浙江理工大学学报,第 51 卷,第 6 期,2024 年 11 月
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673-3851(n).2024.06.012

基于时序 Petri 网的机器人柔性作业车间无死锁调度优化算法

陈海波[1],张超隆[1],董建明[2]
(1.浙江理工大学计算机科学与技术学院(人工智能学院),杭州 310018;
2.浙江工商大学管理工程与电子商务学院,杭州 310018)

摘　要:研究了一类源于芯片生产等智能制造领域的机器人柔性作业车间调度问题。针对该类问题在算法设计过程中,需要同时考虑工件加工、机器人运输和死锁求解等带来的问题,提出了一种基于时序 Petri 网的无死锁调度优化算法。首先,对问题进行时序 Petri 网建模,提出了一种基于 Petri 网变迁串的解表示形式,以方便死锁求解和算法寻优;其次,通过对问题死锁的结构分析和分类,提出了一种死锁判断和求解算法,并证明了算法可在多项式时间内求解任意类型死锁,同时也能在一定程度上保证解的优良结构;最后,提出了问题的一种离散蜂群算法求解方案,在算法寻优过程中进行实时死锁求解以保证解的可行性和较快的收敛速度。不同规模实例的数值实验和与问题最优解及最优解下界的比较分析表明,提出的算法对不同类型实例都体现了较好的性能和较低的时间复杂度。该研究为复杂作业车间实时调度问题的算法研究提供了新思路和方法。

关键词:作业车间调度;离散蜂群算法;单机器人;Petri 网;阻塞和死锁
中图分类号:TP301.6　　**文献标志码**:A　　**文章编号**:1673-3851 (2024)11-0839-12
引文格式:陈海波,张超隆,董建明. 基于时序 Petri 网的机器人柔性作业车间无死锁调度优化算法[J]. 浙江理工大学学报(自然科学),2024,51(6):839-850.
Reference Format: CHEN Haibo,ZHANG Chaolong,DONG Jianming. A deadlock-free scheduling algorithm of robotic flexible job shops based on temporal Petri nets[J]. Journal of Zhejiang Sci-Tech University,2024,51(6):839-850.

A deadlock-free scheduling algorithm of robotic flexible job shops based on temporal Petri nets

CHEN Haibo[1], *ZHANG Chaolong*[1], *DONG Jianming*[2]
(1. School of Computer Science and Technology (School of Artificial Intelligence), Zhejiang Sci-Tech University, Hangzhou 310018, China; 2. School of Management and E-Business, Zhejiang Gongshang University, Hangzhou 310018, China)

Abstract: A robotic flexible job shop scheduling problem originating from intelligent manufacturing fields such as chip manufacturing was studied. To solve this problem, it is necessary to consider the difficulties caused by job processing, robot transportation and deadlock solving. As a result, a deadlock-free scheduling optimization algorithm based on temporal Petri net was proposed. Firstly, the problem was modeled by temporal Petri net, and a solution representation based on Petri net transition string was proposed to facilitate deadlock solving and algorithm optimization. Secondly, through the analysis and classification of the deadlock structure, a deadlock detection and resolution algorithm was proposed, and it was proved that the algorithm could solve any type of deadlock in polynomial time, and at the same time could ensure the good structure of the solution to a certain extent. Finally, a discrete swarm algorithm was proposed to solve the problem. In the optimization process of the algorithm, deadlock was solved to ensure

收稿日期:2024-04-15　　网络出版日期:2024-07-22
基金项目:国家自然科学基金项目(11971435);浙江省"领雁"研发攻关项目(2024C01108)
作者简介:陈海波(1972—),男,杭州人,副教授,博士,主要从事算法设计与计算机语言理论方面的研究。
通信作者:董建明,E-mail:djm226@163.com

the feasibility of the solution and fast convergence speed. The numerical experiments of different scale instances and the comparison with the optimal solution and the lower bound of the optimal solution showed that the proposed algorithm had good performance and lower time complexity for different types of instances. The study provides a new idea and method for the algorithm research of complex job shop real-time scheduling problems.

Key words: job shop scheduling; discrete bee colony algorithm; single robot; Petri nets; blocking and deadlock

0 引 言

随着工业互联网及工业 4.0 的逐步深入,制造业的升级朝着无人化和智能化推进,满足“多样化、小规模、周期可控”的柔性化生产制造将成为趋势。大量无人辅助制造工具,例如自动导引机器人(Automated guided vehicle, AGV)、码垛机器人等设备的使用,大大提高了生产制造的效率,同时也给柔性生产调度提出了更高的要求。调度理论主要研究如何将有限的资源进行优化并合理安排任务的执行顺序使得设定的目标(例如时间最短、成本最低或利润最高等)达到最优,已被大量用于生产调度与排程等诸多领域[1]。柔性作业车间调度问题(Flexible job-shop scheduling psroblem, FJSP)研究复杂柔性车间生产任务的优化调度,是运筹学、调度理论和工业工程等研究的重要问题之一[2]。在传统作业车间调度问题中,工件具有若干道工序,需要在不同的机器上进行加工,且不同工件的工序数量及其在机器上的加工顺序可以不同,同一工件的不同工序不能同时加工。若同一台机器可以加工不同类型的工序,从而产生了一类更为一般的柔性作业车间调度问题。柔性车间作业调度要求机器具有多功能属性,可以用于加工不同类型的工序,所以更加符合小批量定制等新型生产模式的要求。

经典的柔性作业车间调度问题通常不考虑工件(工序)在机器间的移动,并且假设工件(工序)在机器上加工完后即被移走,或具有一个无限大的缓冲区可以存放加工完的工件。在现代智能制造及无人工厂场景下,除了需要优化生产环节,产品的运输及路径的优化调度同样至关重要[3]。特别是在一些典型柔性制造场景下,通常工件(工序)加工完后需要由无人运输工具(例如 AGV)进行运输至下一道工序所需的机器上;由于加工区域狭小,机器通常并没有缓冲空间可以存放加工完的工件,因此工件可能要滞留在被加工机器上,等待运输机器人空闲时将完工工件移出机器并将其运至后续工序机器等待可用时进行加工,这种缓冲约束被称为“阻塞”。当运输工具数量有限且调度不当时,机器的“阻塞”可能会导致“死锁”现象的发生,此时整个加工过程将会停滞。所以,在实时作业车间调度环境下,如何保证机器柔性的情况下,将工件加工和运输工具调度进行协同优化,并实现无死锁调度与算法寻优协同至关重要。

为了适应个性化定制等新型制造模式,在传统的作业车间调度问题(JSP)模型基础上,柔性作业车间调度问题得到了广泛的研究[2]。例如,将传统的每台机器只能加工一类工序,扩展成每台机器可以加工多种类型工序,或者每道工序可以选择多台平行机器中的一台进行加工,甚至考虑加工时间等参数不确定情况下的动态柔性作业车间调度问题[4-5]。典型(柔性)作业车间调度问题假设机器缓冲区是无限的,即工件在机器上加工完成后机器即可释放。事实上,在大量实际制造场景下,由于制造空间狭小等原因,通常机器缓冲区有限甚至无缓冲区,所以已有研究人员开始研究缓冲区的不同设置对算法设计的影响。例如,Liu 等[6]研究了具有 4 种缓冲区约束(即无等待、无缓冲、有限缓冲和无限缓冲约束)情况下的广义 JSP 问题,给出了问题的混合整数规划模型和多个启发式算法。Bektur 等[7]研究了工件具有到达时间且机器具有有限缓冲区情况下的不同类机调度问题,给出了问题的混合整数规划模型、禁忌搜索算法(TS)和模拟退火(SA)算法求解算法。考虑缓冲区有限情况下的模型,需要同时考虑工件的加工和缓冲区的合理使用,所以算法设计变得更加困难。

