控制与决策
Control and Decision

基于生成对抗网络的模糊分布式装配流水车间无死锁调度算法

张广辉, 赵成龙, 魏晨轩, 冯彦翔, 李晓玲

引用本文:
张广辉, 赵成龙, 魏晨轩, 等. 基于生成对抗网络的模糊分布式装配流水车间无死锁调度算法[J]. 控制与决策, 2026, 41(4): 1176-1186.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2025.0472

基于生成对抗网络的模糊分布式装配流水车间无死锁调度算法

张广辉[1†]，赵成龙[1]，魏晨轩[1]，冯彦翔[2]，李晓玲[3]

(1. 河北农业大学 信息科学与技术学院，河北 保定 071001；
2. 西安交通大学 自动化科学与工程学院，西安 710049；
3. 长安大学 电子与控制工程学院，西安 710064)

摘　要: 随着全球化和定制化需求的不断发展, 分布式装配流水车间调度问题 (DAFSP) 受到广泛关注. 为增加现实性, 在 DAFSP 的基础上进一步考虑了模糊加工时间和有限缓冲区引发的生产死锁约束, 研究一种新的模糊分布式装配流水车间无死锁调度问题. 针对该问题, 首先, 建立以最小化最大模糊完工时间的计算模型; 然后, 基于 Petri 网提出一种死锁检测和修复算法, 以避免系统死锁状态; 接着, 基于死锁避免算法和生成对抗网络 (GANs), 提出一种基于 GANs 的模糊分布式装配流水车间无死锁调度算法 (GAN-DSA), 既能够保证系统活性又能实现高效调度; 最后, 通过 32 组测试算例实验验证所提出算法的有效性.

关键词: 分布式装配流水车间；有限缓冲区；模糊时间；Petri 网；无死锁调度；生成对抗网络

中图分类号: TP273　　**文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.0472

引用格式: 张广辉, 赵成龙, 魏晨轩, 等. 基于生成对抗网络的模糊分布式装配流水车间无死锁调度算法 [J]. 控制与决策, 2026, 41(4): 1176-1186.

Generative adversarial networks-based deadlock-free scheduling algorithm for fuzzy distributed assembly flowshops

ZHANG Guang-hui[1†]，*ZHAO Cheng-long*[1]，*WEI Chen-xuan*[1]，*FENG Yan-xiang*[2]，*LI Xiao-ling*[3]

(1. School of Information Science and Technology，Hebei Agricultural University，Baoding 071001，China；2. School of Automation Science and Engineering，Xi'an Jiaotong University，Xi'an 710049，China；3. School of Electronics and Control Engineering，Chang'an University，Xi'an 710064，China)

Abstract: With the continuous development of globalization and customization demands, the distributed assembly flowshop scheduling problem (DAFSP) has received widespread attention. To add the practicality, this paper further considers the constraints of fuzzy processing time and production deadlock caused by finite assembly buffer in the DAFSP, and therefore a novel fuzzy distributed assembly flowshop deadlock-free scheduling problem (FDAFDSP) is studied. To address this problem, the calculational model is first developed to minimize the maximum fuzzy completion time. Secondly, a Petri nets-based deadlock detection and repair algorithm is proposed to avoid system deadlock. Then, based on deadlock avoidance algorithms and generative adversarial networks (GANs), a GANs-based deadlock-free scheduling algorithm (GAN-DSA) is established for the FDAFDSP, which can avoid deadlocks and achieve efficient scheduling. Finally, the effectiveness of the GAN-DSA for solving the FDAFDSP is experimentally validated by comparing it with related algorithms on 32 test instances.

Keywords: distributed assembly flowshops；limited buffers；fuzzy time；Petri net；deadlock-free scheduling；generative adversarial networks

0　引　言

随着全球化生产的不断发展, 传统的集中式生产模式已无法适应市场环境, 越来越多的现代制造企业选择将生产环节分散到世界各地, 以充分利用

收稿日期: 2025-05-06；**录用日期:** 2025-08-21.
基金项目: 河北省自然科学基金项目 (F2024204007)；西安交通大学机械制造系统工程国家重点实验室开放课题项目 (sklms2023002).
责任编委: 王凌.
†通信作者. E-mail: ghzhang@hebau.edu.cn.

各地资源, 降低生产成本[1]. 在此背景下, 分布式装配流水车间作为一种新的生产模式, 受到现代企业的广泛关注. 在该模式下, 产品零件在多个地理位置不同的工厂进行加工, 最后统一进行组装[2-3]. 该模式广泛存在于航空航天、电子装备、汽车制造等高端装备制造领域, 尤其适用于多工序、定制化以及分布式协同生产的复杂产品制造. 目前, 关于分布式装配流水车间调度问题 (DAFSP) 的研究取得了一定成果[4-6], 但是, 现有研究通常理想化处理实际生产中的关键因素, 如仅考虑无限缓冲区和确定性时间加工. 然而, 有限缓冲区和时间模糊性在实际生产中更加普遍, 会引发系统死锁而使得调度计划失效, 严重影响生产效率, 研究有限缓冲区和模糊环境下的分布式装配流水车间无死锁调度问题更具意义.

目前, 关于 DAFSP 的研究主要集中在确定性生产环境, 然而, 实际生产是一个复杂动态的不确定过程, 受到机器老化、机器故障、工人经验等诸多不确定因素的影响[7-8], 这些因素往往导致生产加工时间的不确定性. 为增加问题的现实性, 模糊环境下的 DAFSP 开始受到关注. Cheng 等[7] 为求解模糊 DAFSP, 以最小化最大模糊完工时间为优化目标, 提出了一种基于问题关键工厂和区域生物地理学的混合进化算法; 佘明哲等[9] 以最小化模糊完工时间和模糊总能耗为目标, 通过融合自适应变邻域局部搜索策略, 提出了一种混合交叉熵算法来求解模糊 DAFSP 的低碳调度问题; Li 等[10] 针对模糊加工和装配时间的 DAFSP, 开发了一种改进的帝国主义竞争算法, 利用历史数据, 实现了策略的自适应合作; Zuo 等[11] 提出了一种人工蜂群算法来求解模糊 DAFSP, 通过子种群自适应划分、全局和局部搜索的多样化组合等策略, 实现了算法性能的提升.

在实际生产中, 由于企业生产环境和经济成本等因素, 缓冲区容量通常是有限的, 因此, 研究带有有限缓冲区的 DAFSP 更加符合生产实际. 自 Hatami 等[4] 首次提出 DAFSP 以来, 关于 DAFSP 的研究仍然主要集中于无限缓冲区情形, 求解方法通常采用进化算法[5,12]. 目前, 尽管少数学者研究了阻塞和无等待约束下的 DAFSP[13-15], 但是这些研究具有一定局限性, 主要表现在有限缓冲区约束仅应用于分布式加工阶段, 而装配阶段仍然是无限缓冲区的理想情况. 实际上, 加工阶段的有限缓冲区约束只是暂时阻塞工件的加工, 不会导致生产完全停滞. 然而, 在有限装配缓冲区下, 由于产品的装配操作, 整个生产系统会因发生死锁而完全停滞[16-18]. 考虑到目前普遍缺乏对死锁建模和控制的研究, 现有调度算法难以处理死锁问题, 因此, 亟需建立能够协同解决死锁控制和优化调度的无死锁调度算法.

