

# 无等待流水车间调度问题的优化

潘全科<sup>1),2)</sup> 赵保华<sup>1)</sup> 屈玉贵<sup>1)</sup>

<sup>1)</sup>(中国科学技术大学计算机科学技术系 合肥 230027)

<sup>2)</sup>(聊城大学计算机学院 山东 聊城 252059)

**摘 要** 文中研究了以生产周期为目标的无等待流水车间调度问题. 首先, 结合问题特征, 提出了一种复杂度为  $O(n)$  的快速生产周期算法. 其次, 研究了两种插入邻域结构: 基本插入邻域和多重插入邻域, 并提出了快速基本插入邻域算法和最大多重插入移动算法. 在此基础上, 将离散粒子群算法与上述两种邻域搜索算法相结合, 得到了离散粒子群优化调度算法. 第三, 根据问题生产周期的不规则性, 给出了一种通过延长工序加工时间进一步改进调度方案的方法. 最后, 仿真实验表明了所得算法的可行性和有效性.

**关键词** 无等待流水车间; 生产周期; 粒子群算法; 邻域搜索算法; 不规则性

**中图法分类号** TP18

## Heuristics for the No-Wait Flow Shop Problem with Makespan Criterion

PAN Quan-Ke<sup>1),2)</sup> ZHAO Bao-Hua<sup>1)</sup> QU Yu-Gui<sup>1)</sup>

<sup>1)</sup>(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027)

<sup>2)</sup>(School of Computer Science, Liaocheng University, Liaocheng, Shandong 252059)

**Abstract** This paper aims at finding a permutation of jobs for the no-wait flow shop (NWFS) scheduling problem with the objective of minimizing makespan. First, after investigating the property of the solution of NWFS, a speed-up method with the computational complexity  $O(n)$  is developed for calculating the makespan of a permutation. Second, a discrete particle swarm optimization algorithm (DPSO) is presented for solving this problem. Two kinds of neighborhood, insert neighborhood and multi-insert neighborhood consisting of multi-insert, which performs several inserts simultaneously in a single iteration of algorithm, are fused in the algorithm to balance the exploration and exploitation. A short-cut for insert neighborhood is also proposed. Third, an anomaly in NWFS is studied where increasing processing time of some operations may decrease makespan. Several theorems about this anomaly are reported, and an improvement procedure by slowing down some machines is designed for a permutation. Last, computational tests based on the well known benchmark suites in the literature show that the presented DPSO is effective and efficient on finding optimum or near-optimal solutions, and that slowing down some machines may result in significant reduction of the makespan yielded by DPSO.

**Keywords** no-wait flow shop; makespan; particle swarm optimization; neighborhood search; anomaly

## 1 引言

无等待流水车间(No-Wait Flow Shop, NWFS)调度问题是一类十分重要的调度问题<sup>[1-5]</sup>, 它广泛存在于炼钢、食品加工、化工和制药等领域. 已经证明机床数量大于 2 的 NWFS 是强 NP 难题<sup>[8]</sup>. 新发展起来的粒子群算法(Particle Swarm Optimization, PSO)为解决该类问题提供了新思路. 与进化算法相比, PSO 具有结构简单、容易实现、快速聚合和鲁棒性强等优势<sup>[4]</sup>. 但连续本质决定了它难以直接求解生产调度这类复杂的离散问题. 于是, 文献<sup>[5-7]</sup>结合 PSO 的优化机理和调度问题的特点, 提出了一种离散 PSO(Discrete PSO, DPSO). 该 DPSO 采用自然数编码, 在离散的解空间内执行粒子更新操作, 非常适合于调度问题的求解. 在此基础上, 本文针对 NWFS 提出了一种高性能的 DPSO 调度算法, 并结合其不规则特性, 提出了通过延长工序加工时间进一步改进调度方案的方法. 仿真试验表明了所得算法的可行性和优越性.

## 2 调度模型

### 2.1 问题描述

NWFS 可描述为: 给定  $m$  台机床和  $n$  个工件, 所有工件在各机床上的加工顺序均相同. 同时约定, 一个工件在某一时刻只能在一台机床上加工, 一台机床在某一时刻只能加工一个工件. 由于技术条件的限制, 同一工件的加工必须连续完成, 即同一工件的相邻工序之间没有等待时间. 各工序的加工时间已知. 问题是如何安排生产, 在满足上述要求的条件下得到最小生产周期.

### 2.2 生产周期的计算

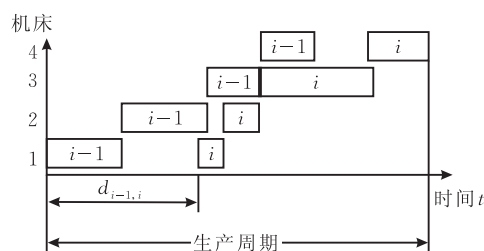
由于有同一工件的工序必须连续生产的限制, 计算 NWFS 的生产周期不同于一般流水车间调度问题. 文献<sup>[3]</sup>给出了 NWFS 生产周期的计算公式: 令  $p_{i,k}$  为工件  $i$  在机床  $k$  上的加工时间,  $\Pi = \{1, 2, \dots, i-1, i, \dots, n\}$  为一个调度,  $d_{i-1,i}$  为相邻两工件  $i-1$  和  $i$  的开工时间之差(如图 1(a))所示; 则  $d_{i-1,i}$  为

$$d_{i-1,i} = \max \left\{ \max_{2 \leq k \leq m} \left\{ \sum_{y=1}^k p_{i-1,y} - \sum_{y=1}^{k-1} p_{i,y} \right\}, p_{i-1,1} \right\} \quad (1)$$

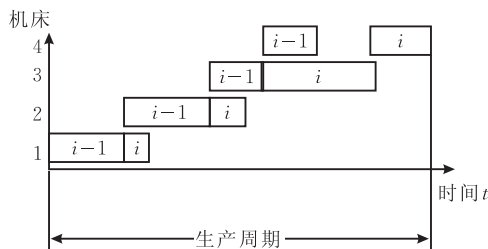
$\Pi$  的生产周期为

$$f(\Pi) = \sum_{i=2}^n d_{i-1,i} + \sum_{k=1}^m p_{n,k} \quad (2)$$

上述  $d_{i-1,i}$  和  $f(\Pi)$  的算法复杂度分别为  $O(m^2)$ ,  $O(nm^2)$ .



