计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(4):10−17 [doi: 10.15888/j.cnki.csa.007335]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

混和进化算法求解具有分段恶化效应的并行机调度问题①

陈海潮, 程文明, 郭　鹏, 王丽敏

(西南交通大学 机械工程学院, 成都 610031)
通讯作者: 陈海潮, E-mail: chenhaichao0607@gmail.com

摘　要: 本文提出了一种新的混合进化算法求解具有线性恶化的并行机调度问题, 目标是使总完工时间最小. 该算法采用对立策略以及最小比率优先规则生成初始种群, 并且引入种群多样度指标加快算法的收敛; 同时加入含有3-opt 扰动算子的变邻域搜索算法对遗传算法得到的结果进行局部搜索. 通过对不同规模算例的实验进行仿真, 其结果与传统 GA 和 VNS 算法相比, 效果均有所提升.

关键词: 并行机调度; 分段恶化效应; 对立学习; 遗传算法; 变邻域搜索

引用格式: 陈海潮,程文明,郭鹏,王丽敏.混和进化算法求解具有分段恶化效应的并行机调度问题.计算机系统应用,2020,29(4):10–17. http://www.c-s-a.org.cn/1003-3254/7335.html

Hybrid Evolutionary Algorithm for Solving Parallel Machine Scheduling Problems with Step-Piece Deteriorating Processing Time

CHEN Hai-Chao, CHENG Wen-Ming, GUO Peng, WANG Li-Min

(School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: A new hybrid evolutionary algorithm is proposed to solve the parallel machine scheduling problems with step-piece deteriorating processing time. The goal is minimizing the total completion time. The algorithm uses opposing strategy and Smallest Rate First (SRF) rule to generate the initial population to improve its quality, and the algorithm considers the population diversity to accelerate the convergence of the algorithm, which improves the calculation efficiency of the algorithm. At the same time, a variable neighborhood search algorithm with 3-opt perturbation operator is added to improve the quality of the results obtained by the genetic algorithm. By simulating the experiments of different scale examples, the results are improved compared with the traditional GA and VNS algorithms.

Key words: parallel machine scheduling problem; step-piece deteriorating processing time; opposition-based learning; genetic algorithm; variable neighborhood search

1　引言

在传统的调度理论中, 通常工件的加工参数事先是完全确定的. 然而在现实生产过程中, 一些工件的实际加工时间会随其开始加工的时间的延后而增长. 这种情况经常发生在许多化学和冶金过程中. 例如, 在钢轧机中, 在轧制之前将锭加热至所需温度. 加热时间取决于锭的当前温度 (这取决于锭等待的时间). 在等待期间, 晶锭冷却, 因此在炉中需要更多的加热时间. 这会导致被加工工件的处理时间的延长, 这种工件加工时间随开始加工时刻变化的现象称之为恶化效应.

本文研究的为带恶化的生产调度问题, 这个问题最早由 Gupta 等[1]和 Browne 等[2]提出, Gupta 等[1]在1988 年引入了带恶化的单机调度问题, 他们认为任务的处理时间为多项式函数, 1990 年Browne 等[2]提出工

① 收稿时间: 2019-08-09; 修改时间: 2019-09-05; 采用时间: 2019-09-23; csa 在线出版时间: 2020-04-05

件的作业处理时间是不间断的, 是具有与开始时间相关的线性函数. 对于处理时间分段线性恶化的单机调度问题, Kunnathur 和 Gupta[3]最早提出并给出了模型. Kubiak 等[4]比 Kunnathur 和 Gupta[3]多引入了恶化工期最大值 H, 如果 H 趋近无穷大, 则恶化造成的增加量就是无限的, 否则增加量就是有限的. Hosseini 和 Farahani[5]他们给定了一个规定的加工时刻 h, 提前加工会得到一个奖励时间, 延后会有一定的恶化时间从而提出了新的分段恶化模型. Wang 等[6]考虑了非线性恶化的单机调度问题. Wei 等[7]和 Wang 等[8]考虑了具有时间和资源相关处理时间的单机调度. Jiang 等[9], Wang[10], Wang 等[11,12]和 Wang 等[13]考虑了具有学习效果和恶化工作的单机调度问题.

在带恶化并行机调度问题的求解方面, Rostami 等[14]考虑了模糊环境下带工件恶化和学习效应的异构并行机调度问题, 提出了分支定界算法以最小化提前延误和最大完工时间. Eduardo 等[15]考虑了带阶梯恶化工件的等同并行机调度, 以最小化总完工时间为目标函数, 提出了两种基于集合划分的数学模型. Bahalke 等[16]考虑了具有一般线性恶化效应和顺序依赖的调整时间的单机最大完工时间最小化调度问题, 并利用遗传算法进行求解. 国内对带恶化并行机调度的问题研究较少, 轩华等[17]研究了以最小化最大完工时间为目标的不相关并行机环境下带恶化工件的车间调度问题, 假定工件在不同机器上有不同的恶化系数, 并设计了两段式编码的改进遗传算法. 郭鹏[18]针对具有阶梯恶化效应的并行机调度问题, 构建了目标总完工时间最小化的混合整数规划模型, 并设计了基于工件排序的变领域搜索算法 VNS.

