文章编号: 1001-0920 (2012) 04-0513-06

基于改进粒子群算法求解柔性作业车间批量调度问题

张 静[a,b], 王万良[a], 徐新黎[a], 王海燕[c]

(浙江工业大学 a. 计算机科学与技术学院; b. 信息工程学院; c. 机械工程学院, 杭州 310023)

摘 要: 基于工序排序和机器分配的粒子编码方式, 提出一种新的粒子位置更新方式, 该方式使得粒子群算法更新可以直接在离散域执行. 通过对工件工序进行多次机器分配来扩大搜索范围, 引入改进的模拟退火算法, 用以增强粒子群算法的邻域搜索能力, 实现全局搜索与局部搜索能力的有效平衡. 最后通过数值算例以及某电声企业纸盆车间批量调度的应用实例验证了所提出算法的有效性和可行性.

关键词: 粒子群算法; 柔性作业车间调度; 批量调度; 模拟退火

中图分类号: TP301.6 **文献标识码:** A

Improved particle swarm algorithm for batch splitting flexible job shop scheduling

ZHANG Jing[a,b], *WANG Wan-liang*[a], *XU Xin-li*[a], *WANG Hai-yan*[c]

(a. College of Computer Science and Technology, b. College of Information Engineering, c. College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou 310023, China. Correspondent: WANG Wan-liang, E-mail: wwl@zjut.edu.cn)

Abstract: A novel particle position updating strategy is presented based on the chromosome coding scheme of operation sequencing and machine allocation. This strategy makes the particle swarm optimization(PSO) algorithm work directly in the discrete domain, and machines are allocated many times to operations to expand search scope. An improved simulated annealing technique is introduced to enhance the neighborhood search of PSO algorithm, which efficiently balances the exploration and exploitation abilities. Finally, a numerical example and an electro-acoustic enterprise cone shop example are given to show the effectiveness and feasibility of the proposed algorithm.

Key words: particle swarm algorithm; flexible job-shop scheduling; the batch splitting scheduling; simulated annealing

1 引 言

作业车间调度问题 (JSP) 可描述为将各个工件的各个工序分配到给定机器的调度问题, 是一类满足任务配置和顺序约束要求的资源分配问题, 是最困难的组合优化问题之一[1]. 因为 JSP 对机器约束的限制严重脱离了生产实践, 所以减少了机器约束的柔性作业车间调度问题 (FJSP) 是实际生产中亟待解决的一类调度问题, 对其展开研究具有重要意义.

目前, 求解 FJSP 常用的方法主要有遗传算法、模拟退火、禁忌搜索和粒子群算法等. 文献 [2] 采用粒子群算法与模拟退火算法相结合的混合算法, 在求解多目标柔性作业车间调度问题上取得了很好的效果, 体现了粒子群算法与模拟退火算法相融合的优越性; [3] 提出了一种双层子代产生模式的改进遗传算法; [4] 和 [5] 分别采用并行的变邻域搜索算法和不同的策略产生初始种群, 增加了种群的多样性, 从而提高了算法的局部搜索能力.

在实际制造业中, 工件的加工往往是批量加工, 并且大批量的生产模式逐渐向多品种、中小批量转化, 因此在 FJSP 中考虑工件批量调度问题更贴近实际, 对工业生产更具指导意义. 相对于传统 JSP, FJSP 的可行解空间更大, 问题的复杂性更高, 其批量调度问题更是复杂的 NP 难题, 然而国内外的相关报道却相对较少. 文献 [6] 研究了多工艺路线的作业车间批量调度问题, 该文采用遗传算法求解, 并仿真验证了分批处理可以达到缩短生产周期的目的. 文献 [7] 采

收稿日期: 2010-10-21; **修回日期:** 2011-01-07.

基金项目: 国家自然科学基金项目(60874074, 61070043); 中国博士后科学基金项目(20090451486); 浙江省自然科学基金项目(Y1090592).

作者简介: 张静(1986–), 女, 博士生, 从事生产调度、智能算法的研究; 王万良(1957–), 男, 教授, 博士生导师, 从事 CIMS, 生产计划与调度等研究.

用混合差分进化算法求解作业车间批量划分和批量调度问题, 有效地提高了生产效率.