综上, 目前关于模糊 DAFSP 的研究还很初步, 鲜有综合考虑模糊时间和有限装配缓冲区的无死锁 DAFSP 的研究. 因此, 本文将综合考虑模糊环境和死锁约束, 研究一种新的模糊分布式装配流水车间无死锁调度问题 (FDAFDSP). 另外, 鉴于传统进化算法在解决复杂组合优化问题时存在计算复杂度高、易于早熟收敛等问题, 且 FDAFDSP 中包含诸多领域知识可以辅助问题求解, 因此, 建立高效的学习型求解方法是本文主要关注的. 生成对抗网络 (GANs)[19] 是一种深度学习模型, 在学习历史数据方面非常高效[20-21], 并在一些组合优化问题上得到了成功应用[22-23]. 因此, 本文提出一种基于 GANs 的无死锁调度算法 (GAN-DSA), 以优化 FDAFDSP 的最大模糊完工时间. 本文主要内容如下:

1) 问题层面: 综合考虑模糊时间和有限缓冲区约束, 研究模糊分布式装配流水车间的无死锁调度问题, 建立一种基于 Petri 网的死锁检测和修复策略, 以有效处理 FDAFDSP 的无死锁调度问题;

2) 算法层面: 提出一种基于生成对抗网络的全局搜索算子和两种无死锁局部搜索算子, 通过嵌入死锁检测和修复策略到算法, 提出一种新的无死锁调度算法, 以实现 FDAFDSP 死锁控制和系统调度的整体优化.

1　问题描述与模型

1.1　问题描述

FDAFDSP 由模糊加工和模糊装配两个连续制造阶段组成, 其中模糊是指加工时间和装配时间具有不确定性. 考虑模糊环境和死锁约束的 FDAFDSP 描述如下.

在加工阶段, n个工件$N=\{J_1,J_2,\ldots,J_n\}$被分配到f个相同的工厂$F=\{F_1,F_2,\ldots,F_f\}$进行加工. 每个工厂是由m台机器$M=\{M_1,M_2,\ldots,M_m\}$构成的流水车间, 每个工件按照机器顺序完成m道工序$\{O_1,O_2,\ldots,O_m\}$.

在装配阶段, 配置一个容量有限的装配缓冲区B_A和一台装配机器M_A. n个工件完成加工后, 按照其在最后一道工序O_m (即其所分配工厂中的最后一台机器) 上的完工时间, 依次进入B_A等待组装, 最终在M_A上组装成L个产品, $L=\{L_1,L_2,\ldots,L_l\}$. 每个产品均有一个装配计划, 决定了装配产品的工件构成. 工件在加工阶段完成加工后, 必须进入装配缓冲区, 在一个产品装配计划内的工件集齐后, 其才可

组装. 所有产品的装配计划集合记为$\mathrm{AP}=\{\mathrm{AP}_1,\dots,\mathrm{AP}_c,\dots,\mathrm{AP}_l\}$, 其中$\mathrm{AP}_c$为产品$L_c$的装配计划. 对于任意$p\neq k$, 有$\mathrm{AP}_p\bigcap\mathrm{AP}_k=\varnothing$, 且$\bigcup_{c=1}^{l}\mathrm{AP}_c=N$, 以确保所有工件同时且只能装配到一个产品.

由于缓冲区容量有限且每个产品有特定的装配计划, 一旦缓冲区占满且不包含任何一个产品装配计划内的全部工件, 将会出现分布式加工阶段的工件无法进入缓冲区, 同时装配缓冲区的工件也无法进行后续装配操作的情况, 致使系统发生死锁, 生产无限堵塞甚至瘫痪. 死锁的产生无法根据调度计划直接判断, 而是依赖于工件加工完成后进入缓冲区的序列. 通过对该序列进行动态的推演和分析, 才能识别因资源冲突导致的死锁.

所提出 FDAFDSP 是一类新的生产调度问题, 它在 DAFSP 的基础上进一步考虑了模糊加工时间和有限装配缓冲区引发的死锁约束, 求解更加复杂和困难, 主要涉及两个子问题: 1) 解的死锁处理问题, 包括解的死锁判断和修复; 2) 工件和产品的生产调度问题, 包括分配工件到分布式工厂, 确定每个工厂的工件加工次序、工件进入装配缓冲区的次序和产品的装配次序. FDAFDSP 的优化目标是在满足模糊加工时间和无死锁约束的前提下, 寻找一个工件组装成产品的无死锁调度方案, 以最小化最大模糊完工时间.

FDAFDSP 除具有处理时间不确定性和有限装配缓冲区引发的死锁约束, 本文对其做出如下假设:

1) 所有工件在每个工厂中的每台机器上均必须按照相同的顺序加工, 且必须通过指定工厂中的所有机器;

2) 任一工件在任意时刻只能在一台机器上加工, 每台机器只能同时加工一个工件;

3) 机器的设置时间包含在加工时间中, 工件运输时间不予考虑;

4) 工件一旦分配到某一工厂, 则不能再转移至其他工厂加工;

5) 一旦某产品装配计划内的全部工件均已到达装配缓冲区, 则立即开始装配, 同时释放所占用的缓冲区容量.

1.2 符号定义

除问题描述中给出的符号, 引入以下符号对 FDAFDSP 进行建模.

i: 工件索引, $i\in 1,2,\dots,n$;

j: 加工机器索引, $j\in 1,2,\dots,m$;

k: 工厂索引, $k\in 1,2,\dots,f$;

c: 产品索引, $c\in 1,2,\dots,l$;

$\pi_{\mathrm{seq}}=[\pi_{\mathrm{seq}}(1),\pi_{\mathrm{seq}}(2),\dots,\pi_{\mathrm{seq}}(n)]$: 工件的加工顺序和进入装配缓冲区的顺序;

$\pi_k=[\pi_k(1),\pi_k(2),\dots,\pi_k(n_k)]$: 分配到工厂$F_k$的工件加工序列;

$\pi=\pi_1,\pi_2,\dots,\pi_f$: 工件在分布式工厂的调度方案;

$\widetilde{\mathrm{CM}}_{\max}$: 全部工件在加工阶段的模糊完工时间;

$\tilde{C}_{\max}$: 全部产品在装配阶段的模糊完工时间, 即最大模糊完工时间;

$\tilde{S}_{i,j}$: 工件J_i在机器M_j上的模糊开始时刻;

$\tilde{C}_{i,j}$: 工件J_i在机器M_j上的模糊完工时刻;

$\tilde{P}_{i,j}$: 工件J_i在机器M_j上的模糊加工时间;

$\widetilde{\mathrm{SA}}_c$: 产品$L_c$在装配机器$M_A$上的模糊开始时刻;

$\widetilde{\mathrm{CA}}_c$: 产品$L_c$在装配机器$M_A$上的模糊完工时刻;

$\widetilde{\mathrm{PA}}_c$: 产品$L_c$的模糊装配时间;

$\sigma=[\sigma(q),\sigma(2),\dots,\sigma(l)]$: 基于工件序列$\pi_{\mathrm{seq}}$和产品装配计划获得的产品装配顺序.

1.3 计算模型

由于 FDAFDSP 中的死锁涉及复杂的资源占用和释放的动态过程, 难以通过不等式约束进行精确定义, 使得数学规划模型在刻画此类问题时存在很大困难, 因此, 本文通过建立递推计算模型来对可行解进行适应度值计算. FDAFDSP 的计算模型如下所示:

$$\begin{cases}\tilde{C}_{\pi_{\mathrm{seq}}(n),m}=\mathrm{Time};\\ \tilde{S}_{\pi_{\mathrm{seq}}(n),m}=\tilde{C}_{\pi_{\mathrm{seq}}(n),m}-\tilde{P}_{\pi_{\mathrm{seq}}(n),m}.\end{cases}\tag{1}$$

$$\begin{cases}\tilde{C}_{\pi_{\mathrm{seq}}(i),m}=\begin{cases}\tilde{S}_{\pi_{\mathrm{seq}}(i+1),m},\ \pi_{\mathrm{seq}}(i),\pi_{\mathrm{seq}}(i+1)\in\pi_k,\\ \quad\forall i\in\{1,2,\dots,n-1\},\ \forall k\in F;\\ \tilde{C}_{\pi_{\mathrm{seq}}(i+1),m}-1,\ \mathrm{otherwise};\end{cases}\\ \tilde{S}_{\pi_{\mathrm{seq}}(i),m}=\tilde{C}_{\pi_{\mathrm{seq}}(i),m}-\tilde{P}_{\pi_{\mathrm{seq}}(i),m}.\end{cases}\tag{2}$$