就目前查阅的文献而言, 现有的研究大多假定工件具有固定的恶化率或者恶化效应为线性恶化以简化问题, 对分段线性恶化的并行机调度的研究还较少. 本文研究具有分段线性恶化的并行机调度问题, 其中工件加工时间 $p_j = a_j + b_j x_{jb}$, a_j 为工件 j 在正常情况下的基本加工时间, b_j 为工件 j 的惩罚时间, x_{jb} 为 0-1 变量, 如果工件 j 在恶化工期 h_j 之前加工则 x_{jb}=0, 否则发生恶化 x_{jb}= 1.

2 问题描述

各参数符号定义如表 1 所示, 并给出以下问题假设:

表 1 模型参数符号及其定义

符号	定义
a_j	工件 j 在正常情况下的基本加工时间
b_j	工件 j 的惩罚时间
h_j	工件 j 的恶化工期
C_k^l	机器 k 第 l 个位置加工工件的完工时间
S_k^l	机器 k 第 l 个位置加工工件开始加工的时刻
p_j	工件 j 实际加工时间
C_j	工件 j 完工时间
x_{jb}	0–1 变量, 如果工件 j 在恶化工期 h_j 之前加工则 x_{jb}=0, 否则发生恶化 x_{jb}= 1
x_{jk}^l	决策变量, 如果工件 j 在机器 k 第 l 个位置加工那么 $x_{jk}^l = 1$, 否则 $x_{jk}^l = 0$

(1) 一台机器不能同时加工多个工件, 并且一个工件也不能被多次加工.

(2) 工件没有优先权.

(3) 机器是连续可用的, 即机器不会在有工件在等待加工时空闲.

(4) 各工件正常情况下加工时间、恶化工期、恶化率已知.

基于以上假设, 问题即是对于 n 个在 m 台机器上加工, 已知正常情况下基本加工时间, 带有不同恶化工期和恶化率的工件, 寻找一个最优调度 S, 使最大完工时间这一优化目标最小.

$$\text{objective}: \quad MinZ = \sum_j^n C_j \tag{1}$$

s.t.

$$p_j = a_j + b_j x_{jb}, \forall j = 1,2,\cdots,n, x_{jb} \in 0,1 \tag{2}$$

$$\sum_{k=1}^m \sum_{l=1}^n x_{jk}^l \le 1, \forall j = 1,\cdots,n \tag{3}$$

$$\sum_{j=1}^m x_{jk}^l \le 1, \forall k = 1,\cdots,m, \forall l = 1,\cdots,n \tag{4}$$

$$S_k^l \ge C_k^{l-1}, \forall k = 1,\cdots,m \tag{5}$$

$$S_k^l \ge C_k^{l-1}, \forall k = 1,\cdots,m, \forall l = 2,\cdots,n \tag{6}$$

$$C_k^l \ge S_k^l + \sum_{j=1}^n p_j x_{jk}^l, \forall k = 1,\cdots,m, \forall l = 1,\cdots,n \tag{7}$$

$$C_j \ge C_k^l - M(1 - x_{jk}^l), \forall j,l = 1,\cdots,n, \forall k = 1,\cdots,m \tag{8}$$

$$s_j \ge S_k^l - M(1 - x_{jk}^l), \forall j,l = 1,\cdots,n, \forall k = 1,\cdots,m \tag{9}$$

$$x_{jk}^l \in \{0,1\}, \forall j,l = 1,\cdots,n, \forall k = 1,\cdots,m \tag{10}$$

$$S_k^l, C_k^l \ge 0, \forall k = 1,\cdots,m, \forall l = 1,\cdots,n \tag{11}$$

其中, 式 (1) 是目标函数, 代表所需求的最小的最大完工时间, 也就是所需要求出的最优工件排序的最后一个工件的完工时间. 式 (2) 定义了工件的实际加工时间, 如果工件在恶化工期前加工, 那么加工时间为基本加工时间, 否则增加一个惩罚时间. 式 (3) 表示同一时间机器上只有一个工件在加工的. 式 (4) 表示机器同一个加工位置只能分配一个工件, 式 (5) 表示机器第一个的工件开始加工的时间非负. 式 (6) 表示机器上 1 位置工件的开工时刻是 l–1 位置的完工时间. 式 (7) 确定了机器 k 在 l 位置工件的完工时间. 式 (8) 和式 (9) 表示工件的完工时刻和开始加工的时间. 式 (10) 表示定义 x_{jk}^{l}为 0–1 变量. 式 (11) 表明机器位置 l 处工件开始加工的时刻和其完工时间是非负的.

