

工件可拒绝的有限等待置换流水车间调度算法

王柏琳[†], 王海凤, 李铁克

- (1. 北京科技大学 东凌经济管理学院, 北京 100083;
2. 钢铁生产制造执行系统技术教育部工程研究中心, 北京 100083)

摘要: 有限等待限定了工件在相邻机器间的等待时间上下限, 普遍存在于中间产品性质不稳定且存在运输作业的车间环境中. 工件可拒绝的有限等待置换流水车间调度是对工件拒绝和工件调度的联合决策, 要求确定拒绝工件集合并给出被接受工件的调度方案. 针对这一联合决策问题, 以最小化总拒绝成本与总拖期成本之和为目标, 并为最大完工时间(Makespan)设置上限约束, 结合问题特征提出一种协同进化遗传算法. 该算法将染色体编码分解为工件拒绝和工件序列两个子集, 基于调度规则生成初始种群, 引入协同进化策略依次进化子集种群, 并提出基于记忆的动态概率参数设计方法以确定遗传算子的执行概率, 设计解码规则以保证解的可行性并优化总成本. 最后, 通过数据实验验证了所提算法及相关策略的可行性和有效性, 并分析了问题参数对算法性能的影响.

关键词: 生产调度; 工件拒绝; 置换流水车间; 有限等待; 遗传算法; 协同进化

中图分类号: TP278

文献标志码: A

Scheduling algorithm for permutation flowshop with limited waiting times and rejection

WANG Bai-lin[†], WANG Hai-feng, LI Tie-ke

- (1. Donlinks School of Economics and Management, University of Science and Technology Beijing, Beijing 100083, China; 2. Engineering Research Center of MES Technology for Iron & Steel Production, Ministry of Education, Beijing 100083, China)

Abstract: Limited waiting time constraint restricts the maximal and minimal waiting times of a job between consecutive machines. This constraint widely exist in the manufacturing environment with transportation requirement and high production continuity due to the unstable physical or chemical nature of intermediate product. The permutation flowshop scheduling problem with rejection is a joint decision of job rejection and production scheduling, which is to determine rejected jobs and the schedule with accepted jobs. This joint decision problem is considered with the objective to minimize the sum of total rejection cost and total tardiness cost, and an upper bound is set for the makespan in this problem. A co-evolutionary genetic algorithm is presented, in which an encoding scheme is proposed to divide a chromosome into two subsets, respectively corresponding to job rejection and job sequencing. The initial population is produced by dispatching rules, and the co-evolutionary strategy is introduced into this algorithm to coevolve the two subset population. Moreover, a memory-based dynamic probability generation method is presented to determine the probabilities to run the genetic operators, and some decoding rules are set to ensure the feasibility of a solution and optimize its total cost. Computational experiments are carried out to verify the effectiveness of this algorithm, and the impact of problem parameters on the performance of this algorithm are also studied in the experiments.

Keywords: production scheduling; job rejection; permutation flowshop; limited waiting times; genetic algorithm; co-evolutionary

0 引言

在中间产品性质不稳定且工序间存在运输作业的工业生产中, 通常要求加工对象在相邻工序间

的等待时间位于给定的上下限内, 即有限等待约束(Limited waiting time constraints). 例如, 钢铁生产中由于高温作业要求, 钢水在相邻工序间只允许等待较短

收稿日期: 2017-08-24; 修回日期: 2017-11-26.

基金项目: 国家自然科学基金项目(71701016); 北京市自然科学基金项目(9174038); 教育部人文社会科学研究青年基金项目(17YJC630143); 中央高校基本科研业务费专项资金项目(FRF-BD-16-006A).

责任编辑: 刘民.

作者简介: 王柏琳(1983—), 女, 讲师, 博士, 从事生产计划与调度、智能优化算法的研究; 李铁克(1958—), 男, 教授, 博士, 从事先进制造管理、生产计划与调度等研究.

[†]通讯作者. E-mail: wangbl@ustb.edu.cn.

时间,且要满足运输时间的要求;食品生产限制了等待时间的上下限,以保证工序间正常衔接且防止食物变质;在半导体晶圆制造中,一片晶圆在一个加工模块中完成加工后必须在规定的时间内取出.已有研究表明,等待时间下限和上限约束均会增加机器空闲时间^[1-2],使机器产能无法充分利用.在订单管理层制定订单接受决策时,若仅考虑产能则会与实际生产存在较大偏差,产生不必要的订单拒绝或拖期惩罚.因此,有必要在订单接受决策时考虑有限等待约束,将订单视为工件,对工件拒绝与生产调度进行联合决策,确定拒绝的工件并给出接受工件的调度方案,此即为工件可拒绝的有限等待调度问题.置换流水车间是最基本的一类多工序生产环境,本文将以此为背景展开研究.

工件可拒绝的调度(Scheduling with rejection)在理论上是一类多决策组合优化问题,在生产实际中能够优化订单和生产管理,近年来引起调度专家和管理者的关注.目前,有关置换流水车间环境的研究较少,主要针对两机或成比例流水车间.对于两机问题,Shabtay等^[3]证明了最小化最大完工时间(Makespan)和工件拒绝成本之和的问题是NP-难的,并给出近似算法;Zhang等^[4]证明了即使工件拒绝成本相同或在一台机器上加工时间相同,问题也是NP-难的;谢谢等^[5]考虑了第1台机器具有不可用区间、工件可中断的情况,针对最小化Makespan与拒绝惩罚之和设计了动态规划算法和启发式算法;苗翠霞等^[6]考虑了工件的退化效应,针对最小化Makespan与拒绝惩罚之和设计了动态规划算法.对于成比例流水车间,Shabtay等^[7]考虑了拒绝成本和调度目标两类指标,分别针对指标加总为单目标、优化第1个目标并为第2个目标设上限约束、优化第2个目标并为第1个目标设上限约束、同时优化双目标这4种指标处理方式,探讨问题的复杂性并给出了近优方案;Li等^[8]以最小化调度成本与拒绝成本之和为目标,其中调度成本分别考虑了最大拖期和总加权完工时间,证明了问题是NP-难的并设计了近优算法.

有限等待置换流水车间调度研究始于2006年,Fondreville等^[1]证明了最小化Makespan问题是强NP-难的.对于交货期相关目标,Fondreville等^[9]针对等待时间上下限相同且最小化最大拖期问题给出了多项式可解的特例,并提出了分支定界法;Dhouib等^[10]考虑最小化拖期工件数和顺序依赖的安装时间,提出了数学模型和模拟退火算法;Hamdi等^[11]探讨最小化总拖期的上下界,设计了拉格朗日松弛算

法.可以看出,目前有限等待置换流水车间调度研究较少,且研究方法以分支定界等精确算法为主,无法应用于大规模的实际生产环境.此外,对于有限等待置换流水车间下的工件拒绝与调度的联合决策,笔者尚未发现相关文献.

本文将研究工件可拒绝的有限等待置换流水车间调度问题,并考虑总成本和Makespan两个优化指标,其中,总成本为拒绝成本与拖期成本之和.考虑到同时将两个指标作为目标函数会加大求解难度,且订单层面的决策更偏重收益,因此本文基于 ε -约束法^[7],将总成本指标作为最小化目标,并为Makespan指标设置上限约束,将问题转化为单目标优化.为了能在短时间内取得问题的优质解,提出一种协同进化遗传算法,结合问题特征和协同进化特点设计编码方案和解码规则,并给出基于记忆的动态概率设计方法以确定遗传算子的执行概率.最后,通过数据实验对算法有效性以及关键问题参数对算法性能的影响进行分析.

