文章编号: 1001-0920(2005)02-0137-05

基于混合微粒群优化的多目标柔性 Job-shop 调度

夏蔚军, 吴智铭
(上海交通大学 自动化系, 上海 200030)

摘 要: 应用传统方法求解多目标柔性 Job-shop 调度问题是十分困难的, 微粒群优化采用基于种群的搜索方式, 融合了局部搜索和全局搜索, 具有很高的搜索效率. 模拟退火算法使用概率来避免陷入局部最优, 整个搜索过程可由冷却表来控制. 通过对这两种算法的合理组合, 建立了一种快速且易于实现的新的混合优化算法. 实例计算以及与其他算法的比较说明, 该算法是求解多目标柔性 Job-shop 调度问题的可行且高效的方法.

关键词: 多目标; 柔性 Job-shop 调度; 微粒群优化; 模拟退火; 混合优化算法
中图分类号: TP18 **文献标识码**: A

Hybrid particle swarm optimization approach for multi-objective flexible job-shop scheduling problems

XIA Wei-jun, WU Zhi-ming
(Department of Automation, Shanghai Jiaotong University, Shanghai 200030, China. Correspondent: XIA Wei-jun, E-mail: weijunxia@sjtu.edu.cn)

Abstract: Particle swarm optimization (PSO) is discussed, which combines local search and global search, possessing high search efficiency. Simulated annealing (SA) as a local search algorithm employs certain probability to avoid becoming trapped in a local optimum. By reasonably hybridizing these two methodologies, an easily implemented hybrid algorithm for the multi-objective Flexible job-shop scheduling problem (FJSP) is presented. The computational results show that the proposed algorithm is a viable and effective approach for the multi-objective FJSP.

Key words: multi-objective; flexible job-shop scheduling; particle swarm optimization; simulated annealing; hybrid optimization algorithm

1 引 言

Job-shop 调度作为生产调度的一个分支, 已经得到了广泛的关注和研究. 在传统的 Job-shop 调度问题中, 仅考虑工件具有唯一确定的加工工艺路线的情况, 使得加工计划和生产调度脱节, 缺乏实用性. 在实际的加工中, 某道工序应该可以在多台机器上加工, 工件具有可选择的加工路线, 即路径柔性已成为生产的实际需求. 采用柔性加工路径可有效平衡机器负载, 缩短加工时间, 减少在制品库存, 大大提高加工系统的整体性能. 柔性 Job-shop 调度问题包含两个子问题: 1) 路径子问题, 即把所有的工序分配给可供选择的机器; 2) 排序子问题, 即对已安排到各机器上的工序进行排序, 使得预先确定的目标达到最优.

目前关于柔性 Job-shop 调度问题的文献较少, 研究方法主要有两种: 1) 分解方法, 分别考虑路径子问题和排序子问题; 2) 集成方法, 同时考虑两个子问题. 分解方法由于贴近实际情况且有利于问题简单化而被大多数研究者采用. Brandimarte[1] 首先采用了分解方法, 即先用分配规则解决路径子问题, 然后

收稿日期: 2004-04-12; **修回日期**: 2004-06-16.
基金项目: 国家自然科学基金项目(70071017).
作者简介: 夏蔚军(1970—), 男, 浙江宁波人, 博士生, 从事 CMS 生产环境下的生产计划与调度、供应链优化等研究; 吴智铭(1936—), 男, 江苏苏州人, 教授, 博士生导师, 从事 CMS 生产环境下的生产计划与调度、智能软计算等研究.

用禁忌搜索算法解决排序子问题. Tung 等[2]发展了一种相似的方法, 并用将其用于柔性制造系统, 取得了良好的效果. 最近, Kacem 等[3,4]提出了一种以局部最小化方法(AL)为分配模型的遗传算法, 并用它求解单目标和多目标柔性 Job-shop 调度问题.

本文则提出一种基于微粒群优化和模拟退火的分解方法, 所考虑的优化目标包括最小化制造周期(最大完工时间)、关键机器负载和机器总负载. 其中微粒群优化用于向各机器分配工序, 模拟退火用于排序各机器上的工序并对排序结果进行评价.

