基于离散粒子群优化的轧辊热处理调度方法①

宋继伟, 唐加福
(东北大学流程工业综合自动化教育部重点实验室, 沈阳 110004)

摘要: 以某轧辊企业铸钢分厂的轧辊热处理调度问题为实际背景, 研究了两阶段及三阶段无等待混合流水车间调度问题. 针对问题中工件加工无等待特点, 设计了分阶段实现的无等待算法; 在此基础上, 结合离散粒子群优化算法对建立的整数规划模型进行优化求解. 通过对真实数据仿真实验所得结果的比较与分析, 验证了算法的可行性和有效性, 并给出了具有实际参考价值的设备改进策略, 对生产决策者合理安排生产具有一定的指导意义.

关键词: 轧辊热处理; 无等待混合流水车间; 离散粒子群优化算法; 分阶段无等待算法
中图分类号: TP301. 6 文献标识码: A 文章编号: 1007- 9807(2010)06- 0044- 10

0 引 言

轧辊生产工艺流程由造型、冶炼、浇铸、热处理等诸多传统工艺组成. 其中, 由于热处理工艺加工周期长、对企业的生产成本起到决定性作用, 使其在轧辊生产工艺流程中处于突出重要的地位. 目前, 随着企业的生产订单不断增加, 造成热处理工艺中的机器数量相对减少, 使得热处理过程成为整个生产工艺流程的瓶颈. 因此, 在现有条件下如何优化企业的资源配置, 提高热处理工艺中机器的利用率, 已经成为降低企业生产成本的关键因素.

轧辊热处理工艺是将浇铸成的毛坯辊根据轧辊生产工艺曲线的要求, 在加热炉中不断的升温、降温, 最终形成轧辊的过程. 轧辊热处理阶段由高温炉、低温炉、地坑三道工序组成, 每一道工序均存在多台并行机器, 毛坯辊加工过程中, 在任意两道工序之间不允许有等待时间. 该问题可归结为无等待混合流水车间 (no-wait hybrid flow shop, NWHFS)调度问题.

目前, 诸多学者对混合流水车间 (HFS)调度问题进行了深入的研究[1- 6]. 该问题即使在仅有两个阶段, 且其中只有一个阶段存在并行机, 目标函数为最小化最大完工时间这种最简单的情况下, 被证明是一个 NP-hard问题[7]. 而本文研究的 NWHFS调度问题以轧辊热处理工艺为实际背景, 在 HFS调度问题的基础上增加了工件的无等待约束条件, 目前, 仅有少数文献对该问题进行了研究. 其中, 文献[8]提出了一种改进的拉格朗日算法对问题的模型进行了求解, 并通过实验证明了该算法的收敛速度优于传统的拉格朗日算法; 文献[9]研究了机器无空闲条件下的 NWHFS调度问题, 并证明了该问题为 NP-hard问题; 文献[10]采用启发式方法对问题进行了研究, 并通过实验结果的比较, 验证了算法的有效性.

但是, 从求解的质量角度考虑, 启发式算法与智能优化算法相比仍有一定的差距. 因此, 为了提高解的寻优质量, 本文应用离散粒子群优化算法 (discrete particle swarm optimization, DPSO)对 NWHFS调度问题进行了研究; 此外, 根据工件加工无等待特点, 设计了分阶段实现的无等待算法, 进而将两种算法相结合对建立的整数规划模型进行求解. 通过仿真实验对结果的比较与分析, 验证

① 收稿日期: 2008- 08- 14; 修订日期: 2010- 03- 30.
基金项目: 国家自然科学基金资助项目 (70721001; 70625001); 教育部新世纪优秀人才支持计划资助项目 (NCET- 04- 280).
作者简介: 宋继伟 (1978—), 男, 辽宁辽阳人, 博士生. Email: givensjw@ sina com

了 DPSO 算法对求解 NWHFS 调度问题的有效性; 最后, 给出了符合轧辊热处理实际生产的设备改进策略, 为生产决策者降低生产成本、合理安排生产提供了有价值的参考策略.

1　问题描述

NWHFS调度问题描述如下:

N个工件在流水线上进行 M 道工序的加工, 每一道工序至少有一台机器且至少有一道工序存在多台机器, 并且同一道工序上的每台机器有相同的处理能力. 一台机器一次至多加工一个工件, 工件可以在相应工序上的任意一台机器上加工, 且至多在一台机器上加工. 一旦工件在第一道工序上开始加工, 则此工件在任意两道工序之间不允许中断.

在进行热处理工艺加工前, 首先, 必须将不同种类的毛坯辊分成多个批次的待加工辊坯族, 且每个批次的待加工辊坯族只能由同一种类的毛坯辊组成. 其次, 根据不同类型轧辊的热处理工艺要求, 待加工的辊坯族选择热处理阶段中任意两道或三道工序进行加工 (即根据轧辊类型的不同, 在进行热处理加工时, 要求其选择不同的加工路径), 并且在加工过程中, 任意两道工序之间工件不允许等待.

由此可见, 本文研究问题的实质是在热处理阶段机器数量一定的情况下, 如何安排一天内多个批次待加工辊坯族的加工顺序, 来缩短辊坯族的加工时间, 从而有效的降低企业的生产成本. 实际生产中轧辊热处理工艺流程如图 1所示.

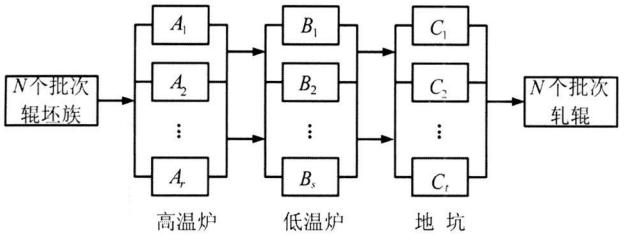

图 1　轧辊热处理工艺流程图

Fig. 1 The work-flow of roller annealing operation

2　模型建立

在热处理工艺加工过程中, 要求同一个批次的辊坯族必须放在同一个加热炉中进行加工. 基于此, 定义放在同一个加热炉中的同一个批次的辊坯族称为一个工件. 同时假设: 1)不考虑工件加工过程中某台设备出现故障对整个生产流程可能造成的影响; 2)不考虑生产过程中加热炉体积大小对整个生产流程可能造成的影响; 3)第一个工件在第一道工序上的开始加工时刻记为 1.