$$\begin{cases}\tilde{C}_{\pi_k(n_k),j}=\tilde{S}_{\pi_k(n_k),j+1};\\ \tilde{S}_{\pi_k(n_k),j}=\tilde{C}_{\pi_k(n_k),j}-\tilde{P}_{\pi_k(n_k),j},\\ \quad\forall k\in\{1,2,\dots,f\},\ \forall j\in\{1,2,\dots,m-1\}.\end{cases}\tag{3}$$

$$\begin{cases}\tilde{C}_{\pi_k(i),j}=\min(\tilde{S}_{\pi_k(i),j+1},\tilde{S}_{\pi_k(i),j});\\ \tilde{S}_{\pi_k(i),j}=\tilde{C}_{\pi_k(i),j}-\tilde{P}_{\pi_k(i),j},\\ \quad\forall i\in\{1,2,\dots,n_k-1\},\ \forall k\in\{1,2,\dots,f\},\\ \quad\forall j\in\{1,2,\dots,m-1\}.\end{cases}\tag{4}$$

$$\widetilde{\mathrm{CM}}_{\max}=\tilde{C}_{\pi_{\mathrm{seq}}(n),m}-\min_{i=1}^{n}\{\tilde{S}_{\pi_{\mathrm{seq}}(i),1}\}. \tag{5}$$

$$\begin{cases}\widetilde{\mathrm{SA}}_{\sigma(1)}=\max\limits_{i\in\mathrm{AP}_{\sigma(1)}}\{\tilde{C}_{i,m}\};\\ \widetilde{\mathrm{CA}}_{\sigma(1)}=\widetilde{\mathrm{SA}}_{\sigma(1)}+\widetilde{\mathrm{PA}}_{\sigma(1)}.\end{cases} \tag{6}$$

$$\begin{cases}\widetilde{\mathrm{SA}}_{\sigma(c)}=\max\{\widetilde{\mathrm{CA}}_{\sigma(c-1)},\ \max\limits_{i\in\mathrm{AP}_{\sigma(c)}}\{\tilde{C}_{i,m}\}\};\\ \widetilde{\mathrm{CA}}_{\sigma(c)}=\widetilde{\mathrm{SA}}_{\sigma(c)}+\widetilde{\mathrm{PA}}_{\sigma(c)},\ \forall c\in\{2,3,\ldots,l\}.\end{cases} \tag{7}$$

$$\tilde{C}_{\max}=\max_{c=1}^{l}\{\widetilde{\mathrm{CA}}_{\sigma(c)}\}-\min_{i=1}^{n}\{\tilde{S}_{i,1}\}. \tag{8}$$

其中: 式 (1) 计算π_{seq}中最后一个工件在分布式工厂的完工时刻, 记为Time, 并据此计算相应开始加工时刻. 式 (2) 用于计算π_{seq}中前$n-1$个工件在分布式工厂的开始时刻和完工时刻. 但是需要注意的是, 若$\pi_{\mathrm{seq}}(i)$和$\pi_{\mathrm{seq}}(i+1)$在同一工厂, 则$\pi_{\mathrm{seq}}(i)$的完工时刻等于$\pi_{\mathrm{seq}}(i+1)$的开始时刻; 否则, 前者的完工时刻等于后者的完工时刻减去 1 个时间单位, 以保证工件按照π_{seq}中的顺序进入装配缓冲区. 式 (3) 和 (4) 计算工件在前$m-1$台机器上的完工时刻和开始时刻. 式 (5) 计算所有工件在加工阶段的最大完工时间. 式 (6) 和 (7) 计算产品的开始时刻和完工时刻. 式 (8) 计算最大模糊完工时间, 即目标值.

1.4　模糊数运算

鉴于 FDAFDSP 的加工时间和装配时间的不确定性, 本文使用三角模糊数对其进行描述和操作, 主要涉及到模糊数的加法运算、比较运算和极值运算[24]. 三角模糊数由三元数组组成, 假如有两个模糊数$\tilde{A}=(a_1,a_2,a_3)$和$\tilde{B}=(b_1,b_2,b_3)$, 加法运算定义为$\tilde{A}+\tilde{B}=(a_1+b_1,a_2+b_2,a_3+b_3)$.

为了比较$\tilde{A}$与$\tilde{B}$的大小关系, 定义以下比较准则: 对于$\tilde{A}$, 令$c_1(\tilde{A})=(a_1+2a_2+a_3)/4$, $c_2(\tilde{A})=a_2$, $c_3(\tilde{A})=a_3-a_1$; 对于$\tilde{B}$, 令$c_1(\tilde{B})=b_1+2b_2+b_3$, $c_2(\tilde{B})=b_2$, $c_3(\tilde{B})=b_3-b_1$. 若 $c_1(\tilde{A})>(<)c_1(\tilde{B})$, 则$\tilde{A}>(<)\tilde{B}$. 若$c_1(\tilde{A})=c_1(\tilde{B})$, 则比较$c_2(\tilde{A})$与 $c_2(\tilde{B})$: 若 $c_2(\tilde{A})>(<)c_2(\tilde{B})$, 则 $\tilde{A}>(<)\tilde{B}$; 若 $c_2(\tilde{A})=c_2(\tilde{B})$, 则比较$c_3(\tilde{A})$与$c_3(\tilde{B})$, 若$c_3(\tilde{A})>(<)c_3(\tilde{B})$, 则$\tilde{A}>(<)\tilde{B}$.

为获得$\tilde{A}$和$\tilde{B}$的极值, 定义如下极值运算: 若$\tilde{A}>(<)\tilde{B}$, 则$\tilde{A}\bigvee(\bigwedge)\tilde{B}=\tilde{A}$; 否则, $\tilde{A}\bigvee(\bigwedge)\tilde{B}=\tilde{B}$, 其中符号$\bigvee(\bigwedge)$为极大 (极小) 运算.

1.5　FDAFDSP 的 Petri 网模型

Petri 网模型能够清楚地表示工件的制造过程, 可以更好地分析和处理死锁问题, 且在描述系统动态行为方面具备良好的表达能力, 已在制造系统死锁问题的研究中得到广泛应用. 因此, 本文采用 Petri 网对产品装配和死锁特征进行建模.

Petri 网可表示为一个四元组$N=(P,T,F,W)$. 其中: P为库所 (place) 集合; T为变迁 (transition) 集合; F为有向弧 (arc) 集合, 即$F\subseteq(P\times T)\bigcup(T\times P)$; W为有向弧的权值. P、T、F分别由圆圈、方框和带箭头的有向弧表示. 对于任意$x\in P\bigcup T$, ${}^\bullet x=\{y\in P\bigcup T|(y,x\in F)\}$表示$x$所有输入元素的集合; 同理, $x^\bullet$为x所有输出元素的集合, 这里的元素包括库所和变迁. Petri 网N的一个状态M是一个映射$M:P\to H$, 其中H为非负整数集合. 对于一个库所$p\in P$和状态M, $M(p)$为在M状态下库所p的令牌数量. N的状态M表示为$M(p)=M(p_1)p_1+M(p_2)p_2+\ldots+M(p_n)p_n$. 含有初始状态$M_0$的 Petri 网记作$(N,M_0)$, 含有目标状态的$M_F$的 Petri 网记作$(N,M_F)$. 若$t\in T$, 满足$\forall p\in{}^\bullet t$, $M(p)>0$, 则称变迁t在状态M下可以触发, 记作$M[t\rangle$. 若在M状态下变迁t触发, 则导致系统状态从M转移至新状态M', 记作$M[t\rangle M'$, 其中t也可以是一个变迁集合, 表示一个变迁序列, 即$T_{\mathrm{seq}}=\{t_1,t_2,\ldots,t_n\}$.