1 问题描述

工件可拒绝的有限等待置换流水车间调度问题描述如下:给定 n 个工件 $\{J_1, \dots, J_n\}$,工件允许拒绝,若拒绝则产生拒绝成本,否则工件被接受并安排生产,此时可能产生拖期成本(工件拒绝决策).被接受的工件以相同的加工路径依次经过 m 台机器 $\{M_1, M_2, \dots, M_m\}$,每台机器上的工件加工序列均相同(置换流水车间).工件在相邻机器间的等待时间必须位于给定的上下限之内(有限等待约束).问题要求确定拒绝工件集合,并对接受工件进行调度,优化目标为最小化总拒绝成本与总拖期成本之和.此外,考虑到接受工件应在计划期内完工,将计划期间长度设为Makespan上限加以约束.

为便于描述,定义以下符号.

1) 下标和参数.

i, j : 工件和机器编号;

n, m : 工件和机器数量;

J_i : 编号为 i 的工件;

M_j : 编号为 j 的机器,即加工路径上的第 j 台机器;

p_{ij} : J_i 在 M_j 上的加工时间;

rej_i, w_i, d_i : J_i 的拒绝成本、单位拖期成本和交货期;

$\theta_{ij}^{\min}, \theta_{ij}^{\max}$: J_i 在 M_j 与 M_{j+1} 之间的等待时间的下限和上限;

$C_{\max}^+$: 最大完工时间上限,即计划期间长度.

2) 决策变量.

RS: 拒绝工件集合;

π : 工件加工序列, $\pi(k)$ 为 π 的第 k 个工件, $|\pi| = n$;

C_{ij} : J_i 在 M_j 的完工时间.

3) 决策变量的衍生变量.

T_i : J_i 的拖期;

f_i : J_i 的成本, 若 J_i 被拒绝则 $f_i = \text{rej}_i$, 否则 $f_i = w_i T_i$;

$C_{\max}$: 最大完工时间 (Makespan).

根据以上符号, 有限等待约束表示为

$$\theta_{ij}^{\min} \leq C_{i,j+1} - p_{i,j+1} - C_{ij} \leq \theta_{ij}^{\max}; \quad (1)$$

总成本目标和 Makespan 约束表示为

$$\begin{aligned} \min f = \sum_{\forall i} f_i = \sum_{i \in \text{RS}} \text{rej}_i + \sum_{k \in \pi} w_k T_k = \\ \sum_{i \in \text{RS}} \text{rej}_i + \sum_{k \in \pi} w_k \max\{C_k - d_k, 0\}, \quad (2) \end{aligned}$$

$$C_{\max} = \max_{i \in \pi} \{C_{im}\} \leq C_{\max}^+. \quad (3)$$

本文的问题是工件拒绝与调度的联合决策, 问题解可表示为决策对 $\langle \text{RS}, \pi \rangle$. 最小化总拖期的有限等待置换流水车间调度问题是本文调度子决策的特例, 具有强 NP-难复杂性^[1], 因此本文问题是强 NP-难的, 精确算法无法在有限时间内获得问题解, 不能满足实际应用需求. 遗传算法 (Genetic algorithm, GA) 等群智能算法为快速求解该类问题提供了一种有效途径.

2 协同进化遗传算法

给定决策对 $\langle \text{RS}, \pi \rangle$, 则接受工件的完工时间可按以下步骤确定, 记为 $\text{Timing}(\pi)$, 复杂度为 $O(|\pi|m)$:

$$C_{\pi(1),1} = p_{\pi(1),1};$$

For $j = 2$ to m do

$$C_{\pi(1),j} = C_{\pi(1),j-1} + \theta_{\pi(1),j-1}^{\min} + p_{\pi(1),j}$$

End For

For $i = 2$ to $|\pi|$ do

$$C_{\pi(i),1} = C_{\pi(i-1),1} + p_{\pi(i),1};$$

For $j = 2$ to m do

$$C_{\pi(i),j} = p_{\pi(i),j} + \max\{C_{\pi(i-1),j}, \\ C_{\pi(i),j-1} + \theta_{\pi(i),j-1}^{\min}\}$$

End For

For $j = m - 1$ to 1 do

$$C_{\pi(i),j} = \max\{C_{\pi(i),j}, C_{\pi(i),j+1} - \\ p_{\pi(i),j+1} - \theta_{\pi(i),j}^{\max}\}$$

End For

End For

显然, 问题的求解关键在于如何生成高质量的决

策对 $\langle \text{RS}, \pi \rangle$. 遗传算法具有快速求解能力, 能够灵活设置染色体编码和遗传算子. 此外, 协同进化是一类针对多决策问题的优化策略, 现已成功应用于遗传算法、粒子群算法、蜂群算法等^[12-14]. 考虑到本文问题的联合决策特征, 本节设计协同进化遗传算法 (Co-evolutionary GA, CGA), 并引入问题特征来优化算法搜索.

2.1 编码方案与初始种群生成

工件可拒绝的有限等待置换流水车间调度问题由工件拒绝和调度决策构成, 而调度的难点则在于工件序列 π 的确定. 因此, CGA 将染色体分解为两个子集 X 和 Y : 子集 $X = \{x_1, \dots, x_n\}$ 为二进制编码, 表示工件拒绝情况, 若 $x_i = 1$ 则拒绝 J_i , 若 $x_i = 0$ 则接受 J_i ; 子集 $Y = \{y_1, \dots, y_n\}$ 为自然数的置换序列编码, 表示工件序列 (n 为工件数). 考虑 5 个工件的问题 ($n = 5$), 若染色体 $[X, Y]$ 为 $X = [0, 1, 0, 0, 1]$, $Y = [3, 4, 2, 1, 5]$, 则对应的决策对为 $\langle \text{RS}, \pi \rangle = \langle \{J_2, J_5\}, \{J_3, J_4, J_1\} \rangle$.

CGA 生成初始种群时, 对于每个染色体, 首先基于调度规则生成工件序列 (子集 Y), 进而确定拒绝工件集合 (子集 X). Hamdi 等^[11] 针对最小化总拖期的有限等待置换流水车间提出了 3 种调度规则: 最短加工时间优先规则 (Shortest processing time, SPT)、基于瓶颈机的调整规则 (Adjustment on the bottleneck machine, ABM) 和预估完工时间规则 (Estimated completion time, ECT), 其中 SPT 和 ECT 分别生成 $m+1$ 和 m 个工件序列, 从中选择最好方案 (m 为机器数). CGA 采用上述 3 种规则以及经典的最早交货期优先规则 (Earliest due date, EDD) 生成工件序列 (子集 Y), 且为了增加种群多样性并减少计算时间, 对 SPT 和 ECT 的 $2m+1$ 个序列不再择优, 而是各自编码为子集 Y , 由此 4 种规则将产生 $2m+3$ 个序列. 限定种群规模 $ps \geq 2m+3$, 随机生成种群中的剩余子集 Y .

给定子集 Y , 根据总成本和 $C_{\max}^+$ 确定拒绝工件集合, 生成子集 X . 具体方法如下: 每次拒绝令总成本下降最多的工件, 直至总成本不再下降或 $C_{\max}$ 满足上限约束. 将该方法命名为 $\text{Initializing_X}(Y)$, 伪代码如下, 其复杂度为 $O(n^2m)$:

$$\pi = Y, \text{RS} = \emptyset;$$

For $i = 1$ to n do $x_i = 0$

End For

$$\text{Timing}(\pi); i^* = \arg \max_{i \in \pi} \{w_i T_i - \text{rej}_i\};$$

While $(\Delta_{i^*} \geq 0 \vee C_{\max} > C_{\max}^+)$ do

$$x_{i*} = 1; RS = RS \cup \{i^*\}; \pi = \pi \setminus \{i^*\};$$

$$\text{Timing}(\pi); i^* = \arg \max_{i \in \pi} \{w_i T_i - \text{rej}_i\};$$

End While

2.2 协同进化策略

协同进化是一种有效解决多决策优化问题的求解策略,它将个体拆分成多个子集,每个子集对应一类决策,通过独立地进化每个子集,再将各子集结合成一个完整的解,对完整解进行评价来实现子集种群的选择^[13]. 本文将协同进化引入算法,理由如下:1) 本文问题为工件拒绝与调度的联合决策,在CGA编码方案中,染色体由子集 X 和 Y 构成,分别对应这两类决策,且任意 X 和 Y 均可组合成一个完整解,这一特征符合协同进化的应用场景;2) 子集 X 和 Y 具有不同的编码特征,子集 X 为二进制编码,子集 Y 则为置换序列编码,因此,为子集 X 和 Y 分别设计遗传算子并独立进化,这种方式比传统的统一进化更为灵活有效.