当缓冲区有限情况下,工件加工完后通常需要借助运输工具(机器人)将完工工件运出缓冲区,并同时运至下一阶段的机器,此时加工与运输协同调度效率显得异常重要。同时,当缓冲区容量为 0 时,即机器无缓冲区时,考虑阻塞和死锁情况下的调度,不仅符合现实而且是解决缓冲区有限且考虑工件运输情况下调度问题的关键[8]。无死锁调度决策在实时调度系统中较为常见。例如,Li 等[9]研究了一类机器无缓冲区的柔性装配系统(AFSSP)调度问题,工件在装配过程中会存在死锁约束。Xing 等[10]研

究了柔性装配系统(FAS)的死锁控制问题,使用 Petri 网(Petri Net)分析了 FAS 问题的活性结构,并设计了 Petri 网控制器以控制 FAS 的无死锁运行。李浚等[11]研究了机器人柔性流水车间调度问题,提出了一种时序 Petri 网监督学习的启发式算法,并且设计了时序 Petri 网的全连接神经网络模型。Luo 等[12]研究柔性制造系统(FMS)中的死锁避免策略,提出了一种类似银行家算法的死锁避免策略,并分析了所设计策略的时间复杂度。这些研究主要集中在装配线调度问题,且研究的核心集中在问题的混合整数规划模型和基于 Petri 网的死锁控制策略两个方面。然而,采用基于 Petri 网的死锁仿真往往会产生状态爆炸等现实问题,因而在实际应用中受到一定的限制。

对于考虑机器人运输的调度问题研究,现实调度系统的机器环境主要涉及车间作业环境。例如,Hurink 等[13]将传统的作业车间调度问题分为两个阶段研究,先确定工件调度,再确定工件的运输路线规划并将机器设置成无限缓冲区,同时给出了问题的一种禁忌搜索算法。Nouri 等[14]研究了一类具有运输时间和多机器人的作业车间调度问题(JSPTMR),该问题将机器设置成无限缓冲区,并给出了一种基于混合多智能体的启发式算法。晏晓凡[15]研究了针对物料机器人指派和作业车间的联合调度问题,该调度环境为无限缓冲区,并且设计了一种改进灰狼优化算法进行求解。上述问题都假设机器的缓冲区无限,此时机器人的运输与工件的加工不会形成死锁。Sun 等[16]则研究了一类机器缓冲区容量为 0 的作业车间调度问题,且问题中同时考虑机器人对工件的运输和死锁情形,并给出了该类问题的一个混合整数规划模型,对小规模问题实例进行了规划模型的求解,但未给出问题的有效算法。

本文将进一步解决文献[16]中提出的问题,即给出该类问题的无死锁调度算法。该类问题在芯片生产等领域具有广泛应用背景。例如,芯片生产分为清洗、光刻、蚀刻等若干步骤单元,其中清洗单元是其中一个至关重要的步骤。在清洗单元,晶圆需要经过浸泡去除有机物、氧化层去除、颗粒去除、金属离子去除等多道工序,且工序间具有严格的先后顺序要求,并在多个清洗站由专门的机器负责完成,如槽式清洗机、DI 水冲洗机、超声波清洗机、干燥设备等。清洗单元对环境洁净度要求极高,例如在英飞凌晶圆处理环境中,28 L 空气中颗粒度需小于 1。因此,清洗单元内部部署机器人,能在保持空气高洁净度的同时,精确地将晶圆移动至不同的清洗机器进行处理。如何优化机器人的调度和晶圆清洗流程的排序是清洗单元高效运行的关键。本文针对该类机器缓冲区容量为 0 且考虑机器人运输的柔性作业车间调度问题(Robotic flexible job-shop scheduling problem, RFJSP),给出了该问题的首个无死锁调度的高效算法。首先对问题进行时序 Petri 网建模,将调度问题的解转化成相应的 Petri 网变迁串;其次,对问题解的结构进行分析,将死锁类型进行分类,并基于 Petri 网变迁串的解结构设计了相应的高效死锁检测和求解算法,并证明算法在多项式时间内可以求解任意死锁类型;最后,设计了该问题的人工蜂群算法解决方案。在蜂群算法设计中,解结构采用 Petri 网变迁串,且当算法运行时遇到死锁导致解不可行时,则采用设计死锁判定和求解算法进行死锁的求解,使算法的寻优过程既能保证解结构的优良特性又始终在可行解中进行。

1　RFJSP 问题的形式化描述及时序 Petri 网建模

首先给出 RFJSP 问题的形式化描述:n 个工件的集合记为 $J=\{j_1,j_2,\cdots,j_i,j_n\}$,需要在 m 台机器上进行加工,机器集合记为 $M=\{M_1,M_2,M_3,\cdots,M_m\}$。对于每个工件 $j_i(i=1,2,3,\cdots,n)$,由 θ_i 个工序组成,所以每个工件也可以表示为 $j_i=\{O_{i,1},O_{i,2},\cdots,O_{\theta_i}\}$,且工序的加工顺序给定。对于每道工序 $O_{i,j}$,需要在对应机器 $M(O_{i,j})\in\{M_1,M_2,M_3,\cdots,M_m\}$ 上加工,加工时间记为 $\rho_{i,j}$。工件在机器上的加工需要满足作业车间环境(Job shop)要求,即每个工件工序的加工顺序可以不同,且同一台机器可以加工同一工件的多道工序(机器柔性),但是同一时刻同一工件的不同工序不能同时加工。

有一台移动机器人记为 R,负责工件在不同机器上的运输,且机器人每次只能运输一个工件,例如将工件 j 从当前工序 $O_{i,j}$ 所在的机器 $M(O_{i,j})$ 运输到下一道工序 $O_{i,j+1}$ 所在的机器 $M(O_{i,j+1})$。每台机器没有缓冲区,即加工完相应工序的工件需要一直占用机器(机器阻塞),直到机器人将工件移出机器。所以,机器人的移动方式有两种,即空载移动和负载移动。空载移动是指机器人在没有携带任何工件的情况下移动,而负载移动是指机器人在携带工件的情况下移动至指定机器。假定,每个工件的第 1 道工序都需要通过移动机器人 R 从初始站 S 中运出,且最后一道工序完工后由移动机器人运输至存

储站 D。机器人在相邻机器间的运输时间记为单位时间,且以跨机器数来计算运输总时间。

问题的优化目标是合理安排工件(工序)在机器上的加工顺序与机器人运输顺序协同,使得最后完工工件的完工时刻(makespan)最早。本文中对 RFJSP 问题的解进行 Petri 网建模,首先介绍了 Petri 网的基本概念,然后设计了 RFJSP 问题的时序 Petri 网模型。

1.1 Petri 网

Petri 网(下称"PN")是一个具有两种节点类型的有向二分图,分别称为"Place"(库所)和"Transition"(变迁)。其中节点通过有向弧连接,且有向弧从一个库所指向一个变迁,或从一个变迁指向一个库所。在图表示中,库所、变迁和弧分别用圆圈、方框和箭头表示[17]。所以,一个 PN 可以表示为一个三元组 $N=(P,T,F)$,其中:P 是一组有限的库所集合;T 是一组有限变迁的集合,满足 $P\neq\varnothing,T\neq\varnothing,P\cap T=\varnothing;F\subseteq(P\times T)\cup(T\times P)$ 表示有向弧的集合[1]。每条弧上可以添加权重,用来表示变迁触发的条件。弧权重为 1 的 PN 称为普通 Petri 网。对于 $x\in P\cup T,x$ 的前集用 $\bullet x=\{y\in P\cup T|(y,x)\in F\}$ 表示,同样 x 的后集可以表示为 $x\bullet=\{y\in P\cup T|(x,y)\in F\}$。给定集合 $Z=\{0,1,2,3,\cdots\}$ 以及,标 $Z_k=\{1,2,3,\cdots,k\}$ 记 ψ 是一个映射 $\psi:(P\to Z)$,给定一个库所 $p\in P$ 和一个标记 ψ,$\psi(p)$ 表示在标记 ψ 时库所 p 中的令牌(token)的数量。带有初始标记 ψ_0 的 PN 称为有标记的 PN,表示为 (N,ψ_0)。