(a) NWFS 调度



(b) 流水车间调度

图 1 两工件的 NWFS 调度和流水车间调度

### 2.3 生产周期的快速算法

结合问题特征, 可简化  $d_{i-1,i}$  的计算. 如图 1 所示, 两个工件的 NWFS 和流水车间调度问题有相同的生产周期. 因此, 可先按照流水车间调度问题求得生产周期, 再根据连续生产的要求从后向前依次求得 NWFS 的各工序开工时间, 进而得到  $d_{i-1,i}$ . 令  $s_{i,y}$ ,  $c_{i,y}$  分别为工序  $o_{i,y}$  的开工、完工时间, 求  $d_{i-1,i}$  的算法如下:

- (1)  $s_{i-1,1} = 0$ ,  $c_{i-1,1} = p_{i-1,1}$ ; 令  $y$  从 2 到  $m$ , 分别计算  $s_{i-1,y} = c_{i-1,y-1}$ ,  $c_{i-1,y} = s_{i-1,y} + p_{i-1,y}$ .
- (2)  $s_{i,1} = c_{i-1,1}$ ,  $c_{i,1} = s_{i,1} + p_{i,1}$ ; 令  $y$  从 2 到  $m$ , 分别计算  $s_{i,y} = \max\{c_{i,y-1}, c_{i-1,y}\}$ ,  $c_{i,y} = s_{i,y} + p_{i,y}$ .
- (3) 令  $y$  从  $m-1$  到 1, 分别调整  $c_{i,y} = s_{i,y+1}$ ,  $s_{i,y} = c_{i,y} - p_{i,y}$ .
- (4)  $d_{i-1,i} = s_{i,1} - s_{i-1,1}$ .

上述算法的复杂度为  $O(m)$ . 若将得到的  $d_{i-1,i}$  代入式(2), 容易求得生产周期, 其复杂度为  $O(nm)$ . 因为  $d_{i-1,i}$  共有  $n(n-1)$  个, 为了提高算法效率, 可预先求出所有  $d_{i-1,i}$ . 这样, 在计算生产周期时,  $d_{i-1,i}$  就可视为常数. 同样的,  $\sum_{k=1}^m p_{n,k}$  也可看作常数. 于是, 式(2)的复杂度就可降低为  $O(n)$ .

3 DPSO 调度算法

3.1 PSO 算法

PSO 是 Kennedy 和 Ebrhart 于 1995 年提出的。在 PSO 中,粒子代表候选解,具有位置和速度两个特征。从初始群体出发,粒子根据自己和同伴的飞行经验不断调整位置和速度,使整个群体逐渐接近最佳解。PSO 的基本步骤为<sup>[4]</sup>:

- 1. 初始化算法参数:惯性系数、社会系数和认知系数。
- 2. 初始化粒子群:粒子、个体极值和全体极值。
- 3. 循环步 4、步 5 直到满足停止条件。
- 4. 对所有粒子执行下列操作:
  - i. 产生新粒子;
  - ii. 更新该粒子的个体极值。
- 5. 更新全体极值。
- 6. 输出全体极值。

3.2 DPSO 算法

PSO 是针对连续函数的优化提出的,其位置矢量和速度矢量的编码以及新粒子的产生策略均具有连续本质,而 NWFS 是复杂的离散问题,故需要专门设计位置矢量编码及其更新策略。

位置矢量编码

对于 NWFS 问题,最直接的编码方法就是采用位置矢量的分量代表工件,而粒子本身表示所有工件的一个排列,即一个调度。如表 1 所示。

表 1 位置矢量及对应的工件排列

粒子维数	位置向量 $X_i$	工件序列
1	3	3
2	1	1
3	2	2
4	4	4
5	6	6
6	5	5

位置更新公式。粒子群算法的实质在于粒子根据自己和同伴的飞行经验不断调整位置和速度,从而向最优位置飞行。粒子的新位置是粒子的速度、个体极值和全体极值相互作用的结果。在位置矢量编码的基础上,定义粒子更新方法如下:

$$X_j^{l+1} = c_2 \otimes g(c_1 \otimes g(\omega \otimes h_{i,i'}(X_j^l), pB_j^l), gB^l)$$

(3)

式中,  $X_j^l$  和  $pB_j^l$  分别为粒子  $j$  在第  $l$  次迭代中的位置及其个体极值;  $gB^l$  为第  $l$  次迭代中的全体极值;  $rand()$  为区间  $[0, 1]$  上的随机数;  $\omega$  为惯性系数;  $c_1$  是认知系数;  $c_2$  是社会系数。

上述位置更新公式由 3 部分构成:

第 1 部分为

$$E_j^l = \omega \otimes h_{i,i'}(X_j^l) = \begin{cases} h_{i,i'}(X_j^l), & rand() < \omega \\ X_j^l, & rand() \geq \omega \end{cases},$$