2 问题描述

多目标柔性 Job-shop 调度问题可描述为: 有 n 个工件要在 m 台机器上加工, m 台机器的集合表示为 $M, M = \{M_1, M_2, \cdots, M_m\}$. 每个工件 j 有 n_j 道工序 $O_{j,1}, O_{j,2}, \cdots, O_{j,nj}$ 需要加工, 工件 j 的任一道工序 i ($O_{j,i}$) 都由机器集合 M 中的一台机器 $M_{j,i}$ ($M_{j,i} \subseteq M$) 来完成. 问题是要分配各工序 $O_{j,i}$ 到各机器并在各机器上排序使得以下各目标最小:

F_1: 制造周期(各机器最大完工时间);

F_2: 机器总负载, 即各机器总的加工时间;

F_3: 关键机器负载, 即加工时间最长的机器负载.

目标函数 $F(c)$ 表示为上面 3 个目标的加权和形式, 即

$$F(c) = w_1F_1(c) + w_2F_2(c) + w_3F_3(c).$$

3 分配算法 - 微粒群优化

3.1 标准微粒群优化算法

微粒群优化是由 Kennedy 和 Eberhart[5] 在 1995 年提出的一种基于种群的进化计算方法, 具有隐含的并行性, 初始种群由随机产生的解组成. 种群中每个个体被称为微粒, 微粒在搜索空间中以一定的速度飞行, 该速度可根据其本身的飞行经验以及同伴的飞行经验进行动态调整. 目前已开发了多种微粒群优化模型, 本文使用的全局优化模型如下[6]:

$$V_{id} = W * V_{id} + C_1 * \mathrm{Rand}() * (P_{id} - X_{id}) + C2 * \mathrm{rand}() * (P_{gd} - X_{id}), \quad (1a)$$

$$X_{id} = X_{id} + V_{id}. \quad (1b)$$

其中: V_{id} 为微粒 i 的速度; X_{id} 为微粒 i 的位置; P_{id} 为微粒 i 所经历的最好位置(p_{best}); P_{gd} 为所有微粒所经历过的最好位置(g_{best}); W 为惯性权重; C_1, C_2 为加速度常数, 表示把微粒拉向 p_{best} 和 g_{best} 的随机项(式(1a) 后两项) 权值; Rand 和 rand 为两个在[0, 1]范围内变化的随机数.

标准微粒群优化算法的执行过程如下: 1) 初始化一群微粒, 随机产生每个微粒的位置和速度; 2) 评价每个微粒的适应度; 3) 对每个微粒, 将其适应度值与自身 p_{best} 作比较, 如果当前值好于 p_{best}, 则将当前值设为该微粒的 p_{best}; 4) 对每个微粒, 将其适应度值与 g_{best} 作比较, 如果当前值好于 g_{best}, 则将当前值设为该群体的 g_{best}; 5) 根据式(1) 变化每个微粒的速度和位置; 6) 如未达到终止条件(通常是足够好的适应度值或预先设定的最大代数), 则返回 2).

3.2 编码与初始种群

成功应用微粒群优化算法的关键问题是: 如何将工件的一个排序编码为搜索空间中的一个解? 即要在问题解和微粒表示之间建立合适的映射. 首先针对每一道工序, 将可用机器按照加工时间递增的顺序进行排序, 如果两台机器加工时间相同, 则原序号较小的机器排在前. 由此得到可加工同一道工序的不同机器的优先水平. 然后根据各工件的工序顺序随机产生微粒的位置坐标, 但每个位置坐标值代表所用机器的优先水平. 因为不同的优先水平对应不同的机器, 所以一个微粒的位置坐标串就相应于所有工序在各机器上的一个分配方案. 下面以表 1 的问题(具有完全柔性, 即对某一工序, 所有机器都可选择) 为例具体说明.

