设定为0，将所有装配工序放入待调度序列 *waitlist*，根据可开工时间将工序升序排序，排序后的序列为P1→P8→P9→P3→P6→P5→P2→P4→P10→P7。随后，算法进入迭代循环：首先确定当前时间 *nowTime* 下的可开工工序，若无则将 *nowTime* 更新为 *waitlist* 中第一个工序的可开工时间；接着，在可选工序中选择装配用时最短的工序进行处理，将所选工序加入 *assemblySchedule* 最后位置，更新 *nowTime* 为该工序的完工时间，并将所选工序从 *waitlist* 中移除。重复此过程直至 *waitlist* 为空。最终得到的调度顺序为P1→P8→P9→P3→P6→P5→P10→P4→P7→P2。

3　算法总体框架与流程

本文提出的GTSAIGS基于分解策略，以工件完工时间作为解耦点，将DAJSP分解为工件加工调度和产品装配调度两阶段问题，两阶段通过工件完工时间与适应度值实现阶段间信息传递。

加工调度由遗传算法与禁忌搜索完成，通过全局搜索确定工件在各工厂的分配及加工顺序，随后对关键路径进行深度搜索以跳出局部最优。加工阶段的输出是所有工件的完工时间集合 $\{C_i\}$。装配调度采用贪心策略求解，将加工阶段输出的 $\{C_i\}$ 作为装配阶段的齐套性约束。贪心算法依据“当前可开工时间装配用时最短工序优先原则”快速生成装配调度方案，进而得到最终最大完工时间 $C_{\max}$，将 $C_{\max}$ 传递给主算法并作为染色体的适应度，指导种群进化。混合算法具体步骤如伪代码3所示：

伪代码3：algorithm(*data*) //混合算法主流程

```
Input：DAJSP问题参数及约束data；
设置问题参数
随机初始化种群popu
Set  counter = 0 // 初始化最优解未改善代数计数器
Set  preC_max = MAX，x_best = ø
While  当前最优解未改善代数小于终止要求  do
    For  x  in  popu  do
        TS(x)//禁忌搜索提升解质量，如伪代码2所示，在此过程中，使用贪心算法快速计算完整调度方案
    End for
    For  x  in  popu  do
        对x主动解码，得到加工工序调度processSchedule
        schedule = Greedy(processSchedule) //使用贪心算法得到完整调度方案
        C_max(x) = max(schedule)
    End for
    if  minC_max < preC_max
        preC_max = minC_max
        counter = 0，更新x_best
    else
        counter++
    End if
    Set  elitePopu将C_max最佳的部分个体直接遗传下去
    Set  selectPopu 通过锦标赛选择算法生成子代种群
    selectPopu.FA=UN(selectPopu.FA)
    //对工厂选择链进行均匀交叉
    selectPopu.PS=POX(selectPopu.PS)
    //对工序顺序链进行POX交叉
    selectPopu.FA=SPR(selectPopu.FA)
    //对工厂选择链进行单点随机变异
    selectPopu.PS=TPE(selectPopu.PS)
    //对工序顺序链进行两点交换变异
    将elitePopu和selectPopu合并为popu
End while
Output：最优解x_best
```

4 实验测试

由于目前尚缺乏统一的DAJSP基准算例，本文采用文献[9]设计的一系列算例对算法进行评估。这一系列算例基于Ta01-Ta40设计，为每个算例新增产品数量、分布式加工车间的数量、每个产品的装配时间、工件和产品之间的包含关系，以适用于DAJSP。随后通过正交试验确定算法参数组合，并在此基础上开展对比实验与结果分析。

4.1 参数设置

正交试验在表5所列参数水平范围内进行，以确定算法的较优参数组合。实验基于DA09算例进行，采用相对偏差(RD)

$$D_R=\frac{\overline{C}_{max}-C_{max\,best}}{C_{max\,best}} \tag{12}$$

描述实验结果，其中，$\overline{C}_{max}$、$C_{maxbest}$分别为参数组合运行5次获得的最大完工时间的平均值和最优值。

根据图7所示主效应图，分别比较*popuSize*、P_c和P_{ER}在不同水平下对应的D_R均值，优先选择能使D_R均值达到最小的参数水平作为候选组合。

表5 混合算法参数表

Tab.5 Hybrid algorithm parameters table

参数	水平值
种群规模*popuSize*	20，50，100
交叉概率P_c	0.85，0.90，0.95
精英保留占比P_{ER}/%	2，5，10，20

由图7可见，*popuSize*=50、P_c=0.9和P_{ER}=5%对应的D_R均值均处于各自因素的最优水平。算法中未参与正交试验的参数给定如下：遗传算法、禁忌搜索算法的最优解连续不更新迭代终止代数分别设为10和500；为避免早熟收敛，若连续5代最优适应度不改善，则将变异概率p_m在当前值基础上增加0.05，直至上限0.50；禁忌表长度$L_{tubu}=10+\left\lceil\frac{n}{N_f m}\right\rceil$；锦标赛规模在2和3之间随机取值。所有算例均采用上述参数配置。