2.1　模型参数定义

模型中主要参数定义如下:

N —— 工件总数, 其中 $i(i = 1, 2, \cdots, N)$ 表示工件号;

M —— 工序数, 其中 $j(j = 1, 2, \cdots, M)$ 表示工序序号;

M_j —— 每道工序的机器数, 其中 $m(m = 1, 2, \cdots, M_j)$ 表示每道工序的机器序号;

K —— 时间水平, 其中 $k(k = 1, 2, \cdots, K)$ 表示工件的加工时刻, K 表示每当加工工件数量变化后, 待加工工件最差调度情况下的最大完成时间.

t_{ij} —— 工件 i在第 j道工序的加工时间;

$C_{\max}$ —— 所有工件的完成时间.

决策变量定义如下:

X_i —— 工件 i的开始加工时刻;

$$Y_{ijk} = \begin{cases} 1 & \text{工件 } i\text{在第 } j\text{道工序上的} \\ & \quad \text{第 } k\text{个时刻被加工;} \\ 0 & \text{否则.} \end{cases}$$

2.2　整数规划模型

$$\min C_{\max} = \min \max_{i=1}^{N}(X_i + \sum_{j=1}^{M} t_{ij}) \tag{1}$$

s. t

$$t_{ij} = \sum_{k=1}^{K} Y_{ijk}, \tag{2}$$

$$i = 1, 2, \cdots, N; \quad j = 1, 2, \cdots, M.$$

$$\sum_{k=1}^{K+1} | Y_{ijk} - Y_{ijk-1} | = 2, \tag{3}$$

$$Y_{ij0} = 0, Y_{ijk+1} = 0, \quad i = 1, 2, \cdots, N; \quad j = 1, 2, \cdots, M.$$

$$\sum_{k=1}^{X_i} Y_{ijk} = 1, \tag{4}$$

$$i = 1, 2, \cdots, N; \quad j = 1.$$

$$\sum_{i=1}^{N} Y_{ijk} \leqslant M_j, \tag{5}$$

$j = 1, 2, \cdots, M;\ k = 1, 2, \cdots, K.$

$$X_i \in \{1, 2, \cdots, K\}, \quad i = 1, 2, \cdots, N. \tag{6}$$

$$Y_{ijk} \in \{0, 1\}, \quad i = 1, 2, \cdots, N;\ j = 1, 2, \cdots, M;\ k = 1, 2, \cdots, K. \tag{7}$$

式(1)为模型的目标函数, 其目的是在工件加工无等待的情况下, 使最大完工时间达到最小; 式(2)和(3)表示工件 i在工序 j上加工的连续性, 且第一道工序开始前的所有时刻及最后一道工序完成后的所有时刻均为 0; 式(4)表明了变量 X_i 和 Y_{ijk} 之间的关系; 式(5)为能力约束, 保证了在任意一道工序上加工的工件数量不超过该道工序上的机器数量; 式(6)和(7)定义了决策变量的取值范围.

3　算法设计

如前所述, 对于本文研究的 NWHFS调度问题, 如何在满足机器能力约束条件下实现无等待策略是整个算法设计的难点之一. 基于此, 本文所设计的算法主要由两部分组成: (1)离散粒子群优化算法; (2)满足机器能力约束条件下的分阶段无等待算法. 下面就两种算法的特点进行详细的说明.

3.1　离散粒子群优化算法

基本的粒子群优化算法(particle swarm optimization, PSO)本质上属于迭代的随机搜索智能优化算法, 于 1995年由 Kennedy和 Eberhart 二人首次提出[11]. 目前, 广泛应用于约束优化、多目标优化、人工神经网络等领域[12]. 随着研究的深入, 一些学者开始研究如何利用 PSO 算法求解离散优化问题. Kennedy于 1997年首次提出了 DPSO 算法, 算法中的粒子采用二进制编码方式, 即粒子位置向量的每一位取值为 0或 1, 并根据仿真实验对结果的比较与分析, 验证了算法具有很好的鲁棒性[13]. 此外, Hu X 等提出了一种采用顺序编码方式的 DPSO算法[14], 并将其应用在求解非线性优化问题中.

对于离散优化问题的求解, DPSO 算法与遗传算法(genetic algorithm)、禁忌搜索(taboo search)等其它智能优化算法相比, 具有结构简单, 控制参数少等优点. 因此, 本文将应用 DPSO 算法求解 NWHFS这类复杂的离散调度问题, DPSO算法主要由顺序编码方式与更新方法及可行解调整策略两部分组成, 算法的流程如图 2 所示.