3 混合进化算法设计

3.1 染色体编码及适应度函数

染色体编码用自然数序列表示工件加工的顺序, 解码时, 按照染色体编码的顺序依次将工件分配到最早可用的机器上. 适应度函数即计算对应染色体调度分配后的工件完工时间总和的倒数. 进化的目标是使适应度函数最大. 例如, 对于针对表 2 算例, 并以 2 台机器 6 个工件举例.

表 2 算例数据

工件 (j)	1	2	3	4	5	6
a_j	30	9	81	95	84	47
b_j	72	85	47	15	71	28
h_j	24	37	11	58	23	40

若某一染色体编码 (根据 SRF 规则生成) 为:

2 1 5 6 3 4

则解码后甘特图如图 1 所示, 且总完工时间为 617.

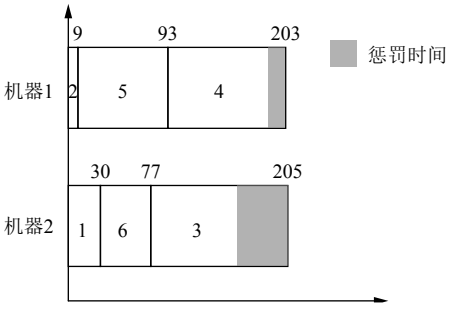

图 1 解码后甘特图

3.2 种群初始化

对立学习 (Opposition-Based Learning, OBL)[19]被应用于多种进化算法以提高性能, 在进化过程中, 不只考虑通常的解, 还考虑其对立解, 以提高收敛速度. 而在种群初始化过程中采用对立策略能够显著增强初始解的质量. 对立策略考虑当前点以及其对立点. 对立点的定义如下: 假设$p(x_1,x_2,\cdots,x_n)$是 N 维空间的一个点, $x_1,x_2,\cdots,x_n,x\in[a_i,b_i],i=1,2,\cdots,N$, 则 p 的对立点可表示为$\tilde{p}(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)$, 其中, $\tilde{x}_i=a_i+b_i-x_i$. 计算随机生成的初始种群中每个个体的对立点并进行计较, 将两者中较好的解保留. 此外根据问题的性质, 基本加工时间较小的和惩罚时间较大的工件应先加工. 因此, 初始种群中的一个解可按照 a_j/b_j 的升序排列获得. 将其称为最小比率优先 (Smallest Rate First, SRF) 规则.

3.3 选择

采用轮盘赌选择方式, 若染色体x_i应度值为$f(x_i)$个体被选中的概率$P(x_i)=f(x_i)/\sum_1^n f(x_i)$, 即适应度越高的个体被选择的概率越大.

3.4 交叉

采用部分匹配交叉. 如两个父代个体为:

p1 1 4 5 7 2 3 6 8 9
p2 2 1 8 3 4 5 9 6 7

随机选择两交叉点位并交换中间的染色体片段:

p1 1 4 | 8 3 4 5 | 6 8 9
p2 2 1 | 5 7 2 3 | 9 6 7

对于交叉点位外的区域出现的遍历重复, 根据匹配区域内的映射关系, 逐一进行交换. 比如, 对于 p1 交叉区域外重复的片段, 存在来自 p2 交叉区域内 7 到 4, 2 到 8 的映射. 即对 p1 匹配区外 4 和 8 分别以 7 和 2 替代. 对于 p2 同理.

即交叉后:

p1 1 7 | 8 3 4 5 | 6 2 9
p2 8 1 | 5 7 2 3 | 9 6 4

3.5 变异

变异采用两点交换变异, 随机选择两交叉点并交换两基因点位. 如某个体 p1 为:

p1 1 4 | 5 7 2 3 6 | 8 9

随机选择两交叉点位, 如 p1 中随机选择 5 工件与 6 工件, 变异后得到个体:

p1 1 4 | 6 7 2 3 5 | 8 9

3.6 种群多样性

在遗传算法不断进行优化迭代的过程中, 在进化的初期, 种群拥有多种个体, 而后由于适者生存, 较高适应度的个体不断被保留, 种群的多样性不断下降, 逐渐趋于成熟, 遗传算法的择优基本依赖于种群中个体的变异, 此时遗传算法的搜索效率大大降低. 因此本文在提出遗传算法中加入种群多样性指标N_p在种群多样性不足, 搜索效率不足时跳出遗传算法, 对目前得到的最优解进行变邻域搜索.

$$N_t = \frac{N}{N_p} \times 100\%$$

其中, N 是种群中基因完全不同的个体, 是种群规模.

3.7 变邻域搜索 VNS

变邻域索算法 (Variable Neighborhood Search, VNS) 是 1997 年由 Mladenovi 和 Hansen[20]提出的一种局部搜索方法, 主要包括

随机扰动、局部搜索和邻域变换等 3 个过程.

本文采用 5 种邻域结构, 单点交换, 单点插入, 两点交换, 两点插入和逆序操作. 具体操作如下所示.

单点交换. 随机选取染色体序列上的两点进行交换.

单点插入. 随机选取染色体序列上一个基因, 将其取出, 插入到另外一个随机点位中.