如果存在一个变迁 $t(t\in T),\forall p\in\bullet t,\psi(p)>0$,那么变迁 t 称为使能状态(可以被触发),用 $\psi[t>$ 来表示。对于普通 PN,在标记 ψ 状态下,使能状态下的变迁 t 被触发从而产生新的标记 ψ',用 $\psi[t>\psi'$ 表示。由于是普通 PN,即弧权重为 1,每次变迁需要 1 个令牌,所以 $\psi'(p)$ 会出现如下 3 种情况。第 1 种情况:$\psi'(p)=\psi(p)-1,\forall p\in\bullet t/t$ 表示变迁 t 的前集的所有库所中的令牌(Token)的数量减少 1 个单位;第 2 种情况:$\psi'(p)=\psi(p)+1,\forall p\in t\bullet/t$,表示变迁 t 的后集的所有库所中的令牌(Token)的数量增加 1 个单位;第 3 种情况:$\psi'(p)=\psi(p),\forall p\notin t\bullet\cup\bullet t$,表示未发生变迁的 t 前后集的库所中的令牌数量不变。如果变迁串 $\lambda=t_1,t_2,t_i,\cdots,t_k$,变迁 t_i 被触发的状态时刻的标记为 $\psi_i[t_i$,当满足条件 $\psi_i[t_i>\psi'_{i+1},i\in Z_k,\psi_1=\psi$,那么就称变迁串 λ 在标识 ψ 下一定是可达的。

时序 Petri 网在普通 Petri 网的基础上引入了时间概念,以更精确地模拟和分析系统中各个事件的时间行为。允许令牌在库所中滞留一段时间以便满足变迁的条件,因此,时序 Petri 网能够更准确地反映系统中的动态行为。

1.2 RFJSP 问题的时序 PN 模型

本文使用时序 PN 对 RFJSP 问题进行建模。首先需要将 RFJSP 问题的元素映射为 PN 的基本元素,即将库所、变迁和弧与 RFJSP 问题中的元素一一对应,并且给定一个 RFJSP 问题的实例,则可以产生一个时序 Petri 网模型与之对应:

a)对于给定问题实例,每道工序对应 2 个状态库所,分别表示工序正在被机器人运输至指定机器状态和工序正在指定机器上进行加工的状态,且这两个库所都带时间窗口,时间窗口包括运输时间和加工时间。

b)每道工序对应 2 个变迁,分别表示工件(工序)开始运输和工件(工序)开始加工。

c)对于每个工件的第 1 道工序和最后一道工序,分别设置 1 个表示开始状态的初始库所和 1 个表示终止状态的存储库所。

d)对于每个工件的最后一道工序,设置 2 个虚拟变迁,分别表示最后一道工序完工后开始运往存储站和最后一道工序运至存储站并释放机器人。

e)整个模型中有 1 个机器人资源库所和 m 个机器资源库所。

f)每个工件的初始库所、机器人资源库所和 m 个机器资源库所初始各有 1 个令牌。所以,对于实例中的每个工件 $j_i,(j_i\in J)$,可以将其每道工序在 PN 模型中的库所和变迁按照规定的加工顺序组成如下"库所-变迁"序列 ξ_i:

$$\xi_i=p_{i,\mathrm{S}},t_{i,1},p_{i,1},t_{i,2},p_{i,2},t_{t,3},\cdots,t_{i,l},p_{i,l},\cdots,t_{i,L},p_{i,L},t_{i,(L+1)},p_{i,\mathrm{D}} \tag{1}$$

其中:$p_{i,\mathrm{S}}$ 表示工件 j_i 的初始库所,$t_{i,1}$ 表示工件 j_i 的第 1 道工序开始运输变迁,$p_{i,1}$ 表示第 1 道工序正在运输状态库所,$t_{i,2}$ 表示第 1 道工序开始加工变迁,$p_{i,2}$ 表示第 1 道工序正在指定机器上加工状态库所。依此类推。$t_{i,L}$ 表示最后一道工序开始运输变迁,$p_{i,L}$ 表示最后一道工序正在运输状态库所,$t_{i,(L+1)}$ 表示最后一道工序运至存储站变迁,$p_{i,\mathrm{D}}$ 表示工件 j_i 完工且已到达存储站状态库所。

表 1 给出了 RFJSP 问题的一个具体建模实例,其中包含 3 台机器 M_1,M_2,M_3,1 台移动机器人 R,4 个工件 j_1,j_2,j_3,j_4,每个工件都有 5 道工序。

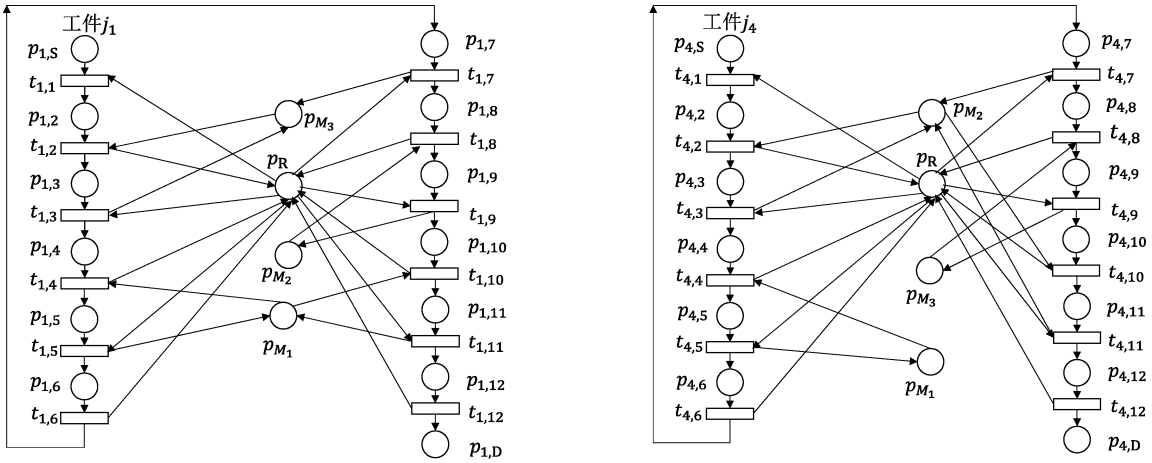
表 1　RFJSP 实例的输入参数

输入参数	j_1	j_2	j_3	j_4
工序	$O_{1,1},O_{1,2},O_{1,3},O_{1,4},O_{1,5}$	$O_{2,1},O_{2,2},O_{2,3},O_{2,4},O_{2,5}$	$O_{3,1},O_{3,2},O_{3,3},O_{3,4},O_{3,5}$	$O_{4,1},O_{4,2},O_{4,3},O_{4,4},O_{4,5}$
工序指定加工机器	M_3,M_1,M_3,M_2,M_1	M_3,M_1,M_3,M_1,M_2	M_3,M_2,M_3,M_1,M_3	M_2,M_1,M_2,M_3,M_2
加工时间	5,9,8,8,7	4,4,6,4,7	10,10,5,4,4	5,5,2,6,2

以工件 j_1 为例,由式(1)得到其“库所-变迁”序列如下:

$$\xi_1=p_{1,S},t_{1,1},p_{1,1},t_{1,2},p_{1,2},t_{1,3},\cdots,t_{1,l},p_{1,l},\cdots,t_{1,11},p_{1,11},t_{1,12},p_{1,D}。$$

其具体的 Petri 网示意图如图 1 所示。



(a) 工件j_1的PN模型　　(b) 工件j_4的PN模型

图 1　工件的 PN 模型示意图

同 j_1 和 j_4,工件 j_2 和 j_3 同样可以得到相应的“库所-变迁”序列和相应的 PN 模型表示。同样以工件 j_1 的第 1 道工序为例,其各库所状态变化和变迁触发规则可描述如下:工件 j_1 初始化在 S 站中,经过变迁 $t_{1,1}$ 请求到机器人 R(如果 R 库所存在令牌),当变迁 $t_{1,1}$ 触发之后令牌转移至 $p_{1,2}$ 状态库所,且工件 j_1 在库所 $p_{1,2}$ 状态中停留一定时间(机器人移动到机器所需时间);当机器 M_3 可用时(机器 M_3 库所存在令牌),则变迁 $t_{1,2}$ 触发并转移至库所 $p_{1,3}$ 进入在机器 M_3 上加工状态,同时归还令牌给机器人 R 库所。令牌在库所 $p_{1,3}$ 需停留一定时间(至少是第 1 道工序的加工时间)来完成工序。工件 j_1 的后续工序可按照上述规则执行。工件 j_2、j_3、j_4 同 j_1。