为建立 FDAFDSP 的 Petri 网模型, 定义库所P_i表示工件i加工完成, P_i'表示工件i在B_A中等待, PL_c表示产品L_c装配完成, B_A为有限装配缓冲区. 变迁T_i表示在分布式工厂加工完成的工件i进入缓冲区B_A, T_c'表示缓冲区中所有属于产品L_c的工件组装成产品L_c, 则 FDAFDSP 的 Petri 网模型N可描述为$(N,M_0)=(P_i\bigcup P_i'\bigcup\mathrm{PL}_c,T_i\bigcup T_c',F,W)$, 建立步骤如下所示:

step 1: 对每个产品L_c的每个工件i构建一条库所变迁路径, 记作$\mathrm{PT}_i=P_{i,1}T_{i,2}P_{i,3}'T_{c,4}'\mathrm{PL}_c$, 库所与变迁间用有向弧连接, 且权重为 1;

step 2: 对于库所B_A, 每个工件进入有限装配缓冲区占用一个资源 (令牌), 需要消耗库所B_A中的一个令牌, 库所B_A与P_i'相连权重为 1;

step 3: 当一个产品L_c的组装变迁$T_{c,4}'$触发时, 表示系统中产品L_c开始组装, 同时释放缓冲区中占用的资源, 将$T_{c,4}'$与B_A用有向弧连接, 权重等于$T_{c,4}'$输入元素的数量.

2　基于 GANs 的无死锁调度算法

2.1　解的表示与种群初始化

考虑到 FDAFDSP 的两阶段特性, 一个解在加工阶段需要同时明确工件到工厂的分配信息和各工厂的工件加工顺序信息, 在装配阶段需要明确工件进入装配缓冲区的次序, 以便装配产品和处理死锁.

因此, 本文提出一种双向量排列编码来表示FDAFDSP的一个解, 具体如下: 一个解由工件优先级向量π_{seq}和工厂分配向量π_{fac}共同表示, 记作$\pi=\{\pi_{\text{seq}},\pi_{\text{fac}}\}$. 其中: π_{seq}为由工件编号构成的一个序列, 即$\pi_{\text{seq}}(i)\in\{1,2,\ldots,n\}$; π_{fac}为由工厂编号构成的一个序列, 即$\pi_{\text{fac}}(k)\in\{1,2,\ldots,f\}$.

对于一个编码$\pi=\{\pi_{\text{seq}},\pi_{\text{fac}}\}$, 可以方便地解码为FDAFDSP的一个可执行调度解, 具体如下: 在加工阶段, 根据π_{fac}依次分配π_{seq}中的工件至各工厂, 再根据π_{seq}的顺序确定每个工厂中所分配工件的加工次序; 在装配阶段, 加工完成的工件按照π_{seq}的顺序依次进入装配缓冲区, 当一个产品装配计划内的全部工件加工完成且均进入装配缓冲区后, 该产品开始准备装配.

为阐述编解码过程, 考虑一个6个工件、2个工厂、2个产品的FDAFDSP例子进行说明, 装配计划为$\text{AP}_1=\{2,4,6\}$和$\text{AP}_2=\{1,3,5\}$. 一个解可表示为$\pi=\{\pi_{\text{seq}},\pi_{\text{fac}}\}$, $\pi_{\text{seq}}=[3,4,2,6,5,1]$和$\pi_{\text{fac}}=[1,2,2,1,1,2]$. 在加工阶段: 首先, 根据$\pi_{\text{fac}}$确定工件$\{1,4,5\}$分配到工厂1, 工件$\{2,3,6\}$分配到工厂2. 然后, 分配给工厂1中的工件按照在π_{seq}中的顺序进行加工, 即$4\to5\to1$; 同理, 工厂2中工件的加工顺序为$3\to2\to6$. 在装配阶段: 加工完成的工件按照π_{seq}的顺序进入装配缓冲区, 即$3\to4\to2\to6\to5\to1$, 然后按照产品装配计划中工件的完成时间, 确定产品的装配次序. 这样, 得到一个完整的问题调度.

当初始化规模为Ps的种群时, 采用随机初始化方法, 即随机生成每个个体的π_{seq}和π_{fac}向量, 其中要对π_{seq}进行死锁检测和修复. 然后选择适应度较优的前ω个个体作为精英种群, 而剩余的其他个体作为保留种群.

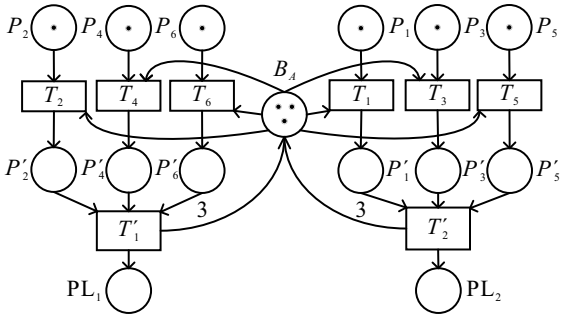
2.2 死锁检测与修复

由于没有考虑资源约束和死锁情况, 算法在进化中的新生成个体可能解码为一个不可行调度解, 需要检测出不可行解并修复其为无死锁的可行解. 基本过程如下: 利用Petri网对新个体的π_{seq}进行死锁检测, 判断该个体是否为可行解, 若为不可行解, 则判断出其中的死锁工件, 然后基于死锁工件对π_{seq}进行调整, 直至消除死锁状态.

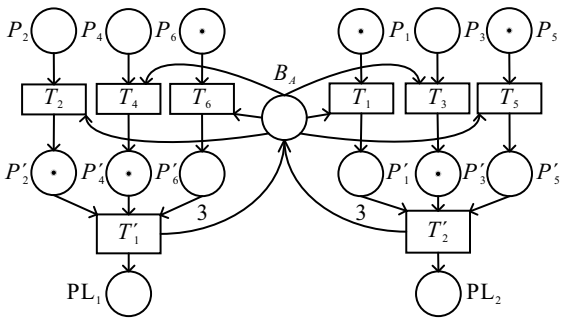
2.2.1 基于Petri网的解的可行性检测

基于问题的Petri网模型, 对一个解的死锁检测, 是对解的双向量编码中向量π_{seq}进行检测. 首先, 根据π_{seq}构建变迁序列, 将π_{seq}中的每个元素i逐个映射为变迁T_i, 并按照π_{seq}中的顺序进行排列, 构成一个变迁序列$T_{\text{seq}}=T_1,T_2,\ldots,T_n$. 然后, 在带有初始状态$M_0$的Petri网$N$中, 逐个激活$T_{\text{seq}}$中的变迁, 不断更新Petri网的状态. 最后, 若所有变迁均能够依次触发成功, 达到目标状态M_F, 即$M_0[T_{\text{seq}}\rangle M_F$, 则该解为可行解; 否则为不可行解.

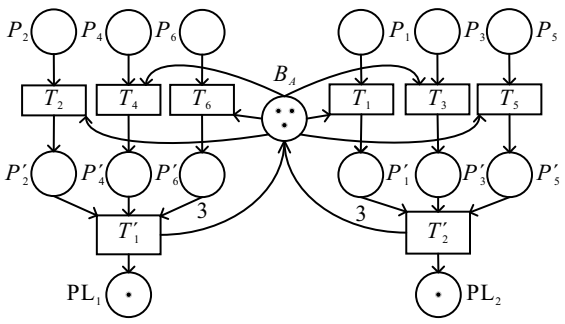
以第2.1节例子中的解$\pi=\{\pi_{\text{seq}},\pi_{\text{fac}}\}$为例来说明可行性检测过程. 其中: $\pi_{\text{seq}}=[3,4,2,6,5,1]$, $\pi_{\text{fac}}=[1,2,2,1,1,2]$. 首先, 构建该问题的初始Petri网$(N,M_0)$, 这里缓冲区大小$B_A=3$, Petri网初始状态定义为$M_0=P_1+P_2+P_3+P_4+P_5+P_6+3B_A$, 如图1(a)所示; 然后, 根据$\pi_{\text{seq}}=[3,4,2,6,5,1]$构建变迁序列$T_{\text{seq}}=\{T_3,T_4,T_2,T_6,T_5,T_1\}$, 并按照顺序逐个尝试触发每个变迁$T_i$. 由于$T_3$的前置库所为$\{P_3,B_A\}$满足$\forall p\in{}^\bullet T_3$, $M(p)>0$, T_3可被触发, Petri网状态更新为$M_1=P_1+P_2+P_4+P_5+P_6+P_3'+2B_A$; 同理, 变迁$T_4$成功触发使得Petri网状态变为$M_2=P_1+P_2+P_5+P_6+P_3'+P_4'+B_A$; 变