两点交换. 首先随机选取染色体序列上的两个基因点, 再随机选择染色体上的另外两个基因点, 两次选择的基因点位上的基因进行互换.

两点插入. 随机取出染色体序列上的两个基因点, 将其分别插入两个随机点位.

逆序操作. 随机选择染色体上的两个基因点位, 将两个基因点位之间的基因全部逆序.

其中, 由于本文染色体解的形式为自然数序列, 其中单点交换与单点插入是所有邻域结构中对解产生的最微小的扰动, 保证了每次搜索的精度, 两点交换与两点插入是单点操作的延伸, 相当于同时进行两次单点操作但不评估首次交换后的结果, 一定程度上防止陷入局部最优. 逆序操作是为了进一步提升算法的性能加入的随机搜索过程, 在以上 4 个领域结构完成后实施.

3.8 搜索扰动

为防止算法陷入局部最优, 当算法连续多代没有改善时, 对当前的解进行扰动产生下一次迭代的初始解, 以减少重复搜索的时间. 本文采用 3-opt 交换算子, 具体实施过程为随机选择染色体上三个基因点位, 将染色体分成 4 段子序列, 保持子序列的方向不变对 4 段子序列重新进行连接, 易知 4 段子序列总共有 4! =24 种排列组合方式, 本算法仅交换中间两段子序列, 首位子序列保持不变. 例如某染色体为:

10 1 4 | 6 7 | 2 3 5 | 8 9

随机选择 4,7,5 工件所在的位置, 交换中间两段子序列后得到:

10 1 4 2 3 5 6 7 8 9

3.9 混合进化算法流程

算法 1. OBGAVNS

Input: 设置种群大小N_p的值, 最大迭代次数$t_{\max}$最大连续非改进迭代数t^*_{nip}, 设置种群多样度阈值$N_{t\min}$, 染色体编码长度 n.

Step 1. 初始种群由N_p-1个随机生成的个体以及一个由 SRF 规则生成的个体组成. 并根据N_p个个体生成其相应的对立种群.

Step 2. 使用适应度函数对初始种群所有N_p个体及其对立种群进行评估并比较种群中每一个原个体与其对立点的适应度值, 若对立点更优, 则替代原个体. 以此更新初始种群P. 并记录最优个体X_{best}.

Step 3. 设代数 t=1, 连续非改进迭代数t_{nip}=0;

Step 4. 当$t_{\text{nip}}<t^*_{\text{nip}}$时

Step 5. 轮盘赌选择适应度较高的个体, 将选出的个体放入育种种群P'中; /% 选择操作 %/

Step 6. 对种群P'中的每个个体, 按照一定的交叉概率p_c进行部分匹配交叉操作. /% 交叉操作 %/

Step 7. 对交叉操作后的每个个体, 以一定的变异概率p_m进行两点交换变异. /% 突变操作 %/

Step 8. 使用适应度函数计算育种种群P'的每个个体的适应度值, 将育种种群P'中个体全部插入到原种群中并淘汰适应度低的个体, 使种群中个体数目维持在N_p. 若最优个体X_{best}得到更新, 则t_{nip}=0, 否则$t_{\text{nip}}=t_{\text{nip}}+1$.

Step 9. 评估种群多样度N_t, 若小于种群多样度阈值$N_{t\min}$, 则跳出遗传算法循环进行 VNS 搜索, 否则回到 Step 4.

VNS: 以最优个体X_{best}作为初始解

Input 最优个体X_{best}, 最大迭代次数$iter_{\max}$, 最大连续非改进迭代数t^*_{nip}.

Step 10. 令$iter$=0, t_{nip}=0.

Step 11. 进行 N 次单点交换邻域搜索, 每当邻域解有所改善, t_{nip}=0, 将邻域解视为当前解, 并重复此步骤. 若无改善$t_{\text{nip}}=t_{\text{nip}}+1$.

Step 12. 进行 N 次单点插入邻域搜索, 每当邻域解有所改善, t_{nip}=0, 将邻域解视为当前解, 并重复此步骤. 若无改善$t_{\text{nip}}=t_{\text{nip}}+1$.

Step 13. 进行 N 次两点交换邻域搜索, 每当邻域解有所改善, t_{nip}=0, 将邻域解视为当前解, 并重复此步骤. 若无改善$t_{\text{nip}}=t_{\text{nip}}+1$.

Step 14. 进行 N 次两点交换邻域搜索, 每当邻域解有所改善, t_{nip}=0, 将邻域解视为当前解, 并重复此步骤. 若无改善$t_{\text{nip}}=t_{\text{nip}}+1$.

Step 15. 进行 N 次逆序操作邻域搜索, 每当邻域解有所改善, t_{nip}=0, 将邻域解视为当前解, 并重复此步骤. 若无改善$t_{\text{nip}}=t_{\text{nip}}+1$. $iter=iter+1$

Step 16. 若$t_{\text{nip}}>\frac{1}{2}t^*_{\text{nip}}$, 对当前解进行 3-opt 扰动, 并记录当前$X_{\text{best}}$.