如上所述,协同进化的个体评价方法比较特殊,需要将当前进化的子集与其他子集个体结合,形成完整的问题解进行评价,而选择哪些其他子集个体进行组合,这是协同进化策略的一个关键点. 常用的选择方法有两种,选择其他子集种群的最优个体或随机选择个体. 已有研究表明,在其他子集种群中选择最优个体进行协同进化能够取得较好的结果^[12],目前已在多篇文献中采用^[14]. 这种最优个体选择策略的实质是通过评价当前子集与其他子集种群中最优个体的协同效果,来寻找与此最优个体具有更佳协同性的个体,这意味这二者的组合会产生更优的解,以此保证进化过程的持续优化. 因此,CGA算法在进行子集评价时,将选择另一子集种群中的最优个体进行问题解的组合与评价.

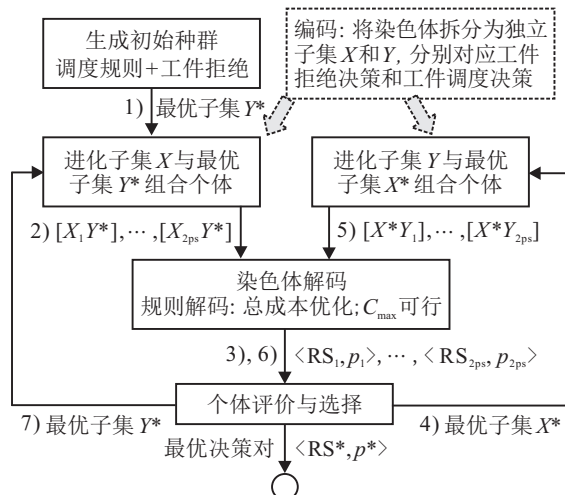


图1 CGA协同进化策略示意

图1给出了CGA协同进化示意,每次迭代首先对子集 X 的种群 pop_X 执行遗传操作,得到子代集合 off_X ,将 $\text{pop}_X \cup \text{off}_X$ 中的子集 X 与当前最优子集 Y^* 组合,形成个体集合并进行评价与选择,得到新种群 pop_X ;然后,采用相同的方法进化子集 Y 种群 pop_Y ,至此完成一代完整的进化.

2.3 遗传算子及执行概率设计

CGA的子集 X 和 Y 采用不同的编码方式,本节将根据编码特征分别为其设计遗传算子. 子集 X 为经典的二进制编码,将采用传统的交叉和变异算子(记为 $X_Crossover$ 和 $X_Mutation$). 子集 Y 为置换序列,若采用传统算子则容易产生基因值相同的非法解,因此,本文采用广泛应用于该类编码的单元交叉算子、插入变异算子和交换变异算子来生成子代子集 Y ,记为 $Y_Crossover$ 、 Y_Insert 和 Y_Swap . 图2给出了这5种算子的示意.

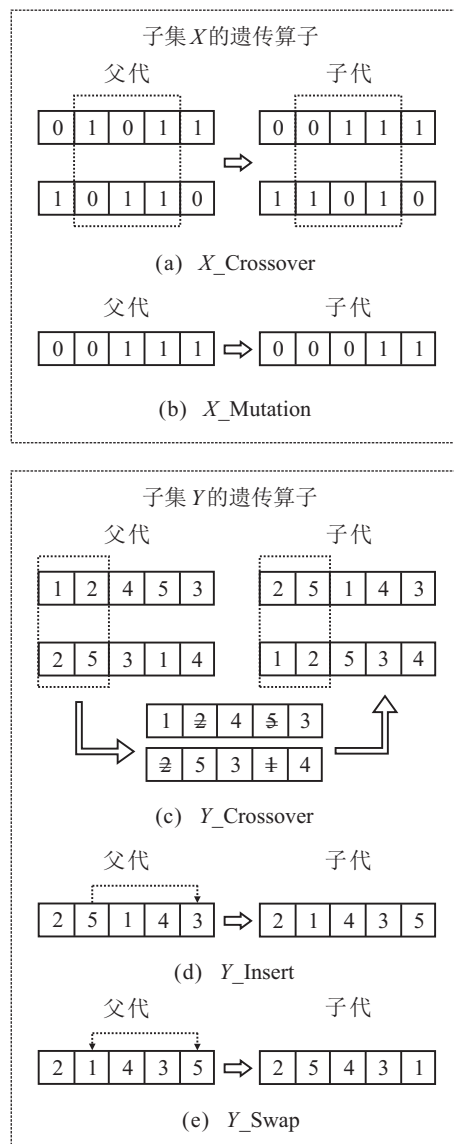


图2 子集 X 和 Y 的遗传算子

给定遗传算子,如何确定执行该算子的概率,这是遗传操作的另一关键问题.算子执行概率一般被预先指定,不根据实际进化情况自动调整.然而,对于不同的算例以及相同算例下的不同进化阶段,遗传算子的执行效果往往是不同的,因此算子概率应具有自适应能力.本文提出一种基于记忆(Memory)的动态概率设计方法,能够根据实际进化情况动态调整.在该方法中,每种遗传算子有两个执行概率:指定概率 P^s 和记忆概率 P^m ,其中 P^m 通过记忆遗传算子在进化过程中的优化效果进行动态调整.此外,参考Pan^[14]在蜂群算法中提出的自适应邻域算子,设定控制概率 P^c ,每次遗传操作时若生成的随机数位于 $[0, P^c]$,则采用记忆概率 P^m ,否则采用指定概率 P^s .

基于记忆的动态概率设计思想源于近年发展起来的基于知识/机器学习的调度算法^[15-16],即对优质染色体或有效的参数设置进行记忆,进而基于统计学习方法来学习基因知识或参数知识,指导算法搜索.在参数知识学习方面,Xing等^[15]针对蚁群算法设置参数组合,并记忆各组合的成功次数来更新参数组合的选择概率,从而降低算法参数对优化的敏感度.

借鉴上述记忆学习策略,CGA的动态概率采用统计学方法,记录并学习各算子对算例及其进化过程的适应程度,进而判定当前阶段的算子记忆概率.该方法的基本思路如下:首先,记忆遗传算子在各代的执行情况,持续更新各算子的成功概率,其中,若由该算子生成的子代被选择至新种群中,则记为一次成功.若某算子的成功概率较高,则被认为由该算子进化后得到的子代被选择至新种群的可能性较大,进而为该算子赋予较高的记忆概率 P^m .可见,记忆概率 P^m 需记忆遗传算子的累计执行次数和累计成功次数,其中,算子执行次数可以在遗传操作时进行累计更新,成功次数则在选择过程中进行判定和累加,进而采用以下规则更新 P^m .