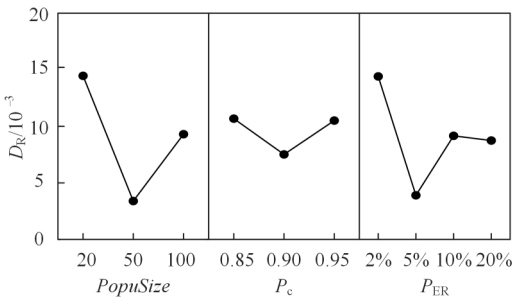


图7 算法参数的主效应图

Fig.7 Main effects plots for algorithm parameters

4.2 算法性能对比及部件有效性消融实验

对比算法选取遗传算法与禁忌搜索相结合的GATS。与本文方法不同，GATS采用三层编码，除工厂选择链和工序顺序链外，还包含产品装配顺序链。

在GATS中，产品装配顺序链同时参与交叉、变异和邻域移动，其中，装配顺序链的交叉算子和变异算子与工序顺序链相同，并在邻域搜索中采用N7邻域结构。

为保证对比的可解释性，本文设置两类参照：①CPLEX求解MILP模型获得的最优解，标记为MILP1；②文献[9]在相同算例上的结果(MILP2与GA-VNS)作为外部对照。为进行深入分析各组件的有效性，设计了3种消融算法：遗传算法(GA)、融合贪心策略的遗传算法(GA-Grd)、GATS。

基于上述算例和参数的所有算法均使用MATLAB R2023a编程实现。实验在一台配置i5-9300H CPU、8GB RAM、Windows 10 64位操作系统的计算机上进行。为确保实验的公平性，

所有算法均在完全相同的软硬件环境下运行，每个算例均重复运行20次以获取统计结果。结果按算例的产品数划分。见表6～表8，表中，*best*列表示各算法在20次运行中求得的最佳结果，*average*列表示各算法20次运行中求得的平均结果，*time*列表示各算法的平均耗时(s)。数值保留到小数点后两位。

根据实验结果，按产品数量分组对比了算法GTSAIGS和GATS的平均耗时和C_{max}平均结果，使用耗时减少率R_T和结果增加率R_R

$$R_T=\frac{averageTime_{GATS}-averageTime_{GTSAIGS}}{averageTime_{GATS}} \quad (13)$$

$$R_R=\frac{averageC_{maxGATS}-averageC_{maxGTSAIGS}}{averageC_{maxGATS}} \quad (14)$$

表示混合算法的效率提升和结果优劣。

表6　算法的消融实验及对比实验的结果(具有2个产品的算例)

Tab.6　Results of ablation experiments and comparative experiments of the algorithm (the case with 2 products)

算例	规模				MILP1	GA			GA-Grd			GATS			文献[9]结果		GTSAIGS		
	N_j	n	m	N_p	*best*	*best*	*average*	*time*	*best*	*average*	*time*	*best*	*average*	*time*	MILP2	GA-VNS	*best*	*average*	*time*
DA01	2	15	15	2	**2048**	2421	2498.50	2.90	2324	2448.20	2.60	2052	2075.05	231.15	**2048**	2074	**2048**	2073.35	246.00
DA02	2	15	15	2	**1560**	1828	1911.15	2.64	1744	1871.20	1.76	**1560**	1562.00	51.77	**1560**	**1560**	**1560**	1560.00	52.71
DA03	2	15	15	2	**2001**	2256	2398.05	2.25	2292	2391.20	1.41	2003	2024.25	168.94	**2001**	2021	2003	2020.85	162.84
DA04	2	15	15	2	**1852**	2163	2269.00	2.19	2065	2203.20	2.17	**1852**	1867.20	261.17	**1852**	**1852**	**1852**	1854.80	266.14
DA05	2	15	15	2	**2073**	2262	2393.15	2.76	2205	2337.60	2.11	**2073**	2073.00	26.40	**2073**	**2073**	**2073**	2073.00	25.08
DA11	3	20	15	2	**2111**	2377	2456.50	2.07	2307	2416.05	1.58	**2111**	2117.50	137.05	**2111**	2169	**2111**	2124.40	128.80
DA12	3	20	15	2	**1609**	1879	1947.40	1.85	1805	1921.95	1.86	**1609**	1609.00	29.35	**1609**	**1609**	**1609**	1609.00	27.31
DA13	3	20	15	2	**1969**	2252	2362.65	3.35	2267	2348.55	1.72	**1969**	2010.25	55.11	**1969**	**1969**	**1969**	1988.90	58.35
DA14	3	20	15	2	**1789**	2162	2253.35	2.81	2087	2200.75	1.84	**1789**	1807.15	255.03	**1789**	**1789**	**1789**	1801.80	250.87
DA15	3	20	15	2	**2086**	2415	2491.00	2.31	2375	2446.05	2.11	2096	2100.30	59.32	**2086**	2100	**2086**	2086.00	61.56
DA21	3	20	20	2	**2403**	2768	2842.85	2.01	2611	2775.30	2.14	**2403**	2403.00	71.22	**2403**	**2403**	**2403**	2403.00	69.06
DA22	3	20	20	2	**1793**	2154	2243.40	2.64	2101	2230.55	1.71	1846	1858.35	46.77	**1793**	1806	1859	1859.00	48.02
DA23	3	20	20	2	**2288**	2627	2721.15	2.85	2558	2670.85	2.30	**2288**	2293.05	58.36	**2288**	**2288**	**2288**	2297.50	56.74
DA24	3	20	20	2	**2186**	2486	2633.90	2.93	2589	2663.45	1.68	**2186**	2186.00	52.26	**2186**	**2186**	**2186**	2186.00	53.01
DA25	3	20	20	2	**2364**	2726	2835.70	1.88	2649	2788.50	1.85	**2364**	2365.05	86.60	**2364**	2373	**2364**	2364.25	82.42