1.3　解的表示形式

本文根据 RFJSP 问题的时序 PN 模型表示定义 RFJSP 问题解的形式。确定 RFJSP 问题解主要有两个方面:一是确定工件(工序)之间的加工顺序,二是确定机器人运输工件的顺序(运输路径)。但是,在单机器人和机器零缓冲区情况下,任意可行解中工件的加工顺序与机器的运输顺序是一致的。所以,考虑使用 PN 模型中的变迁串来表示 RFJSP 问题的解(可行或不可行)。例如,针对表 1 实例,给定工序的加工顺序的局部为($O_{1,1},O_{4,1},O_{1,2},O_{1,3},O_{4,2},\cdots$),可以构造如下变迁串 λ 来表示问题的解:

$$\lambda=t_{1,1},t_{1,2},t_{4,1},t_{4,2},t_{1,3},t_{1,4},t_{1,5},t_{1,6},t_{4,3},t_{4,4},\cdots。$$

上述变迁编号如图 1 所示,并依此规则可以将完整解进行表示。为了保证解的基本可行性,即满足每个工件工序的加工顺序必须满足指定的顺序,在构造变迁串时,同一工件工序的变迁串标号保持递增顺序以保证解满足工序先后顺序的硬约束。同时,可以根据变迁串对于初始标记是否可达来判断当前变迁串所对应的解是否可行。例如,若初始标记 $\psi=p_{1,S}+p_{2,S}+p_{3,S}+p_{4,S}+p_{M_1}+p_{M_2}+p_{M_3}+p_R$,则上述变迁串 λ 所表示的局部解是可行的。

同时,对于任一工序都包含两个变迁,即工件开始运输变迁和工序开始加工变迁(对于每个工件的最后一道工序另外分别增加到虚拟存储站的两个变迁)。不难发现,当单机器人情况下,工序的运输变迁和加工变迁在变迁串中必须相邻且都先后被触发;否则,该变迁串所表示的解不可行且一定会发生

死锁(关于死锁的描述见 2.1),因为运输变迁触发后若不能及时触发加工变迁,则机器人无法释放且产生死锁。所以,本文将解的变迁串表示成运输变迁和加工变迁成对出现的形式,以避免死锁发生。例如上述变迁串 λ 就是成对出现,所以可以进一步表示成如下形式:

$$\lambda=(t_{1,1},t_{1,2}),(t_{4,1},t_{4,2}),(t_{1,3},t_{1,4}),(t_{1,5},t_{1,6}),(t_{4,3},t_{4,4}),\cdots。$$

2 死锁的判定与求解

2.1 阻塞与死锁

在 RFJSP 问题中,由于机器上没有缓冲区,所以当工件加工完成后无法及时将工件运离机器,则机器将会出现阻塞。若机器阻塞导致后续工序无法继续被加工则产生死锁。下面根据不同的死锁结构和求解死锁的不同方案将 RFJSP 问题可能存在的死锁分成两类如下。

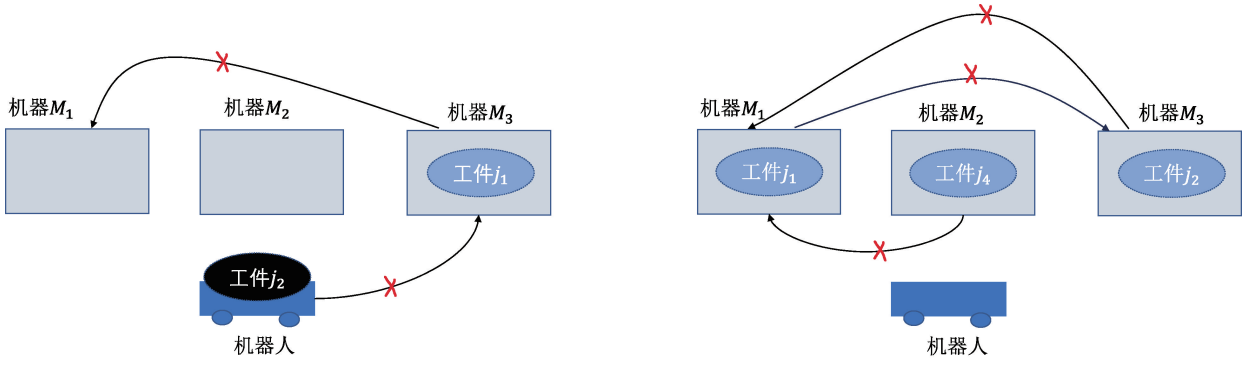
第 1 类死锁:以表 1 为例,假设工序加工顺序的局部为$(O_{1,1},O_{2,1},O_{1,2},\cdots)$,初始标记 $\psi_0=p_{1,\mathrm{S}}+p_{2,\mathrm{S}}+p_{3,\mathrm{S}}+p_{M_1}+p_{M_2}+p_{M_3}+p_{\mathrm{R}}$,变迁串 $\lambda_1=t_{1,1},t_{1,2},t_{2,1},t_{2,2},\cdots$可以表示该加工顺序所表示的解。对于 λ_1 的子串$(t_{1,1},t_{1,2},t_{2,1})$对于标记 ψ_0 是可达的,即 $\psi_0[\lambda_1>\psi_1,\psi_1=p_{1,3}+p_{2,2}+p_{3,\mathrm{S}}+p_{4,\mathrm{S}}+p_{M_1}+p_{M_2}$ 而对于 λ_1 的子串$(t_{1,1},t_{1,2},t_{2,1},t_{2,2})$,对于标记 ψ_0 不可达,所以该解不可行且在变迁 $t_{2,2}$ 处发生死锁。

因为工序 $O_{1,1}$ 已经在机器 M_3 上加工完成,此时等待机器人将其卸载,并运送至下一台机器 M_1 上继续加工第 2 道工序 $O_{1,2}$。但是此时,工件 j_2 的工序 $O_{2,1}$ 需要紧跟 $O_{1,1}$ 之后加工并已经请求到机器人(负载状态)。这导致了工件 j_1 阻塞在机器 M_3 上(等待机器人将其运走),工件 j_2 占用机器人等待机器 M_3 空闲,从而导致死锁发生,如图 2(a)所示。该死锁发生的原因是 $O_{1,1}$ 和 $O_{2,1}$ 需要使用同一台机器且工序 $O_{2,1}$ 的加工顺序不应排在 $O_{1,1}$ 之后,同时可以简单通过将工序 $O_{2,1}$ 后移至后面相应位置进行加工即可求解死锁。本文记符合该特征类别的死锁为“死锁 I”。

第 2 类死锁:以表 1 实例为例,假设工序加工顺序的局部为$(O_{1,1},O_{1,2},O_{2,1},O_{4,1},\cdots)$,初始标记 $\psi_0=p_{1,\mathrm{S}}+p_{2,\mathrm{S}}+p_{3,\mathrm{S}}+p_{4,\mathrm{S}}+p_{M_1}+p_{M_2}+p_{M_3}+p_{\mathrm{R}}$。

变迁串 $\lambda_1=t_{1,1},t_{1,2},t_{1,3},t_{1,4},t_{2,1},t_{2,2},t_{4,1},t_{4,2},t_{2,3},t_{2,4},\cdots$可以表示该加工顺序所表示的解。对于 λ_1 的子串$(t_{1,1},t_{1,2},t_{1,3},t_{1,4},t_{2,1},t_{2,2},t_{4,1},t_{4,2},t_{2,3})$对于标记 ψ_0 是可达的,即 $\psi_0[\lambda_1>\psi_1$,且 $\psi_1=p_{1,5}+p_{2,4}+p_{4,3}+p_{M_3}$。而对于 λ_1 的子串$(t_{1,1},t_{1,2},t_{1,3},t_{1,4},t_{2,1},t_{2,2},t_{4,1},t_{4,2},t_{2,3},t_{2,4})$则对于标记 ψ_0 不可达,所以该解不可行且在变迁 $t_{2,4}$ 处发生死锁,如图 2(b)所示。