记忆规则1:

If 累计执行次数 = 0,

Then $P^m = P^s$;

记忆规则2:

If 累计执行次数 $\neq 0 \wedge$ 累计成功次数 = 0,

Then $P^m = 1/(2ps \cdot t)$, t 为进化代数;

记忆规则3:

If 累计成功次数 $\neq 0$,

Then $P^m = \text{累计成功次数} / \text{累计执行次数}$.

前两项规则确保遗传算子在从未执行或成功的情况下仍有运行机会.在协同进化中,每更新一类子

集的遗传算子记忆概率后,将对概率进行总和和标准化处理.例如,若某代进化子集 X 后,根据累计执行和成功次数得到的交叉和变异算子的记忆概率分别为0.3和0.1,则用于子集 X 下次进化的记忆概率为其标准化值0.75和0.25.

2.4 解码、评价与选择

如2.1节所述,CGA染色体可直接解码为决策对 (RS, π) .然而,遗传操作具有随机性,得到的决策对可能违背Makespan上限约束,且总成本尚有优化余地.因此CGA设计了解码规则,工件按 π 中顺序依次执行以下规则,生成可行且较优的解.

解码规则1:

If $C_{\pi(i),m} > C_{\max}^+$,

Then 拒绝 $J_{\pi(i)}$;

解码规则2:

If $\text{rej}_{\pi(i)} \leq w_{\pi(i)} T_{\pi(i)}$,

Then 拒绝 $J_{\pi(i)}$.

规则1保证了解的可行性.此规则仅拒绝第1个违背Makespan上限的工件 $J_{\pi(i)}$,这样若有加工时间短或拖期成本小的工件位于 $J_{\pi(i)}$ 之后,则该工件有被接受的机会.规则2是对总成本的优化.这里没有采用复杂度为 $O(n^2m)$ 的Initializing_X(Y),而是基于贪婪思想,在对 π 的解码过程中,每确定一个调度工件,即调用解码规则2进行拒绝判定,以此提高算法效率.CGA的解码过程如下,记为Decoding($[XY]$),复杂度为 $O(nm)$:

令 $\pi = \emptyset$; $RS = \emptyset$; 令 $k = 1, 2, \dots, n$,执行以下步骤:

Step 1: $i = y_k, r = x_i$. k 为工件序列位置 i 的工件编号, r 为 J_i 的拒绝情况.

Step 2: 若 $r = 1$,则 $RS = RS \cup \{i\}$,转Step 4; 否则,转Step 3.

Step 3: 计算 C_{ij} 和复杂度 $O(m)$.若 $\pi_i = \emptyset$,则 $C_i = p_i$,对于 $j = 2, \dots, m$,计算 $C_{ij} = C_{i,j-1} + \theta_{i,j-1}^{\min} + p_{ij}$; 否则,对于 $j = 2, \dots, m$,计算 $C_{ij} = \max\{C_{\pi(|\pi|),j}, C_{i,j-1} + \theta_{i,j-1}^{\min}\} + p_{i,j}$,进而对于 $j = m-1, \dots, 1$,计算 $C_{ij} = \max\{C_{ij}, C_{i,j+1} - p_{i,j+1} - \theta_{ij}^{\max}\}$. 转Step 4.

Step 4: 基于解码规则的工件拒绝再判定.若 $C_{im} > C_{\max}^+$,则 $RS = RS \cup \{j\}$ (规则1); 否则, $T_i = \max\{C_{im} - d_i, 0\}$.若 $\text{rej}_i \leq w_i T_i$,则 $RS = RS \cup \{j\}$,否则 $\pi_i = \pi_i \cup \{j\}$ (规则2).

染色体解码后将进行个体评价.根据本文问题所考虑的优化指标,将总成本作为主评价函数,

Makespan 作为平局判定函数 (Tie-breaker), 采用以下规则对个体 A 和 B 进行比较.

评价规则:

If $f(A) \leq f(B) \vee$
 $(f(A) = f(B) \wedge C_{\max}(A) < C_{\max}(B)),$

Then 个体 A 优于个体 B .

基于评价规则, CGA 采用父子锦标赛方法进行个体选择, 即每次分别从父代集合和子代集合中随机抽取一个个体进行比较, 保留较优个体.

2.5 算法步骤

CGA 算法步骤如下:

Step 1: 初始化.

1) 输入算法参数. 若种群规模 $ps < 2m + 3$, 则 $ps = 2m + 3$.

2) 调用 SPT、ABM、ECT、EDD 规则生成 $2m + 3$ 个子集 Y , 随机生成剩余子集 Y , 形成种群 pop_Y .

3) 对于 $y = 1, \dots, ps$, 执行

$$X = \text{Initializing_X}(pop_Y(y)),$$

$$pop_X = pop_X \cup \{X\},$$

记录最优个体 $[X^*Y^*]$.

Step 2: 进化 pop_X .

1) 若随机数 $\leq$ 控制概率 P^c , 则按记忆概率 $P^m(X_Crossover)$ 和 $P^m(X_Mutation)$ 调用两种遗传算子, 生成子代集合 off_X ; 否则, 按指定概率 $P^s(X_Crossover)$ 和 $P^s(X_Mutation)$ 调用两种算子生成 off_X . 更新算子累计执行次数.

2) 对于 $x = 1, \dots, ps$, 执行

$$(\text{Decoding})([pop_X(x)Y^*]),$$

$$(\text{Decoding})([off_X(x)Y^*]).$$

3) 调用父子锦标赛法, 根据评价规则选择个体, 更新 pop_X 和遗传算子累计成功次数; 根据记忆规则 1~记忆规则 3, 计算 $P^m(X_Crossover)$ 和 $P^m(X_Mutation)$, 并进行总和标准化处理, 更新最优子集 X^* .

Step 3: 进化 pop_Y .

1) 若随机数 $\leq$ 控制概率 P^c , 则按记忆概率 $P^m(Y_Crossover)$ 、 $P^m(Y_Insert)$ 和 $P^m(Y_Swap)$ 调用 3 种算子, 生成子代集合 off_Y ; 否则, 按指定概率 $P^s(Y_Crossover)$ 、 $P^s(Y_Insert)$ 和 $P^s(Y_Swap)$ 调用算子生成 off_Y , 更新算子累计执行次数.

2) 对于 $y = 1, \dots, ps$, 执行

$$\text{Decoding}([X^*pop_Y(y)]),$$

$$\text{Decoding}([X^*off_Y(y)]).$$

3) 调用父子锦标赛法选择个体, 更新 pop_Y 和遗传算子累计成功次数; 调用记忆规则 1~记忆规则 3 和总和标准化方法计算 $P^m(Y_Crossover)$ 、 $P^m(Y_Insert)$ 和 $P^m(Y_Swap)$, 更新最优子集 Y^* .

Step 4: 若不满足终止准则, 则转 Step 2; 否则, 输出 $[X^*Y^*]$ 对应的决策对 (RS^*, π^*) .

3 数据实验

3.1 实验设计

为检验 CGA 算法及其关键策略的有效性, 实验设置了以下 4 种对比算法:

1) CGA_P: 协同进化遗传算法, 遗传算子采用指定概率, 以检验 CGA 动态概率方法的有效性.

2) CGA_I: 协同进化遗传算法, 初始种群随机生成, 采用 Decoding() 解码, 用来检验 CGA 初始种群生成方法的有效性.

3) CGA_D: 协同进化遗传算法, 初始种群随机生成, 解码时仅保留规则 1 来保证解的可行性. 与 CGA_I 对比可检验 CGA 解码规则的有效性, 与 CGA 对比检验 CGA 引入问题特征的优化效果.

4) TGA: 传统遗传算法, 将子集 X 和 Y 作为一条染色体同时进化, 其他操作与 CGA 相同, 用来检验 CGA 协同进化策略的有效性.