Step 17. 若$t_{\text{nip}}>t^*_{\text{nip}}$或者$iter>iter_{\max}$, 结束 VNS 搜索, 否则回到 Step 12 记录此时X_{best}.

Out: 算法中的最佳个体 X 及其目标函数值.

4 算例仿真及结果分析

根据数学模型, 首先采用 Gurobi 对小规模算例进行计算, 验证所提出算法的有效性, 设定最大计算时间为 1800 s. 此外, 为了验证所以出的算法性能, 同时在较大规模的算例中将其与传统 GA 算法以及 VNS 算法进行比较. 所有的算法均在处理器为 Intel(R) Core(TM)i7-9750H CPU@2.60 GHZ, 内存 8.00 GB 微机 Python 3.7 上实现.

为了确定算法的参数, 根据文献[21]提供的各项参数进行取值设置.

小规模算例工件数$n \in \{6,8,10,12\}$和机器数$m \in \{2,3\}$; 大规模算例工件数$n \in \{30,50,100\}$和机器数$m \in \{5,10,20\}$.

算例中加工参数按以下规则随机产生:

基本加工时间为[1,100]之间均匀分布的随机数;

惩罚加工时间为[1,100]之间均匀分布的随机数;

恶化工期取来自不同区间$H_1(0,D_{0.5}]$, $H_2(D_{0.5},D]$, 和$H_3(0,D]$的均匀分布的随机数, 其中$D_f = \sum_{i=1}^{f\times n} a_i/m$ $(0 \le f \le 1)$.

传统 GA 算法参数设置: 种群规模$N_p = 60$, 最大连续非改进迭代数$t^*_{\text{nip}} = 60$, 最大迭代次数$t_{\max} = 1000$, 交叉概率$p_c = 0.65$, 变异概率$p_m = 0.01$.

VNS 算法参数设置: 取最大迭代次数$iter_{\max} = 200$, 最大连续非改进迭代数$t^*_{\text{nip}} = 20$. 初始解由最小比率优先 (SRF) 规则生成.

OBGAVNS 算法参数设置: 为与传统 GA 算法以及 VNS 算法形成有效对照, 所采用的参数与对比参数大部分相同, 即种群规模$N_p = 60$, 最大连续非改进迭代数$t^*_{\text{nip}} = 60$, 最大迭代次数$t_{\max} = 1000$, 交叉概率$p_c = 0.65$, 变异概率$p_m = 0.01$. VNS 优化部分的最大迭代次数$iter_{\max} = 200$, 最大连续非改进迭代数$t^*_{\text{nip}} = 20$, 此外另取种群多样度阈值$N_{t\min} = 0.05$.

为了方便分析算法的性能, 本文将还目标值转化为相对百分比偏差 (Relative Percentage Deviation, RPD)[18], 以这个值作为参数分析的相应变量. *RPD* 值由下式决定:

$$RPD = \frac{Z(A) - Z(B)}{Z(B)} \times 100\% \tag{12}$$

式 (12) 中 $Z(A)$ 是算例计算目标值, $Z(B)$ 是算例在各种算法多次计算中所能获得的最优目标值.

小规模算例对比结果如表 3 所示, 表中数据均为计算 10 次后的平均值. 但由于 Gurobi 的性能限制, 随着算例规模的增大, Gurobi 逐渐无法在设定的时间内求问题的解, 因此表中只列出 Gurobi 在设定时间内能求得解的算例. 同时将各算法在算例中得到的最小 *RPD* 值加粗.

根据小规模算例测试结果, 所采用来对比的 OBGAVNS 算法、传统 GA 算法以及 VNS 算法是有效的并且与 Gurobi 这类数学规划求解器相比, 在计算时间上具有明显的优势. 同时 OBGAVNS 算法的计算结果与 VNS 算法以及传统 GA 算法相比, 在大部分算例中都具有最小的 RPD 值, 其中 OBGAVNS 算法的平均 RDP 值为 0.23%, 而 VNS 算法平均 RPD 值为 0.53%, 传统 GA 算法平均 RPD 值为 0.58%, 表现出较好的计算性能.

大规模算例如表 4 所示, 其中, RPDa 为所用算法计算结果平均值的相对百分比偏差, RPDm 为所用算法计算结果最小值的相对百分比偏差. 对表格中 OBGAVNS 算法在对比中最优部分进行加粗.

通过对比 3 种算法的 RPD 值可知, 所设计的 OBGAVNS 算法的所有算例的平均 RDP 值为 0.36%, 最小 RPD 值为 0.04%, VNS 算法的平均 RPD 值为 0.68%, 最小 RPD 值为 0.21%, 传统 GA 的平均 RPD 值为 3.37%, 最小 RPD 值为 1.83%. 从所有算例的平均 RPD 值的计算结果看, OBGAVNS 算法的计算性能效果优于传统 GA 与 VNS. 此外, 在大部分的算例中, OBGAVNS 算法在 3 种算法中都取得了最优的计算结果.