因为,当变迁 $t_{1,1},t_{1,2},t_{1,3},t_{1,4},t_{2,1},t_{2,2},t_{4,1},t_{4,2},t_{2,3}$ 被触发后,此时机器 M_1 和 M_3 有工件在加工或被阻塞,M_2 空闲且工序 $O_{2,2}$ 对应工件 j_2 已申请到机器人的运输。此时,由于工序 $O_{2,2}$ 需要 M_1,而 M_1 被阻塞,所以此时表示 $O_{2,2}$ 开始加工的变迁 $t_{2,4}$ 无法被触发,从而产生死锁。事实上,此时机器人去装载任意一个工件都将会造成死锁,且该类死锁无法通过简单移动 $t_{2,4}$ 及其后面的变迁位置来求解,需要调整 $O_{2,2}$ 之前已经加工的工序的顺序。记该类特征的死锁类型为“死锁 Ⅱ”。



(a) 死锁I (b) 死锁 II

图 2 死锁类型示意图

2.2 死锁类型的判定及求解算法

给定 RFJSP 问题解的一个变迁串,需要首先判断是否有死锁且分析死锁类型,然后求解死锁。假设给定问题实例解的一个变迁串表示形式为 $\lambda=(t_{1,1},t_{1,2}),(t_{4,1},t_{4,2}),(t_{1,3},t_{1,4}),(t_{1,5},t_{1,6}),(t_{4,3},t_{4,4}),\cdots$。由于每道工序的运输变迁和加工变迁成

对出现,为了算法描述方便,将解的形式简化表示成 $\lambda=\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_{\theta+n-1},\alpha_{\theta+n}$,其中 α_i 表示第 i 对变迁串,即表示第 i 顺序被加工的工序,n 和 θ 分别为工件总数和工序总数。本文设计死锁判定和求解算法如下,记为算法 DF(λ)。

算法　DF(λ)

输入:RFJSP 问题实例解的变迁串表示 λ,给定初始标记 ψ_0。

Step1. 对变迁串 λ 从左到右进行遍历,依据初始标记 ψ_0,判断当前变迁串是否可被触发。

Step2. 若当前变迁对 α_i 无法被触发,此时根据如下规则判断死锁类型及求解当前死锁。

Step2.1. 若从变迁对 α_{i+1} 开始,依次往后遍历,当遍历到的变迁对记为 α_l 时,将 α_l 插入到 α_i 之前,即第 i 个变迁串位置时,可使 α_l 被触发,则将其插入并保持 α_i 及之后的变迁对的相对顺序不变,并对变迁对下标重新编号。继续从 α_{i+1} 位置开始遍历,进行 Step2 操作寻找下一个死锁变迁对。α_i 处发生了死锁Ⅰ。

Step2.2. 若从变迁对 α_{i+1} 开始,依次往后遍历,不存在某个变迁对 α_l,使得将其插入至第 i 个变迁串位置时可被触发,则说明当前 α_i 处发生了死锁Ⅱ,继续按照如下规则解锁:当该情况发生时,说明当前从 α_i 开始之后的变迁串所对应工序所需的机器都处于阻塞状态,此时记这些被阻塞的机器中,最早被阻塞的机器所对应的被触发变迁对为 α_k,其对应的工序为 $O_{s,t}$。

Step2.2.1. 若 $O_{s,t}$ 是工件 j_s 的最后一道工序,则将完工工件返回存储站 D 的变迁对插入到 α_k 之后,使得机器 $M(O_{s,t})$ 被释放,α_k 之后的其他变迁对相对位置不变,并对变迁对重新编号。从变迁对 α_{k+1} 开始回到 Step2 继续往后遍历。

Step2.2.2. 若 $O_{s,t}$ 不是工件 $O_{s,t+1}$ 的最后一道工序,则将工序 $O_{s,t+1}$ 所对应的变迁对插入到 α_k 之后,使得机器 $M(O_{s,t})$ 被释放,α_k 之后的其他变迁对相对位置不变,并对变迁对重新编号。从变迁对 α_{k+1} 开始回到 Step2 继续往后遍历。

Step3. 若遍历到最后一个变迁对 $\alpha_{\theta+n}$,则遍历结束,此时得到的变迁串即为无死锁变迁串,并将其输出,算法结束。

引理 1　对于 RFJSP 问题任意解的变迁串 λ,在初始标记 ψ_0 下,若 λ 中存在死锁,算法 DF(λ)一定可以在多项式时间内求解死锁,并将 λ 转化成无死锁的可行解。

证明　算法 DF(λ)的基本思想是,从左到右遍历变迁串 λ,若当前遍历位置的变迁串发生死锁Ⅰ,则将当前位置之后的某个变迁对插入到当前位置之前并被触发,然后继续往后遍历。显然,若遍历过程中,每次遇到的死锁都是死锁Ⅰ,即每次都能在当前死锁变迁串位置之后找到变迁对插入到当前位置之前并被触发,当遍历结束即可获得一个无死锁的可行解且遍历过程可在多项式时间内完成。

若当遍历过程中,当前位置的变迁串无法被触发,且将当前位置之后的任意一个变迁对插入到当前位置之前都无法被触发,即此时发生死锁Ⅱ;按照算法 DF(λ),此时将需要在当前位置之前的某个位置之后插入一个变迁对,即 Step2.2 中的操作。为了证明 Step2.2 可求解死锁Ⅱ,且可在多项式时间内完成,只需证明 Step2.2 可行,且每次求解死锁Ⅱ都可以使得无死锁变迁对增加即可。

首先,证明 Step2.2 的可行性。当死锁Ⅱ发生时,当前变迁串 α_i 及之后所有变迁串所对应的工序所需的机器都被阻塞,记这些机器的集合为 M_z,即当前变迁串之前变迁串所对应的工序在这些机器上加工,并且之后并没有同一工件的工序被加工而使这些机器被释放。显然按照 Step2.2,在 M_z 中找到一台最早被阻塞的机器(记为 M^k)和相应被触发的变迁串(记为 α_k),若在 M^k 上被触发变迁串 M^k 所对应工序是对应工件的最后一道工序,则可将对应工件的返回存储站 D 变迁对插入到该变迁串之后即可,此时插入变迁串一定可被触发且同时释放机器 M^k,即 Step2.2.1 可行。若变迁串 α_k 所对应工序不是对应工件的最后一道工序,则必然可以找到其下一道工序所对应的变迁串(记为 α_x),并将其插入到该变迁串之后即可,此时 α_x 一定可被触发且同时释放机器 M^k。因为 α_x 在原位置一定未被触发,即 $i\leqslant x$,且 M^k 是 M_z 中最早被阻塞的机器且 α_x 所需机器(记为 M^c)也在 M_z 中且在 α_k 前未被阻塞,所以此时 α_x 必可被触发,即 Step2.2.2 可行。所以 Step2.2 可行。

其次,当每执行一次 Step2.2,插入的变迁串 α_x 及之前变迁串都可被触发。若继续遍历过程中遇到死锁Ⅰ,则通过 Step2.1 使得可被触发的变迁串增加。若继续遇到死锁Ⅱ,则假设原变迁串中机器 M^c 在变迁对 α_c 处被阻塞,如图 3 所示。显然,除了 α_c 以外,原串变迁中不可被触发的变迁依然在变迁对 α_i 之后,且被阻塞的第 1 台机器所在的位置至少在 α_x 的新位置,即第 $k+1$ 变迁对位置。所以继续

执行 Step2.2,新的插入位置至少会在变迁串 α_{k+1} 之后,即 α_x 的新位置之后,所以每执行一次 Step2.2,无死锁变迁对的长度至少增加 1。显然,Step2.2 执行的次数是多项式时间的,所以综上可得算法可在多项式时间内完成。