表7　算法的消融实验及对比实验的结果(具有3个产品的算例)

Tab.7　Results of ablation experiments and comparative experiments of the algorithm (the case with 3 products)

算例	规模				MILP1	GA			GA-Grd			GATS			文献[9]结果		GTSAIGS		
	N_j	n	m	N_p	*best*	*best*	*average*	*time*	*best*	*average*	*time*	*best*	*average*	*time*	MILP2	GA-VNS	*best*	*average*	*time*
DA06	2	15	15	3	**2165**	2424	2525.60	2.59	2376	2495.40	1.61	2167	2192.75	68.35	**2165**	2197	**2165**	2172.15	61.42
DA07	2	15	15	3	**1774**	2075	2174.80	2.95	2043	2144.85	1.83	**1774**	1774.00	91.20	**1774**	**1774**	**1774**	1774.00	83.40
DA08	2	15	15	3	**1601**	1909	1966.40	2.45	1854	1923.95	1.53	**1601**	1601.00	30.16	**1601**	**1601**	**1601**	1601.00	28.21
DA09	2	15	15	3	**1929**	2218	2336.70	2.79	2224	2311.65	1.96	1931	1953.20	222.22	**1929**	1931	**1929**	1953.75	210.45
DA10	2	15	15	3	**2417**	2725	2835.00	2.02	2662	2790.60	1.01	**2417**	2432.10	75.46	**2417**	**2417**	**2417**	2437.30	67.33
DA16	3	20	15	3	**2172**	2449	2569.40	3.00	2363	2508.00	2.09	2182	2184.25	52.89	**2172**	**2172**	2188	2188.00	47.59
DA17	3	20	15	3	**1808**	2144	2225.10	2.31	2017	2150.90	2.60	**1808**	1808.00	34.71	**1808**	**1808**	**1808**	1808.00	32.68
DA18	3	20	15	3	**1541**	1845	1920.80	3.11	1720	1874.25	2.10	**1541**	1541.00	147.37	**1541**	1571	**1541**	1541.00	142.52
DA19	3	20	15	3	**1812**	2115	2273.30	1.50	2007	2173.90	2.19	**1812**	1812.00	51.39	**1812**	**1812**	**1812**	1812.00	47.34
DA20	3	20	15	3	**2454**	2664	2761.40	2.70	2511	2695.70	2.05	**2454**	2454.15	19.30	**2454**	**2454**	**2454**	2454.00	18.28
DA26	3	20	20	3	**2489**	2857	2951.35	2.97	2745	2926.80	1.53	**2489**	2510.45	131.52	**2489**	**2489**	2494	2508.45	122.85
DA27	3	20	20	3	**2154**	2498	2616.35	3.24	2466	2590.15	1.64	**2154**	2154.00	46.11	**2154**	**2154**	**2154**	2154.00	40.54
DA28	3	20	20	3	**1860**	2139	2321.60	1.50	2142	2228.85	1.67	**1860**	1860.00	24.39	**1860**	**1860**	**1860**	1860.00	22.27
DA29	3	20	20	3	**2129**	2541	2684.80	2.63	2525	2616.90	2.12	2145	2153.75	163.01	**2129**	2145	**2129**	2149.95	148.00
DA30	3	20	20	3	**2403**	3025	3120.85	2.36	2893	3065.90	1.69	**2640**	2640.00	34.26	**2640**	**2640**	**2640**	2640.00	32.38
DA31	4	30	15	3	**2352**	2741	2831.50	3.01	2641	2803.15	1.91	**2352**	2352.00	21.80	**2352**	**2352**	**2352**	2352.00	21.89
DA32	4	30	15	3	**2174**	2531	2729.50	2.89	2544	2666.00	2.56	**2174**	2174.00	91.74	2192	**2174**	**2174**	2174.00	82.6
DA33	4	30	15	3	**2063**	2428	2561.40	3.22	2332	2523.80	2.18	**2063**	2071.45	81.30	**2063**	**2063**	**2063**	2087.60	76.68
DA34	4	30	15	3	**2113**	2498	2600.60	4.10	2410	2565.25	3.23	**2113**	2113.00	53.42	**2113**	**2113**	**2113**	2113.00	48.13
DA35	4	30	15	3	**2471**	2813	2977.50	2.65	2786	2909.00	1.99	**2471**	2471.00	76.84	2503	2511	**2471**	2471.15	72.82