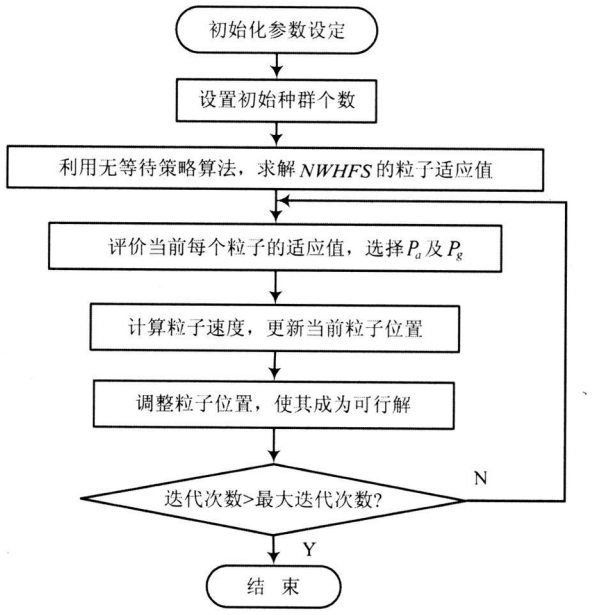

图 2　DPSO算法流程图

Fig. 2 The flow chart of DPSO

3.1.1　顺序编码方式与更新方法

本文设计的 DPSO 算法的位置编码采用顺序编码方式, 粒子的速度与位置更新方法采用基本 PSO 算法的更新方法. 其中, $X_a^b = (x_{a1}^b, x_{a2}^b, \cdots, x_{aN}^b)$ 表示第 a 个粒子在第 b次迭代时的位置, 即 NWHFS调度问题的一个解; N 表示粒子群中粒子具有的维数, 即待加工的工件数; x_{ai}^b表示第 a个粒子在第 b次迭代时的第 i个待加工工件的工件号; $V_a^b = (v_{a1}^b, v_{a2}^b, \cdots, v_{aN}^b)$ 表示第 a个粒子在第 b次迭代时的速度. 在每次迭代过程中, 粒子根据以下公式进行更新:

$$V_a^{b+1} = wV_a^b + c_1\xi(P_a^b - X_a^b) + c_2\eta(P_g^b - X_a^b) \tag{8}$$

$$X_a^{b+1} = X_a^b + V_a^{b+1} \tag{9}$$

式(8)和(9)分别为每个粒子速度和位置更新方程. 其中, $P_a = (p_{a1}, p_{a2}, \cdots, p_{aN})$ 表示第 a 个粒子自身找到的当前最佳位置, 即个体最优解; $P_g = (p_{g1}, p_{g2}, \cdots, p_{gN})$ 表示整个群体找到的当前最佳位置, 即全局最优解. w为惯性权重, 一般在 0.1~0.9之间取值, 结合本问题的特点, 通过大量的实

验比较, 证明当 w 取值为 0.6时粒子具有较好的平衡性. c_1 和 c_2 为学习因子, 通常取值为 2, 但文献[15]表示, 在 $c_1 \geqslant c_2$, 且 $c_1 + c_2 \leqslant 4$的情况下, PSO 算法会有更好的表现, 因此, 本文将 c_1, c_2 分别设定为 2.2和 1.6. ξ和 η是 0~1之间的伪随机数. 此外, 在粒子更新过程中, 粒子的最大速度 V_{max} 决定了粒子在一次迭代中的最大移动距离, 由于文中的工件总数为 N, 且粒子编码采用顺序编码方式, 因此, 粒子的初始化范围为 $[1, N]$, 这里并设定粒子的最大速度 $V_{max} = N - 1$.

3.1.2　可行解调整策略

为了保证粒子在每次迭代过程中所经过的每个位置 $X_a^b = (x_{a1}^b, x_{a2}^b, \cdots, x_{aN}^b)$ 均为顺序编码方式, 即每次更新之后的位置编码仍然为一个可行解, 本文在 DPSO 设计过程中设计了一个可行解调整策略. 下面通过一个实例对该可行解调整策略进行具体说明:

假设待加工工件的数量 $N = 6$, 随机生成粒子的一个初始位置 $X_a = (1, 4, 6, 2, 3, 5)$, 即 NWHFS调度问题的一个初始解. 在进行某次迭代后, 粒子通过位置的更新方程计算, 得到一个新的位置 $X'_a = (-1, -5, 11, 2, 8, -2)$, 很显然更新后的 X'_a不是一个顺序编码, 即不是一个可行解. 因此, 需要调整 X'_a, 使其成为一个可行解. 可行解调整的具体步骤如下所示:

步骤 **1**　令 $X'_a = (-1, -5, 11, 2, 8, -2)$中每一个小于“1”的值等于“1”, 每一个大于“6”$(N = 6)$的值等于“6”, 其余的值不变, 得到 $X'_a = (1, 1, 6, 2, 6, 1)$;

步骤 **2**　令 $X'_a = (1, 1, 6, 2, 6, 1)$中重复的值任意保留一个原值, 其余重复的值取“0”, 得到 $X'_a = (1, 0, 6, 2, 0, 0)$;

步骤 **3**　随机产生一个与 X'_a有相同维数的可行解 $X''_a = (2, 3, 1, 4, 5, 6)$. 令 X''_a中与 X'_a中相同的值取“0”, 其余值不变, 得到 $X''_a = (0, 3, 0, 4, 5, 0)$;

步骤 **4**　将 $X''_a = (0, 3, 0, 4, 5, 0)$ 中的非“0”值, 依次替换 $X'_a = (1, 0, 6, 2, 0, 0)$中的“0”值, 得到调整后的可行解 $X'_a = (1, 3, 6, 2, 4, 5)$.

通过以上的 4个步骤的调整, 粒子 $X_a = (1, 4, 6, 2, 3, 5)$ 经过一次迭代后, 得到新的粒子 $X'_a = (1, 3, 6, 2, 4, 5)$ 仍是一个可行解.

3.2　分阶段无等待算法

NWHFS调度问题的难点在于如何在满足机器能力约束的条件下, 保证在任意两道工序之间工件加工不被中断. 为了解决此难题, 本文设计了一种分阶段实现的无等待算法, 其具体思想如下:

按照工件的初始加工顺序, 在满足每道工序中机器能力约束条件下, 依次计算工件在每道工序的完成时间, 并将其完成时间按照从小到大进行排序, 以此加工顺序安排工件进行下一道工序的加工. 此外, 为了保证任意两道工序间工件加工无等待, 对于正在加工工件的这道工序, 将每次在满足该道工序机器能力约束条件下的加工工件的最早完成时间, 与下一个待加工工件在前一道工序的完成时间相比较, 取两者中较大的值作为下一个待加工工件在该道工序的开始时间, 并对后续待加工工件在前一道工序的完成时间进行相应的调整, 直到所有工件在该道工序上加工完成. 最后, 通过计算得到每个工件在该道工序的完成时间. 以此类推, 按照上述方法可依次求解待加工工件在每道工序上的完成时间. 以两阶段 NWHFS 调度问题为例, 分阶段无等待算法的流程如图 3所示.