这是由于, 与 VNS 算法相比, OBGAVNS 算法的 VNS 部分初始解由遗传算法提供, 其初始解质量优于一般的 VNS 算法. 此外, 与一般的 VNS 算法不同, OBGAVNS 算法给 VNS 寻优部分提供的初始解带有一定的随机性, 在每次重复搜索中都会有所不同, 这为 VNS 算法提供了更大的搜索域, 此外在 OBGAVNS 算法中还加入了 3-opt 扰动算子能够使算法在 VNS 寻优部分能够跳出局部最优解.

表 3 小规模算例对比

	m	n	MIN	Gurobi		GA		OBGAVNS		VNS	
				RPD	Time (s)	*RPD*	Time (s)	*RPD*	Time (s)	*RPD*	Time (s)
H_1	2	6	617.00	**0**	1.34	**0.00**	0.63	**0.00**	0.16	**0.00**	0.02
		8	821.00	**0**	55.67	2.27	1.30	**0.00**	0.42	0.44	0.03
		10	1069.00	-	1800.00	0.53	0.97	**0.08**	0.58	0.11	0.08
		12	2643.00	-	1800.00	0.49	1.68	**0.04**	0.83	0.81	0.12
	3	6	468.00	**0**	7.30	**0.00**	0.60	**0.00**	0.19	**0.00**	0.02
		8	462.00	**0**	1530.68	**0.00**	1.36	**0.00**	0.32	**0.00**	0.04
		10	917.00	-	1800.00	**0.00**	1.99	**0.00**	0.55	**0.00**	0.07
		12	1450.00	-	1800.00	0.78	2.65	0.12	0.83	**0.00**	0.12
H_2	2	6	552.00	**0**	1.26	**0.00**	0.21	**0.00**	0.17	**0.00**	0.01
		8	735.00	**0**	81.54	**0.00**	1.17	**0.00**	0.38	**0.00**	0.02
		10	927.00	-	1800.00	**0.00**	0.43	0.08	0.44	0.60	0.02
		12	2526.00	-	1800.00	0.03	0.57	**0.00**	0.97	0.23	0.05
	3	6	453.00	**0**	6.37	**0.00**	0.26	3.31	0.19	**0.00**	0.01
		8	461.00	**0**	904.47	**0.00**	0.31	**0.00**	0.30	**0.00**	0.01
		10	804.00	-	1800.00	3.31	0.49	**0.82**	0.67	2.41	0.03
		12	1349.00	-	1800.00	0.44	0.59	**0.00**	1.02	0.13	0.06
H_3	2	6	566.00	**0**	1.60	**0.00**	0.23	**0.00**	0.20	0.02	0.01
		8	794.00	**0**	111.02	0.74	0.30	**0.04**	0.36	0.57	0.01
		10	965.00	-	1800.00	**0.00**	0.40	**0.00**	0.65	**0.00**	0.03
		12	2652.00	-	1800.00	1.84	0.54	**0.78**	1.02	1.33	0.06
	3	6	468.00	**0**	12.23	**0.00**	0.26	**0.00**	0.20	**0.00**	0.01
		8	461.00	-	1800.00	**0.00**	0.31	**0.00**	0.32	**0.00**	0.01
		10	866.00	-	1800.00	0.99	0.45	**0.00**	0.76	0.65	0.03
		12	1418.00	-	1800.00	2.39	0.60	**0.18**	1.11	0.99	0.06
平均值			1018.50	**0.00**	1125.38	0.58	0.76	**0.23**	0.53	0.35	0.04

随着算例规模的增加, 算法的计算时间也在同步增加, 所提出的 OBGAVNS 算法中由于有更多的操作算子, 因此较对比的算法而言需要更多的计算时间, 从大规模算例统计的结果看, OBGAVNS 算法平均计算时间是 3 种算法中最长的, 传统 GA 时间最短, 这是由于传统 GA 虽然具有较好的全局搜索能力, 但欠缺局部搜索能力, 因此容易陷入局部最优过早收敛. OBGAVNS 算法与 VNS 算法相比, 除了初始解生成多花费的时间外, 花费的时间主要是在扰动部分, 每次扰动后都需要花费额外的一部分时间进行寻优, 但与算法性能的提升来说, 仍在可接受的范围内, 并且随算例规模的增加, 计算时间的百分比差距也有所减少. 例如, S_1 算例中 5 台机器 30 个工件 OBGAVNS 算法计算时间为 VNS 算法的 5.52 倍, 而增加到 5 台机器 100 个工件时计算时间缩短为 2.97 倍.

为了更直观的展示算法的性能, 以算例 H1 区间 5 台机器 50 个工件为例给出迭代中的个体分布图如 2 所示. 根据程序计算结果, 其中 0 到 152 代为遗传算法部分, 个体数为种大小, 在 152 代由于到达种群多样性指标阈值因而进入变邻域搜索部分, 对单一个体进行 VNS 优化, 并在连续非改进次数达到$\frac{1}{2}t^*_{\text{nip}}$时进行 3-opt 扰动 (在图 2 中表现为适应度值变差), 开始重新搜索如此循环, 最终到达设定的最大迭代次数或连续非改进次数停止搜索, 输出迭代中最好的解.