表8 算法的消融实验及对比实验的结果(具有4个产品的算例)

Tab.8 Results of ablation experiments and comparative experiments of the algorithm (the case with 4 products)

算例	规模				MILP1	GA			GA-Grd			GATS			文献[9]结果		GTSAIGS		
	N_j	n	m	N_p	*best*	*best*	*average*	*time*	*best*	*average*	*time*	*best*	*average*	*time*	MILP2	GA-VNS	*best*	*average*	*time*
DA36	4	30	15	4	**1873**	2345	2455.05	3.20	2234	2403.95	1.82	**1873**	1873.00	66.45	2044	**1873**	**1873**	1873.00	61.26
DA37	4	30	15	4	**2637**	3021	3121.10	3.18	2934	3051.25	2.00	**2637**	2637.15	35.94	**2637**	**2637**	**2637**	2637.00	32.72
DA38	4	30	15	4	**2519**	2831	3007.65	3.38	2780	2929.55	1.80	**2519**	2519.00	39.90	**2519**	**2519**	**2519**	2519.00	35.62
DA39	4	30	15	4	**2165**	2571	2679.40	2.95	2498	2594.20	2.41	**2165**	2165.85	108.62	2184	**2165**	**2165**	2165.00	98.24
DA40	4	30	15	4	**1681**	2059	2159.3	2.37	1958	2091.2	2.43	1695	1706.15	92.61	1757	1695	1695	1709.50	80.41

如图8所示，与GA和GATS相比，GTSAIGS在产品较多的算例求解时间上表现出明显的优势。如图8a所示，产品数量为2的15个算例中，GTSAIGS在9个算例上的计算耗时均有缩短，且最大完工时间的均值在8个算例中优于GATS求得的结果。图9a、图9b分别为算例DA02和DA03求得C_{maxbest}所用时间对比图(以GTSAIGS平均耗时为基准时间)，产品较少导致GTSAIGS的时间优势不明显，两种算法求得C_{maxbest}所用时间相近。图8c所示的20个算例中，GTSAIGS在19个算例中均缩短了求解时间，且与GATS相比，最大完工时间的均值没有明显增大。图8b所示的5个算例中，GTSAIGS只有一个算例的$C_{\max}$均值增大0.19%，在其余算例中的耗时都缩短。如图9c～图9f所示，两种算法在20次求得最优解的耗时对比中，GTSAIGS的耗时较GATS明显缩短。GTSAIGS求得DA35算例最优解的甘特图为图10。