本文将离散粒子群算法 (DPSO) 与模拟退火算法 (SA) 相结合, 构造了改进型的 PSO (IPSO), 用以解决 PSO 的早熟收敛问题, 并通过算例仿真和实例应用验证了所提出算法的可行性和有效性.

2 柔性作业车间批量调度问题描述

考虑如下 FJSP: 有 n 个工件在 m 台机器上加工, 第 i 个工件包含 n_i 道工序, 工序的加工时间随机器的性能不同而变化, 并且每种工件可分成若干批次, 各个子批量当作一个整体来处理. 需要解决的问题是: 如何为每个工件的每道工序选择最适合的机器, 并确定每台机器上各个工件工序的最佳加工顺序, 找到最小加工时间长度 (Makespan).

记工件为 J_i, 机器为 M_k, n_i表示工件 J_i的工序总数, p_i为工件 J_i划分的批量数. 其中: $i\in N=\{1,2,\cdots,n\}, k\in W=\{1,2,\cdots,m\}$. 容易看出, 若 $p_i=1,\forall i\in N$, 则该问题为单件 FJSP; 否则, 为多件 FJSP.

3 IPSO 求解 FJSP

PSO 是由 Kennedy 等人[8]于 1995 年提出的, 它在解决连续优化问题上得到了成功的应用, 但在组合优化领域的应用却非常有限. 为此, 本文提出一种新的 RPX (rand-point preservation crossover)交叉算子, 用于基于机器编码的调整, 以增大解的搜索范围, 进而提高获得最优解的概率; 采用文献 [9] 提出的 POX (precedence preserving order based crossover)交叉算子, 用于基于工序编码的调整, 以避免非法解的出现, 提高搜索效率, 进而得到一种新型的 DPSO; 引入基于机器负载的 SA 来增强邻域搜索, 从而得到一种新型的 IPSO, 用以求解 FJSP.

3.1 编 码

求解 FJSP, 首先需对其进行编码. 采用 2 个 l 维向量表示粒子的位置, 分别记作 A 向量和 B 向量, 其中 $l=\sum_{i=1}^{n}n_i$. A 向量表示基于工序的编码, B 向量表示基于机器的编码. A 向量的各个元素为工件号, 并表示工序加工的先后顺序; B 向量的各个元素为机器号, 表示相应工序加工的机器. 为方便描述每道工序的加工时间, 记第 i 个工件的第 j 道工序为 O_{ij}, j 表示工件 i 的第 j 道工序, $i\in N, j\in\{1,2,\cdots,n_i\}$. 为每道工序分配加工机器 M_k, 则加工时间可用表格清楚地表示. 一个由 3 工件 6 机器 10 工序组成的 FJSP 算例[10]的加工时间如表 1 所示, 相应随机产生的粒子编码如表 2 所示. 例如, 表 2 中的粒子的第 2 维为 (2, 1), 它表示第 2 个工件的第 1 道工序在机器 1 上加工, 容易看出在表 1 中对应为 O_{21} 和 M_1, 加工时间为 15.

表 1 柔性作业车间调度问题算例

		M_1	M_2	M_3	M_4	M_5	M_6
J_1	O_{11}	10	7	6	13	5	1
	O_{12}	4	5	8	12	7	11
	O_{13}	9	5	6	12	6	17
	O_{14}	7	8	4	10	15	3
J_2	O_{21}	**15**	12	8	6	10	9
	O_{22}	9	5	7	13	4	7
	O_{23}	14	13	14	20	8	17
J_3	O_{31}	7	16	5	11	17	9
	O_{32}	9	16	8	11	6	3
	O_{33}	6	14	8	18	21	14

表 2 粒子编码表达式

粒子维数	1	2	3	4	5	6	7	8	9	10
A 向量	1	2	1	1	2	3	2	1	3	3
B 向量	4	1	5	2	6	4	3	2	1	5

对于多件 FJSP 批量调度问题, 采用等量划分原则, 即每个工件的各子批次的批量大小相等, 并将划分后的工件当作一个独立的工件, 从而保持进化操作与未进行批量划分一致. 因此, 容易得到批量划分后粒子长度为

$$l^{'}=\sum_{i=1}^{n}(n_i\times p_i).$$

若上例中将工件 J_1 划分为 2 个批次, 则工件 J_1 的第 1 批和第 2 批可分别编码为 1 和 2. 此时, 表示 J_1 的粒子长度增加为 $4\times2=8$. 可以看出, 批量划分将增加粒子的维数, 加大求解的复杂性.