(a) 初始状态M_0



(b) 死锁状态M_S



(c) 目标状态M_F

图1 Petri网模型的3种状态

迁T_2触发使得 Petri 网状态变为$M_3 = P_1 + P_5 + P_6 + P_3' + P_4' + P_2'$; 到变迁$T_6$时, 由于其前置库所中的$B_A$无可用令牌, 导致变迁无法触发. 系统无法达到图 1(c) 所示目标状态$M_F = \mathrm{PL}_1 + \mathrm{PL}_2$, 而是进入图 1(b) 所示的死锁状态$M_s = P_1 + P_5 + P_6 + P_2' + P_4' + P_3'$, 故该解为一个不可行解.

2.2.2　死锁工件的判断

对于一个不可行解$\pi = \{\pi_{\mathrm{seq}}, \pi_{\mathrm{fac}}\}$, 其死锁原因是工件按照$\pi_{\mathrm{seq}}$中的顺序进入装配缓冲区时, 由于装配计划限制和缓冲区容量有限, 在缓冲区容量已满且不包含任一产品装配计划内的所有工件时, 缓冲单元被占用无法释放, 导致工件无限堵塞并引发系统死锁. 由此可见, 当死锁发生时, 判断哪些工件无限占用缓冲单元, 是修复死锁的关键. 本文将这类造成系统持续堵塞、缓冲单元无法释放的工件定义为死锁工件. 而工件进入装配缓冲区的顺序是由π_{seq}决定的, 若能够准确识别出π_{seq}中导致缓冲区堵塞的死锁工件, 对死锁工件及其后续工件的调度顺序进行合理调整, 则可有效解除缓冲区堵塞状态, 从而恢复系统的可行性. 识别死锁工件的步骤如下.

step 1: 当系统死锁后, Petri 网进入死锁状态M_s. 在该状态下, 可通过 Petri 网中缓冲区对应库所P_i'中所持有的令牌, 识别出当前堵塞在装配缓冲区中的工件.

step 2: 按照π_{seq}中的顺序依次判断当前滞留在缓冲区中的工件是否有进入装配缓冲区的条件. 当一个工件进入缓冲区时, 占用一个缓冲单元, 记录当前装配缓冲区中剩余位置的数量, 记为B_S. 同时确定工件所属产品, 统计在当前缓冲区中组装成该产品的剩余工件数量, 记为P_x. 当$B_S \geqslant P_x$时, 该工件可以进入缓冲区; 否则, 该工件为死锁工件, 后面的工件不用再进行判断.

如图 1(b) 所示: 工件$\{J_3, J_4, J_2\}$堵塞在装配缓冲区中, $\{P_3', P_4', P_2'\}$这 3 个库所均各持有一个令牌, 系统死锁而无法装配. 首先, 工件 3 进入至缓冲区时, 占用一个缓冲单元, 缓冲区剩余位置为 2, 记为$B_S = 2$. 工件 3 属于产品 2, 装配成产品 2 还需要 2 个工件, 记为$P_x = 2$, 当$B_S \geqslant P_x$时, 该工件可以进入. 然后, 工件 4 进入至缓冲区时, 占用一个缓冲单元, 缓冲区剩余位置为 1, 记为$B_S = 1$. 工件 4 属于产品 1, 装配产品 1 还需要 2 个工件, 记为$P_x = 2$, 当$B_S < P_x$时, 工件 4 不可以进入装配缓冲区, 工件 4 为死锁工件.

2.2.3　死锁修复

本文设计一种基于死锁工件的死锁修复方法, 能够在较少调整工件次数下将不可行解转化为可行解. 对于一个不可行解$\pi = \{\pi_{\mathrm{seq}}, \pi_{\mathrm{fac}}\}$, 修复步骤如下.

step 1: 当系统陷入死锁时, Petri 网为死锁状态M_s, 根据第 2.2.2 节中所提出死锁工件判定方法, 判断缓冲区中的死锁工件, 死锁工件在π_{seq}的索引记作J_d. π_{seq}中位于死锁工件前的工件为非死锁工件, 记为$D_{\mathrm{job}} = \{\pi_{\mathrm{seq}}(1), \pi_{\mathrm{seq}}(2), \ldots, \pi_{\mathrm{seq}}(J_d)\}$.

step 2: 判断D_{job}中工件的所属产品, 并统计组装成产品所需剩余工件数量. 优先选择所需剩余工件数量最少的产品作为修复对象, 若有多个产品所需剩余工件数量最少且相同, 则随机选择其中一个产品作为修复对象.

step 3: 创建修复对象产品的所需剩余工件集合, 从该集合中随机选择一个工件, 在π_{seq}中与死锁工件交换位置, 交换后的新装配顺序记为π'_{seq}.

step 4: 根据第 2.2.1 节所提出方法判断π'_{seq}是否可行, 能否完成所有产品的装配. 若能够完成, 则π'_{seq}为修复的可行解; 否则, 重复 step 1 ～ step 4, 直至将π修复为可行解.

2.3　基于 GANs 的全局搜索

GANs 是一种深度学习模型, 由生成器 (Generator, G) 和判别器 (Discriminator, D) 两个神经网络组成, 通过二者不断对抗和学习来实现问题的求解[22-23]. 本文提出一种基于 GANs 的全局搜索算子, 首先通过生成器与判别器的不断对抗对 GANs 进行训练, 然后利用训练后的生成器生成具有一定质量和多样性的新个体.

2.3.1　GANs 的训练机制

GANs 的训练将执行 epochs 代, 每代的训练机制如图 2 所示. 首先, 传统的 GANs 用于处理图像等连续数据建模, 不能直接处理组合优化问题中的离散数据形式. 为使得 GANs 能够有效学习调度问题中离散的工件序列, 本文将精英种群中每个个体的工件序列π_{seq}编码为 one-hot 矩阵, 再将 one-hot 矩阵作为真样本输入至判别器进行训练. 该编码中, 用 0-1 矩阵描述π_{seq}. 其中: 每行对应一个工件编号, 每列表示其在加工序列中的位置, 从而保留了工件的排列顺序信息. 图 3 为一个π_{seq}的 one-hot 矩阵编码示例. 然后, 生成器接收从标准正态分布中随机采样得到的长度为N的一维随机噪声向量, 输出$N \times N$的实数概率矩阵, 这里每行表示一个工件在所有加工

位置上的概率分布. 对概率矩阵执行最大值选择操作, 得到一个对应的 one-hot 矩阵, 即确定概率矩阵每行的最大值, 将最大值位置设置为 1, 其余位置设置为 0. 将该 one-hot 矩阵作为假样本输入至判别器, 同时, 也保证了输入判别器的真样本与假样本在数据形式上的一致性. 图 3 为一个概率矩阵到 one-hot 矩阵编码的示例. 接着, 判别器对真样本和假样本进行判断, 并采用二元交叉熵损失函数计算损失值. 最后, 根据损失值反向传播以优化生成器和判别器的网络参数.