图 3中的参数说明如下:

M_1 —— 工序 1的机器数;

M_2 —— 工序 2的机器数;

t_{n1} ——S_1 中第 n个待加工工件在工序 1上的加工时间; $(n = M_1 + 1, \cdots, N)$

t_{n2} ——S_2 中第 n个待加工工件在工序 2上的加工时间; $(n = M_2 + 1, \cdots, N)$

T_{n1} ——S_1 中第 n个待加工工件在工序 1上的完成时刻; $(n = M_1 + 1, \cdots, N)$

$T_{n1'}$ ——S_2 中第 n个待加工工件在工序 1上的完成时刻; $(n = M_2 + 1, \cdots, N)$

$T_{n'1'}$ ——S_2 中第 n'个待加工工件在工序 1上的完成时刻; $(n' = M_2 + 1, \cdots, N)$

T_{n2} ——S_2 中第 n个待加工工件在工序 2上的完成时刻. $(n = M_2 + 1, \cdots, N)$

综合上述两种算法的概述与分析, 本文在对 NWHFS调度问题进行优化求解过程中, 首先, 根据加工工件总数 N 随机产生与 DPSO算法种群数目相同的初始解 (即工件的初始加工顺序); 其次, 根据 NWAG计算每个初始解的最大完成时间

C_{max}; 再次, 根据 DPSO 算法求出此次迭代过程中所有初始解的最小 C_{max} (即局部最优解); 最后, 根据可行解调整策略将新产生的粒子调整为可行解, 并做为下一次迭代的初始解. 如此循环, 直到求得最后一次迭代的最小 C_{max} 值 (即全局最优解).

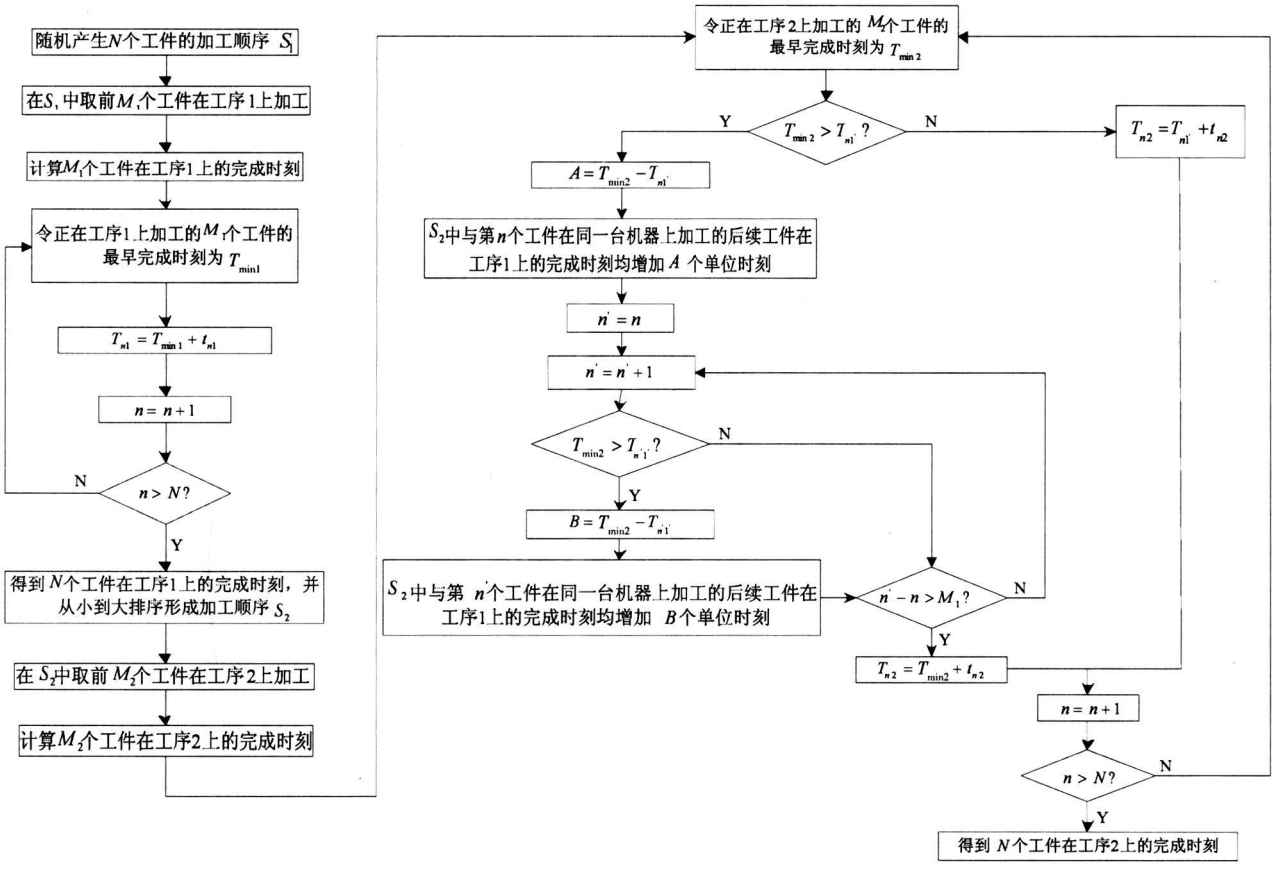

图 3　两阶段无等待算法流程图

Fig 3 The flow chart of two-stage no-wait algorithm

4　实例分析

目前, 轧辊热处理实际生产中采用的调度方法为最长加工时间 (longest total processing time, LTPT) 启发式算法, 即总加工时间最长的辊坯族先加工. 为了验证 DPSO 算法及分阶段无等待算法的可行性和有效性, 本文还引入了生产调度中常用的先到先加工 (first coming first processing FCFP) 启发式算法, 通过对三种算法的实验结果进行比较与分析, 为生产决策者提出了具有实际参考价值的设备改进策略.

