文章编号: 1000-6788(2000)09-0041-07

一种新的求解 Flow Shop 问题的启发式算法

韦有双, 杨湘龙, 冯允成
(北京航空航天大学管理学院, 北京 100083)

摘要: 同顺序 Flow Shop 问题是一个著名的NP 难题, 至今尚未找到有效算法. 总体来讲, 求解该问题的启发式算法主要可分为规则式算法和迭代式算法两种. 对该问题有很多求解目标, 如最小加工周期(min makespan), 工件的最小平均在系统的停留时间(min mean flow time)等. 本文以求解最小加工周期为目标, 基于目前已知的性能最好的算法NEH 算法的基本思想, 提出了一种新的启发式算法-组合指标算法. 大量的数据实验表明, 新的算法具有很好的计算结果, 而且这种算法可以说是给出了求解 Flow shop 问题的一种新的思路和方向.

关键词: 排序; 最优化算法; 仿真; 加工车间的作业排序
中图分类号: TP391.9

Heuristic Algorithms for Flow Shop Scheduling Problem

WEI You-shuang, YANG Xiang-long, FENG Yun-cheng
(School of Management, Beijing University of Aeronautics and Astronautics, Beijing 100083)

Abstract: Flow shop scheduling problem is a well-known NP-hard problem. In this paper we first summarize the exist heuristic algorithms for flow shop problem which the objective is minimize makespan and then on the basis of NEH algorithm, we propose a new heuristic algorithm for flow shop to minimize the makespan. The numerical experiments show that the proposed algorithm has very good performance.

Keywords: scheduling; sequencing; flow shop scheduling problem; makespan

同顺序 Flow shop (Permutation Flow shop)问题可定义为: 已知有 m 台机床, n 种工件, 每个工件的加工工序都相同, 并且按照相同的顺序在各个机器上加工, 每个工件的加工时间是确定的. 不考虑复杂情况, 假设每种机床只有一台, 并且在有工件未加工完时, 后续工件不允许抢先占用机床. 问题的目标是最小化总加工周期, 即所有工件的最大完工时间. 该问题一般表示为 $n/m/P/F_{\max}$. 为描述问题, 定义 t_{ij} 为工件 j 在机器 i 上的加工时间, 令 S_{ij} 和 T_{ij} 分别表示工件 j 在机器 i 上的开工时间和完工时间, 于是有:

$$T_{ij} = S_{ij} + t_{ij} \tag{1}$$

$$S_{ij} = \max\{T_{i-1,j}, T_{i,j-1}\} \tag{2}$$

公式(2)表明工件 j 在它的前道工序 $i-1(T_{i-1,j})$ 及排在它前面的工件 $j-1(T_{i,j-1})$ 加工完之前是不能被加工的. 不断计算公式(1)(2)直到 $j=n$, $i=m$. 最后得到 T_{nm} 即为当前给定工件序的加工周期. 本文的目标就是要寻找一个工件序, 在这个工件序下, T_{nm} 为最小, 即求最小加工周期对应的加工顺序.

这个问题已经被证明[1]了是NP 难题. 其搜索空间为 $n!$. 当 n 较小时, 可采用穷举法或分支定界法等来求最优解. 然而当 n 大于 15 时, $n!$ 就已经是一个天文数字, 再采用穷举法或分支定界法等来求最优解已经几乎是不可能的. 因此目前的算法都是在合理的时间内寻找满意解.

在过去的几十年里发表了很多文章及书籍, 提出了很多算法. 这些算法可大致分为列举法、启发式算

收稿日期: 1999-01-28
资助项目: 国家自然科学基金(79430022); 航空基础科研基金资助(98J51094)

法两类．在启发式算法中, 又可分为规则式算法和迭代式算法两类．规则式启发式算法是指将所有工件按指定的一些规则排序, 然后计算出总加工时间即可; 而迭代式启发式算法基本上是结合某种规则不断地在整个问题空间中寻找一个新的优势点(序), 计算它的总加工周期, 比较优劣, 反复迭代直至满足某个条件为止．如遗传算法、模拟退火算法、Tabu 搜索算法等．

本文只讨论规则式启发式算法, 第一部分是现有的启发式算法综述, 第二节是本文提出的新的启发式算法, 第三节是实验结果对比, 第四节是算法分析．

1 现有启发式算法综述

启发式算法以其计算量小、算法简单并且能得到较好的解而吸引着很多的研究者．在过去的 40 年中, 出现了很多的启发式算法, 下面就简要地对现有求解最小加工周期的一些重要的算法做一简要综述．

1) Palmer 算法[2]

1965 年 D. S. Palmer 提出 Palmer．该算法提出按所谓斜度指标(slope order index)来排序工件．工件 j 的斜度指标按下式计算:

$$\lambda_j = \sum_{i=1}^{m}\left(i - \frac{m+1}{2}\right) t_{ij} \quad j = 1, 2, \cdots, n$$

将工件按 λ_j 降序排列求解, 即可得到近优解．

2) Gupta 算法[2]

该算法类似 Palmer, 但其斜度指标的定义不同, 他将著名的 Johnson 规则[3]中关于三种机器问题的一些条件考虑到斜度指标中．

3) CDS 算法[4]

1970 年 Campbell, Dudek 和 Smith 提出 CDS 算法．该算法将 $n/m/P/F_{\max}$ 问题虚拟为 $(m-1)$ 个两台机器问题, 应用 Johnson 规则, 得到 $(m-1)$ 个加工顺序, 然后取其最优者为近优解．

4) RA (Rapid Access Procedure) 算法[5]