表 4 大规模算例对比

	m	n	MIN	GA			OBGAVNS			VNS		
				RPD_a	RPD_m	Time (s)	RPD_a	RPD_m	Time (s)	RPD_a	RPD_m	Time (s)
		30	4548	5.48	1.31	3.93	**0.81**	**0.00**	15.74	2.29	1.13	2.85
	5	50	10 839	4.79	2.30	14.41	**0.87**	**0.00**	74.52	1.18	0.01	27.28
		100	39 293	6.38	3.82	63.65	0.95	0.35	696.04	**0.79**	**0.00**	233.62
		30	2173	4.23	2.39	3.54	**0.36**	**0.00**	21.81	1.51	0.41	1.90
H_1	10	50	5208	3.69	0.04	17.47	**0.89**	**0.00**	88.26	1.14	0.21	14.65
		100	16 440	6.38	3.69	94.67	**0.52**	**0.00**	862.01	0.61	0.25	401.28
		30	1464	0.08	**0.00**	3.11	**0.00**	**0.00**	23.00	**0.00**	**0.00**	1.66
	20	50	3890	3.52	1.95	20.98	**0.55**	**0.00**	175.22	1.39	0.62	19.14
		100	10 993	4.53	3.17	155.38	**0.30**	**0.05**	807.20	0.45	0.00	467.07
		30	4003	0.70	0.15	3.50	**0.05**	**0.00**	15.18	0.24	0.05	1.42
	5	50	10 070	1.25	0.60	14.04	**0.23**	**0.00**	73.44	0.39	0.22	13.60
		100	38 027	2.61	2.01	66.15	**0.22**	**0.00**	311.89	0.28	0.05	248.09
		30	2114	0.07	**0.00**	2.44	**0.00**	**0.00**	19.14	**0.00**	**0.00**	1.18
H_2	10	50	4894	2.83	1.49	12.22	**0.19**	**0.00**	103.34	0.87	0.37	15.51
		100	15 434	3.97	2.21	94.89	**0.40**	**0.00**	342.27	0.67	0.18	287.08
		30	1464	**0.00**	**0.00**	2.16	**0.00**	**0.00**	9.53	**0.00**	**0.00**	1.31
	20	50	3694	0.24	0.03	16.52	**0.00**	**0.00**	164.44	0.04	**0.00**	14.18
		100	10 625	2.52	1.72	157.71	**0.27**	0.13	535.08	0.39	**0.00**	425.80
		30	4041	1.96	0.82	4.95	**0.39**	**0.00**	14.91	0.94	0.57	1.27
	5	50	10 493	2.68	1.04	14.54	**0.28**	**0.00**	65.36	0.61	0.21	12.00
		100	42 589	9.81	6.34	54.66	**0.72**	0.58	567.74	0.96	**0.00**	286.74
		30	2209	1.09	0.45	4.98	**0.07**	**0.00**	20.04	0.40	0.09	1.25
H_3	10	50	5190	4.29	2.62	18.77	**0.43**	**0.00**	104.29	0.86	0.17	14.31
		100	17 885	7.37	4.58	107.54	**0.63**	**0.00**	627.03	1.05	0.50	343.18
		30	1464	**0.00**	**0.00**	2.44	**0.00**	**0.00**	14.14	**0.00**	**0.00**	1.42
	20	50	3799	3.69	1.87	25.01	**0.31**	**0.00**	181.25	0.77	0.37	23.49
		100	11 819	6.78	4.74	140.35	**0.35**	**0.00**	663.80	0.48	0.25	499.19
平均值			10 543	3.37	1.83	41.48	**0.36**	**0.04**	244.32	0.68	0.21	124.5

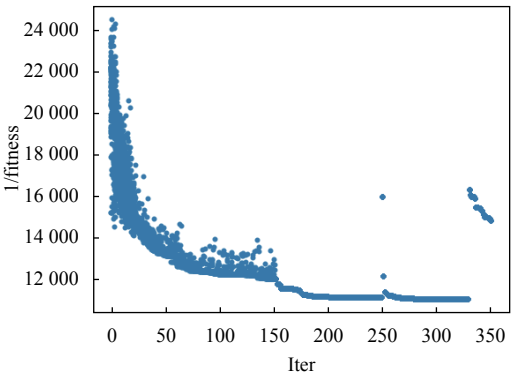

图 2 OBGAVNS 算法迭代个体分布图

5 结论与展望

本文针对具有线性恶化的并行机调度问题, 提出了一种混合进化算法 OBGAVNS, 该算法采用工件序号编码以及机器先空闲先分配的解码规则, 用对立策略以及最小比率优先规则生成初始种群以提高初始种群的质量, 并且引入种群多样度指标加快算法的收敛; 针对遗传算法局部搜索能力不强的特点, 加入带有 3-opt 扰动算子的 VNS 变邻域搜索算法对遗传算法得到的结果进行优化. 通过对不同规模的算例进行仿真实验, 结果表明了所提出 OBGAVNS 算法的有效性, 并且其性能较传统 GA 与 VNS 算法均有所提升. 接下来的研究工作将考虑更复杂的机器环境, 如异速并行机、流水车间等, 并添加考虑能耗与调整时间等约束使之更贴近生产实际.