4 1　实验参数设置

根据实际调研情况得知, 铸钢分厂一天中最多对 20多个批次的辊坯族进行热处理加工, 其中, 需要进行两道和三道工序加工的辊坯族数量的比例大约为 2∶1 基于此, 为了真实的反映实际生产情况, 本实验设定一天中最多有 30个批次的待加工辊坯族, 其中, 需要进行两道及三道工序加工的辊坯族最多分别为 20个批次和 10个批次, 这样足以符合并满足实际生产的需要. 每个批次辊坯族的具体加工时间如表 1、表 2所示.

本文求解 NWHFS问题的目标为最小化最大完成时间, 因此, 将所有工件的最大完成时间作为 DPSO 算法的适值函数, 即 $fitness = C_{max}$, 这样, 每次迭代过程中将适应值最小的粒子作为该代的最好解, 则最后一代得到的全局最好解即为所求的最优调度方案. 另外, 根据实例中规模的不同, DPSO 算法的参数设置有所不同. 其中, 对于三阶段的所有批次及两阶段中的前 10个批次, DPSO 算法的种群规模为 20, 最大迭代次数为 200 当两阶段的批次数取值为 10 ~ 20之间时, DPSO 算法

的种群规模为 50, 最大迭代次数为 500.

表 1　经两道工序加工的辊坯族加工时间

Table 1 Processing time for the rough roller family of two-stage

辊坯族	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
高温炉 (天)	3	4	7	7	6	6	13	15	5	4	14	13	13	6	2	11	15	9	4	6
低温炉 (天)	4	4	6	8	8	9	16	15	6	5	13	12	12	5	3	10	12	7	5	7

表 2　经三道工序加工的辊坯族加工时间

Table 2 Processing time for the rough roller family of three-stage

辊坯族	1	2	3	4	5	6	7	8	9	10
高温炉 (天)	8	7	9	13	14	15	15	6	7	8
低温炉 (天)	9	7	6	12	15	14	15	8	9	6
地　　坑 (天)	2	2	1	3	3	2	2	1	1	2

4.2　实验结果及分析

由于实际生产中低温炉的数量为 10台左右, 高温炉的数量约为低温炉的一半, 地坑的数量为 4个左右. 因此, 为了使实验更加符合实际生产情况, 本文在实验过程中严格按照实际数量及比例对每道工序的机器数量进行选取. 算法利用 VC++ 6.0实现, 在 CPU 为 Pentium Ⅳ3.0G, 内存为 1G的计算机上运行, 实验结果如表 3和表 4所示.

由表 3, 表 4可以看出, 无论是经两道工序还是三道工序加工的辊坯族, DPSO 得到的最好适应值均优于 LTPT及 FCFP求得的最好适应值.

此外, 为了进一步证明 DPSO 算法的有效性, 本文在同一台计算机上利用商业优化软件 CPLEX10.1对表 3和表 4中的小规模问题 (10个以下工件数的实例) 进行了精确求解, 并将其结果与 DPSO 算法求得的最好解进行了比较, 比较结果如表 5和表 6所示.

表 3　经两道工序加工的辊坯族实验结果

Table 3 Experimental results for the rough roller family of two-stage

实例	N × M1 × M2	算法最好适应值			最好解 (工件加工顺序)
		FCFP	LTPT	DPSO	
1	6 × 2 × 2	29	28	26	{2, 1, 5, 6, 4, 3}
2	6 × 2 × 3	27	25	23	{4, 1, 6, 5, 3, 2}
3	6 × 3 × 3	23	22	21	{1, 2, 5, 4, 6, 3}
4	8 × 2 × 3	48	43	40	{1, 7, 5, 8, 4, 3, 6, 2}
5	8 × 3 × 3	41	38	35	{1, 7, 6, 8, 5, 3, 4, 2}
6	8 × 3 × 4	41	33	31	{5, 7, 8, 4, 1, 2, 6, 3}
7	10 × 2 × 4	48	42	41	{8, 7, 2, 9, 6, 4, 5, 3, 10, 1}
8	10 × 3 × 3	41	41	37	{5, 3, 6, 8, 4, 7, 10, 9, 2, 1}
9	10 × 3 × 6	41	32	31	{7, 8, 5, 10, 1, 2, 6, 4, 3, 9}
10	12 × 4 × 4	43	43	40	{2, 8, 4, 3, 6, 5, 9, 7, 11, 12, 10, 1}
11	12 × 4 × 8	42	35	33	{7, 11, 4, 8, 12, 1, 2, 3, 6, 5, 9, 10}
12	12 × 5 × 10	37	31	31	{7, 11, 4, 1, 8, 12, 2, 3, 5, 6, 9, 10}
13	16 × 5 × 5	45	48	40	{5, 2, 4, 11, 3, 6, 9, 7, 8, 10, 14, 12, 13, 16, 15, 1}
14	16 × 6 × 6	39	42	36	{7, 3, 10, 4, 1, 2, 6, 14, 8, 11, 5, 13, 12, 9, 16, 15}
15	16 × 5 × 10	45	36	35	{1, 10, 2, 7, 13, 8, 11, 12, 3, 16, 5, 6, 4, 9, 14, 15}
16	20 × 4 × 6	56	51	50	{2, 13, 9, 5, 8, 3, 6, 17, 20, 7, 10, 12, 11, 16, 4, 18, 15, 19, 14, 1}
17	20 × 7 × 7	45	43	37	{19, 17, 2, 3, 4, 6, 9, 1, 5, 10, 8, 7, 15, 13, 11, 20, 12, 14, 16, 18}
18	20 × 5 × 10	51	41	40	{8, 7, 11, 17, 12, 13, 16, 18, 4, 6, 5, 3, 20, 9, 10, 14, 15, 2, 1, 19}