算法采用 C# 语言编程, 运行环境为 Intel Core i5-6300U / CPU 2.40 GHz 2.50 GHz / RAM 8.00 GB. 问题规模设置为工件数 $n = 20, 50, 100, 150, 200$, 机器数 $m = 2, 5, 10$. 根据问题规模分为 15 组实验, 每组随机生成 50 个算例. 参考文献 [11] 的实验设计, 算例参数设置为: $p_{ij} \in \text{DU}[20, 50]$, 其中 $\text{DU}[a, b]$ 为位于 $[a, b]$ 的离散均匀分布; $\theta_{ij}^{\max} \in \text{DU}[0, 14]$ 、 $\theta_{ij}^{\min} \in \text{DU}[0, 7]$; $\text{rej}_i \in \text{DU}[1, 20]$; $C_{\max}^+ \in [U[0.8, 1] \times 35(n + m - 1)]$; $w_i \in U[0.5, 1.5]$, $d_i \in \text{DU}[0.2C_{\max}^-, C_{\max}^-]$, 其中 $C_{\max}^-$ 为不考虑工件拒绝时的 Makespan 下界, 公式如下^[11]:

$$C_{\max}^- =$$

$$\max \left\{ \max_i \sum_{j=1}^m (p_{ij} + \theta_{ij}^{\min}), \max_j \left[\sum_{i=1}^n p_{ij} + \right. \right.$$

$$\left. \min_i \sum_{k=1}^{j-1} (p_{ik} + \theta_{ik}^{\min}) + \min_i \sum_{k=j+1}^m (p_{ik} + \theta_{ik}^{\min}) \right] \right\}. \quad (4)$$

5 种遗传算法的参数设置如下: 基于计算时间和求解质量的折衷考虑, 设种群规模 $ps = n$. 对于子集 X , 设 $P^s(X_Crossover) = 0.8$, $P^s(X_Mutation) = 0.2$ ^[17]; 对于子集 Y , 考虑到其变异算子实质为

插入邻域和交换邻域,是调度领域的经典邻域构造方法^[18],本文适当提高了变异概率,设定为 $P^S(Y_Crossover) = 0.6, P^S(Y_Insert) = P^S(Y_Swap) = 0.2$. 对于控制概率 P^c ,参考Pan^[14]将控制概率设为0.75,本文针对问题规模 $20 \times 2, 50 \times 5$ 和 100×10 分别生成50个算例,对 $P^c = 0.7, 0.75$ 和 0.8 三种情况下的CGA成本相对于三者最低成本的偏离率进行计算,分别为5.37、4.48、3.77,进而选定 $P^c = 0.8$ (CGA_P中 $P^c = 0$). 算法终止准则为最大进化500代.

算法性能从有效性和求解效率两个方面衡量,其中求解效率的评价指标为CPU时间. 本节提出 $g(\text{Alg})$ 函数对算法Alg取得的总成本和Makespan进行综合评价(式(5)),并采用 $g(\text{Alg})$ 的相对偏离率(Percentage relative difference, PRD)衡量算法有效性(式(6)). 如式(5)所示, $g(\text{Alg})$ 首先对Makespan进行极差标准化处理,进而与总成本加总,其中 $\max C_{\max}$ 和 $\min C_{\max}$ 分别为5种算法取得的Makespan最大和最小值. 可见, $g(\text{Alg})$ 既能反映算法对Makespan的优化效果,又能保证总成本优化目标的主导地位. 在PRD的计算公式(6)中, g^* 为5种算法取得的 $g(\text{Alg})$ 最小(最好)值,有

$$g(\text{Alg}) = \begin{cases} f(\text{Alg}), & \max C_{\max} = \min C_{\max}; \\ f(\text{Alg}) + \frac{C_{\max}(\text{Alg}) - \min C_{\max}}{\max C_{\max} - \min C_{\max}}, & \\ \text{otherwise.} & \end{cases} \quad (5)$$

$$\text{PRD}(\text{Alg}) = \frac{g(\text{Alg}) - g^*}{g^*} \times 100. \quad (6)$$

3.2 算法性能对比与分析

表1给出了算法的PRD均值和标准差. 此外,本文采用方差分析(Analysis of variance, ANOVA)对每组实验进行PRD均值差异性的显著性检验,设 $\alpha = 0.05$,对应的 F 临界值为2.41. 表1的最后两列给出了每组实验的 F 与 P 值. 由此可得以下结论:

1) CGA的PRD值最低,其次为CGA_P;CGA_I在工件数较少时($n \leq 100$)优于TGA,但随着工件数增加将逐渐劣于TGA;CGA_D最差. 这表明CGA能取得相对较优的解. 每组实验进行ANOVA分析得到的 P 值均接近于0,明显小于 $\alpha(0.05)$,说明算法的求解质量存在显著差异. 此外,表1中CGA的标准差也是最小的,说明CGA不仅求解质量较高,而且求解性能相对稳定.

2) CGA_P的PRD值是CGA的平均1.50倍,最高2.06倍,这说明CGA基于记忆的动态概率设计方法比传统的指定概率具有更好的优化效果.

3) CGA_I的PRD值是CGA的平均5.88倍,最高15.86倍,说明CGA基于调度规则能够生成高质量的初始解,提高算法的有效性. 此外,CGA_D的PRD值是CGA_I的平均3.15倍,且是5种算法中求解质量最差的,这说明CGA在遗传算法中引入问题特征来设计求解规则的方法是可行且有效的.

4) TGA的PRD值是CGA的平均5.24倍、CGA_P的平均3.77倍. 显然,基于协同进化的CGA和CGA_P优于采用传统遗传策略的TGA算法. 此外,虽然CGA_I的初始解劣于TGA基于调度规则生成的初始解,但二者求解性能相近,这也说明了CGA协同进化策略的有效性.

表1 5种算法的PRD均值(Avg)、标准差(Std)与ANOVA分析结果(终止准则:进化500代)

$n \times m$	CGA		CGA_P		CGA_I		CGA_D		TGA		ANOVA($\alpha = 0.05$)	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	F	P -value
20×2	4.27	6.36	6.18	9.67	6.33	10.07	29.55	33.74	8.10	11.15	18.47	2.79×10^{-13}
20×5	3.16	5.37	5.32	7.70	4.50	8.44	23.27	17.93	6.21	7.62	32.24	1.37×10^{-21}
20×10	1.67	2.59	3.16	4.34	1.95	4.47	7.20	11.50	2.86	3.55	6.52	5.32×10^{-5}
50×2	3.08	4.08	4.19	5.35	8.61	10.53	26.47	20.18	11.34	10.33	33.03	4.97×10^{-22}
50×5	3.63	5.06	7.20	6.98	7.24	8.00	26.38	18.92	12.36	8.30	35.46	2.32×10^{-23}
50×10	2.93	4.02	3.67	4.28	5.07	5.30	20.64	11.10	8.69	7.38	55.44	3.02×10^{-33}
100×2	2.18	3.25	2.37	2.89	11.58	7.47	26.91	15.69	9.79	4.70	74.01	4.85×10^{-41}
100×5	1.88	3.25	3.01	4.12	10.50	7.33	31.29	18.29	11.08	4.39	80.09	2.30×10^{-43}
100×10	1.59	2.79	3.28	4.32	9.15	6.67	24.50	11.73	9.84	5.70	84.64	4.83×10^{-45}
150×2	1.12	1.69	1.44	2.24	16.17	7.23	37.14	16.02	10.41	4.72	161.18	2.20×10^{-67}
150×5	2.12	2.86	2.33	3.15	14.68	6.25	33.05	12.22	13.03	6.38	161.05	2.37×10^{-67}
150×10	1.64	2.77	2.95	3.66	10.10	5.63	26.88	8.82	12.38	6.38	148.56	2.76×10^{-64}
200×2	1.53	2.31	2.01	2.87	24.24	11.21	68.19	29.31	11.68	5.06	186.98	3.32×10^{-73}
200×5	1.94	2.86	2.43	3.48	20.68	7.81	44.75	15.32	15.67	5.29	222.36	2.81×10^{-80}
200×10	1.60	2.15	1.96	3.25	11.24	5.23	30.65	9.95	12.29	6.24	192.74	2.01×10^{-74}
平均值	2.29	3.72	3.44	5.17	10.80	9.61	30.46	21.99	10.38	7.37	714.34	0