表示粒子对自身飞行速度的思考。其中,  $h_{i,i'}(X_j^l)$  表示粒子的速度。它的实现方法为:先产生两个不同的随机数  $i$  和  $i'$ , 然后交换位置矢量的第  $i, i'$  分量。

第 2 部分为

$$F_j^l = c_1 \otimes g(E_j^l, pB_j^l) = \begin{cases} g(E_j^l, pB_j^l), & rand() < c_1 \\ E_j^l, & rand() \geq c_1 \end{cases},$$

表示粒子根据  $pB_j^l$  调整位置。其中  $Y_j^l = g(E_j^l, pB_j^l)$  的实现方法为:先从  $E_j^l$  中随机抽取一段,放在  $pB_j^l$  的前面或后面;再从  $pB_j^l$  删除重复工件。

第 3 部分为

$$X_j^{l+1} = c_2 \otimes g(F_j^l, gB^l) = \begin{cases} g(F_j^l, gB^l), & rand() < c_2 \\ F_j^l, & rand() \geq c_2 \end{cases},$$

表示粒子根据  $gB_j^l$  调整位置。

对  $gB^l$  的局部搜索

利用粒子群的优化原理,采用上述离散位置矢量编码并在离散域内执行粒子更新操作的算法就称为 DPSO。该 DPSO 算法虽然收敛快,但求解质量不高。分析算法原理知,DPSO 的信息传递是单向的:  $gB^l$  将信息传给其它粒子,带动其它粒子在较好的区域搜索。因此,可通过提高  $gB^l$  的局域搜索能力改进算法性能。同时,为了防止算法陷入局部最优,需要采取如下措施:

(1) 先对  $gB^l$  执行扰动,再对扰动结果执行邻域搜索算法。

(2) 采用概率接收准则接收所得结果,即如式下成立,则用所得结果取代  $gB^l$ 。

$$rand() \leq \min\{1, \exp(-\Delta/t)\}$$

(4)

式中,  $\Delta$  为邻域搜索算法的输出结果与  $gB^l$  的目标值之差,  $t$  是温度参数。

DPSO 的实现步骤

- 1. 初始化算法参数:惯性系数、社会系数、认知系数,扰动长度和温度参数。
- 2. 初始化粒子群:粒子、个体极值和全体极值。
- 3. 循环步 4、5 和 6 直到满足停止条件。
- 4. 对所有粒子执行下列操作:
  - i. 采用式(3)产生新粒子;
  - ii. 更新该粒子的个体极值。
- 5. 更新全体极值。
- 6. 对全体极值执行局部搜索算法。
- 7. 输出全体极值。

3.3 邻域搜索算法

研究表明,对于作业调度这类复杂问题,插入邻域搜索算法性能较高<sup>[3]</sup>。本文采用两种插入邻域算

法,DPSO的每次迭代随机选择一种执行.

3.3.1 插入邻域及其搜索算法

**定义 1.** 在工件排列中随机选择不同的两位置  $i,i'$ ,把位置  $i$  的工序插入位置  $i'$ ,称为插入移动,记为  $Insert(i,i')$ .由该移动得到的新排列称为原排列的邻居.所有这样的邻居构成插入邻域.

插入邻域搜索算法如下.

1. 令  $i'=1$ .从当前排列中取出第  $i'$  个工件,将其分别插入另外  $n-1$  个位置,并计算生产周期.
2.  $i'=i'+1$ .如  $i'\leq n$ ,返回步 1.
3. 如果邻域内的最优排列优于当前排列,则用其置换当前排列,并返回步 1;否则算法结束.

上述邻域算法的复杂度为  $O(n^3)$ . 为了提高效率,根据邻域解的相似性可降低其复杂度. 设含有  $n-1$  个工件的一个排列为  $\{1,2,\cdots,i-1,i,\cdots,n-1\}$ ,其生产周期为  $f$ . 若将工件  $i'$  插入位置  $i$  后(见图 2),则所得排列的生产周期为

$$f' = \begin{cases} f + d_{i',1}, & i = 1 \\ f + d_{i',i} + d_{i-1,i'} - d_{i-1,i}, & 1 < i < n \\ f + d_{n-1,i'} - \sum_{j=1}^m p_{n-1,j} + \sum_{j=1}^m p_{i',j}, & i' = n \end{cases} \quad (5)$$

采用式(5)计算生产周期,可将插入邻域搜索算法的复杂度降为  $O(n^2)$ .

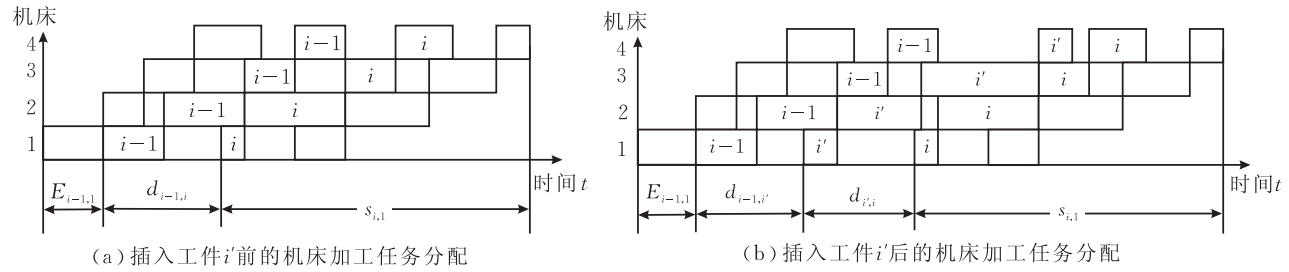


图 2 插入工件  $i'$  前后总流水时间的变化

3.3.2 块结构及其性质

**定义 2.** 在同一机床上,4 个或 4 个以上的连续加工的工序称为块. 除去块的第一工序和最后工序后剩余的工序,称为块内工序.

**定理 1.** 改变块内工序所对应工件的次序不会缩短生产周期.

证明. 如图 3 所示,工序  $o_{i-2,k}, o_{i-1,k}, o_{i,k}, o_{i+1,k}$ , 和  $o_{i+2,k}$  构成块. 生产周期  $L' = l_1 + l_2 + l_3$ . 改变块内工序对应工件的加工顺序后,  $l_1$  和  $l_3$  不变,而  $l_2$  不可能减小,故  $L'$  不会减小.