表 4　经三道工序加工的辊坯族实验结果

Table 4 Experimental results for the rough roller family of three-stage

实例	N × M1 × M2 × M3	算法最好适应值			最好解 (工件加工顺序)
		FCFP	LTPT	DPSO	
1	6× 2× 2× 4	53	49	47	{5, 2, 1, 4, 6, 3}
2	6× 2× 3× 3	53	46	44	{5, 2, 6, 4, 1, 3}
3	6× 3× 3× 2	41	41	39	{2, 1, 5, 6, 4, 3}
4	8× 2× 2× 2	68	61	59	{8, 2, 6, 5, 4, 7, 1, 3}
5	8× 2× 3× 2	63	57	53	{5, 6, 8, 4, 7, 1, 2, 3}
6	8× 3× 3× 2	55	48	46	{4, 5, 3, 8, 7, 6, 2, 1}
7	8× 3× 6× 4	53	43	43	{5, 6, 7, 4, 8, 1, 2, 3}
8	10× 2× 4× 2	63	61	60	{7, 6, 9, 5, 4, 1, 2, 8, 3, 10}
9	10× 3× 3× 2	56	55	49	{7, 6, 1, 8, 5, 4, 2, 10, 3, 9}
10	10× 3× 6× 2	53	45	45	{2, 3, 5, 6, 7, 4, 9, 1, 10, 8}
11	10× 5× 10× 2	41	36	35	{7, 4, 6, 10, 5, 1, 2, 3, 8, 9}
12	10× 5× 10× 4	41	34	33	{5, 6, 7, 10, 4, 8, 1, 3, 2, 9}

表 5　经两道工序的 **DPSO** 与 **CPLEX** 比较结果

Table 5 Comparative results of two-stage for DPSO and CPLEX

实例	N × M1 × M2	适应值		运行时间 (秒)		精确解 (工件加工顺序)
		DPSO	CPLEX	DPSO	CPLEX	
1	6× 2× 2	26	26	0.015	0.015	{2, 1, 5, 6, 4, 3}
2	6× 2× 3	23	23	0.015	0.015	{4, 1, 6, 5, 3, 2}
3	6× 3× 3	21	21	0.015	0.015	{1, 2, 5, 4, 6, 3}
4	8× 2× 3	40	40	0.043	0.450	{1, 7, 5, 8, 4, 3, 6, 2}
5	8× 3× 3	35	35	0.043	0.450	{1, 7, 6, 8, 5, 3, 4, 2}
6	8× 3× 4	31	31	0.045	0.450	{5, 7, 8, 4, 1, 2, 6, 3}
7	10× 2× 4	41	**40**	0.091	38.266	{5, 8, 7, 10, 6, 9, 4, 3, 2, 1}
8	10× 3× 3	37	37	0.091	38.266	{5, 3, 6, 8, 4, 7, 10, 9, 2, 1}
9	10× 3× 6	31	31	0.091	38.436	{7, 8, 5, 10, 1, 2, 6, 4, 3, 9}

表 6　经三道工序的 **DPSO** 与 **CPLEX** 比较结果

Table 6 Comparative results of three-stage for DPSO and CPLEX

实例	N × M1 × M2 × M3	适应值		运行时间 (秒)		精确解 (工件加工顺序)
		DPSO	CPLEX	DPSO	CPLEX	
1	6× 2× 2× 4	47	47	0.093	0.015	{5, 2, 1, 4, 6, 3}
2	6× 2× 3× 3	44	44	0.093	0.015	{5, 2, 6, 4, 1, 3}
3	6× 3× 3× 2	39	39	0.093	0.015	{2, 1, 5, 6, 4, 3}
4	8× 2× 2× 2	59	59	0.116	0.510	{8, 2, 6, 5, 4, 7, 1, 3}
5	8× 2× 3× 2	53	53	0.116	0.510	{5, 6, 8, 4, 7, 1, 2, 3}
6	8× 3× 3× 2	46	**45**	0.117	0.510	{1, 4, 5, 7, 6, 2, 3, 8}
7	8× 3× 6× 4	43	43	0.116	0.510	{5, 6, 7, 4, 8, 1, 2, 3}
8	10× 2× 4× 2	60	60	0.221	42.395	{7, 6, 9, 5, 4, 1, 2, 8, 3, 10}
9	10× 3× 3× 2	49	**48**	0.222	42.395	{1, 5, 8, 4, 9, 7, 6, 2, 10, 3}
10	10× 3× 6× 2	45	45	0.306	43.266	{2, 3, 5, 6, 7, 4, 9, 1, 10, 8}
11	10× 5× 10× 2	35	35	0.306	43.266	{7, 4, 6, 10, 5, 1, 2, 3, 8, 9}
12	10× 5× 10× 4	33	33	0.307	43.266	{5, 6, 7, 10, 4, 8, 1, 3, 2, 9}

从上述表中数据可知, 除了表 5中的实例 7及表 6中的实例 6和 9之外, DPSO求得的最好解均与 CPLEX求得的精确解相同, 而且从表中还清楚的看到, 随着问题规模的增大, CPLEX 求解时间远大于 DPSO. 这也进一步证明了 DPSO算法对于求解 NWHFS调度问题的有效性.

为了更直观的说明辊坯族在整个加工过程中所处的位置, 以及验证三种算法分别求得的最好适应值的准确性, 本文以表 3中的实例 6为例, 给出了三种算法分别求得的最好适应值对应的调度甘特图, 如图 4所示.