在算法效率方面,表2给出了算法在求解小、中、大3种规模组以及所有实例的平均CPU时间. 由于CGA_P与CGA的区别仅在于 P^c 不同,计算时间与CGA基本相同,这里不再列出. 由表2可知,算法计算效率由高至低分别为TGA、CGA_D、CGA_I、CGA. 其中:1) 3种协同进化遗传算法CGA、CGA_I和CGA_D的计算时间相差不大,说明初始解生成方法和解码方案的运算时间较快,对算法效率影响不大;2) 协同进化遗传算法的计算时间多于TGA,这是因为每次进化要两个子集分别执行遗传和选择操作,而TGA是将两个子集作为一个染色体进化;3) CGA虽然计算时间相对较长,但即使是对于200个工件10台机器的问题,也能在25 s内取得问题解,完全能够满足实际

应用的需要.

表2 算法的CPU时间

$n \times m$	s			
	CGA	CGA_I	CGA_D	TGA
20×2	0.15	0.14	0.14	0.06
100×5	3.66	3.58	3.29	1.48
200×10	23.43	22.48	18.97	9.61
实例均值	6.91	6.64	5.88	2.89

由表2可知,在相同进化代数下,3种协同进化遗传算法的计算时间与TGA存在明显差距. 换言之,在相同计算时间下,TGA的进化代数为协同进化遗传算法的2倍以上. 因此可能会存在以下情况:1) 若以TGA计算时间为终止准则,则CGA代数过少,很可能处于快速收敛的初期阶段,搜索到的解可能劣于

表3 5种算法的PRD均值(Avg)、标准差(Std)与ANOVA分析结果(终止准则:计算时间)

$n \times m$	CPU时间/s	CGA		CGA_P		CGA_I		CGA_D		TGA		ANOVA($\alpha = 0.05$)	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	F	P-value
20×2	(t_T) 0.05	12.33	13.98	9.55	13.75	13.42	14.03	30.46	34.11	12.87	11.77	9.30	5.13×10^{-7}
	(t_C) 0.12	6.42	9.81	8.13	12.19	9.93	12.15	29.70	28.81	8.41	13.17	16.79	3.55×10^{-12}
20×5	(t_T) 0.06	6.86	6.73	9.44	13.42	6.99	6.94	26.76	24.44	8.10	9.34	18.96	1.35×10^{-13}
	(t_C) 0.15	4.05	7.50	5.54	6.72	4.79	6.82	19.42	13.83	4.12	5.95	29.44	5.38×10^{-20}
20×10	(t_T) 0.09	2.58	3.88	4.37	6.24	2.45	3.10	8.84	9.32	2.88	4.10	10.85	4.12×10^{-8}
	(t_C) 0.23	1.90	3.29	3.23	4.02	2.22	3.79	7.34	6.36	2.33	3.35	13.68	4.42×10^{-10}
50×2	(t_T) 0.26	9.53	7.87	9.28	7.95	21.67	22.42	49.74	62.46	12.16	7.73	15.96	1.26×10^{-11}
	(t_C) 0.65	4.45	5.54	7.22	6.27	7.81	6.48	29.67	27.72	8.95	7.87	27.73	5.31×10^{-19}
50×5	(t_T) 0.32	9.24	6.39	9.88	7.34	16.96	12.63	37.07	22.20	11.05	6.17	43.69	1.17×10^{-27}
	(t_C) 0.87	4.05	4.28	5.46	4.78	8.20	8.02	23.06	21.85	6.81	5.81	24.23	6.63×10^{-17}
50×10	(t_T) 0.43	8.12	8.07	9.59	6.55	13.90	8.18	31.22	13.49	9.08	7.35	56.88	6.86×10^{-34}
	(t_C) 1.17	3.56	4.70	8.04	6.03	6.32	6.24	23.06	13.33	4.94	3.89	54.21	1.10×10^{-32}
100×2	(t_T) 1.01	7.18	3.96	7.73	5.31	32.09	14.22	63.46	29.00	9.72	4.47	134.02	1.77×10^{-60}
	(t_C) 2.54	2.27	2.93	4.10	4.75	11.33	7.97	26.45	16.95	4.89	3.68	62.08	3.65×10^{-36}
100×5	(t_T) 1.27	12.26	6.31	13.73	7.75	30.22	9.43	52.22	15.52	13.75	6.10	158.25	1.11×10^{-66}
	(t_C) 3.41	3.32	3.64	3.96	5.57	11.74	6.89	24.73	11.80	4.65	4.77	81.11	9.54×10^{-44}
100×10	(t_T) 1.72	11.59	7.16	10.85	8.77	21.36	7.51	36.94	13.36	10.68	5.65	81.55	6.56×10^{-44}
	(t_C) 4.66	3.19	4.02	4.28	3.92	6.71	5.67	19.35	10.44	3.70	4.42	59.94	3.06×10^{-35}
150×2	(t_T) 2.66	10.00	5.12	9.37	4.79	48.97	22.84	124.87	47.20	11.11	6.17	219.29	1.07×10^{-79}
	(t_C) 6.23	2.11	2.73	3.02	3.74	17.96	7.84	31.30	15.59	3.85	3.09	121.48	5.81×10^{-57}
150×5	(t_T) 3.13	14.44	6.02	13.58	6.36	40.91	10.32	86.98	20.96	14.48	4.97	389.63	6.72×10^{-105}
	(t_C) 7.83	2.03	2.84	3.27	3.39	15.30	6.88	28.37	11.89	3.94	3.54	144.20	3.59×10^{-63}
150×10	(t_T) 4.44	13.07	5.97	13.20	6.18	29.71	7.16	57.34	16.60	13.40	4.86	218.44	1.54×10^{-79}
	(t_C) 11.88	2.62	3.51	3.23	3.55	10.25	4.48	23.30	7.90	3.71	3.08	164.26	4.10×10^{-68}
200×2	(t_T) 4.97	9.91	4.22	10.42	4.16	74.29	29.23	180.31	39.16	12.33	4.15	560.59	5.52×10^{-122}
	(t_C) 11.79	1.75	2.40	2.27	3.02	22.69	8.07	49.75	20.10	3.66	3.16	218.63	1.42×10^{-79}
200×5	(t_T) 6.18	17.29	5.15	16.13	6.36	48.60	9.57	107.43	25.87	14.98	4.55	466.01	3.20×10^{-113}
	(t_C) 15.48	1.70	2.38	3.07	3.29	18.07	5.84	35.29	11.41	3.06	2.79	280.05	4.14×10^{-90}
200×10	(t_T) 9.09	16.52	8.12	14.54	7.19	34.41	7.94	71.50	15.69	11.88	5.70	338.84	1.50×10^{-98}
	(t_C) 23.28	2.34	3.10	3.04	3.40	13.63	6.04	26.93	8.20	2.74	2.33	220.42	6.47×10^{-80}
平均值	(t_T) 2.38	10.73	7.88	10.78	8.38	29.06	22.96	64.34	52.31	11.23	7.10	587.55	0
	(t_C) 6.02	3.05	4.75	4.52	5.72	11.13	8.89	26.51	18.56	4.65	5.70	698.71	0

TGA; 2) 若以CGA计算时间为终止准则, 则TGA可能会由于两倍于CGA的代数而有机会搜索到更优的解. 基于以上考虑, 下面以计算时间为终止准则, 进一步检验CGA的有效性.