表 1 例子问题

		机器				优先水平			
		M_1	M_2	M_3	M_4	1	2	3	4
J1	$O_{1,1}$	2	3	4	1	M_4	M_1	M_2	M_3
	$O_{1,2}$	3	1	8	2	M_2	M_4	M_1	M_3
J2	$O_{2,1}$	1	4	1	2	M_1	M_3	M_4	M_2
	$O_{2,2}$	5	3	2	9	M_3	M_2	M_1	M_4
	$O_{2,3}$	3	1	1	4	M_2	M_3	M_1	M_4
J3	$O_{3,1}$	7	6	3	5	M_3	M_4	M_2	M_1
	$O_{3,2}$	4	5	6	2	M_4	M_1	M_2	M_3

这是一个 3 工件、4 机器问题, 表的左边是原始数据, 包括工件号、工序和在不同机器上的加工时间, 表的右边是相应于每一工序的机器优先水平, 且优先水平 1 > 2 > 3 > 4. 一个随机产生的微粒位置坐标串及其含义如表 2 所示.

表 2 一个随机的微粒位置表示

工序	Job1		Job2			Job3	
	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{2,3}$	$O_{3,1}$	$O_{3,2}$
微粒位置	2	1	3	2	2	4	4
加工机器	M_1	M_2	M_4	M_2	M_3	M_1	M_3

根据前述方法产生初始微粒的位置, 可以很容易地收缩搜索空间, 这有助于改善搜索速度和解的质量. 例如在表 1 中, 可以限定优先水平小于 4, 那么在虚线右边的机器就不能被选择, 即 M_3 不能用来加工 $O_{1,1}$, M_3 也不能用来加工 $O_{1,2}$ 等. 因为每个微粒的位置坐标表示加工机器的优先水平, 所以每个位

置坐标应该是整数, 但经式(1) 计算后, 可能会出现小数, 如 3. 238, 这是没有意义的. 因此, 可将其取整到最接近的整数, 连续优化问题便转换成了离散优化问题.

3.3　参数设置

在式(1a) 中, 惯性权重(W) 是关系到微粒群优化算法搜索能力的重要参数. 一个大的惯性权值有助于搜索一个新的区域, 而一个小的惯性权值有助于在当前区域仔细搜索. 因此, 考虑将惯性权值从相对较大的值线性地减小到较小的值. 这样, 开始时算法具有很强的全局搜索能力, 而接近结束时具有更好的局部搜索能力. 具体计算公式如下:

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{\mathrm{iter}_{\max}} * \mathrm{iter}. \qquad (2)$$

其中: $W_{\max}$ 为惯性权重的初始值, $W_{\min}$ 为惯性权重的最终值, $\mathrm{iter}_{\max}$ 为最大迭代次数(或代数), iter 为当前迭代次数(或代数).

在计算实例中, 惯性权重的初始值为 1. 2, 然后根据式(2) 线性地减少到 0. 4. 在式(1a) 中, 加速度常数 C_1 和 C_2 根据以前经验取为 2. 0.

在式(1) 的计算过程中, 速度 V_{id} 和位置 X_{id} 的绝对值有可能很大, 使得微粒一下子飞过问题空间. 因此, 应将 V_{id} 和 X_{id} 的值限定在一定范围内, 即设定最大速度 $V_{\max}$ 和最大位置 $X_{\max}$. 在本文中, $V_{\max}$ 被设定为优先水平的最大值(mpl), 因为速度可为负值, 所以 V_{id} 是[- mpl, mpl] 之间的值. $X_{\max}$ 也被设定为 mpl. 因为 X_{id} 代表优先水平而只能为正值, 所以 X_{id} 是区间[1, mpl] 内的值.

3.4　适应度函数

适应度作为对微粒的绩效评价, 通常用评价函数 $f: S \rightarrow R^+$ (S 为候选排序集合, R^+ 为正实数集合) 来表示, 将目标函数值映射为适应度值是适应度函数的一个特征. 本文定义适应度函数(fit(c)) 等于目标函数($F(c)$), 即 fit(c) = $F(c)$, 这样, 适应度值小的微粒将更具优越性而被保留.