根据上述定理,可在插入邻域算法中除去不必要的移动,进一步提高算法效率.

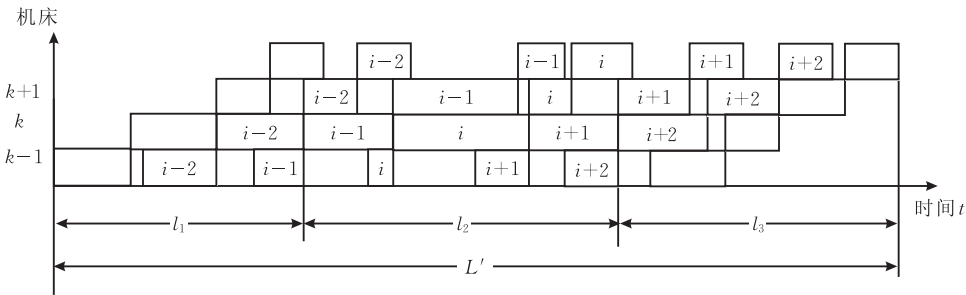
3.3.3 多重插入移动

**定义 3**<sup>[3]</sup>. 两个  $Insert$  移动 ( $v_1 = Insert(x_1, y_1)$  和  $v_2 = Insert(x_2, y_2)$ ) 是相互独立的,如果  $x_1, y_1$  与  $x_2, y_2$  不相邻,即满足下列 3 个条件之一:

(1)  $\begin{cases} \max(x_1, y_1) + 1 < \min(x_2, y_2) \\ \text{or} \\ \max(x_2, y_2) + 1 < \min(x_1, y_1) \end{cases};$

(2)  $\begin{cases} \min(x_1, y_1) + 1 < \min(x_2, y_2) \wedge \max(x_2, y_2) + 1 < \max(x_1, y_1) \\ \text{or} \\ \min(x_2, y_2) + 1 < \min(x_1, y_1) \wedge \max(x_1, y_1) + 1 < \max(x_2, y_2) \end{cases};$

(3)  $\begin{cases} \min(x_1, y_1) + 1 < \min(x_2, y_2) \wedge \min(x_2, y_2) + 1 < \max(x_1, y_1) \wedge \max(x_1, y_1) + 1 < \max(x_2, y_2) \\ \text{or} \\ \min(x_2, y_2) + 1 < \min(x_1, y_1) \wedge \min(x_1, y_1) + 1 < \max(x_2, y_2) \wedge \max(x_2, y_2) + 1 < \max(x_1, y_1) \end{cases}.$



容易证明,两个独立的移动具有下列性质.

**性质 1.** 设  $v_1$  和  $v_2$  是工件排列  $\Pi$  的两个独立移动,且  $\Pi$  经过移动  $v_1, v_2$  分别得到  $\Pi_{v_1}, \Pi_{v_2}$ . 若对  $\Pi$  同时执行这两个移动得到  $\Pi_{v_1+v_2}$ , 则有下列式成立

$$\Delta = \Delta_1 + \Delta_2 \quad (6)$$

式中,  $\Delta_1 = f(\Pi_{v_1}) - f(\Pi)$ ,  $\Delta_2 = f(\Pi_{v_2}) - f(\Pi)$ ,  $\Delta = f(\Pi_{v_1+v_2}) - f(\Pi)$ .

**定义 4**<sup>[3]</sup>. 由若干个互相独立的插入移动一次同时执行而构成的移动称为多重插入移动.

**推论 1.** 多重插入移动对生产周期的改变等于构成该多重移动的所有独立移动对生产周期的改变之和.

**定义 5**<sup>[3]</sup>. 记  $PZ$  为工件排列  $\Pi$  的所有使生产周期减小的插入移动集合,  $PX$  是由  $PZ$  的多个相互独立的插入移动构成的子集, 则  $PX$  中的移动可构成多重插入移动. 如果把  $PZ$  中的任一其他移动加入  $PX$  中, 使  $PX$  中的移动不再相互独立, 则称该多重插入移动称为最大多重插入移动. 显然, 多重移动对生产周期的改进大于简单移动, 这样对工件排列执行一次搜索就能使之较快地移动到较好区域.

令  $PZ = \{v_1, v_2, \dots, v_h\}$ , 下列算法可得到一个最大多重移动.

1. 在  $PZ$  中求与  $v_e$  不相互独立的移动的个数  $q(v_e)$ ,  $e=1, 2, \dots, h$ .
2. 求  $v_{e'} \in PZ$  且  $q(v_{e'}) = \max_{e=1, 2, \dots, h} q(v_e)$ .
3. 若  $q(v_{e'}) > 0$ , 令  $PZ = PZ - \{v_{e'}\}$ , 转步 2.
4. 若  $q(v_{e'}) = 0$ , 算法结束, 则  $PZ$  中的移动构成一个最大多重移动.