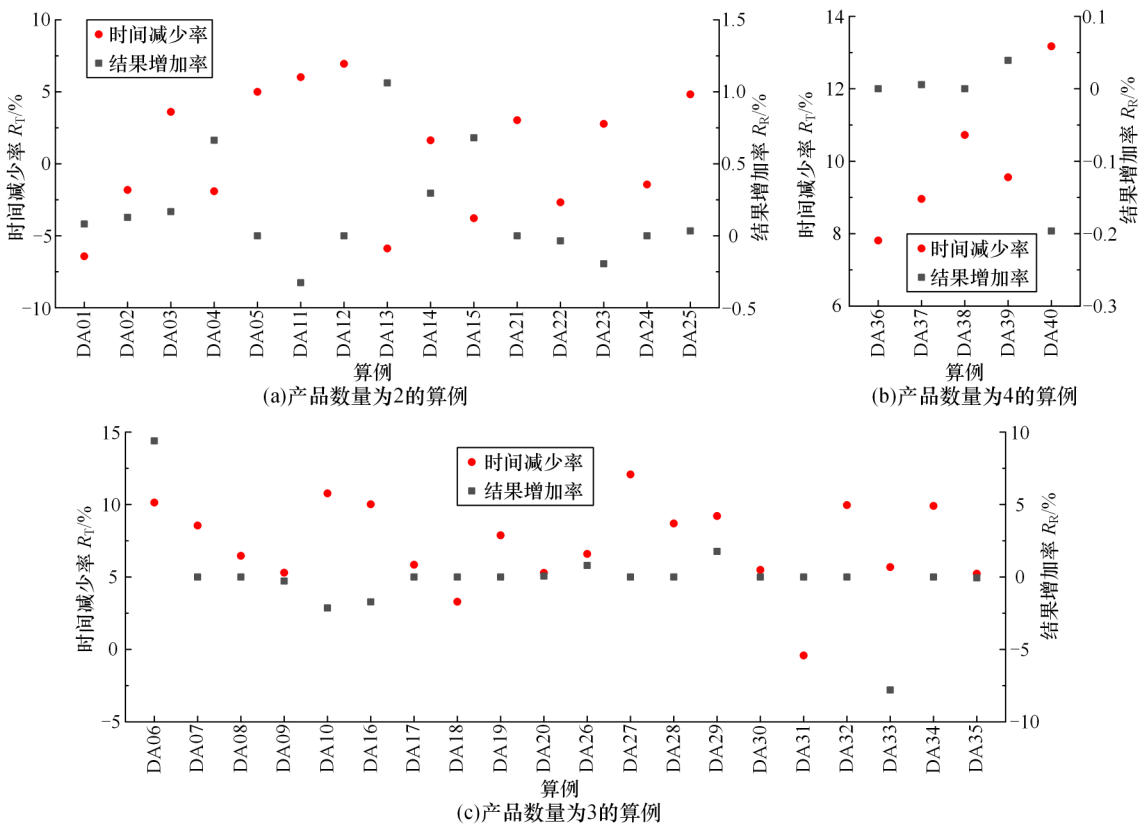


图8 不同算例下GTSAIGS相较于GATS的耗时及结果

Fig.8 Computation time and solution quality between GTSAIGS and GATS for different instances

4.3 结果分析

通过试验结果可知，采用CPLEX对DAJSP的MILP模型求解可得到问题的最优解，但巨大的计算耗时凸显了精确算法在求解此类NP-hard问题时的局限，证明了开发高效启发式算法的必要性。GATS和GTSAIGS在求解质量上均远远优于不包含禁忌搜索的GA和GA-Grd。对比结果显示，引入禁忌搜索可显著提高解质量；仅依遗传算法的全局搜索时，受限于局部搜索能力，难以找到高质量的解。

分解策略和贪心算法是算法设计的核心创新点，两项策略的有效性在不同规模的算例中得到

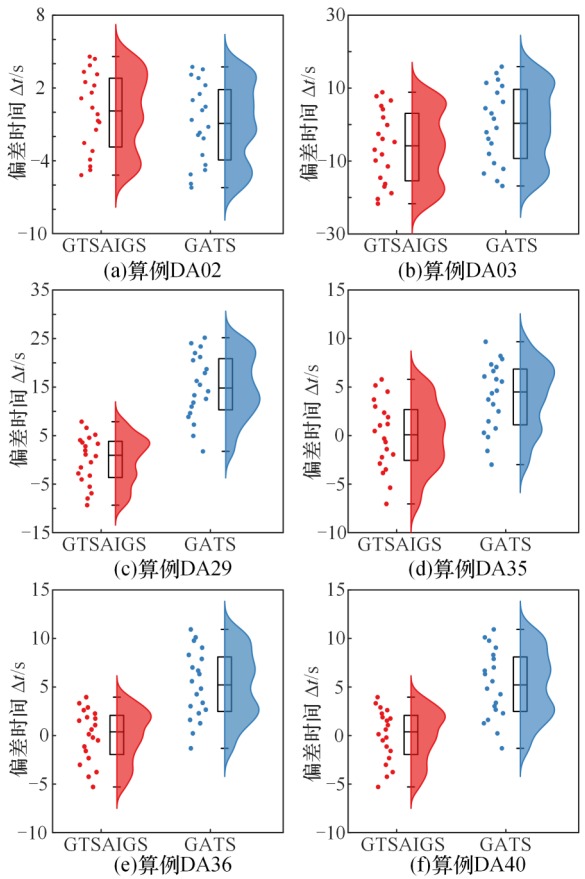


图9　GTSAIGS与GATS在不同算例下的运行时间

Fig.9　Running time of GTSAIGS and GATS in different instances

验证。对比平均性能可发现GTSAIGS在大部分算例上的求解质量优于GA，且耗时更少。综合来看，贪心算法将装配阶段从迭代优化中剥离，使启发式算法能专注于工厂分配和加工排序，避免在更庞大解空间中的无效搜索。因此贪心策略在所提算法中实现了提质、增效的统一。