3.2 种群初始化

根据上述编码方式, 采用文献 [11] 提出的初始种群定位法生成初始种群. 该方法为工序分配机器时考虑了机器负载, 操作过程如下:

1) 为每道工序在可行机器集中选取具有最短加工时间的机器;

2) 更新机器负载, 将该最短加工时间加到时间表中同一列其他项上.

例如, 对表 2 的粒子进行解码, 得到 Makespan = 54, 而采用文献 [11] 提出的初始种群定位法, 则初始 B 向量变为 $[6,4,1,2,5,3,5,6,6,1]$, 得 Makespan = 22. 可以看出, 相比于随机产生的粒子, 本文方法所得结果更接近最优解, 从而提高了算法的收敛速度.

3.3 粒子位置更新

基于文献 [12] 提出的位置更新公式, 通过加入 RPX 交叉算子, 可得到如下 DPSO 位置更新公式:

$$X_i^{k+1}=$$
$$c_2\otimes f\{c_1\otimes g[c_1\otimes q(w\otimes h(X_i^k),pB_i^k),pB_i^k],gB^k\}. \tag{1}$$

其中: X_i^k 为粒子的位置; w 为惯性权重; c_1 为认知系数; c_2 为社会系数; $w, c_1, c_2 \in [0,1]$; pB_i^k 为 k 代粒子个体最优值; gB^k 为 k 代全局最优值; h, q, g 和 f 为操作算子.

位置更新公式 (1) 由以下 3 部分组成:

1) 第 1 部分

$$E_i^k = w \otimes h(X_i^k) = \begin{cases} h(X_i^k),\ r < w; \\ X_i^k,\ \text{otherwise}. \end{cases} \tag{2}$$

其中 r 为区间 $[0,1]$ 上的随机数.

式 (2) 表示粒子对先前状态的思考. 对 $h(X_i^k)$ 粒子的 2 个向量分别调整, 实现方法如下: 在 A 向量中随机交换染色体中 2 个不同基因的位置; 在 B 向量中随机选择 1 个与其不等的机器号进行替换.

2) 第 2 部分

$$F_i^k = c_1 \otimes q(E_i^k, pB_i^k) = \begin{cases} q(E_i^k, pB_i^k),\ r < c_1; \\ E_i^k,\ \text{otherwise}. \end{cases} \tag{3}$$

$$G_i^k = c_1 \otimes g(F_i^k, pB_i^k) = \begin{cases} g(F_i^k, pB_i^k),\ r < c_1; \\ F_i^k,\ \text{otherwise}. \end{cases} \tag{4}$$

式 (3) 和 (4) 表示粒子根据自身最优位置 pB_i^k, 分别采用 POX[9] 交叉算子和本文提出的 RPX 交叉算子进行 2 次调整. 对同一种工序编码的向量进行多次机器分配, 得到粒子的 A 向量和 B 向量的多种组合, 从而扩大解的搜索范围, 防止算法陷入局部最优.

$q(E_i^k, pB_i^k)$ 表示 POX 交叉算子, 操作过程如下: 将工件集随机分为 2 个非空互补子集 Q_1 和 Q_2; 复制 E_i^k 包含 Q_1 的工件到子代 1, 复制 pB_i^k 包含 Q_1 的工件到子代 2, 并保留它们的位置, 同时复制对应位置的机器号; 复制 pB_i^k 包含 Q_2 的工件到子代 1, 复制 E_i^k 包含 Q_2 的工件到子代 2, 并保留它们的顺序, 同时复制对应位置的机器号; 最后从子代中随机选择 1 个作为后代. POX 交叉过程使后代保留了部分工件在机器上的位置, 并使子代继承父代在每台机器上工序的次序, 具有很好的保优作用, 同时从交叉后的 2 个粒子中随机选择 1 个作为后代, 从而保持了种群的多样性.