4　排序算法 - 模拟退火

模拟退火算法是一种增强版的局部搜索算法或者循环改进算法, 通过重复对一个初始解作小的、局部的改进, 并以一定的概率接受更差的解, 使算法避免陷入局部最优, 从而达到更好的解. 通过微粒群优化算法将工序分配到各机器之后, 就可通过模拟退火算法完成各工序在机器上的排序.

4.1　模拟退火算法

由一个初始解 S 开始, 算法在 S 的邻域内产生一个新解 S', 计算目标函数的差值 $\Delta = f(S') - f(S)$; 对于最小化问题, 如果 $\Delta < 0$, 则接受新解; 如果 $\Delta \geq 0$, 以概率 $\exp(-\Delta/T)$ 接受新解, 其中 T 为当前温度(控制参数). 模拟退火算法通常由一个很高的温度开始, 之后温度逐渐降低. 在同一温度, 要进行一定数量的迭代, 称为(同温度) 内循环长度(或者次数). 当终止条件满足时, 算法停止.

4.2　邻域解

在模拟退火算法中, 邻域的选取在很大程度上影响算法的效果. 选择一个大的邻域(包含大量候选解) 将会增加找到最优解的机会, 但所需的计算时间也会相应增加. 本文使用"成对交换"这种简单方法产生邻域解. 假设在某台机器上安排有 p 道工序, 按照以下方式进行多对顺序交换:

$(1 \leftrightarrow 2), (2 \leftrightarrow 3), (3 \leftrightarrow 4), \cdots, (p-1 \leftrightarrow p)$.

由于微粒的位置编码是基于工序顺序的, 所以不能直接用于成对交换, 必须转换为基于机器顺序的编码. 仍以表 2 为例, 表 3 给出转换之后的表示形式.

表 3　转换后的基于机器顺序的编码

机器	M_1	M_2	M_3	M_4
工序	$O_{1,1}O_{3,1}$	$O_{1,2}O_{2,2}$	$O_{2,3}O_{3,2}$	$O_{2,1}$

4.3　冷却进度表

模拟退火算法的搜索过程是由冷却进度表进行控制的. 冷却进度表由几个参数和原则组成, 主要包括: 初始温度 T_0, 内循环长度 L, 温度下降原则和终止条件等.

初始温度 T_0 应足够高, 以使所有可能的解能以同等的机会被搜索到, 这对计算时间有很大影响. 在本文中, $T_0 = \Delta f_{\max}$, $\Delta f_{\max}$ 是两个邻域解之间适应度的最大差值, 并通过多次实验确定 T_0.

内循环长度 L 表示在同一温度下进行的迭代次数, 在本文中, 根据邻域解的产生方法, L 由安排在机器上的工序数决定.

在模拟退火算法中, 若使整个冷却过程不需很长时间, 则温度应以某种方式下降. 目前比较通用的温度下降方式是 $T_k = B * T_{k-1}$. 其中: k ($k = 1, 2, \cdots$) 为内循环序号, 即第 k 次内循环; B 为小于 1 的参数, 称为温度递减率, B 越大, 意味着搜索过程越慢.

本文采用一种简单的方式来使算法终止, 即设定终止温度 T_{end}. 如果当前温度 $T_k < T_{\mathrm{end}}$, 则算法停止. T_{end} 为一接近零的实数, 它直接影响算法的搜索"粒度", 较小的 T_{end} 意味着在算法接近终止时对问题空间进行更仔细地搜索.

5　混合优化算法

具体的混合优化算法流程如下所示:

开始:

Step 1: 确定参数

* 设定种群数, 最大循环代数, 给参数 W_{max}, W_{min}, C_1, C_2 赋值, 令当前代数 generation = 0;

* 通过实验确定 T_0, T_{end}, B 的值

Step 2: 分配与排序

随机产生每个微粒的位置和初始速度;

用模拟退火算法评价每个微粒的适应度;

用每个微粒的自身位置初始化 p_{best};

用种群中适应度最小的微粒初始化 g_{best};

当前代数未达到设定的最大代数时

{ generation: = generation + 1;