进一步发现，随着产品数量增加，GTSAIGS算法的耗时优势更为明显，同时大部分算例的C_{max}均值不劣于GATS，证明了GTSAIGS的可靠性和稳定性。深入分析可知，对比算法GATS采用传统的启发式算法求解装配工序的调度解，在每次禁忌搜索的迭代中，对装配机器上的关键块生成众多的邻域解，解码和生成邻域解等高耗时步骤的使用频率也随之增加，最终导致求解时间的延长。算法GTSAIGS基于分解策略，采用贪心算法独立求解装配工序调度解，消除了装配工序迭代优化的耗时，能快速求得装配阶段的调度方案，不仅降低了装配工序的求解时间复杂度，而且保证了结果的可靠性。

4.4　实例测试

为对所提算法的实际应用性能进行深入评估，本节采用不同算法对实际生产案例进行求解和对比。案例来自一家大型数控机床及自动化生产线制造商，其生产模式是典型的分布式生产与装配，该制造商在不同的工业园区有4个工厂，每个工厂的生产设备相同且均配备20台专用设备，包括重型数控加工中心、中型数控车床及铣床、轻型精密设备和检测装置。案例包含45个数控机床零部件，最终产品为8种不同型号的数控机床。该场景下的调度问题可建模为一个以最小化最大完工时间为目标的分布式装配作业车间调度问题。该问题包含45个工件、4个分布式加工工厂、每个工厂20台机器及8个产品（算例请参考：https://github.com/dqgo/DAJSP-instance）。分别采用GTSAIGS和其他算法对案例进行测试，各算法独立运行10次并求得结果的均值和最优值，MILP模型由CPLEX求解运行1次并记录求解时间，结果如表9所示。

表9　实际生产案例的求解结果表

Tab.9　Solution results table for actual production instance

算法	*best*	*average*	*time*
MILP	15166	-	7160.98
GA	16915	17633.3	3.67
GA-Grd	16358	17085.2	3.34
GATS	15234	16839.6	312.68
GTSAIGS	15166	16170.0	217.99

4.5　管理启示

从计算复杂性视角看，生产调度中的加工阶段与装配阶段的决策特性存在显著差异。前者可行的分配与排序方案呈指数级增长，需要大量计算资源进行全局寻优。后者是在上游决策确定后的类单机调度问题，可高效确定一个较优解。基于本文对DAJSP的解耦问题、分层求解思想，为生产管理者提供以下决策建议：

（1）决策资源的分层配置。将复杂的优化资源集中用于加工阶段的生产计划与排序；在装配阶段实施简单高效的算法，如本文改进的贪心算法快速排程。这种模式不仅降低了现场管理的复杂性，还提高了调度系统的响应速度与稳定性。

（2）制造部门的决策解耦。将复杂的调度问题降维处理，允许企业管理者对不同部门进行独立的资源配置和优化。不同生产区域，如工厂、部门和产线都可独立进行生产计划，各生产单元的管理者可专注于内部资源的局部最优配置，做到单元内部最优调度。而后，与其他部门及上层管理者进行协同调度，在大规模的跨部门、产线和工