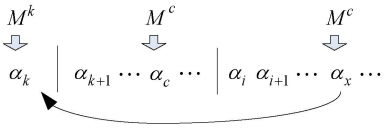


图 3 算法 DF(λ) Step2.2 执行示意图

综上引理 1 得证。

2.3 目标函数计算

本文考虑 RFJSP 问题的目标函数是极小化最后完工工件的完工时间,也就是需要首先计算最后一个变迁串所对应的工序的开工时间。按照 1.3 中对变迁串的简化定义,假设当前解的变迁串为 $\lambda=\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_{\theta+n-1},\alpha_{\theta+n}$,其中:$\alpha_{\theta+n-1}$ 为最后加工的工序所对应的运输变迁和加工变迁对,$\alpha_{\theta+n}$ 则表示最后加工的工序所对应的工件运输至存储站所对应的变迁对。

假设变迁对 α_k 所对应的工序为 $O_{i,j}$,并令 $f(O_{i,j})$表示 α_k 中加工变迁的触发时间,即工序 $O_{i,j}$ 在机器上的开工时间。令 $O_{i,j}$ 为 α_{k-1} 所对应的工序,α_k'为工序 $O_{i,j-1}$ 所对应的变迁对,则由变迁触发的顺序规则和同一工件工序加工的顺序约束,可得 $f(O_{i',j'})\leqslant f(O_{i,j})$,$f(O_{i,j-1})+\rho_{i,j-1}\leqslant f(O_{i,j})$。

所以 $O_{i,j}$ 的开工时间 $f(O_{i,j})$可计算如下:

$$f(O_{i,j})=\begin{cases}\max\{f(O_{i,j})+rt_3+rt_4,rt_1+rt_2\},\\ \quad j=1;\\ \max\{f(O_{i,j})+rt_3+rt_4,f(O_{i,j-1})+\rho_{i,j-1}+rt_4\},\\ \quad j>1\end{cases}\tag{2}$$

其中:rt_1 表示机器人从机器 $M(O_{i',j'})$到初始站 S 的运输时间,rt_2 表示机器人从初始站 S 到机器 $M(O_{i,j})$ 的运输时间,rt_3 表示机器人从机器 $M(O_{i',j'})$到 $M(O_{i,j-1})$的运输时间,rt_4 表示机器人从机器 $M(O_{i,j-1})$到 $M(O_{i,j})$的运输时间。当 $O_{i,j}$ 是解中第 1 个被加工的工序时,$rt_1=0$,即机器人初始位置在初始站 S。

根据式(2),对解的变迁串从左到右进行遍历并计算每个工序的开工时间,则计算得到最后一个开工工序的开工时间时,即可得到解的目标函数值。

2.4 问题最优解的下界估计

记 RFJSP 问题的最优目标函数值为 $C^*_{\max}$,下面给出最优解的一个下界估计如下。

引理 2 问题 RFJSP 的最优目标函数值满足 $LB_{\text{RFJSP}}\leqslant C^*_{\max}$,且

$$LB_{\text{RFJSP}}=\max\{\max_{1\leqslant k\leqslant m}(\sum_{i=1}^{n}\mu_{ki})+\min\_rt_{\text{S}}+\min\_rt_{\text{D}},\max_{1\leqslant i\leqslant n}(\sum_{k=1}^{m}\mu_{ki}+\sum_{j=1}^{\theta_i}rt_{ij})\}\tag{3}$$

其中:n 表示工件总数,m 表示机器总数,μ_{ki} 表示工件 j_i 在机器 M_k 上加工时间的总和,$\min\_rt_{\text{S}}$ 表示从初始站到最近机器的运输时间,$\min\_rt_{\text{D}}$ 表示从存储站到最近机器的运输时间,$\sum_{j=1}^{\theta_i}rt_{ij}$ 则表示工件 j_i 的所有相邻工序之间的运输时间之和。

证明 因为表达式 $\max_{1\leqslant k\leqslant m}(\sum_{i=1}^{n}\mu_{ki})$ 表示每台机器上需要加工的工序总时间的最大值,显然在最优解中也至少需要在机器上加工这些时间。每台机器上的第 1 道工序和最后一道工序需要分别从来自初始站和运输至存储,需要至少 $\min\_rt_{\text{S}}+\min\_rt_{\text{D}}$ 时间。所以最优目标函数值 $\max_{1\leqslant k\leqslant m}(\sum_{i=1}^{n}\mu_{ki})+\min\_rt_{\text{S}}+\min\_rt_{\text{D}}\leqslant C^*_{\max}$。

由于问题 RFJSP 的机器环境是作业车间环境,即同一工件工序的加工不能重叠,且相邻工序的加工之间需要通过机器人运输,所以对于每个工件 j_i,其所有工件的加工时间之和为 $\sum_{k=1}^{m}\mu_{ki}$,其所有涉及的运输时间为 $\sum_{j=1}^{\theta_i}rt_{ij}$,在最优解中,至少需要 $\sum_{k=1}^{m}\mu_{ki}+\sum_{j=1}^{\theta_i}rt_{ij}$ 时间才能将工件加工完并运送至存储站。所以最优目标函数值 $\max_{1\leqslant i\leqslant n}(\sum_{k=1}^{m}\mu_{ki}+\sum_{j=1}^{\theta_i}rt_{ij})\leqslant C^*_{\max}$。

综上引理 2 得证。

3 离散人工蜂群算法设计

本文在对问题 RFJSP 进行时序 PN 建模后,将运用离散人工蜂群算法(Discrete artificial bee colony, DABC)[18]对问题进行求解。在求解过程中,将设计算法对产生的每个解进行死锁判定,并对产生死锁的不可行解进行求解,以使得解变成无死

锁的可行解。离散人工蜂群算法是一种用于在离散参数空间中进行全局寻优的群体智能优化算法,其灵感来自工蜂的觅食行为。DABC 算法中用食物源的位置表示解,且 DABC 算法中存在 3 种类型的个体:雇佣蜂、观察蜂和侦察蜂。雇佣蜂在其邻域内寻找更好的解。基于雇佣蜂的状态,观察蜂选择若干解并进行进一步搜索,以提高解群体的质量。侦察蜂为陷入局部最优解的个体生成新的解。DABC 算法中有 3 个控制参数,即蜜源的数量 SN(初始解的数量)、迭代次数、侦察峰出动的标准。本文设计的基于 RFJSP 问题的 DABC 算法如下。

3.1　种群初始化

种群初始化需要设定一些参数,主要包括算法迭代次数,解空间(蜜源(SN))的数量,雇佣蜂、观察蜂、侦察蜂的数量,以及解被搜索阈值($Limit$)。在 DABC 算法中,每一个解都是一组由 PN 生成的变迁序。在 RFJSP 问题中,一个解就是一个编码。例如假设有两个工件分别为 j_i 和 j_j,j_i 和 j_j 均只有两道工序,那么根据 1.3 中形式化描述,j_i 在 Petri 网的变迁串为

$$\lambda_1=(t_{i,1},t_{i,2}),(t_{i,3},t_{i,4}),(t_{i,5},t_{i,6})。$$

同样,工件 j_j 在 Petri 网的变迁串为

$$\lambda_j=(t_{j,1},t_{j,2}),(t_{j,3},t_{j,4}),(t_{j,5},t_{j,6})。$$

那么该编码方式为 $\omega=\lambda_i\cup_+\lambda_j$,其中使用符号“$\cup_+$”表示集合的随机交错合并,$\lambda_i$ 中的变迁对与 λ_j 中的变迁对随机合并,但是属于 λ_i 中的变迁对必须保持在 λ_i 中的相对顺序。例如

$$\omega_1=(t_{i,1},t_{i,2}),(t_{j,1},t_{j,2}),(t_{i,3},t_{i,4}),(t_{j,3},t_{j,4}),(t_{i,5},t_{i,6}),(t_{j,5},t_{j,6}),$$