用式(1a) 和式(1b) 产生下一代种群;

评价新种群{ 用模拟退火算法计算微粒的适应度;

通过比较找到新的 g_{best} 和 p_{best};

更新种群的 g_{best} 和微粒的 p_{best}; }

}

Step 3: 输出优化结果

结束

模拟退火算法子程序如下:

编码转换(转换成基于机器顺序的编码)

{$T_k = T_0$;

当前温度 $T_k > T_{end}$ 时

{产生 S 的一个邻域解 S';

计算 S' 的适应度;

评价 S'{$\Delta = f(S') - f(S)$;

如果 (min [1, exp(- Δ/T_k)] > random [0, 1])

{接受 S';}

如果满足条件, 更新迄今找到的最好解;

}

$T_k = B * T_{k-1}$;

}

}

由程序流程可以看出, 在整个搜索过程中, 微粒群优化为模拟退火算法提供初始解, 每个微粒的适应度值由模拟退火算法进行评价. 事实上模拟退火算法相当于嵌入在微粒群优化算法中的子程序, 微粒群优化算法用模拟退火算法的评价结果继续进化

6 计算实例

本文用两个实例(用"问题 $n \times m$"表示)[3] 来说明所提出算法的有效性. 两个例子(问题 8 × 8[3] 和问题 15 × 10[4]) 是无缓冲区的柔性 Job-shop 调度问题. 混合优化算法的参数设置如表 4 所示

表 4 混合优化算法参数

微粒群优化算法		模拟退火算法		
种群数	100		$n \times m$	
最大代数	50		8 × 8	15 × 10
W_{max}	1.2	T_0	3	10
W_{min}	0.4	T_{end}	0.01	0.01
C_1/C_2	2.0	B	0.9	0.95

例 1 问题 8 × 8

原始数据参见文献[3], 这是一个 8 工件、8 机器、27 工序、部分柔性的例子, 即某些工序只有部分机器可供选择, 符号"X"表示该机器不可选. 我们只需将"X"转换为无穷大(如: 9 999), 这意味着加工时间无限长. 根据前述方法, 该机器的优先水平最低而无法被选择. 通过这样的处理, 部分柔性问题就转化为完全柔性问题而便于解决.

应用新算法得到的解如下:

优化解 1:

$W_{id} = 75$, Max$(W_k) = 12$, Makespan = 15;

优化解 2:

$W_{id} = 73$, Max$(W_k) = 13$, Makespan = 16.

其中: W_{id} 表示机器总负载, Max(W_k) 表示关键机器负载. 图 1 和图 2 为这两个解的甘特图(数字的含义为"工件, 工序", 阴影部分表示机器闲置).

新算法与其他算法的比较如表 5 所示. 其中: "TD"表示 Chetouane[3] 的"暂时分解"方法, "GA"表示经典的遗传算法, "AL"和"AL + CGA"表示文献[3] 中的方法.

例 2 问题 15 × 10

该问题属于大规模完全柔性调度的例子, 具有 15 工件、10 机器、56 工序(具体见文献[4]).