### 3.4 DPSO 的初始化

初始群体对算法的性能有较大影响, 好的初始群体能加快搜索效率和提高求解质量<sup>[4]</sup>. 本文产生初始群体的算法如下:

1. 令  $j=0$ .
2.  $Z = \{z_1, z_2, \dots, z_n\}$  表示所有工件的集合.
3. 令  $i=0, \pi_i = z_j, Z = Z - \{z_j\}, \Pi = \{\pi_i\}$ .
4. 取  $z_x = \arg \min_{z_x \in Z} d_{\pi_i, z_x}$ , 令  $\pi_{i+1} = z_x, Z = Z - \{z_x\}, \Pi = \Pi \cup \{\pi_{i+1}\}$ .
5.  $i=i+1$ , 重复步 4 直到  $Z$  为空, 则得到一个工件排列.
6. 如  $j < n$ , 则  $j=j+1$ , 转步 2.

### 3.5 DPSO 的参数设计

DPSO 有 5 个参数: 惯性系数、社会系数、认知系数、扰动长度和温度系数. 以随机产生的 360 个实例为试验数据来确定算法参数. 先通过参数变化对算法性能的影响确定主要参数, 再确定主要参数的取值. 最后将各参数确定为: 惯性系数为 0.2, 社会

系数为 0.8、认知系数 0.8、扰动长度为 12、温度系数为 0.8.

## 4 调度结果的改进

在 NWFS 中, 延长某些工序的加工时间可缩短生产周期(如图 4 所示). 而延长工序加工时间又可以通过降低机床的加工速度来实现, 这在很多生产过程中是允许的. 因此, 可通过降低机床速度进一步改进调度方案<sup>[8-9]</sup>.

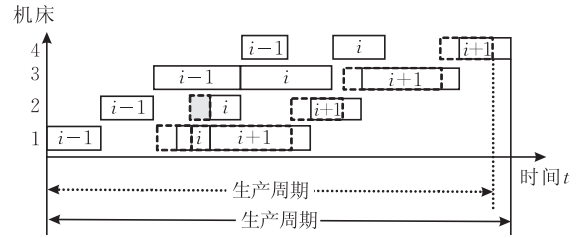


图 4  $p_{i,2}$  增加后生产周期的减小

**定义 6.** 同一机床上连续加工的两个工序  $o_{i-1,k}$  和  $o_{i,k}$  称为结, 记为  $(o_{i-1,k}, o_{i,k})$ .

**定理 2.** 设工件  $i-1$  和  $i$  之间只存在结  $(o_{i-1,b}, o_{i,b})$ , 工件  $i$  和  $i+1$  之间只存在结  $(o_{i,a}, o_{i+1,a})$ , 若存在整数  $k$ , 使得  $a < k < b$ , 则增加  $p_{i,k}$  可使  $c_{i+1,m}$  减小.

证明. 如图 5 所示,

$$c_{i+1,m} = s_{i,a} + p_{i,a} + \sum_{y=a}^m p_{i+1,y},$$

$$s_{i,a} = \begin{cases} s_{i,b} + \sum_{y=b}^{a-1} p_{i,y}, & a > b \\ s_{i,b} - \sum_{y=b-1}^a p_{i,y}, & a < b-1 \\ s_{i,b} - p_{i,a}, & a = b-1 \\ s_{i,b}, & a = b \end{cases}$$

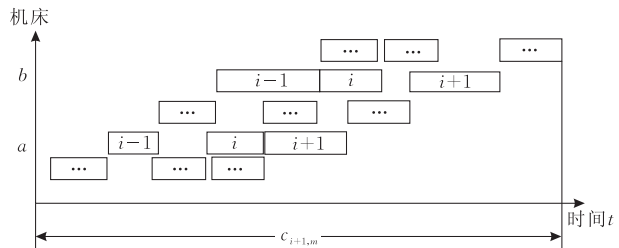


图 5 NWFS 的机床加工任务分配

由于结  $(o_{i-1,b}, o_{i,b})$  的存在,  $s_{i,b}$  保持不变. 当且仅当  $a < b-1$  时, 增加  $p_{i,k}$  ( $a < k < b$ ) 才能减小  $c_{i+1,m}$ .

**定理 3.** 设工件  $i-1$  和  $i$  之间只存在结  $(o_{i-1,b}, o_{i,b})$ , 工件  $i$  和  $i+1$  之间只存在结  $(o_{i,a}, o_{i+1,a})$ , 且存在整数  $k$ , 使得  $a < k < b$ , 则  $p_{i,k}$  的最大增加量为  $\Delta = \min\{\min_{y \leq k}(s_{i,y} - c_{i-1,y}), \min_{y \geq k}(s_{i+1,y} - c_{i,y})\}$ .

证明.

(1) 设  $p_{i,k}$  增加  $\Delta$ , 则工件  $i$  的各工序开工时间和完工时间分别变成:

$$\begin{aligned} s'_{i,y} &= \begin{cases} s_{i,y}, & y > k \\ s_{i,y} - \Delta, & y \leq k \end{cases}, \\ c'_{i,y} &= \begin{cases} c_{i,y}, & y \geq k \\ c_{i,y} - \Delta, & y < k \end{cases}. \end{aligned}$$

当  $y \leq k$  时,  $o_{i,y}$  开工提前, 但需满足:

$$s'_{i,y} - c_{i-1,y} \geq 0,$$

即  $s_{i,y} - c_{i-1,y} \geq \Delta$ .