$g(F_i^k, pB_i^k)$ 表示针对 B 向量的 RPX 交叉算子, 操作过程如下: 随机产生由 $(0,1)$ 区间小数构成的数组 R, R 的长度与粒子维数相等. 若数组 R 中的随机数小于 p ($p \in (0,1)$), 则复制 pB_i^k 对应工序的机器号到 F_i^k. 其中 $p = p_{\max} - [(p_{\max} - p_{\min})/\text{iter}] * \text{gen}$, iter 为迭代次数, gen 为当前运行代数. 该方法相对于传统基于位置的交叉方法, 可根据所需情况控制交叉的概率, p 的自适应调整有利于在全局范围内产生更好的搜索能力, 进化后期加强局部搜索能力, 对提高获取最优解的概率有一定的帮助.

显然, 第 2 部分的结果是一个工序和机器分配序列, 避免了非法解的出现, 从而可以提高算法的搜索效率.

3) 第 3 部分

$$X_i^k = c_2 \otimes f(G_i^k, gB^k) = \begin{cases} f(G_i^k, gB^k),\ r < c_2; \\ G_i^k,\ \text{otherwise}. \end{cases} \tag{5}$$

式 (5) 表示粒子根据全局最优位置 gB^k 进行调整, 其中 $f(G_i^k, gB^k)$ 的操作过程与 $g(F_i^k, pB_i^k)$ 相同. 显然, 第 3 部分的结果仍然是一个可行工序序列.

3.4 邻域搜索策略

采用上述编码和位置更新公式, 可得到适用于 FJSP 的 DPSO 算法, 这种 DPSO 算法保留了 PSO 收敛速度快、实现简单等优点, 但同时也保留了易陷入局部最优的缺点. 而 SA[13] 算法具备较强的邻域探索能力, 为此, 本文将 DPSO 与基于机器负载产生邻域解的 SA 相结合, 以得到 IPSO, 从而有效提高算法的搜索性能. 操作过程如下: 计算每台机器的工作负载, 从负载最大的机器上随机选择一个工件的工序, 然后将具有最小负载的机器分配给它. SA 只对向量 B 进行调整, 使向量 A 分配到更合适的机器, 从而更好地平衡机器负载. 但是, SA 的加入必然增加搜索时间, 因此采用一种随机选择部分个体进行模拟退火的方法, 以增加种群的多样性.

3.5 IPSO 执行流程

采用 IPSO 求解 FJSP 的具体步骤如下:

Step 1: 确定参数. 设定种群规模 P, 迭代次数 G, 问题规模 D, 惯性常量 w, 学习因子 c_1 和 c_2, 初始温度 T_0, 终止温度 T_{end}, 退火率 B_0.

Step 2: 种群初始化. 随机产生基于工序的编码, 采用文献 [11] 提出的初始定位法初始化粒子的位置 $\{X_1, X_2, \cdots, X_P\}$, 计算每个粒子的适应值, 记录初始局部最优值 $pB_i = X_i$ 和全局最优值 $gB = \min\ pB_i, i \in \{1, 2, \cdots, P\}$.

Step 3: 根据式 (3) 更新粒子的位置, 同时更新局部最优值 pB_i 和全局最优值 gB. 操作过程如下:

If $f(X_i) < f(pB_i)$, then $pB_i = X_i$;

If $f(pB_i) < f(gB)$, then $gB = pB_i$.

Step 4: 若未达到最优适应值, 则产生一个随机整数 $r = [1, P/2]$, 从种群中随机选择 r 个粒子, 对其进行基于机器负载的模拟退火邻域搜索算法, 同时更新粒子位置、局部最优位置和全局最优位置.

Step 5: 判断是否达到终止条件 (如迭代次数>

G). 若满足, 则输出最优值; 否则, 转 Step 3.

3.6 算法复杂度分析

从3.5节的算法流程可以看出, IPSO增加的计算量主要发生在位置更新和局部邻域搜索的迭代过程. IPSO在Step 3位置更新中需对A向量调整2次, 对B向量调整4次, 因而其时间复杂度为$G\times O(P\times 4D)$, 计算适应值和保留最优个体的时间复杂度为$G\times O(P\times D)+G\times O(P)$, 邻域搜索过程的最坏时间复杂度为$G\times O(P\times D)+G\times O(P)$, 得到IPSO的时间复杂度为

$$6\times G\times O(P\times D)+G\times O(P)\approx G\times O(P\times D).$$

PSO的计算量主要是更新位置和速度, 时间复杂度为$G\times O(P\times 3D)\approx G\times O(P\times D)$. 可以看出, IPSO并没有明显增加算法的计算时间复杂度.