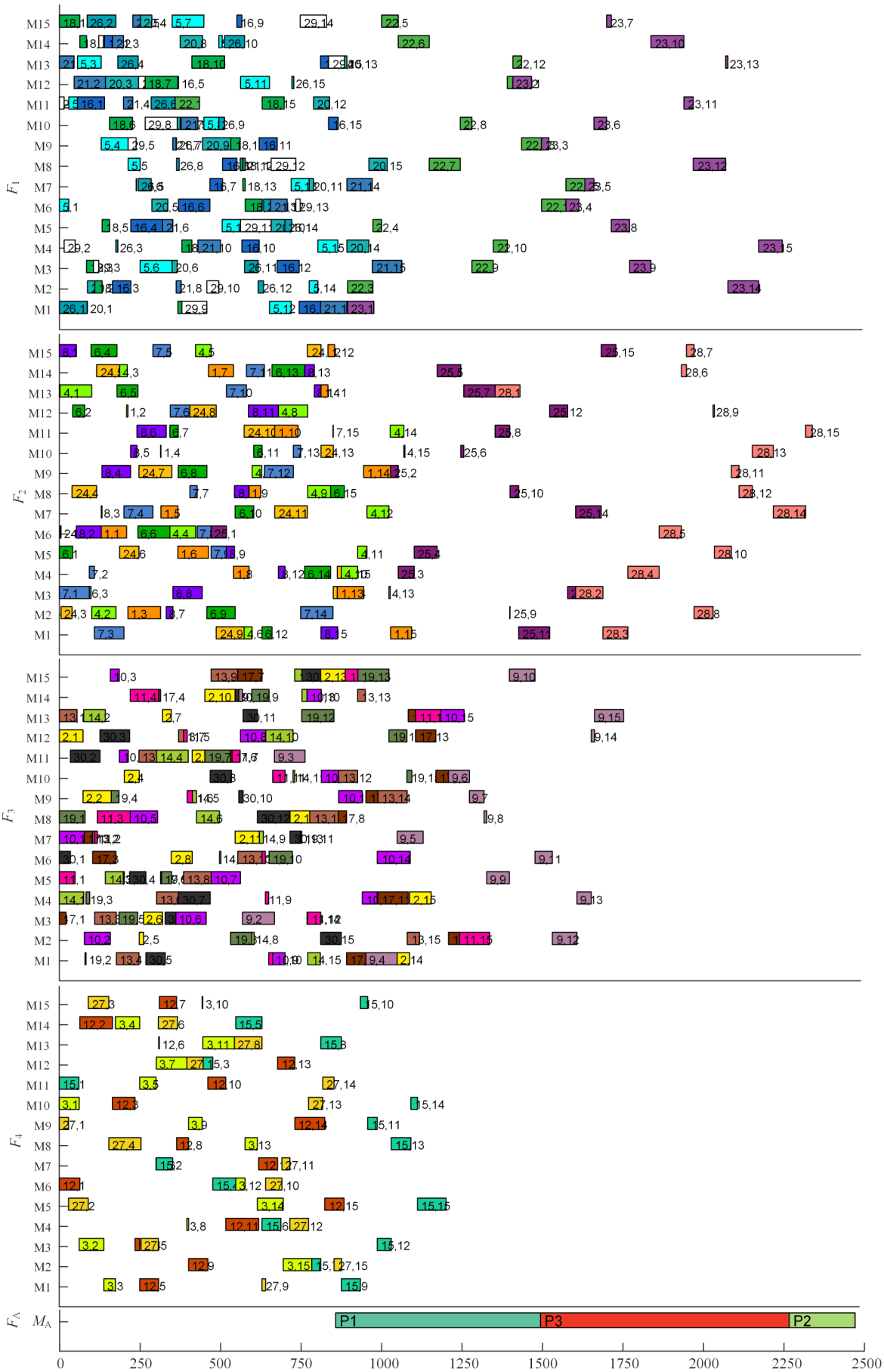


图10 算例DA35最优解的甘特图

Fig.10 Gantt chart of the optimal solution of instance DA35

厂的宏观角度下调整。这种分级管理模式不仅降低了现场管理的复杂度，还赋予了各部门应对突发扰动的灵活性，从而实现大规模复杂生产环境下的整体效益最大化。

5　结论

针对分布式装配作业车间调度问题提出了混合算法GTSAIGS，算法通过引入贪心算法将装配阶段从复杂的迭代优化中剥离，从而将遗传算法与禁忌搜索的搜索资源集中于更具组合爆炸特征的加工子问题，缩短装配阶段的优化迭代耗时。实验结果证实，分解策略应用的混合算法在保证解质量的同时，降低了计算复杂度，展现出优秀的求解性能。

下一步的研究包括设计更合理的邻域结构，提出面向工件跨工厂移动的快速评估方法，进一步提高算法计算速度，以及对包含节能的多目标DAJSP进行优化。

参考文献：

[1] ZHANG Sicheng, LI Xiang, ZHANG Bowen, et al. Multi-objective Optimisation in Flexible Assembly Job Shop Scheduling Using a Distributed Ant Colony System [J]. European Journal of Operational Research, 2020, 283(2): 441-460.

[2] SHI Fei, ZHAO Shikui, MENG Yue. Hybrid Algorithm Based on Improved Extended Shifting Bottleneck Procedure and GA for Assembly Job Shop Scheduling Problem [J]. International Journal of Production Research, 2020, 58(9): 2604-2625.

[3] 李仲华，唐红涛，张伟. 基于改进鲸鱼动态调度算法求解带外协的柔性装配作业车间调度[J]. 机床与液压，2024，52(20)：1-9.
LI Zhonghua, TANG Hongtao, ZHANG Wei. Solving the Scheduling of Flexible Assembly Workshops with Outsourcing Cooperation Based on Improving Whale Dynamic Scheduling Algorithm [J]. Machine Tool & Hydraulics, 2024, 52(20): 1-9.