(2)  $p_{i,k}$  增加  $\Delta$  后, 工件  $i+1$  的各工序开工时间变为

$$s'_{i+1,y} = s_{i+1,y} - \Delta,$$

$$\text{而 } s'_{i+1,y} - c'_{i,y} = \begin{cases} s_{i+1,y} - c_{i,y} - \Delta, & y \geq k \\ s_{i+1,y} - c_{i,y}, & y < k \end{cases}.$$

当  $y \geq k$  时,  $o_{i,y}$  与  $o_{i+1,y}$  的相对位置发生了变化. 其变化应满足:

$$s'_{i+1,y} - c'_{i,y} \geq 0,$$

即  $s_{i+1,y} - c_{i,y} \geq \Delta$ .

故  $p_{i,k}$  的最大增加为

$$\Delta = \min\{\min_{y \leq k}(s_{i,y} - c_{i-1,y}), \min_{y \geq k}(s_{i+1,y} - c_{i,y})\}.$$

**推论 2.** 在定理 3 的条件下,  $c_{i+1,m}$  的减小为  $\Delta = \min\{\min_{y \leq k}(s_{i,y} - c_{i-1,y}), \min_{y \geq k}(s_{i+1,y} - c_{i,y})\}$ .

**定理 4.** 设工件  $i-1$  和  $i$  之间存在多个结, 机床序号最小的结记为  $(o_{i-1,b_{\min}}, o_{i,b_{\min}})$ . 工件  $i$  和  $i+1$  之间也存在多个结, 机床序号最大的结记为  $(o_{i,a_{\max}}, o_{i+1,a_{\max}})$ . 若存在  $k$  ( $a_{\max} < k < b_{\min}$ ), 则增加  $p_{i,k}$  可减小  $c_{i+1,m}$ .

证明. 工件  $i$  的各工序开工时间可表达为

$$s_{i,y} = \begin{cases} s_{i,b_{\min}} + \sum_{z=b_{\min}}^{y-1} p_{i,z}, & y \geq b_{\min} \\ s_{i,b_{\min}} - \sum_{z=y}^{b_{\min}} p_{i,z}, & y < b_{\min} \end{cases}.$$

可见, 只有当  $b_{\min} > k > y$  时,  $p_{i,k}$  增加才能减小  $s_{i,y}$ .

再由  $c_{i+1,m} = s_{i+1,m} + p_{i+1,m} = s_{i+1,a_{\max}} +$

$\sum_{z=a_{\max}}^m p_{i+1,z} = s_{i,a_{\max}} + p_{i,a_{\max}} + \sum_{z=a_{\max}}^m p_{i+1,z}$  知, 要减小  $c_{i+1,m}$ , 则需要减小  $s_{i,a_{\max}}$ . 而当  $k < a_{\max}$  时,  $p_{i,k}$  增加不可能使  $s_{i,a_{\max}}$  减小. 故只有存在整数  $k$  满足  $a_{\max} < k < b_{\min}$ , 当  $p_{i,k}$  增加时, 才能减小  $c_{i+1,m}$ .

**推论 3.** 设工件  $i-1$  和  $i$  之间机床序号最小的结为  $(o_{i-1,b_{\min}}, o_{i,b_{\min}})$ ,  $i$  和  $i+1$  之间机床序号最大的结为  $(o_{i-1,a_{\max}}, o_{i,a_{\max}})$ , 且存在整数  $k$  ( $a_{\max} < k < b_{\min}$ ). 如果通过增加某一工序的加工时间使  $c_{i+1,m}$  减小, 则最大减小量为  $\Delta c_{i+1,m} = \max_{a_{\max} < k < b_{\min}} \{\min_{y \leq k}(s_{i,y} - c_{i-1,y}), \min_{y \geq k}(s_{i+1,y} - c_{i,y})\}$ .

**推论 4.** 令  $\Delta_i = \{\lambda_{i,1}, \dots, \lambda_{i,a_{\max}}, \dots, \lambda_{i,k}, \dots, \lambda_{i,b_{\min}-1}\}$ , 其中  $\lambda_{i,k} = s_{i,k} - c_{i-1,k}$ ;

$\Delta_{i+1} = \{\lambda_{i+1,a_{\max}+1}, \dots, \lambda_{i+1,k}, \dots, \lambda_{i+1,b_{\min}}, \dots, \lambda_{i+1,m}\}$ , 其中  $\lambda_{i+1,k} = s_{i+1,k} - c_{i,k}$ ; 则

$$\Delta c_{i+1,m} = \max_{a_{\max} < k < b_{\min}} \{\min\{\lambda_{i,1}, \dots, \lambda_{i,a_{\max}}, \dots, \lambda_{i,k}, \lambda_{i+1,k}, \dots, \lambda_{i+1,b_{\min}}, \dots, \lambda_{i+1,m}\}\}.$$

求  $\Delta c_{i+1,m}$  的算法如下.

1. 令  $\beta$  为集合  $\Delta_i \cup \Delta_{i+1}$  中元素的最小值.
2. 若在集合  $\Delta_i$  中存在  $\lambda_{i,k} = \beta$  (若存在多个这样的元素, 取机床序号最小的元素), 则删去集合  $\Delta_i$  中所有机床序号大于或等于  $k$  的元素.
3. 若在集合  $\Delta_{i+1}$  中存在  $\lambda_{i+1,k} = \beta$  (若存在多个这样的元素, 取机床序号最大的元素), 则删去集合  $\Delta_{i+1}$  中所有机床序号小于或等于  $k$  的元素.
4. 若集合  $\Delta_i \cup \Delta_{i+1}$  中元素个数小于  $m+1$ , 则  $\Delta c_{i+1,m} = \beta$ , 算法结束; 否则转步 1.