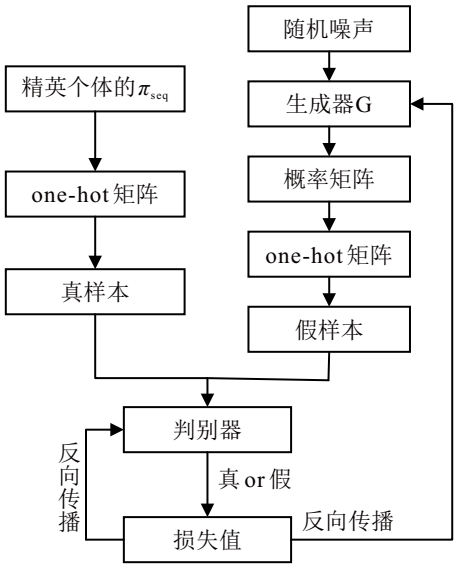


图2 GANs 的训练机制

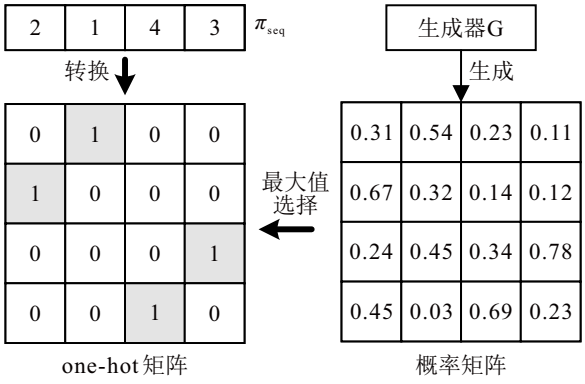


图3 GANs 的 one-hot 矩阵编码

2.3.2 基于生成器的新个体生成

在 GANs 训练完成后, 生成器能够有效学习并重构精英个体中工件序列π_{seq}所隐含的优良排列分布信息, 实现对真实数据分布的隐式建模. 基于生成器生成新种群个体的具体过程如下: 首先, 基于从正态分布中的随机采样, 得到长度为工件数量N的一维随机噪声向量, 将其输入给训练后的生成器, 生成一个$N\times N$的概率矩阵并基于第 2.3.1 节的方法转化为一个 one-hot 矩阵, 再采用图 3 所示的从π_{seq}到 one-hot 矩阵的逆操作, 生成一个工件序列π_{seq}; 然后, 随机生成一个工厂序列π_{fac}, 与π_{seq}组合构成一个新个体$\pi=\{\pi_{seq},\pi_{fac}\}$. 需要注意的是, 由于新生成的个体可能存在死锁, 需要基于第 2.2 节所提出方法进行死锁检测和修复. 新个体的生成过程如图 4 所示.

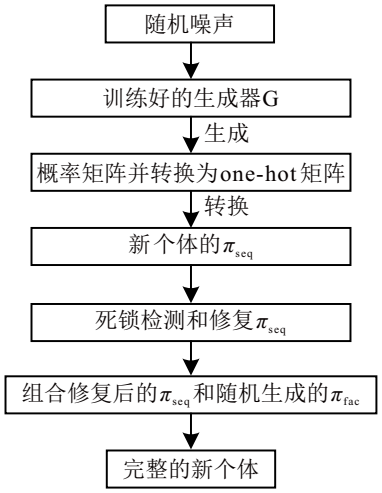


图4 基于生成器的新个体生成

2.4 局部搜索

对于 FDAFDSP, 传统的工厂间插入与交换操作, 以及工厂内部插入与交换操作同样适用, 但是这些操作具有较强的扰动性, 易破坏原有解的可行性, 加剧死锁问题. 若只是将这些操作简单应用于 FDAFDSP, 则必须进行大量的死锁检测和修复, 从而增加了计算负担. 为此, 本文提出两种专门面向 FDAFDSP 的局部搜索策略, 在不破坏解可行性的前提下, 提升局部搜索的优化性能, 从而提升算法整体性能.

所提出破坏性局部搜索策略和无死锁局部搜索策略将采取串行方式, 在每代中, 先对精英个体和新生成的子代种群使用破坏性局部搜索, 以引导算法跳出局部最优. 为进一步提升个体质量, 在不破坏解可行性的前提下, 对精英个体执行无死锁局部搜索策略, 从而实现更高效的局部寻优. 随后, 将处理过的精英个体与子代种群合并, 筛选出适应度较高的个体, 构成下一代种群.

2.4.1 破坏性局部搜索

破坏性局部搜索使用 4 种基于关键工厂的邻域算子. 在 FDAFDSP 中, 关键工厂的识别依赖于解的π_{seq}向量, 由于该向量中的最后一个工件必然是最晚完成加工的工件, 结合π_{fac}查找该工件所分配的工厂编号, 即确定该解的关键工厂. 4 个邻域算子描述如下:

N_1: 从关键工厂随机抽取一个工件, 插入至关

键工厂中任一位置, 并对新解进行死锁修复;

N_2: 从关键工厂随机选择两个工件, 互相交换位置, 并对新解进行死锁修复;

N_3: 随机选择一个非关键工厂, 从关键工厂中随机选取一个工件, 插入至非关键工厂, 并对新解进行死锁修复;

N_4: 从关键工厂和一个非关键工厂各选取一个工件, 交换其位置并对新解进行死锁修复.

变邻域搜索算子的执行过程如下.

step 1: 初始化迭代计数器$\text{gen}=0$, 算子标志变量$t=1$, 确定局部搜索次数 LS.

step 2: 当$\text{gen}<\text{LS}$ 时, 重复执行 step 3 和 step 4.

step 3: 根据当前t的取值 (1, 2, 3, 4), 依次执行对应的变邻域算子$N_1 \sim N_4$, 生成新解并进行死锁修复, 确保解的可行性.

step 4: 若新解优于当前解, 则跳出循环, 返回当前最优解; 否则, 令$\text{gen}++$, $t++$. 当$t>4$时, 令$t=1$.

2.4.2　无死锁局部搜索

无死锁局部搜索使用两种邻域算子, 具体描述如下:

N_5: 随机选择一个产品, 根据该产品装配计划的工件, 随机选择两个并在π_{seq}中交换位置;

N_6: 随机选择π_{seq}中的一个工件, 依次变更π_{fac}中该工件所属的工厂, 选择适应度值最小的工厂.

由于这两种邻域算子不会破坏解的可行性, 无需进行死锁修复操作. 无死锁局部搜索算子的执行过程与破坏性局部搜索算子完全相同, 只是使用N_5和N_6邻域算子, 具体过程不再详述.

2.5　GAN-DSA 算法流程

基于上述设计, GAN-DSA 的算法流程如图 5 所示. 首先, 随机初始化生成规模为 Ps 的初始种群, 基于个体适应度值筛选出规模为$\boldsymbol{\omega}$的精英种群, 记剩余的$\text{Ps}-\boldsymbol{\omega}$个体为保留种群; 然后, 将精英种群作为真样本数据输入给 GANs, 以执行全局搜索, 进而对产生的新个体执行破坏性局部搜索, 形成子代种群; 同时, 精英种群自身依次执行破坏性局部搜索和无死锁局部搜索, 得到改进后的精英种群; 最后, 将子代种群、改进精英种群与保留种群合并, 对合并种群进行死锁检测和修复, 基于适应度值筛选出较优的 Ps 个个体作为新一代种群. 该过程不断循环, 直至满足终止条件.

所提出算法中, 工件数量为n, 在每代迭代过程中, 个体适应度评估复杂度为$O(n)$; 变邻域搜索对个体执行$O(n)$次邻域操作, 每次评估复杂度为$O(n)$, 复杂度为$O(n^2)$; GANs 的训练复杂度在固定种群规模下为$O(n^2)$, 因此算法复杂度为$O(n^2)$.