[4] 石飞，赵诗奎. 基于工序约束链编码的遗传算法求解产品综合调度问题[J]. 中国机械工程，2017，28(20)：2483-2492.
SHI Fei, ZHAO Shikui. Product Comprehensive Scheduling Problems Solved by Genetic Algorithm Based on Operation Constraint Chain Coding[J]. China Mechanical Engineering, 2017, 28(20): 2483-2492.

[5] CHAOUCH I, DRISS O B, GHEDIRA K. A Modified Ant Colony Optimization Algorithm for the Distributed Job Shop Scheduling Problem[J]. Procedia Computer Science, 2017, 112: 296-305.

[6] ŞAHMAN M A. A Discrete Spotted Hyena Optimizer for Solving Distributed Job Shop Scheduling Problems[J]. Applied Soft Computing, 2021, 106: 107349.

[7] JIANG Enda, WANG Ling, PENG Zhiping. Solving Energy-efficient Distributed Job Shop Scheduling via Multi-objective Evolutionary Algorithm with Decomposition[J]. Swarm and Evolutionary Computation, 2020, 58: 100745.

[8] WANG Sihan, LI Xinyu, GAO Liang, et al. An Improved Genetic Algorithm for Distributed Job Shop Scheduling Problem[C]//Intelligent Computing Theories and Application. Cham: Springer, 2021: 37-47.

[9] TIAN Shichen, ZHANG Chunjiang, FAN Jiaxin, et al. A Genetic Algorithm with Critical Path-based Variable Neighborhood Search for Distributed Assembly Job Shop Scheduling Problem[J]. Swarm and Evolutionary Computation, 2024, 85: 101485.

[10] YANG Zhijie, HU Xinkai, LI Yibing, et al. A Q-learning-based Improved Multi-objective Genetic Algorithm for Solving Distributed Heterogeneous Assembly Flexible Job Shop Scheduling Problems with Transfers[J]. Journal of Manufacturing Systems, 2025, 79: 398-418.

[11] 杨绍文，钱斌，胡蓉，等. 混合三维分布估计算法求解分布式加工装配和车辆配送集成调度问题[J]. 电子学报，2024，52(3)：909-923.
YANG Shaowen, QIAN Bin, HU Rong, et al. Hybrid Three-dimensional Estimation of Distribution Algorithm for Integrated Scheduling Problem of Distributed Production Assembly and Vehicle Delivery[J]. Acta Electronica Sinica, 2024, 52(3): 909-923.

[12] 王玉芳，葛嘉荣，缪昇，等. 一种求解柔性作业车间的改进遗传算法[J]. 重庆理工大学学报(自然科学版)，2021，35(9)：152-159.
WANG Yufang, GE Jiarong, MIAO Sheng, et al. An Improve Genetic Algorithm for Solving Flexible Job Shop Scheduling Problem[J]. Journal of Chongqing Institute of Technology, 2021, 35(9): 152-159.

[13] 刘设，李佳欣，田志强，等. 考虑客户满意度的模糊柔性作业车间调度优化方法[J/OL]. 中国机械工程[2026-03-17]. https://link.cnki.net/urlid/42.1294.TH.20260121.1635.018.
LIU She, LI Jiaxin, TIAN Zhiqiang, et al. Fuzzy Flexible Job-shop Scheduling Optimization Approach Considering Customer Satisfaction[J/OL]. China Mechanical Engineering. [2026-03-17]. https://link.cnki.net/urlid/42.1294.TH.20260121.1635.018.

[14] 周豪，张超勇，刘辉，等. 改进Jaya算法求解混合流水车间调度问题[J]. 中国机械工程，2024，35(8)：1462-1471.
ZHOU Hao, ZHANG Chaoyong, LIU Hui, et al. Improved Jaya Algorithm for Solving HFSPS[J]. China Mechanical Engineering, 2024, 35(8): 1462-1471.

（编辑　张　洋）

作者简介：仵政源，男，2001年生，硕士研究生。研究方向为车间生产调度和智能优化算法。发表论文1篇。E-mail：1426095214@qq.com。赵诗奎*（通信作者），男，1984年生，教授、博士研究生导师。研究方向为车间生产调度、智能优化算法。发表论文65篇。E-mail：me_zhaosk@ujn.edu.cn。

本文引用格式：

仵政源，赵诗奎，解瑞建，等. 基于分解策略混合算法的DAJSP求解[J]. 中国机械工程，2026，37(7)：1673-1685.

WU Zhengyuan, ZHAO Shikui, XIE Ruijian, et al. A Hybrid Algorithm Based on Decomposition Strategy for DAJSP Solution[J]. China Mechanical Engineering, 2026, 37(7): 1673-1685.