## 5 算法性能评价

为了评价算法性能, 采用 23 个典型调度问题<sup>[10-11]</sup>做测试数据, 在处理器为 PIV 3.0GHz、内存为 512MB 的 PC 机上做仿真实验. 每个算例独立运行 20 次, 所得计算结果如表 2 所示. 在表 2 中, “相对偏差”指的是相对 RAJ 算法所得解的偏差, “改进结果”系指延长工序加工时间后平均偏差的改进结果. TS、TS+M 和 TS+MP 是文献[3]提出的解决 NWFS 的 3 种禁忌搜索算法, 其计算结果为在 P II 1000MB 的 PC 机上的执行结果. HPSO 是文献[4]提出的解决 NWFS 的混合粒子群算法, 其计算结果为在处理器为 PIV 2.2GHz、内存为 512MB 的 PC 机上的运行结果.

表 2 各算法所得结果比较

(时间单位:s)

调度	问题	RAJ 的结果		TS 的结果		TS+M 的结果		TS+MP 的结果		HPSO 的结果		DPSO 的结果		
名称	$m \times n$	生产周期	偏差	时间	偏差	时间	偏差	时间	平均偏差	平均时间	平均偏差	均方差	平均时间	改进结果
Rec01	5×20	1590	-4.03	0.2	-3.96	0.2	-3.96	0.2	-3.39	3.9	-4.03	0.00	0.00	-4.25
Rec03	5×20	1457	-6.59	0.2	-6.59	0.2	-6.59	0.2	-6.15	4.8	-6.59	0.00	0.00	-6.73
Rec05	5×20	1637	-7.39	0.2	-7.64	0.2	-7.70	0.2	-7.15	4.1	-7.70	0.00	0.02	-7.83
Rec07	10×20	2119	-3.63	0.2	-3.63	0.2	-3.63	0.2	-3.11	6.6	-3.63	0.00	0.00	-8.40
Rec09	10×20	2141	-4.62	0.2	-4.58	0.2	-4.58	0.2	-4.26	6.7	-4.62	0.00	0.01	-6.91
Rec11	10×20	1946	-3.34	0.2	-3.34	0.2	-3.34	0.2	-2.3	7.0	-3.34	0.00	0.00	-5.65
Rec13	15×20	2709	-6.05	0.3	-6.05	0.3	-6.05	0.3	-5.47	11.0	-6.05	0.00	0.01	-7.09
Rec15	15×20	2691	-5.91	0.3	-6.02	0.3	-5.91	0.3	-5.69	8.6	-6.02	0.00	0.00	-8.62
Rec17	15×20	2740	-5.58	0.3	-5.58	0.3	-5.58	0.3	-5.42	8.6	-5.58	0.00	0.00	-7.96
Rec19	10×30	3157	-9.72	0.4	-9.25	0.4	-9.38	0.4	-8.5	23.0	-9.71	0.08	0.05	-11.41
Rec21	10×30	3015	-6.31	0.4	-6.30	0.4	-6.17	0.4	-5.33	24.0	-6.40	0.07	0.06	-9.07
Rec23	10×30	3030	-10.76	0.4	-10.73	0.4	-10.89	0.4	-9.72	24.0	-10.89	0.02	0.05	-14.52
Rec25	15×30	3835	-5.97	0.5	-6.31	0.5	-6.21	0.5	-5.17	32.0	-6.28	0.05	0.03	-8.79
Rec27	15×30	3655	-5.64	0.5	-6.10	0.5	-5.83	0.5	-5.04	39.0	-6.11	0.07	0.06	-9.46
Rec29	15×30	3583	-7.94	0.5	-8.28	0.5	-7.94	0.5	-6.93	31.0	-8.07	0.15	0.03	-11.63
Rec31	10×50	4631	-5.90	1.1	-6.13	1.1	-6.22	1.1	-5.2	122.0	-6.58	0.12	0.30	-8.89
Rec33	10×50	4770	-5.51	1.1	-6.31	1.1	-6.37	1.1	-4.08	116.0	-6.61	0.21	0.22	-8.69
Rec35	10×50	4718	-6.08	1.1	-6.17	1.1	-5.91	1.1	-5.13	105.0	-6.47	0.26	0.26	-8.98
Rec37	20×75	8979	-9.41	2.5	-9.49	2.6	-9.36	2.6	-8.2	635.0	-10.02	0.14	0.77	-12.94
Rec39	20×75	9158	-7.00	2.5	-6.99	2.6	-6.91	2.6	-5.67	897.0	-7.23	0.18	0.89	-10.81
Rec41	20×75	9344	-8.78	2.5	-8.57	2.6	-8.82	2.6	-6.77	883.0	-8.93	0.15	0.87	-12.47
Hel1	10×100	780	-8.08	3.8	-8.21	3.9	-8.33	3.9	—	—	-8.87	0.20	1.24	-10.33
Hel2	10×20	189	-5.29	0.2	-5.29	0.2	-5.29	0.2	—	—	-5.29	0.00	0.00	-7.78
平均			-6.50	0.9	-6.59	0.9	-6.56	1.3	-5.65	142.49	-6.74	0.07	0.21	-9.10

由表 2 知：

(1) 比较 DPSO 与 TS、TS+M 和 TS+MP. 鉴于 TS+M 是 3 种禁忌搜索算法中的最好算法<sup>[3]</sup>，在此仅比较 DPSO 与 TS+M. 从各测试算例来看，DPSO 对 14 个算例的计算结果优于 TS+M，7 个算例的计算结果 DPSO 等于 TS+M，只有 3 个算例的计算结果劣于 TS+M；从问题规模来看，对于工件数量大于 50 的算例，DPSO 所得结果均优于 TS+M，而且随着问题的规模增大，DPSO 的优越性越来越明显；从总体平均结果来看，DPSO 优于 TS+M. 就所需计算时间而言，尽管 DPSO 采用的处理器 3 倍于 TS+M 的处理器，但是 DPSO 的平均时间不到 TS+M 的  $\frac{1}{3}$ . 因此可以说，在同样计算时间内，DPSO 有更高的求解质量，即 DPSO 优于 TS+M.