4 实验结果与分析

为了验证所提出算法的有效性, 首先对4个不同的算例进行仿真, 然后将所提出的算法应用于求解实际生产车间批量调度问题. 采用VC 6.0编程, 运行环境为P 4 CPU, 主频2.66 GHz, 内存4 GB. 算法参数取多次实验中效果较好的值, 设置如下: 种群规模$P=100$, 最大迭代次数$G=1\,000, w=0.95, c_1=0.5, c_2=0.8, T_0=3, T_{\rm end}=0.01, B=0.9$.

4.1 算例测试

采用文献[11]中的4个FJSP算例, 分别为问题4×5 (T-FJSP), 8×8 (P- FJSP), 10×7 (T-FJSP)和10×10 (T-FJSP), 其中: “问题$n\times m$”表示n个工件, m台机器; T-FJSP表示完全FJSP; P-FJSP表示部分FJSP. 将IPSO分别与文献[2]提出的混合粒子群算法(HPSO)以及文献[14]提出的GENACE算法测试的最优值进行比较, 结果如表3所示. 从表3可以看出, 改进的算法所获得的最优解等于或者小于其他4个算法所得到的最优解, 对于两类FJSP均具有更好的寻优能力, 在解的质量上有所提高.

表 3 文献[11]算例测试的最优值比较

问题$n\times m$	文献[11]	文献[14]	文献[2]	IPSO
4 × 5	16	11	—	11
8 × 8	16	—	15	14
10 × 7	15	12	—	11
10 × 10	7	7	7	7

对每个算例独立运行10次, 取平均值, 然后与文献[3]和[15]分别提出的改进的遗传算法(IGA)实验测试的平均值进行比较, 结果如表4所示. 从表4可以看出, 对于问题4×5和问题10×7, IPSO均能稳定地寻找到最优值11; 而对于问题8×8和问题10×10, IPSO所求得的平均值比[3]和[15]小很多, 说明IPSO具有更高的稳定性和可靠性.

表 4 文献[11]算例测试的平均值比较

问题$n\times m$	文献[15]	文献[3]	IPSO
4 × 5	—	—	11
8 × 8	17.9	16.2	14.4
10 × 7	—	—	11
10 × 10	9.1	7.8	7.2

局部搜索操作能有效提高寻优能力, 但也消耗了更多的时间. 因此, 为更公平地验证算法, 可考虑在相同的时间内IPSO和DPSO能够成功搜索到最优解的次数作为评价指标. 独立运行20次, 设定相同的时间20 s, 两种算法达到最优解的次数如图1所示.

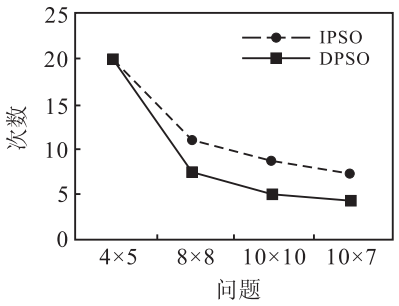

图 1 相同时间内搜索性能比较

从图1可以看出, DPSO只能在小规模问题4×5上稳定求得最优值, 在其他3个问题上, IPSO在相同时间内达到最优解的次数明显增加. 因此, SA的加入使算法能够保持更高的稳定性, 搜索性能得到增强. 实验结果显示, 将IPSO用于求解FJSP时, 在解的质量上有较大提高.

4.2 实例应用

考虑某电声企业纸盆车间. 该问题具有8个工件, 16个资源(其中12台机器, 4个人员), 44个工序, 属于较大规模P-FJSP的例子. 其机器人员信息表和工件单件加工时间分别如表5和表6所示, 其中“—”表示该资源不能用于加工对应工序. 参数设置与算例测试一致.

首先考虑不进行批量划分的情况, 然后采用等量划分, 即每个工件划分为2批, 同一工件的不同批次视为2个不同工件处理. 采用所提出的算法进行优化, 记为IPSO$_p$, 以验证其在批量调度中的可行性. 将IPSO, 批量划分后的IPSO$_p$和未加入局部搜索的DPSO进行比较, 对其分别独立运行20次, 结果如表7所示. 其中: Best, Avg, Worst, Var分别表示最优解、平均值、最差解和方差; Time为运行20次的平均时间. 可以看出, 所提出的算法在寻优能力上具有明显优势, 但在时间消耗方面则有待进一步改进. 与IPSO相比, 进行批量划分后的IPSO$_p$对调度结果有所改进, 说明其在批量调度方面是可行的.