针对上述两种假设情况, 这里分别以TGA和CGA进化500代的时间作为5种算法的终止准则进行实验. 同样, 根据问题规模设置15组实验, 每组随机生成50个算例. 实验方法如下: 针对每个算例, 运行TGA和CGA, 记录两种算法的计算时间, 分别记为 t_T 和 t_C ; 以 t_T 为终止时间, 运行CGA、CGA_P、CGA_I和CGA_D; 以 t_C 为终止时间, 运行CGA_P、CGA_I、CGA_D和TGA; 根据式(5)和(6)计算这10次算法运行的PRD值, 其中 g^* 为10次算法运行取得的 $g(\text{Alg})$ 最小值. 实验结果见表3.

由表3可得以下结论:

1) 当计算时间较短时(终止准则为TGA进化500代时间), CGA的PRD整体均值低于其他算法, 其次为CGA_P和TGA. 有8组实验其CGA的解稍劣于CGA_P或TGA, 原因分析如下: i) 由表2和表3可知, TGA在进化500代时CGA仅进化了200代左右, 此时CGA尚处于快速优化阶段, 而TGA已趋于平稳, 因而CGA的解与TGA差距较小, 甚至劣于TGA; ii) CGA的记忆概率是对进化代数的持续累积, 当代数较少时累积的记忆较少, 对遗传算子执行效果的学习不准确, 得到的解可能会劣于基于指定概率的CGA_P.

2) 当计算时间较长时(终止准则为CGA进化500代时间), 虽然CGA_I、CGA_D和TGA的进化代数多于CGA, 但4种对比算法的求解质量依然明显劣于CGA. 这说明CGA能在较少代数内快速收敛至较优的解, 而其他对比算法在500代后的优化效果有限, 在大多数情况下, 即使放宽进化代数也很难搜索到优于CGA的解.

3) 当计算时间以CGA为基准时, 这5种算法的优化效果均明显优于以TGA计算时间为终止准则的情况. 以规模最大的实验组 200×10 为例, CGA的计算时间(25 s)与TGA(10 s)同处于秒级, 但相对于 t_T , CGA、CGA_P、CGA_I、CGA_D和TGA在以 t_C 为终止时间时PRD均值的改进倍数分别达到6.06、3.78、1.52、1.66、3.44, 优化效果均十分明显. 因此, 若以计算时间作为终止准则, 则将终止时间设置的稍长一些(如CGA的计算时间 t_C)会更适于实际应用要求.

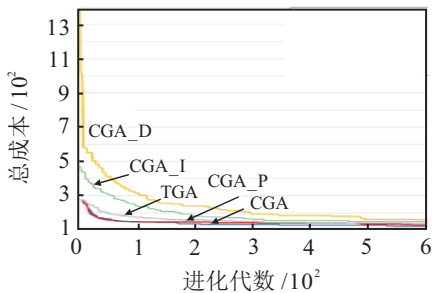
为进一步验证以上结论并检验算法的收敛性, 下

面对算法的收敛情况进行对比分析. 针对 $n \times m = 100 \times 5$ 的随机实例, 令每种算法迭代2000代, 以保证其充分收敛. 各算法取得的最小成本以及取得该成本的进化代数见表4. 可以看出: 1) CGA具有最快的收敛速度, 且取得了最低成本123; CGA_P也达到最低成本, 但比CGA多运行了1112代. 这与上述结论2)一致. 2) TGA在1106代时收敛至成本127, 此时CGA约进化到440~450代, 其成本值为129, 略高于TGA; 反之, CGA在544代收敛至成本123, 此时TGA进化到1360代左右, 早已收敛至127, 劣于123. 这与上述结论1)中的分析i)一致.

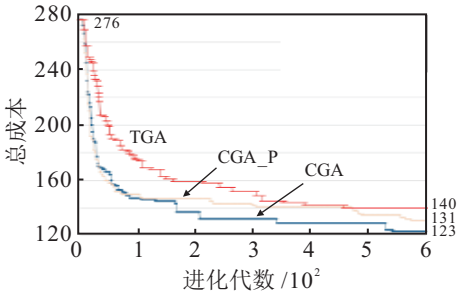
表4 算法的收敛值及其相应代数

指标	CGA	CGA_P	CGA_I	CGA_D	TGA
最小成本	123	123	144	146	127
相应代数	544	1656	1955	1341	1106
初始最低成本	276	276	474	1380	276
600代最低成本	123	131	148	160	140

为便于观察各算法的收敛情况, 图3给出了算法的收敛曲线. 为了能够清晰看出算法的收敛趋势, 且考虑到在500~600代之间出现了收敛算法, 600代以后其他算法的收敛速度也明显放缓, 因此, 图3的收敛曲线给到了600代. 由图3(a)可以看出: CGA、CGA_P和TGA基于调度规则生成的初始种群质量较优; CGA_I的初始种群虽然是随机生成的, 但采用了解码规则进行优化, 得到的最优初始解明显好于CGA_D. 以上结果验证了CGA初始解生成方法和解码规则的有效性.



(a) 5种算法收敛曲线



(b) 相同初始解的算法收敛曲线

图3 算法收敛曲线对比

为便于进一步观察,图3(b)单列了CGA、CGA_P和TGA的收敛曲线.可以看出:两种协同进化算法的收敛速度明显快于TGA,说明协同进化策略能够加快收敛;此外,CGA在进化前期虽求解质量稍逊于CGA_P,但从第81代开始优于CGA_P,且随着代数增加,差距愈发明显.这是因为随着累积记忆的增多,对算子执行效果的学习逐渐准确,记忆概率的优势也就愈发明显,这也与上述结论1)中的分析ii)一致.以上表明,CGA具有良好的收敛性和求解质量,这是初始解生成方法、解码规则、协同进化和记忆概率共同作用的结果.

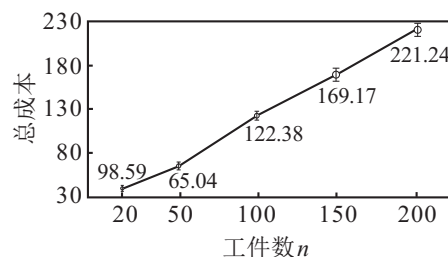
3.3 问题参数对CGA性能的影响

工件数、机器数和等待时间的限制区间是工件可拒绝的有限等待置换流水车间调度问题的3个关键参数.对于工件数的影响,本节对 $m=5$ 的5个实验组 $n=20, 50, 100, 150, 200$ 的总成本和单工件平均成本进行均值及其95%置信区间分析,如图4(a)和图4(b)所示.对于机器数的影响,对 $n=100$ 的3个实验组 $m=2, 5, 10$ 进行总成本均值及其95%置信区间分析,如图4(c)所示.对于等待时间限制的影响,本节设置4组实验,其等待时间上下限分别为 $\{\theta_{ij}^{\min}=7, \theta_{ij}^{\max}=14\}$ 、 $\{\theta_{ij}^{\min}=0, \theta_{ij}^{\max}=14\}$ 、 $\{\theta_{ij}^{\min}=7, \theta_{ij}^{\max}=+\infty\}$ 、 $\{\theta_{ij}^{\min}=0, \theta_{ij}^{\max}=+\infty\}$, $\forall i, j$,分别对应等待时间上下限约束、仅等待时间上限约束、仅等待时间下限约束和不存在等待时间约束4种情况.每组实验按照3.1节的实例参数设置随机生成100个问题规模为 100×5 的实例,共计400个实例.4组实验的总成本均值及其95%置信区间如图4(d)所示.