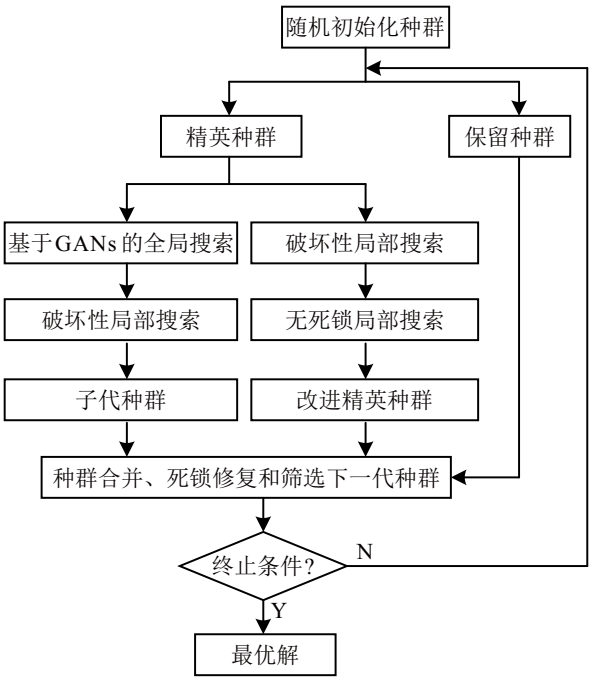


图5　GAN-DSA 流程

3　实验分析

3.1　实验设置

为验证所提出算法 GAN-DSA 的有效性, 共设计 32 个测试算例. 表 1 为算例的规模参数, 其中n、f、m和l分别对应工件、工厂、机器和产品的数量, 共有 32 种组合. 工件和产品的处理时间分别在$[1,99]$和$1.2\times[1,99]$内, 且服从均匀分布, 每个产品L_c的装配计划AP_c随机生成, 但是要确保每个产品至少有一个工件. 缓冲区B_A的容量大小在区间$[b,1.5b]$随机选定, 其中$b=\max\{|\text{AP}_c|, c\in L\}$, 即表示装配计划内工件数最多的产品所需要的工件数量.

表1　算例规模设置

工件n	工厂f	机器m	产品l	$n\times f\times m\times l$
15, 20	2, 4	6, 8	3, 4	$2\times2\times2\times2=16$
80, 100	8, 10	10, 12	8, 10	$2\times2\times2\times2=16$

为公平起见, 所有算法在每个算例下独立运行 10 次, 每次运行的终止时间为$\text{Timelimit}=n\times f\times m\times l\times C$, 其中比例因子$C=90$ ms. 实验环境中, CPU 为英特尔 i5-13400F@2.5 GHz, 内存为 16 GB, 所有算法均在 Python3.12.1 的环境下进行编程. 所提出算法与两种算法 ICA[8] 和 EABC[9] 进行比较, 对比算法的参数根据各自文献中的设置进行配置.

算法评价指标采用相对百分比误差 (RPE), 如下所示. 每个算法在每个算例下独立运行 10 次的最优 RPE (bRPE) 和平均 RPE (aRPE) 将被记录, 作为评价算法的最终指标:

$$\mathrm{RPE} = (f_{\mathrm{algorithm}} - f_{\mathrm{best}})/f_{\mathrm{best}} \times 100\%. \quad (9)$$

其中: $f_{\mathrm{algorithm}}$为当前算法对一个算例的计算结果, f_{best}为所有算法对相同算例取得的最优结果.

3.2 参数设置

所提出算法中, 参数可分成两类, 分别为 GANs 的超参数和调度算法的参数. GANs 的超参数依据已有研究经验, 并通过多组实验调优得出最优配置. 在本研究中, GANs 模型采用 Pytorch 库实现建模, 生成器和判别器均采用全连接神经网络进行建模. GANs 中的超参数包括神经网络层数、各神经层的激活函数、学习率、批次大小. 表 2 为本文中 GANs 超参数设置.

表2 GANs 参数设置

参数名称	参数值
判别器隐藏层	2
生成器隐藏层	2
判别器隐藏层激活函数	LeakyRelu
生成器隐藏层激活函数	LeakyRelu
判别器输出层激活函数	Sigmoid
生成器输出层激活函数	Tanh
学习率	0.0002
批次大小	32

在 GAN-DSA 算法中存在 4 个参数, 分别为种群规模 Ps、精英种群规模ω、局部搜索次数 LS 和 GANs 的训练次数 epochs, 采用 Taguchi 设计实验方法[1] 来确定参数值. 最优的参数组合为$\mathrm{Ps} = 50$, $\omega = 20$, $\mathrm{LS} = 350$, $\mathrm{epochs} = 400$.

3.3 与已有算法结果对比分析

由于目前没有研究模糊分布式装配流水车间无死锁调度问题的算法, 为了分析 GAN-DSA 的性能, 选取两种求解模糊分布式装配流水车间调度问题的算法作为比较对象, 分别为帝国主义竞争算法 (ICA)[10] 和增强型人工蜂群算法 (EABC)[11]. 原算法均未考虑死锁问题, 在不影响算法原有设计思想和结构的前提下, 将所提出死锁检测和修复策略嵌入至算法的解码过程, 以实现无死锁处理. 计算结果如表 3 所示.

表3 算法对比结果

算例		GAN-DSA		EABC		ICA	
		bRPE	aRPE	bRPE	aRPE	bRPE	aRPE
n	15	0.00	23.35	0.57	23.88	0.29	25.88
	20	0.00	14.31	1.94	16.51	1.81	16.08
	80	0.00	9.88	1.43	11.34	1.43	11.34
	100	0.00	8.15	0.37	9.71	0.71	9.30
f	2	0.00	21.83	1.29	23.76	0.81	25.94
	4	0.00	21.80	0.43	22.72	0.14	22.09
	8	0.00	7.70	3.57	10.03	2.25	9.15
	10	0.00	8.30	2.65	10.00	0.71	9.19
m	6	0.00	11.90	0.57	13.29	0.29	15.15
	8	0.00	9.15	1.06	10.35	0.95	10.11
	10	0.00	7.10	1.43	8.71	1.43	8.25
	12	0.00	6.48	1.77	8.27	1.77	7.86
l	3	0.00	13.82	0.57	15.43	0.29	15.21
	4	0.00	13.60	1.25	14.69	0.08	16.23
	8	0.00	7.89	1.43	10.11	1.43	9.42
	10	0.00	4.14	0.17	5.35	0.17	5.16

由表 3 可知, 在 32 个算例中: GAN-DSA 的 bRPE 始终为 0, 表明其在所有规模的测试算例中均找到了当前最优解; GAN-DSA 在 aRPE 和 bRPE 两项指标上均表现出显著优势, GAN-DSA 的平均 aRPE 为 11.84%, 相较于 EABC 的 13.38% 与 ICA 的 13.52%, 分别降低了 1.54% 和 1.68%.

在工件数n从 15 增加至 100 的过程中, GAN-DSA 的 aRPE 降幅达到 15.2%, 优于 EABC 的 14.17%, 虽然略低于 ICA 的 16.58%, 但是在$n \geqslant 80$的情形下, GAN-DSA 的误差波动控制在 1.73% 以内, 显著小于 EABC 与 ICA, 表明了 GAN-DSA 在大规模算例下具备更强的鲁棒性.

针对工厂数量f的变化, GAN-DSA 在f从 2 扩展至 10 的过程中, aRPE 下降了 13.53%, 优于 EABC 与 ICA, 同时在$f = 8$的场景下, 其 aRPE 仅为 7.70%, 显著高于 EABC 与 ICA 的 10.03%、9.15%. 工厂数量越多, 解空间越大, GAN-DSA 在高维复杂解空间中具有越好的搜索能力.

随着产品数量从 3 增加至 10, GAN-DSA 的 aRPE 由 13.82% 逐步下降至 4.14%, 降幅达 70.0%, 明显优于 EABC 与 ICA. 特别是在$l = 8$和$l = 10$的场景中, GAN-DSA 的 bRPE 依然保持为 0, 表明 GAN-DSA 在多产品复杂场景下具有高效的调度能力和避免死锁能力.

综上, GAN-DSA 在不同测试算例中表现出了显著的性能优势, 尤其是在多维复杂调度环境下, 该算法展现出了较好的优化能力, 适用于大规模、多工厂、产品多样和有限缓冲等复杂制造场景.