表 5　工件单件加工时间信息表

工件	批量	工序 (操作人员)	机器												操作人员			
			M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	P_1	P_2	P_3	P_4
布边纸盆 (J_1)	900	上胶	—	—	—	—	—	—	—	—	—	—	5	5	—	—	—	—
		落料	—	—	—	—	—	—	—	2	4	1	—	—	—	—	—	—
		热压	—	—	—	—	—	5	4	—	—	—	—	—	—	—	—	—
		冲切	—	—	—	—	—	—	—	2	2	2	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	14	12
布边纸盆 (J_2)	900	涂里圈	—	—	—	—	—	—	—	—	—	—	5	4	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	4	3	—	—
		压搭纸盆	—	5	2	3	1	—	—	—	—	—	—	—	—	—	—	—
		正封口	—	4	6	3	3	—	—	—	—	—	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	11	9
布边纸盆 (J_3)	800	涂里圈	—	—	—	—	—	—	—	—	—	—	4	4	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	5	5	—	—
		压搭纸盆	—	2	5	7	5	—	—	—	—	—	—	—	—	—	—	—
		反封口	—	3	4	9	7	—	—	—	—	—	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	7	8
布边纸盆 (J_4)	500	涂里圈	—	—	—	—	—	—	—	—	—	—	3	6	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	6	3	—	—
		压搭纸盆	—	5	4	6	6	—	—	—	—	—	—	—	—	—	—	—
		正封口	—	3	5	4	3	—	—	—	—	—	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	9	10
泡沫边纸盆 (J_5)	800	涂里圈	—	—	—	—	—	—	—	—	—	—	7	5	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	4	3	—	—
		压搭纸盆	—	8	3	2	1	—	—	—	—	—	—	—	—	—	—	—
		正封口	—	4	7	7	5	—	—	—	—	—	—	—	—	—	—	—
		反封口	—	3	3	2	3	—	—	—	—	—	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	10	8
布边纸盆 (J_6)	500	涂里圈	—	—	—	—	—	—	—	—	—	—	4	6	—	—	—	—
		上胶	—	—	—	—	—	—	—	—	—	—	6	8	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	4	5	—	—
		压搭纸盆	—	5	4	2	2	—	—	—	—	—	—	—	—	—	—	—
		反封口	—	3	1	4	2	—	—	—	—	—	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	11	10
纸小双盆 (J_7)	900	做铜丝网	—	—	—	—	—	—	—	2	2	2	—	—	—	—	—	—
		捞浆成形	2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	3	4	—	—
		冲切	—	—	—	—	—	—	—	3	4	4	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	6	9
纸锥 (J_8)	1 000	涂里圈	—	—	—	—	—	—	—	—	—	—	4	5	—	—	—	—
		上胶	—	—	—	—	—	—	—	—	—	—	9	6	—	—	—	—
		中检	—	—	—	—	—	—	—	—	—	—	—	—	1	5	—	—
		热压	—	—	—	—	—	5	6	—	—	—	—	—	—	—	—	—
		正封口	—	2	4	3	2	—	—	—	—	—	—	—	—	—	—	—
		反封口	—	2	2	2	2	—	—	—	—	—	—	—	—	—	—	—
		成检	—	—	—	—	—	—	—	—	—	—	—	—	—	—	10	9

表 6　机器、人员信息表

资源名称	捞浆机	简易压力机	可倾压力机	切边机	涂胶机	中检操作人员	成检操作人员
编号	M_1	M_2, M_3, M_4, M_5	M_6, M_7	M_8, M_9, M_{10}	M_{11}, M_{12}	P_1, P_2	P_3, P_4

表 7　求解电声企业实例比较结果

	Best	Avg	Worst	Var	Time / s
DPSO	37.8	39.41	40.6	0.90	6
IPSO	37.8	39.04	40.0	0.88	10
IPSO$_p$	36.95	39.03	40.3	0.87	13