其中:ω_1 表示一个解(蜜源)。采用随机集合交错合并方式可以一定程度保证解空间的多样性。

3.2　雇佣蜂阶段

雇佣蜂通过局部搜索策略在当前蜜源附近探索新蜜源,以搜索更好的解。根据邻近蜜源的优先级雇佣蜂将旧蜜源更新为新蜜源,具体操作如下:

a)在 ω 中任意交叉两个变迁对 ∂_i,∂_j,其他元素相对位置不发生变化得到 ω_{new}。

b)ω_{new} 在经过交叉操作之后可能存在死锁情况,调用 DF 算法将 ω_{new} 修正为 ω'_{new}。

c)求解 ω'_{new} 的目标函数,将新解 ω'_{new} 与旧解 ω 比较,如果新解更优,则 ω'_{new} 取代 ω 的位置,否则保持不变。

3.3　观察蜂阶段

雇佣蜂将蜜源信息与观察蜂共享之后,观察蜂将对蜜源区域进行搜索,观察蜂的出动有助于提高搜索精度和全局收敛性,使用可变邻近搜索的搜索方法,在解 ω'_{new} 基础上进行交换和随机插入。

3.4　侦察蜂阶段

随着算法迭代次数增加,蜜源发生变化的概率逐渐减小。为了避免陷入局部最优,必须对蜜蜂所携带的蜜源限定次数,同时提高种群的多样性。如果蜜源在限定次数内未发生任何变化,那么该蜜源的携带者就会转化为侦察蜂,蜜源会被重新初始化。

4　数值实验及结果分析

本文针对不同问题规模和输入参数的问题实例进行大规模数值实验,以验证算法在不同情况下的效果和时间复杂度。数值实验在配备了具有 3.2 GiHz 频率中央处理器(CPU)的个人 PC 机上进行,且该机配置了 8 GiB RAM。算法实现采用 Python 语言编程,并运用了 Python 的 threading 模块多线程技术进行并行处理。由于该问题现有文献没有给出基准实例且尚无算法结果,因此本文采用相关问题[16]类似方法,随机生成不同类型的实例以测试 DABC 算法的性能。生成 8 种不同类型的实例如表 2 所示,其中:第 1 列表示类型索引;第 2 列用一个三元组(n,θ,m)表示该类型实例的规模,分别表示工件的数量、工件工序数量的上界和机器数量。同时,工序加工时间也分两种类型,分别取[1,10]与[10,100]之间的随机整数。机器人在任意相邻两台机器之间移动的时间是 1,且移动时间依据相隔的机器数递增。针对每种类型,本文随机构建 10 个不同的实例,其中每个实例的工序数量取$(1,\theta)$之间的随机数。上述 8 种不同类型分别针对机器数量、工件数量、工序数量和加工时间大小这 4 个体现问题规模的参数出发,进行例举和组合,从而分析算法在不同规模特征实例下的效果。

表 2　8 种类型的实例规模参数

实例	规模	实例	规模
EX1	(6,6,6)	EX5	(15,5,5)
EX2	(6,12,6)	EX6	(15,10,6)
EX3	(10,10,5)	EX7	(30,5,5)
EX4	(10,15,5)	EX8	(30,20,6)

同时,本文 DABC 算法的相关参数设置如下:迭代次数 $N_{\text{iter}}=100$,种群数目 $SN=50$,$limit$ 数目 $=10$。为了防止执行算法出现栈溢出,设置深度优先搜索树层数最大为 20,CPU 限制时间为 3600 s。为了验证算法的性能,比较对象是 RFJSP 问题最优

解的下界。通过算法解 C^{DABC} 与下界 LB_{RFJSP} 之间的相对误差 $Gap=\dfrac{C^{DABC}-LB_{RFJSP}}{LB_{RFJSP}}\times 100\%$ 来表征算法 DABC 的算法解相对问题最优解的性能。分别用 Max_Gap、Min_Gap、$\overline{Gap}$、Av_time 分别表示在同一类实例中算法解目标函数的最大相对误差、最小相对误差、平均相对误差、平均运行时间。算法求解不同类型实例的结果如表 3—表 5 和图 4 所示。

表 3 实例 EX1 $\rho\in[1,10]$的实验结果

具体实例	LB_{RFJSP}	最优解	算法解 C^{DABC}	Gap_1/%	Gap_2/%
EX1_1	59	78	78	0	32.2
EX1_2	65	88	88	0	35.3
EX1_3	95	126	130	3.1	36.8
EX1_4	94	128	134	4.6	36.1
EX1_5	105	138	145	5.1	38.1
EX1_6	123	166	174	4.8	35.0
EX1_7	115	158	175	10.7	52.1
EX1_8	147	193	204	5.6	38.7
EX1_9	182	244	252	3.2	38.4
EX1_10	177	237	256	8.0	44.6

表 4 实例 EX1—EX8 $\rho\in[1,10]$的实验结果

实例类型	Max_Gap/%	Min_Gap/%	$\overline{Gap}$/%	Av_time/s
EX1	44.6	32.2	38.7	3.5
EX2	49.1	43.5	47.8	8.6
EX3	48.2	45.6	46.9	76.9
EX4	53.5	47.3	51.7	291.0
EX5	57.3	51.6	55.9	144.0
EX6	50.1	48.9	49.8	284.0
EX7	58.9	55.3	57.4	1189.0
EX8	65.4	60.7	63.7	2867.0

表 5 实例 EX1—EX8 $\rho\in[10,100]$的实验结果

实例类型	Max_Gap/%	Min_Gap/%	$\overline{Gap}$/%	Av_time/s
EX1	40.2	35.8	38.9	4.7
EX2	46.3	41.6	44.1	11.4
EX3	51.2	44.8	47.4	90.2
EX4	55.8	50.5	53.6	407.0
EX5	56.7	51.4	54.2	352.0
EX6	54.6	49.7	52.9	1144.0
EX7	61.2	55.1	58.3	1883.0
EX8	66.3	59.9	62.4	2968.0

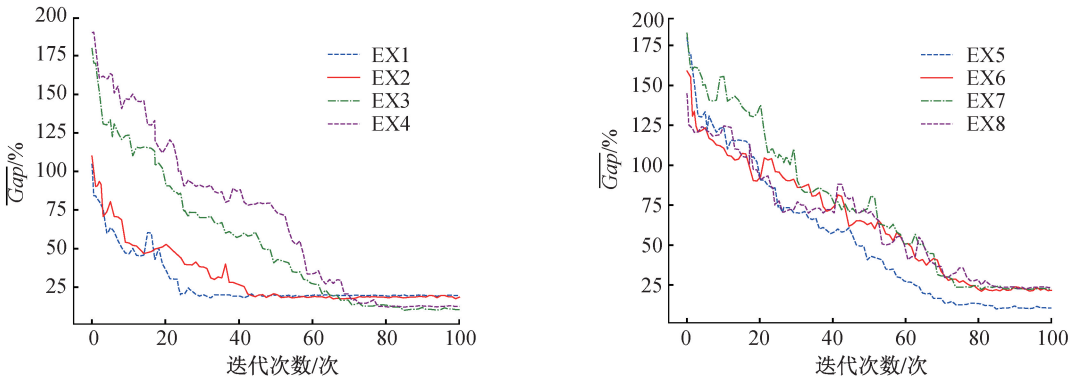
首先,对于小规模实例类型 EX1,本文对随机产生的 10 个实例,首先通过运用 CPLEX 求解器对问题 RFJSP 的整数规划模型[16]进行求解,得到这些实例的最优解。同时,计算这些实例的最优解下界。分别计算本文算法解与最优解之间的相对误差 Gap_1 和与下界之间的相对误差 Gap_2,具体结果如表 3 所示。从表 3 可知,对于这些实例,每个实例运行 10 次后取平均后的算法解呈现非常好的性能。相对于最优解而言,平均相对误差都小于 10%。事实上,对于这些实例,在每个实例的 10 次算法执行中都曾得到最优解,所以算法具有较好的全局寻优性。同时,从表 3 中可以看到与下界比较得到的相对误差 Gap_2 与 Gap_1 比较,大了近 30%,原因是问题的复杂性导致下界的估计相对最优解而言偏离较大。所以,下面对于规模较大实例的分析中,由于无法在短时间内得到最优解,所以本文只分析算法解与下界之间的相对误差一定程度上也可以较好刻画解的质量。