(2) 比较 DPSO 与 HPSO. DPSO 对所有算例的计算结果均优于 HPSO，而且随着问题的规模增大，DPSO 对 HPSO 的优越性也增加. 虽然 DPSO 采用的处理器略优于 HPSO 的处理器，但其计算时间要远远小于 HPSO 的计算时间. 因此可知，DPSO 优于 HPSO.

(3) DPSO 所得偏差均方差较小，其中 10 个算例的均方差为 0，最大均方差为 0.26%. 这表明 DPSO 算法有较强的鲁棒性.

(4) 从 DPSO 的“改进结果”来看，每个算例的生产周期都得到了较大改进，且随着工件数量的增加，算例的改进幅度越来越大. 总体而言，“改进结果”的平均偏差降低了 35%. 这表明延长某些工序的加工时间，能有效地缩短生产周期.

## 6 结束语

本文研究了以生产周期为目标的无等待流水车间优化调度问题，把该类问题的优化分成两步：首先采用离散粒子群调度算法搜索全局最优解，然后通过延长工序加工时间进一步改善所得调度方案. 其中，结合问题特征，提出了一种生产周期的快速算法和一种插入移动邻域的快速搜索法；分析了可一次同时执行多个独立移动的多重插入移动邻域，并提出了相应的算法. 研究了该类问题的不规则特性，并提出了通过延长工序加工时间改进调度方案的算法. 仿真实验表明了所得算法的可行性和优越性.

## 参 考 文 献

[1] Aldowaisan T, Allahverdi A. New heuristic for no-wait flowshops to minimize makespan. Computer & Operation Research, 2003, 30(5): 1219-1231

- [2] Schuster C J, Framinan J M. Approximative procedure for no-wait job shop scheduling. *Operations Research Letters*, 2003, 31(4): 308-18
- [3] Grabowski J, Pempera J. Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers & Operations Research*, 2005, 32(8): 2197-2122
- [4] Liu B, Wang L, Jin Y H. An effective hybrid particle swarm optimization for no-wait flow shop scheduling. *International Journal of Advanced Manufacturing Technology*, 2007, 31(9-10): 1001-1011
- [5] Pan Q K, Tasgetiren M F, Liang Y C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem with makespan criterion//*Proceedings of the International Workshop on UK Planning and Scheduling Special Interest Group*. London: City University, 2005: 31-41
- [6] Pan Q K, Tasgetiren M F, Liang Y C. Minimizing total earliness and tardiness penalties with a common due date on a single-machine using a discrete particle swarm optimization algorithm//*Aat Colong Optimization and Swarm Intelligence. Lecture Notes in Computer Science* 4150, 2006: 460-467
- [7] Pan Q K, Wang L. No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. *International Journal of Advanced Manufacturing Technology*, 2007, doi:10.1007/s00170-007-1252-0
- [8] Spieksma F C R, Woeginger G J. The no-wait flow-shop paradox. *Operations Research Letters*, 2005, 33(6): 603-608
- [9] Kalczynski P J, Kamburowski J. On no-wait and no-idle flow shops with makespan criterion. *European Journal of Operational Research*, 2007, 138(3): 677-685
- [10] Reeves C. A genetic algorithm for flowshop sequencing. *Computers & Operations Research*, 1995, 22(1): 5-13
- [11] Heller J. Some numerical experiments for an  $M \times J$  flow shop and its decision-theoretical aspects. *Operations Research*, 1960, 8(2): 178-184



**PAN Quan-Ke**, born in 1971, Ph.D., professor. His research interests focus on scheduling theory and method.

**ZHAO Bao-Hua**, born in 1947, professor, Ph. D. supervisor. His research interests include protocols theory and engineering, and algorithm of design in wireless sensor networks.

**QU Yu-Gui**, born in 1945, professor, Ph. D. supervisor. Her research interests include signal processing and designing of embedded techniques.

## Background

Production scheduling plays a key role in the manufacturing systems of enterprises for maintaining a competitive position in fast-changing markets, so it is very important to develop effective, efficient and advanced manufacturing and scheduling technologies and approaches. No-wait flow shop scheduling problem is one of the most important scheduling problems, which has important applications in different industries including chemical processing, food processing, concrete ware production, and pharmaceutical processing. In the past decades, most research focused on developing heuristic algorithms for this problem. Particle swarm optimization (PSO) algorithm is one of the latest metaheuristic methods in the literature. However, the applications of PSO algorithm on combinatorial optimization problems are still considerably limited. The major obstacle of successfully applying PSO al-

gorithm to combinatorial problems is due to its continuous nature. To remedy this drawback, Pan, Tasgetiren & Liang presented a novel variant of PSO algorithm, called DPSO algorithm. In this paper, the authors apply DPSO to solve the no-wait flow shop scheduling problem with makespan criterion and propose an improvement procedure by slowing down some machines for a permutation. Computational tests show the effectiveness and efficiency of the presented method. This research is partially supported by National Science Foundation of China under grants 90104010 and Postdoctoral Science Foundation of China under grants 20070410791. The work group have published more than 60 papers in this field, and most of them have been cited by SCI/EI. This paper deepens research theory and contribute to the two projects.