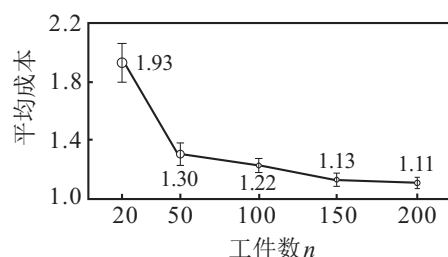
由图4(a)和图4(c)可以看出,随着工件数或机器数的增长,CGA的总成本均呈明显上升趋势.由图4(b)可以看出,随着工件数的增加,CGA中单工件平均成本是明显减少的.这说明虽然工件数增加会导致总成本上升,但好的算法是能够降低单工件平均成本的.

由图4(d)可以看出,等待时间限制会导致总成本增加,且等待时间上限的约束作用强于等待时间下限,当等待时间上下限同时存在时,总成本受到的影响最大.这是因为等待时间上限和下限均会引起工件完工时间的增加,进而拖期或Makespan超出上限的可能性增大,最终导致总成本增加.这也说明,对于存在等待时间限制(特别是上限)的车间环境,若仅根据机器能力进行工件拒绝决策,则会

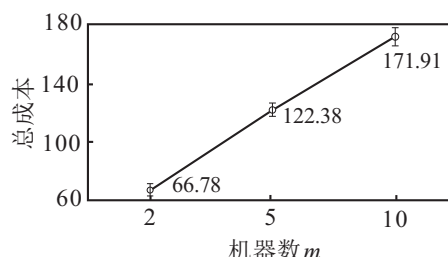
导致实际成本与预期成本偏差过大(参见实验组 $\{\theta_{ij}^{\min}=0, \theta_{ij}^{\max}=+\infty\}$ 与其他组的成本差异),因此有必要进行工件拒绝与工件调度的联合决策.



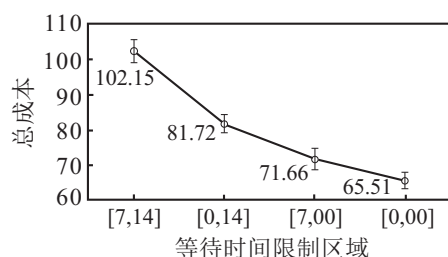
(a) 工件数变动下的总成本



(b) 工件数变动下的平均成本



(c) 机器数变动下的总成本



(d) 不同等待时间限制下的总成本

图4 不同参数值下CGA成本及其95%置信区间

4 结论

本文研究了一类工件可拒绝的有限等待置换流水车间调度问题,以最小化总拒绝成本和总拖期成本之和为优化目标,并为Makespan设定了上限约束.结合问题特征,提出了一种协同进化遗传算法,算法将染色体编码分解为两个子集,分别对应工件拒绝情况和工件调度序列,并基于调度规则生成初始种群,进而采用协同进化策略对两个子集种群进行协同式进化操作.在遗传操作方面,根据两个子集的编码特征分别为其设计遗传算子,并提出了基于记忆的动态概率设计方法;在个体选择方面,提出解码规则来进一步优化总成本和Makespan,并给出了个体评价规则

和父子锦标赛选择方法. 基于大量随机生成的实例展开数据实验, 实验结果表明, 当工件数或机器数较多、存在等待时间上下限特别是等待时间上限时, 总成本将增加, 而对于各种问题规模和等待时间限制下的算例, 本文提出的协同进化遗传算法均获得了很好的求解效果.

参考文献(References)

- [1] Fondrevelle J, Oulamara A, Portmann M C. Permutation flowshop scheduling problems with maximal and minimal time lags[J]. *Computers & Operation Research*, 2006, 33(6): 1540-1556.
- [2] 王柏琳, 李铁克, 孙彬. 基于TSP方法求解等待时间受限的置换流水车间调度[J]. *控制与决策*, 2012, 27(5): 768-772.
(Wang B L, Li T K, Sun B. TSP-based heuristic algorithm for permutation flowshop scheduling with limited waiting time constraints[J]. *Control and Decision*, 2012, 27(5): 768-772.)
- [3] Shabtay D, Gasper N. Two-machine flow-shop scheduling with rejection[J]. *Computers & Operation Research*, 2012, 39(5): 1087-1096.
- [4] Zhang L, Lu L, Li S. New results on two-machine flow-shop scheduling with rejection[J]. *J of Combinatorial Optimization*, 2016, 31(4): 1493-1504.
- [5] 谢谢, 孔祥玉, 郑勇跃. 二机流水作业带不可用区间、工件可拒绝的调度问题[J]. *沈阳大学学报*, 2014, 26(6): 473-478.
(Xie X, Kong X Y, Zheng Y Y. Two-machine flow-shop scheduling problem with unavailable interval and rejection[J]. *J of Shenyang University*, 2014, 26(6): 473-478.)
- [6] 苗翠霞, 孟凡晓. 基于退化效应的两台机器流水作业可拒绝排序[J]. *运筹学学报*, 2017, 21(2): 66-72.
(Miao C X, Meng F X. Two-machine flow-shop scheduling with deterioration and rejection[J]. *OR Transactions*, 2017, 21(2): 66-72.)
- [7] Shabtay D, Oron D. Proportionate flow-shop scheduling with rejection[J]. *J of the Operational Research Society*, 2016, 67(5): 752-769.
- [8] Li S S, Qian D L, Chen R X. Proportionate flow shop scheduling with rejection[J]. *Asia-Pacific J of Operational Research*, 2017, 34(4): 1-13.
- [9] Fondrevelle J, Oulamara A, Portmann M C, et al. Permutation flow shops with exact time lags to minimise maximum lateness[J]. *Int of J Production Research*, 2009, 47(23): 6759-6775.
- [10] Dhoub E, Teghem J, Loukil T. Lexicographic optimization of a permutation flow shop scheduling problem with time lag constraints[J]. *Int Trans on Operational Research*, 2013, 20(2): 213-232.
- [11] Hamdi I, Loukil T. Minimizing total tardiness in the permutation flowshop scheduling problem with minimal and maximal time lags[J]. *Operational Research*, 2015, 15(1): 1-20.
- [12] 张凯波, 李斌. 合作型协同演化算法研究进展[J]. *计算机工程与科学*, 2014, 36(4): 674-684.
(Zhang K B, Li B. Research overview of cooperative coevolutionary algorithms[J]. *Computer Engineering & Science*, 2014, 36(4): 674-684.)
- [13] 王凌, 沈婧楠, 王圣尧, 等. 协同进化算法研究进展[J]. *控制与决策*, 2015, 30(2): 193-202.
(Wang L, Shen J N, Wang S Y, et al. Advances in co-evolutionary algorithms[J]. *Control and Decision*, 2015, 30(2): 193-202.)
- [14] Pan Q K. An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling[J]. *Europent of Operation Research*, 2016, 250(3): 702-714.
- [15] Xing L N, Chen Y W, Wang P, et al. A knowledge-based ant colony optimization for flexible job shop scheduling problems[J]. *Applied Soft Computing*, 2010, 10(3): 888-896.
- [16] Wang X, Tang L. A machine-learning based memetic algorithm for the multi-objective permutation flowshop scheduling problem[J]. *Computers & Operation Research*, 2017, 79(3): 60-77.
- [17] Joo C M, Kim B S. Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability[J]. *Computers & Industrial Engineering*, 2015, 85(C): 102-109.
- [18] Reza Hejazi S, Saghafian S. Flowshop-scheduling problems with makespan criterion: A review[J]. *Int J of Production Research*, 2005, 43(14): 2895-2929.

(责任编辑: 李君玲)