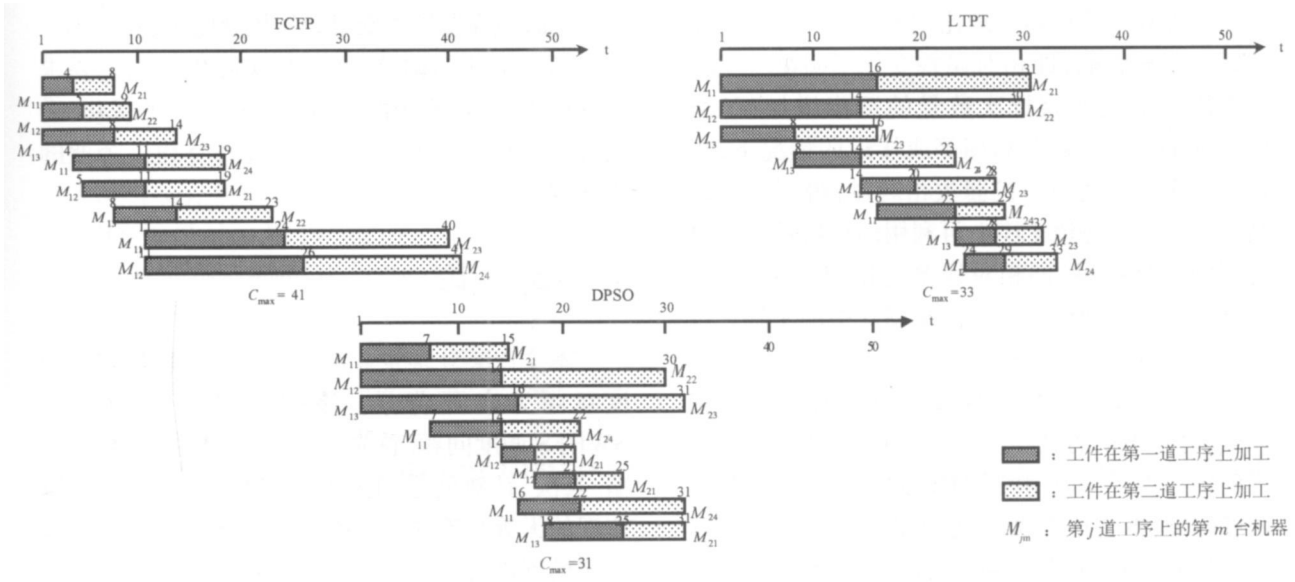

图 4　一个实例调度的甘特图

Fig. 4 A Gantt chart of the example scheduling

从图 4中可以清楚的得知辊坯族在加工过程中的详细调度过程, 其中, 包括每个辊坯族在每台机器上的具体加工时间、以及在每道工序中的哪一台机器上被加工. 此外, 甘特图中得到的辊坯族最大完成时间与算法求得的最好适应值完全相同, 进而, 验证了三种算法分别求得的最好适应值的准确性.

在上述基础上, 为了真实的反映出三种算法之间的优劣, 本文将表 3、表 4中三种算法分别求得的最好适应值进行了相对误差 (relative error, RE) 的比较, 结果如图 5、图 6所示, 图中横坐标表示实例编号, 纵坐标表示 FCFP、LTPT求得的最好适应值与 DPSO 求得的最好适应值的相对误差. 相对误差计算公式如下所示:

$$RE = \frac{FCFP\,(LTPT) - DPSO}{DPSO} \times 100\% \quad (10)$$

由图 5、图 6可以看出, FCFP与 LTPT相比, 前者最好适应值的相对误差明显偏大, 这说明从解的质量角度考虑, LTPT 优于 FCFP. 此外, 通过计算得知, 对于两道工序 18个辊坯族及三道工序 12

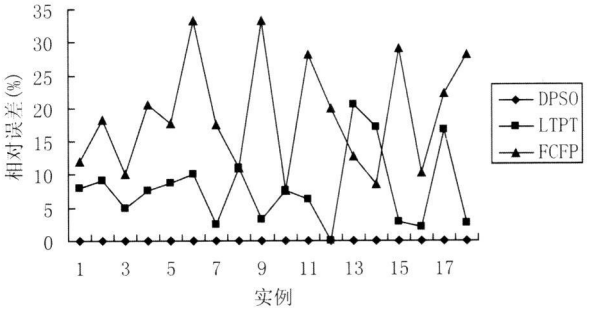

图 5　两道工序的最好适应值相对误差

Fig. 5 Relative error for the optimal fitness of two-stage

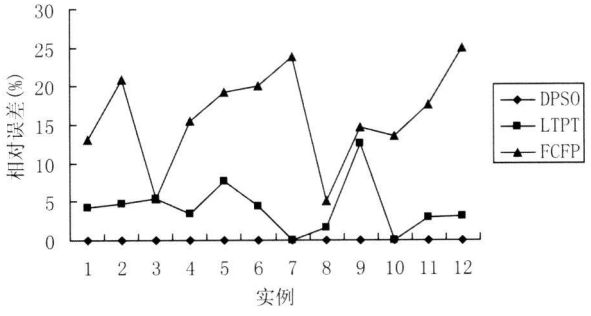

图 6　三道工序的最好适应值相对误差

Fig. 6 Relative error for the optimal fitness of three-stage

个辊坯族的加工实例, *LTPT* 与 *DPSO* 求得的最好适应值之间的平均相对误差分别为 7.8% 和 4.2%. 这说明 DPSO 与 LTPT 相比, 前者对解的质量有了进一步的提高.

另外, 通过对图 5及表 3的分析可得, 在待加工辊坯族批次数量较少的情况下 ($N < 10$), 每道工序上可利用的机器数的比例关系对最终的优化结果无明显影响, 即相对误差变化不明显. 但是, 随着待加工辊坯族批次数量的逐步增加 ($N \geqslant 10$), 在加工的机器总数相等或接近的情况下, 当每道工序上可利用的机器数相同时, 得到的优化结果明显优于每道工序上可利用的机器数不同时得到的优化结果, 即相对误差变化较为明显 (如实例 13, 14, 17所示). 同理, 从图 6中的实例 9也同样可以看到这一规律.