由表 4 中可知,算法对于每一类实例在工件加工时间较小情况下(且与运输时间差距较小)的实例体现了较好的效果,与最优解下界比较得到的相对误差都在 50%左右。即使对于规模最大的两个类型 EX7 和 EX8,算法的性能较规模较小实例并未下降太多。事实上,由于算法解的比较对象是问题的下界,而问题下界比最优解小很多,所以算法解相对实际最优解的相对误差将会远小于 50%(可以参考算法在相对规模较小实例中的表现,见表 3)。对于算法执行的时间效率,本文算法尤为出色,对于小规模与中规模实例,算法都在 10 min 内得到满意结果,对于大规模实例,算法也可以在 1 h 内得到满意解。

从表 5 中可知,对于每一类实例在工件加工时间较大情况下,此时运输时间相对于加工时间较小,算法同样也体现了较好的求解效果。无论实例规模大小,工件加工时间的变大对算法的影响较小,与表 4 中体现的性能差别不大。表 4 和表 5 中的不同实例不仅体现了工件加工时间和问题规模对算法的影响,同时对加工时间与运输时间差距不同情况下的算法分析,一定程度上也说明本文算法的鲁棒性。当工件加工时间与运输时间差距较小时,机器人运输策略除了影响死锁的产生以外,对最终的目标函数也会产生较大影响。当运输时间远小于工件加工时间时,机器人运输策略对于目标函数值的影响较小,主要起到对工件加工顺序的支持和死锁的控制作用。从表 4 和表 5 中可以看到,本文算法受运输时间大小的影响较小。

进一步对本文算法的收敛性能进行分析。通过对每一类实例算法的迭代次数与解的质量进行分析,如图 4 所示。图 4(a)图像针对的是工件加工时

间 $\rho\in[1,10]$情况下的实例类型 EX1—EX4 进行求解的结果,图 4(b)图像针对的是工件加工时间 $\rho\in[10,100]$情况下实例类型 EX5—EX8 的计算结果。从图 4(a)的迭代过程曲线可知,对于实例类型 EX1—EX4,当算法迭代次数在 45 次的时候,算法即收敛到较好解。从图 4(b)的迭代过程曲线可知,对于较大规模实例类型 EX5—EX8,算法在迭代 80 次左右时也收敛到较好满意解。所以本文算法具有满意的收敛速度,并在算法的寻优效果及收敛速度上达到了较好的平衡。



(a) 实例 EX1—EX4　　(b) 实例EX5—EX8

图 4　不同实例的算法迭代收敛曲线

5　结　论

本文研究了一类依赖机器人运输的柔性作业车间调度问题,针对该类在芯片制造等领域具有广泛应用背景的实时调度问题,现有文献只给出了问题的整数规划模型,本文研究的目的是给出该问题的一个高效无死锁调度算法。该研究的难点在于最优调度方案的搜索过程中,如何保证解的可行性即如何检测并求解死锁,并保证死锁求解与算法寻优具有较好的收敛性。本文算法的基本思路是首先对问题进行时序 Petri 网建模,目的不是简单使用传统的 Petri 网进行死锁仿真,而是给出解的一种适合死锁判断并求解的形式,同时方便进行解的解码与迭代。其次,通过对问题死锁产生的原因和类型进行分析,给出通用的死锁判定和求解算法,并证明算法对于任意可能发生的死锁都能在多项式时间内进行求解。最后,给出问题的蜂群算法求解方案。在求解过程中,对于产生的死锁解,通过快速检测和求解,在保证解的可行性情况下尽可能保留原有解的结构。通过理论分析和大规模数值实验分析,得到本文的死锁求解算法不仅能够有效求解任意死锁,同时极大地保留了工序之间的相对顺序,使得蜂群算法在寻优过程中能够更多保留优质解的特性,从而加快算法收敛。通过不同规模实例的数值实验和与问题最优解及其下界的比较,也验证了本文算法在不同输入参数变化情况下都体现了较好的性能及收敛速度,特别是对于较大规模实例的求解,在收敛速度和解的质量上体系了较好的平衡,具有较好的实际应用前景。

本文结果是该类问题首个有效的算法结果。未来可以优化算法,进一步降低死锁求解的时间复杂度,提高启发式算法寻优的效率。此外,研究多机器人情形下问题的模型,并设计高效算法,将使本文研究更具现实意义和实用价值。

参考文献:

[1] 胡觉亮,李志林,董建明. 工件加工需要额外资源的平行机调度问题[J]. 浙江理工大学学报(自然科学版),2022,47(5):791-797.

[2] Ali Chaudhry I, Ali Khan A. A research survey: review of flexible job shop scheduling techniques [J]. International Transactions in Operational Research, 2016, 23(3): 551-591.

[3] 曹珍,韩曙光. 考虑充电调度的电动无人车配送路径规划问题研究[J]. 浙江理工大学学报(自然科学),2023,49(6):784-794.

[4] Yan P Y, Liu S Q, Sun T F, et al. A dynamic scheduling approach for optimizing the material handling operations in a robotic cell[J]. Computers & Operations Research, 2018, 99: 166-177.

[5] Wei L X, He J X, Guo Z Y, et al. A multi-objective migrating birds optimization algorithm based on game theory for dynamic flexible job shop scheduling problem [J]. Expert Systems with Applications, 2023, 227: 120268.

[6] Liu S Q, Kozan E, Masoud M, et al. Job shop scheduling with a combination of four buffering constraints [J]. International Journal of Production Research, 2018, 56(9):

3274-3293.

[7] Bektur G, Saraç T. A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server[J]. Computers & Operations Research, 2019, 103: 46-63.

[8] Drobouchevitch I G, Neil Geismar H, Sriskandarajah C. Throughput optimization in robotic cells with input and output machine buffers: A comparative study of two key models[J]. European Journal of Operational Research, 2010, 206(3): 623-633.

[9] Li X L, Xing K Y, Lu Q C. Hybrid particle swarm optimization algorithm for scheduling flexible assembly systems with blocking and deadlock constraints[J]. Engineering Applications of Artificial Intelligence, 2021, 105: 104411.

[10] Xing K Y, Wang F, Zhou M C, et al. Deadlock characterization and control of flexible assembly systems with Petri nets[J]. Automatica, 2018, 87: 358-364.

[11] 李浚,罗继亮,李旭航,等. 基于Petri网和监督学习的机器人柔性流水车间调度方法[J/OL]. 控制理论应用: 1-9(2024-04-18)[2024-05-01]. http://kns.cnki.net.elib.zstu.edu.cn:80/kcms/detail/44.1240.tp.20240415.2248.004.html.

[12] Luo J C, Liu Z Q, Zhou M C. A Petri net based deadlock avoidance policy for flexible manufacturing systems with assembly operations and multiple resource acquisition[J]. IEEE Transactions on Industrial Informatics, 2019, 15(6): 3379-3387.

[13] Hurink J, Knust S. Tabu search algorithms for job-shop problems with a single transport robot[J]. European Journal of Operational Research, 2005, 162(1): 99-111.

[14] Nouri H E, Driss O B, Ghédira K. Hybrid metaheuristics for scheduling of machines and transport robots in job shop environment[J]. Applied Intelligence, 2016, 45(3): 808-828.

[15] 晏晓凡. 改进灰狼优化算法求解机器人作业车间调度问题[J]. 计算技术与自动化,2023,42(2):158-163.

[16] Sun Y G, Chung S H, Wen X, et al. Novel robotic job-shop scheduling models with deadlock and robot movement considerations[J]. Transportation Research Part E: Logistics and Transportation Review, 2021, 149: 102273.

[17] 李大成, 罗继亮, 孙莎莎, 等. 基于平行Petri网的制造系统调度与控制一体化方法[J]. 自动化学报, 2023, 49(4): 845-856.

[18] Bai D Y, Bai X Y, Li H R, et al. Blocking flowshop scheduling problems with release dates[J]. Swarm and Evolutionary Computation, 2022, 74: 101140.

(责任编辑:康 锋)