4 结 论

本文研究了带有模糊时间和有限装配缓冲区的分布式装配流水车间调度问题, 由于系统死锁的存

在, 导致鲜有算法能够直接处理该问题. 通过建立基于 Petri 网的死锁检测和修复算法、基于生成对抗网络的全局搜索算子、两种不引发死锁的局部搜索算子, 本文提出了一种基于生成对抗网络的无死锁求解算法 GAN-DSA. 实验结果表明, GAN-DSA 在求解所研究问题时, 性能显著优于现有对比算法. 下一步工作将重点探索 GAN-DSA 的训练机制, 以提升模型训练的稳定性和训练效果, 同时改进个体编码方式, 使得 GAN-DSA 可以同时学习工厂分配和加工序列, 增强算法对领域知识的学习能力.

参考文献 (References)

[1] Zhang G H, Liu B, Wang L, et al. Distributed co-evolutionary memetic algorithm for distributed hybrid differentiation flowshop scheduling problem[J]. IEEE Transactions on Evolutionary Computation, 2022, 26(5): 1043-1057.

[2] Zhang G H, Xing K Y, Cao F. Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan[J]. International Journal of Production Research, 2018, 56(9): 3226-3244.

[3] 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.
(Zhang Z Q, Qian B, Hu R, et al. Multidimensional estimation of distribution algorithm for low carbon scheduling of distributed assembly permutation flow-shop[J]. Control and Decision, 2022, 37(5): 1367-1377.)

[4] Hatami S, Ruiz R, Andrés-Romano C. The distributed assembly permutation flowshop scheduling problem[J]. International Journal of Production Research, 2013, 51(17): 5292-5308.

[5] Wang S Y, Wang L. An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016, 46(1): 139-149.

[6] 郎峻, 顾幸生. 多目标协同正弦优化算法求解分布式流水车间调度问题[J]. 控制理论与应用, 2024, 41(6): 1029-1037.
(Lang J, Gu X S. Multi-objective collaborative sine optimization algorithm for the distributed flow-shop scheduling[J]. Control Theory & Applications, 2024, 41(6): 1029-1037.)

[7] Cheng L, Wang L, Cai J. A regional biogeography-based optimization algorithm for the distributed assembly permutation flow-shop scheduling problem with fuzzy processing time[J]. Journal of Intelligent & Fuzzy Systems, 2024, 46(2): 3827-3841.

[8] 王静, 雷德明. 考虑批处理机的绿色模糊混合流水车间调度问题[J]. 控制与决策, 2024, 39(10): 3413-3421.
(Wang J, Lei D M. Research on energy-efficient fuzzy hybrid flow shop scheduling with batch processing machines [J]. Control and Decision, 2024, 39(10): 3413-3421.)

[9] 佘明哲, 钱斌, 胡蓉, 等. 混合交叉熵算法求解模糊分布式装配流水线低碳调度问题[J]. 控制理论与应用, 2020, 37(10): 2081-2092.
(She M Z, Qian B, Hu R, et al. Hybrid cross-entropy algorithm for fuzzy distributed assembly permutation flow-shop low-carbon scheduling problem[J]. Control Theory & Applications, 2020, 37(10): 2081-2092.)

[10] Li M, Su B, Lei D M. A novel imperialist competitive algorithm for fuzzy distributed assembly flow shop scheduling[J]. Journal of Intelligent & Fuzzy Systems, 2021, 40(3): 4545-4561.

[11] Zuo Y D, Wang P, Fan Z, et al. Minimizing fuzzy makespan in a distributed assembly flow shop by using an efficient artificial bee colony algorithm[J]. Journal of Intelligent & Fuzzy Systems, 2023, 45(4): 7025-7046.

[12] Zhang G H, Xing K Y, Zhang G Y, et al. Memetic algorithm with meta-Lamarckian learning and simplex search for distributed flexible assembly permutation flowshop scheduling problem[J]. IEEE Access, 2020, 8: 96115-96128.

[13] Shao Z S, Shao W S, Pi D C. Effective constructive heuristic and metaheuristic for the distributed assembly blocking flow-shop scheduling problem[J]. Applied Intelligence, 2020, 50(12): 4647-4669.

[14] Shao W S, Pi D C, Shao Z S. Local search methods for a distributed assembly no-idle flow shop scheduling problem[J]. IEEE Systems Journal, 2019, 13(2): 1945-1956.

[15] Zhao F Q, Xu Z S, Wang L, et al. A population-based iterated greedy algorithm for distributed assembly no-wait flow-shop scheduling problem[J]. IEEE Transactions on Industrial Informatics, 2023, 19(5): 6692-6705.

[16] Zhang G H, Wang L, Xing K Y. Dual-space co-evolutionary memetic algorithm for scheduling hybrid differentiation flowshop with limited buffer constraints[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(11): 6822-6836.

[17] 李晓玲, 冯彦翔, 张广辉, 等. 混合离散蛙跳算法求解柔性装配系统调度问题[J]. 控制理论与应用, 2025, 42(4): 816-826.
(Li X L, Feng Y X, Zhang G H, et al. Hybrid discrete shuffled frog leaping algorithm for the scheduling problem of flexible assembly systems[J]. Control Theory & Applications, 2025, 42(4): 816-826.)

[18] 康苗苗, 邢科义, 郜振鑫. 柔性制造系统的改进粒子群无死锁调度算法[J]. 控制与决策, 2014, 29(8): 1345-1353.
(Kang M M, Xing K Y, Gao Z X. Deadlock-free modified particle swarm optimization scheduling algorithm for flexible manufacturing systems[J]. Control and Decision, 2014, 29(8): 1345-1353.)

[19] Alqahtani H, Kavakli-Thorne M, Kumar G. Applications of generative adversarial networks (GANs): An updated

review[J]. Archives of Computational Methods in Engineering, 2021, 28(2): 525-552.

[20] Wang W, Wang C, Cui T, et al. Study of restrained network structures for Wasserstein generative adversarial networks (WGANs) on numeric data augmentation[J]. IEEE Access, 2020, 8: 89812-89821.

[21] Yu R R, Xu Y H, Peng H W, et al. Enhancing population diversity by integrating iterative local search with deep convolutional generative adversarial networks (GANs)[C]. Proceedings of the 26th International Conference on Pattern Recognition. Montreal, 2022: 4722-4728.

[22] Lemtenneche S, Cheriet A, Bensayah A. Introducing generative adversarial networks on estimation of distribution algorithm to solve permutation-based problems[C]. International Symposium on iNnovative Informatics of Biskra. Biskra, 2022: 1-5.

[23] Lemtenneche S, Cheriet A, Abdellah B. Permutation-based optimization using a generative adversarial network[C]. Proceedings of the Genetic and Evolutionary Computation Conference Companion. Lille, 2021: 159-160.

[24] Li J Q, Pan Q K. Chemical-reaction optimization for solving fuzzy job-shop scheduling problem with flexible maintenance activities[J]. International Journal of Production Economics, 2013, 145(1): 4-17.

作者简介

张广辉 (1981−), 男, 教授, 博士, 主要研究方向为计算智能理论与方法、分布式柔性制造系统建模、控制、调度, E-mail: ghzhang@hebau.edu.cn;

赵成龙 (1999−), 男, 硕士生, 主要研究方向为智能算法与生产调度, E-mail: zhaocl2023@hotmail.com;

魏晨轩 (2000−), 男, 硕士生, 主要研究方向为智能优化理论及算法, E-mail: chenxuanwei2023@hotmail.com;

冯彦翔 (1987−), 男, 副教授, 博士, 主要研究方向为复杂柔性制造系统建模、控制、调度、智能优化理论及应用, E-mail: fengyanxiang@xjtu.edu.cn;

李晓玲 (1991−), 女, 副教授, 博士, 主要研究方向为柔性制造系统调度、智能优化理论及应用, E-mail: xiaolingli@chd.edu.cn.