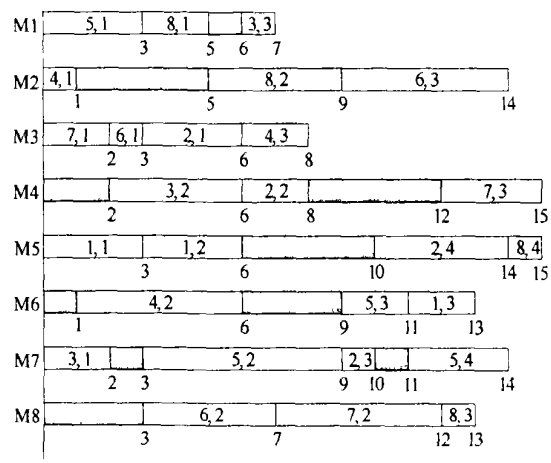

图 1 问题 8 × 8 优化解 1

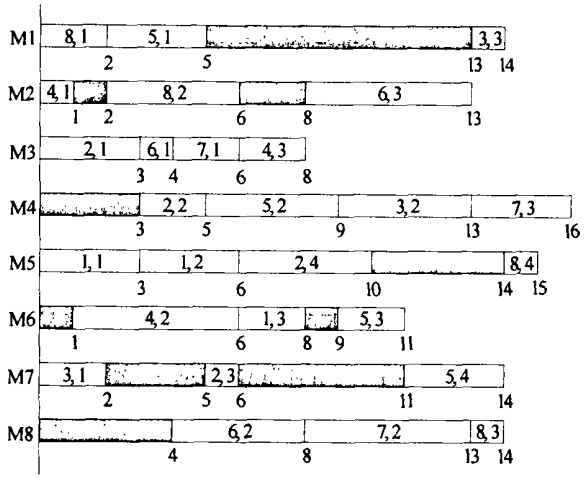

图 2　问题 8 × 8 优化解 2

表 5　不同算法结果比较

	TD	GA	AL	AL + CGA		PSO + SA	
制造周期	19	16	16	15	16	15	16
总负载	91	77	75	79	75	75	73

新算法获得的优化解为W_{td} = 91, Max(W_k) = 11, Makespan = 12. 其甘特图见图 3.

表 6 列出了与文献[4] 中的结果比较, 可以看出: 新算法使制造周期下降了11个单位, 目标函数值下降了约 10%. 这说明新算法对大规模问题同样能取得良好的效果.

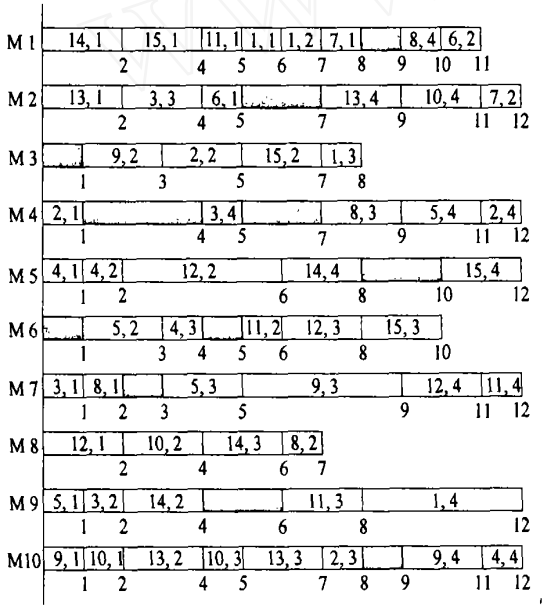

图 3　问题 15 × 10 优化解

表 6　针对问题 15 × 10 的结果比较

	AL + CGA		PSO + SA
Makespan	23	24	12
W_{td}	95	91	91
Max(W_k)	11	11	11

7　结　　语

本文提出了一种基于微粒群优化和模拟退火的混合优化新方法, 并将其应用于求解多目标柔性 Job-shop 调度问题. 通过计算实例说明: 该方法虽不能保证得到问题的最优解, 但可在较短时间内获得较好质量的解, 表现出了良好的求解性能和应用前景.

参考文献(References)

[1] Brandimarte P. Routing and scheduling in a flexible job shop by taboo search [J]. *Annals of Operations Research*, 1993, 41(22): 157-183.

[2] Tung L F, Li L, Nagi R. Multi-objective scheduling for the hierarchical control of flexible manufacturing systems [J]. *The Int J of Flexible Manufacturing Systems*, 1999, 11(4): 379-409.

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

[4] Kacem I, Hammadi S, Borne P. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic [J]. *Mathematics and Computers in Simulation*, 2002, 60(3-5): 245-276.

[5] Kennedy J, Eberhart R. Particle swarm optimization [A]. *Proc of the IEEE Int Conf on Neural Network* [C]. Perth, 1995: 1942-1948.

[6] Shi Y, Eberhart R. Empirical study of particle swarm optimization [A]. *Proc of Congress on Evolutionary Computation* [C]. Washington, 1999: 1945-1950.