综上可得, 无论经两道或三道工序加工的辊坯族, 对于求得的最好适应值, DPSO 整体上优于当前实际生产中应用的 LTPT. 此外, 针对 NWHFS 调度问题, 本文还根据不同的实际工艺要求进行了大量的实验, 但由于篇幅问题, 在此不再赘述.

通过对上述实验结果的分析及总结, 对轧辊热处理生产给出如下参考策略:

(1) 当辊坯族的批次数量 $N \geqslant 10$ 的时候, 为了更加有效的提高热处理工艺的生产效率, 建议生产决策者要尽量保证可利用的低温炉和高温炉的数量相一致;

(2) 结合目前轧辊热处理过程中高温炉数量少于低温炉数量的这一实际情况, 建议生产决策者在考虑生产成本的前提下, 适当增加高温炉的数量, 或者将低温炉进行技术改进使其具有高温炉的功能, 从而, 使高温炉和低温炉的数量比例趋于平衡.

通过上述的调整, 在满足实际生产工艺要求的前提下, 企业资源配置将得到进一步的优化, 进而提高了热处理工艺生产效率, 并节约了企业的生产成本.

5 结　论

本文以某轧辊企业铸钢分厂的轧辊热处理调度问题为实际背景, 该问题可归结为一类 NWHFS 调度问题. 首先, 建立了整数规划数学模型. 其次, 针对问题的无等待特性, 设计了分阶段实现的无等待算法, 并将其与 DPSO 算法相结合对该模型进行求解. 最后, 通过仿真实验及其结果的比较分析, 验证了算法的可行性和有效性, 并根据实验分析得到的结论, 提出了符合轧辊热处理实际生产的设备改进策略, 为企业决策者合理安排生产提供了有价值的参考策略.

参 考 文 献:

[1] Hung-Tso Lin, Ching-Jong Liao. A case study in a two-stage hybrid flow shop with setup time and dedicated machines[J]. International Journal of Production Economics, 2003, 86(2): 133-143.

[2] Hua Xuan, Lixin Tang. Scheduling a hybrid flowshop with batch production at the last stage[J]. Computer & Operations Research, 2007, 34(9): 2718-2733.

[3] Chen Lu, et al. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal[J]. European Journal of Operational Research, 2007, 181(1): 40-58.

[4] Santos D L, Hunsucker J L, Deal D E. Global lower bounds for flow shops with multiple processors[J]. European Journal of Operational Research, 1995, 80(1): 112-120.

[5] Moursli O, Pochet Y. A branch-and-bound algorithm for the hybrid flowshop[J]. International Journal of Production Economics, 2000, 64(1-3): 113-125.

[6] Rajendran C, Chaudhuri D. Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan[J]. International Journal of Production Economics, 1992, 27(2): 137-143.

[7] Gupta J N D. Two-stage hybrid flowshop scheduling problem[J]. Journal of the Operational Research Society, 1988, 39(4): 359-364.

[8] 轩　华, 唐立新. 实时无等待 HFS 调度的一种拉格朗日松弛算法[J]. 控制与决策, 2006, 21(4): 376-380.

Xuan Hua, Tang Lixin. Lagrangian relaxation algorithm for real-time hybrid flowshop scheduling with no-wait in process

[J]. Control and Decision, 2006, 21(4): 376- 380. (in Chinese)

[9]Wang Zhen-bo, Xing Wen-xun, Bai Feng-shan. No-wait flexible flow shop scheduling with no-idle machines[J]. Operations Research Letters, 2005, 33(6): 609- 614.

[10]Guirchoun S, Martineau P, Billaut J-C. Total completion time minimization in a computer system with a server and two parallel processors[J]. Computers & Operations Research, 2005, 32(3): 599- 611.

[11]Kennedy J, Eberhart R C. Particle swarm optimization[C] // IEEE International Conference on Neural Networks. Piscataway: IEEE Service Center, 1995: 1942- 1948.

[12]肖人彬, 陶振武. 群集智能研究进展[J]. 管理科学学报, 2007, 10(3): 82- 96.
Xiao Ren-bin, Tao Zhen-wu. Research progress of swarm intelligence[J]. Journal of Management Sciences in China, 2007, 10(3): 82- 96. (in Chinese)

[13]Kennedy J, Eberhart R C. A discrete binary version of particle swarm algorithm[C] // Proceedings of the 1997 Conference on System, Man, and Cybernetics. Piscataway: IEEE Service Center, 1997: 4104- 4108.

[14]Hu X, Eberhart R. Sloving constrained nonlinear optimization problems with particle swarm optimization[C] // The 6th world Multiconference on Systemics, Cybernetics and Informatics. Orlando, Florida: IEEE Service Center, 2002: 203 - 206.

[15]Parsopoulos K E, Vrahatis M N. Recent approaches to global optimization problems through particle swarm optimization[J]. Natural Computing, 2002, 1(2- 3): 235- 306.

Roller annealing scheduling method based on discrete particle swarm optimization

SONG Jiwei, TANG Jiafu

Key Laboratory of Integrated Automation of Process Industry MOE, Northeastern University, Shenyang 110004, China

Abstract A two-stage and three-stage no-wait hybrid flow shop (NWHFS) scheduling problem is considered against a background of the scheduling problem of roller annealing operation in cast steel plant of a roller corporation. For the no-wait constraint between two sequential operations of a job, the no-wait algorithm of grading (NWAG) is designed. On this basis, we tried to combine NWAG with the discrete particle swarm optimization (DPSO) algorithm to solve the built integer programming model. In the simulation experiment with the real data, the feasibility and the effectiveness of the algorithm are demonstrated by the comparisons and analyses of results, and the equipment reformation strategies of actual reference value are given as well which is beneficial for the policy-maker to arrange production reasonably.

Key words annealing operation of roller; no-wait hybrid flow shop; DPSO; no-wait algorithm of grading