5　结　论

本文针对 FJSP 调度问题的特点, 设计了基于工序和机器分配编码的粒子编码方式; 改进了 PSO 的进化方式, 以避免进化过程中非法解的出现, 提高算法

搜索效率; 设计了基于机器负载的模拟退火算法, 以弥补PSO易陷入局部最优的缺点; 通过多次调整基于机器编码的粒子编码来增大算法的搜索空间, 以有效平衡其局部搜索和全局搜索能力. 用算例测试了改进的PSO, 并与其他算法进行了比较分析, 以验证改进算法的有效性和可行性. 最后将改进算法应用于某电声企业实际车间问题, 并对其批量调度问题进行求解, 获得了更符合实际的效果, 其结果对生产实践具有一定指导作用. 但是, 未对文中涉及的批量调度进行批量划分优化, 后续工作中将对其展开研究.

参考文献(References)

[1] 王万良, 吴启迪. 生产调度智能算法及其应用[M]. 北京: 科学出版社, 2007: 12-13.
(Wang W L, Wu Q D. Production scheduling intelligent algorithm and application[M]. Beijing: Science Press, 2007: 12-13.)

[2] Xia W J, Wu Z M. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Computers and Industrial Engineering, 2005, 48(2): 409-425.

[3] 张超勇, 饶运清, 李培根. 柔性作业车间调度问题的两级遗传算法[J]. 机械工程学报, 2007, 43(4): 119-124.
(Zhang C Y, Rao Y Q, Li P G. Two stage genetic algorithm for flexible job shop scheduling problem[J], Chinese J of Mechanical Engineering, 2007, 43(4): 119-124.)

[4] Yazdani M, Amiri M, Zandieh M. Flexible jobshop scheduling with parallel variable neighborhood search algorithm[J]. Expert Systems with Applications, 2010, 37(1): 678-687.

[5] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem[J]. Computers and Operations Research, 2008, 35(10): 3202-3212.

[6] 潘全科, 朱剑英. 多工艺路线的批量生产调度优化[J]. 机械工程学报, 2004, 40(4): 236-239.
(Pan Q K, Zhu J Y. Optimization method for a job-shop scheduling problem with alternative machines in the batch process[J]. Chinese J of Mechanical Engineering, 2004, 40(4): 236-239.)

[7] Zhao Y W, Wang H Y, Xu X L, et al. A new hybrid parallel algorithm for consistent-sized batch splitting job shop scheduling on alternative machines with forbidden intervals[J]. Int J of Advanced Manufacturing Technology, 2010, 48(9): 1091-1105.

[8] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Piscataway, 1995: 1942-1948.

[9] Shi G Y. A genetic algorithm applied to a classic job shop scheduling problem[J]. Int J of Systems Science, 1997, 28(1): 25-32.

[10] Noureddine L, Ihsen S, Slim H. Ant systems and local search optimization for flexible job shop scheduling production[J]. Int J of Computers Communications and Control, 2007, 2(2): 174-184.

[11] Kacem I, Hammadi S, Borne P. Approach by localization and multi-objective evolutionary optimization for flexible job shop scheduling problems[J]. IEEE Trans on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2002, 32(1): 1-13.

[12] 潘全科, 王文宏, 朱剑英, 等. 基于粒子群优化和变邻域搜索的混合调度算法[J]. 计算机集成制造系统, 2007, 13(2): 323-328.
(Pan Q K, Wang W H, Zhu J Y, et al. Hybrid heuristics based on particle swarm optimization and variable neighborhood search for job shop scheduling[J]. Computer Integrated Manufacturing Systems, 2007, 13(2): 323-328.)

[13] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing science[J]. J of Statistical Physics, 1983, 220(4598): 671-680.

[14] Ho N B, Tay J C. GENACE: An efficient cultural algorithm for solving the flexible job-shop problem[C]. Proc of the Congress on Evolutionary Computation. Portland, 2004: 1759-1766.

[15] 张超勇, 刘琼, 邱浩波, 等. 考虑加工成本和时间的柔性作业车间调度问题研究[J]. 机械科学与技术, 2009, 28(8): 1005-1011.
(Zhang C Y, Liu Q, Qiu H B, et al, On flexible job shop scheduling problem considering operation cost and time[J]. Mechanical Science and Technology for Aerospace Engineering, 2009, 28(8): 1005-1011.)