参考文献

1 Gupta JND, Gupta SK. Single facility scheduling with nonlinear processing times. Computers & Industrial Engineering, 1988, 14(4): 387–393.

2 Browne S, Yechiali U. Scheduling deteriorating jobs on a single processor. Operations Research, 1990, 38(3): 495–498. [doi: 10.1287/opre.38.3.495]

3 Kunnathur AS, Gupta SK. Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. European Journal of Operational Research, 1990, 47(1): 56–64. [doi: 10.1016/0377-2217(90)90089-T]

4 Kubiak W, van de Velde S. Scheduling deteriorating jobs to minimize makespan. Naval Research Logistics, 1998, 45(5): 511–523. [doi: 10.1002/(SICI)1520-6750(199808)45:5<511::AID-NAV5>3.0.CO;2-6]

5 Farahani MH, Hosseini L. Minimizing cycle time in single machine scheduling with start time-dependent processing times. The International Journal of Advanced Manufacturing Technology, 2013, 64(9–12): 1479–1486.

6 Wang JB, Wang MZ. Single-machine scheduling with nonlinear deterioration. Optimization Letters, 2012, 6(1): 87–98. [doi: 10.1007/s11590-010-0253-3]

7 Wei CM, Wang JB, Ji P. Single-machine scheduling with time-and-resource-dependent processing times. Applied Mathematical Modelling, 2012, 36(2): 792–798. [doi: 10.1016/j.apm.2011.07.005]

8 Wang XR, Wang JJ. Single-machine scheduling with convex resource dependent processing times and deteriorating jobs. Applied Mathematical Modelling, 2013, 37(4): 2388–2393. [doi: 10.1016/j.apm.2012.05.025]

9 Jiang ZY, Chen FF, Kang HY. Single-machine scheduling problems with actual time-dependent and job-dependent learning effect. European Journal of Operational Research, 2013, 227(1): 76–80. [doi: 10.1016/j.ejor.2012.12.007]

10 Wang JB. A note on scheduling problems with learning effect and deteriorating jobs. International Journal of Systems Science, 2006, 37(12): 827–833. [doi: 10.1080/00207720600879260]

11 Wang JB, Hsu CJ, Yang DL. Single-machine scheduling with effects of exponential learning and general deterioration. Applied Mathematical Modelling, 2013, 37(4): 2293–2299. [doi: 10.1016/j.apm.2012.05.022]

12 Wang JB, Liu L, Wang C. Single machine SLK/DIF due window assignment problem with learning effect and deteriorating jobs. Applied Mathematical Modelling, 2013, 37(18–19): 8394–8400.

13 Wang XY, Wang JJ. Scheduling problems with past-sequence-dependent setup times and general effects of deterioration and learning. Applied Mathematical Modelling, 2013, 37(7): 4905–4914. [doi: 10.1016/j.apm.2012.09.044]

14 Rostami M, Pilerood AE, Mazdeh MM. Multi-objective parallel machine scheduling problem with job deterioration and learning effect under fuzzy environment. Computers & Industrial Engineering, 2015, 85: 206–215.

15 Lalla-Ruiz E, Voß S. Modeling the parallel machine scheduling problem with step deteriorating jobs. European Journal of Operational Research, 2016, 255(1): 21–33. [doi: 10.1016/j.ejor.2016.04.010]

16 Bahalke U, Yolmeh AM, Shahanaghi K. Meta-heuristics to solve single-machine scheduling problem with sequence-dependent setup time and deteriorating jobs. The International Journal of Advanced Manufacturing Technology, 2010, 50(5–8): 749–759.

17 轩华, 秦莹莹, 王薛苑, 等. 带恶化工件的不相关并行机调度优化. 系统仿真学报, 2019, 31(5): 919–924.

18 郭鹏. 具有分段恶化效应生产过程的智能优化调度研究[博士学位论文]. 成都: 西南交通大学, 2014.

19 Mahdavi S, Rahnamayan S, Deb K. Opposition based learning: A literature review. Swarm and Evolutionary Computation, 2018, 39: 1–23. [doi: 10.1016/j.swevo.2017.09.010]

20 Mladenović N, Hansen P. Variable neighborhood search. Computers & Operations Research, 1997, 24(11): 1097–1100.

21 Wu CC, Shiau YR, Lee LH, *et al*. Scheduling deteriorating jobs to minimize the makespan on a single machine. The International Journal of Advanced Manufacturing Technology, 2009, 44(11–12): 1230–1236.