1977 年 Dannenbring 提出 RA 算法．该算法是基于 CDS 算法的一个改进．该算法的目标是更快、更简单地算出一个近优解．它不再计算 $m-1$ 个虚拟的两台机器问题, 而只计算一个, 其加工时间由一个加权机制来确定．Dannenbring 近一步提出了两个修正过程 RACS (Rapid access with close order search) 和 RAES (Rapid access with extensive search) 以改进 RA 算法的解的质量．

5) 关键工件法[6]

1983 年陈荣秋提出关键工件法．该算法以总加工时间最长的工件作为关键工件．算法如下:

① 计算每个工件的总加工时间, 找出总加工时间最长的工件 C;

② 对其余工件, 若 $t_{1j} \leq t_{mj}$, 则按 t_{1j} 非减的顺序排列成部分顺序 S_a; 若 $t_{1j} > t_{mj}$, 则按 t_{mj} 非增的顺序排列成部分顺序 S_b;

③ 顺序 (S_a, C, S_b) 即为所求近优序．

6) NEH 算法[7]

1983 年 M. Nawaz, E. E. Enscore 和 I Ham 提出 NEH 算法．该算法是目前已知的求解最小加工周期时性能最好的算法之一．该算法基于这样的一个基本假设: 一个工件的总加工时间越长, 它应具有越高的优先权．算法具体描述如下:

① 对每个工件 j 计算它的总加工时间 T_j, $\quad T_j = \sum_{i=1}^{m} t_{ij}$

② 将工件按 T_j 递减顺序排列, 得到序列 P;

③ 选出在 P 序列中位于第 1, 第 2 位的两个工件, 分别计算这两个工件的两种排序下的 $F_{\max}$, 取其 $F_{\max}$ 小的排列, 将这两个工件的相对位置固定下来, 放入序列 Q 中; 令 $k=3$;

④ 从 P 序列中选出第 k 位的工件, 将其插入到 Q 序列中的 k 个位置中, 找出 $F_{\max}$ 最小的排列顺序, 放入序列 Q 中;

⑤ 若 $k = n$, 转步骤⑥, 否则, 令 $k = k + 1$, 转步骤④;

⑥ 序列 Q 即为近优排列 .

7) InsertionMethod 算法[8]

1989 年 Marino Widmer 和 Alian Hertz 提出 InsertionMethod 算法 . 该算法借鉴了求解 TSP 问题的思想, 给出了工件之间的"距离", 然后再确定近优序 . 具体算法如下:

① 定义工件间的距离:

$$d_{ab} = t_{a1} + \sum_{j=2}^{m} (m - j) \left| t_{aj} - t_{b,j-1} \right| + t_{bm}$$

② 找出 a, b 工件, 使得 $d_{ab} = \min_{i,j} d_{ij}$, 令 $Path = a \rightarrow b$;

③While 还有工件尚未排好 do

- 随机选择一个尚未排列的工件;
- 插入该工件, 使其对 path 的增量最小;

End While

8) SL 算法[9]

1993 年 S. Sarin 和 M. Lefoka 提出 SL 算法 . 该算法考虑重点是, 尽量减少各工件在最后一台机器上加工的空闲时间, 以期达到所有工件加工周期尽量小的目的 . 该算法实现较为复杂, 且在机器数较小时该算法性能很差 . 但在机器数很大时, 该算法有非常好的性能表现 . 具体算法可参考文献[12] .

9) SWH 算法[9]

1996 年沈英俊等提出 SWH 算法 . 该算法基于 SL 算法, 在外层加了一个 n (工件数) 次循环, 依次将每个工件作为第一位工件, 然后用 SL 算法计算, 最后在 n 个结果中取最优 . 该算法的计算结果肯定比 SL 算法好, 但其所需计算时间为 SL 算法的 n 倍 .

10) WSH 算法[9]

1996 年沈英俊等提出WSH 算法 . 该算法定义了一个新的排序指标:

$$\lambda_j = \frac{\sum_{i=1}^{m} i \cdot t_{ij}}{\sum_{i=1}^{m} t_{ij}}, \qquad j = 1, 2, \cdots, n$$

将工件按 λ_j 降序排列求解, 即可得到近优解 .

2　新的启发式算法

在第二节中介绍的各种算法中, 作者进行了大量的实验 .结果表明, NEH 算法在机器数不大 ($m < 200$) 且机器数与工件数之比 ($\rho = m/n$) 小于 2 时, 计算结果大部分都是好的; 当机器数很大 (如大于 200) 且 ρ 大于 2 时 SL 算法和WSH 算法结果很好 . 但在实际问题当中, 一般机器的种类不会很多, 而且当前的一些标准算例其给出的机器数也都较小, 所以在算法比较时暂不考虑 SL 算法, 本文将以 NEH 算法作为比较对象 .

仔细研究已有的成果可以看到, 由于 Flow Shop 问题的复杂性, 单靠给定一个规则然后按其排一次序是不可能得到很好的解的 . 性能很好的算法如 NEH, CDS 等都是在某种规则的基础上按一定的方式进行有限的迭代 . 已经发表的研究结果还表明, 工件的总加工时间及第一台机器和最后一台机器上的工件的加工时间对总加工周期具有较大的影响 . 一般来说, 当工件 j 的总加工时间越大, 它应优先往前排; 当工件 j 的 t_{1j} 较大时, 应将 j 尽量往前排; 当 t_{mj} 较小时, 应将 j 尽量往后排 . 然而这大都是些实验观察结果, 真正找到一种有理论依据的排序标准依然是非常困难的 .

既然前人的已有经验表明寻找某一个性能很好的通用排序指标目前看来是不太现实的, 那么能否多设计几个排序指标, 让每个排序指标均有其针对性, 比如重点考虑前面的机器上的加工时间的影响, 或重点考虑后面的机器上的加工时间的影响, 然后分别按照 NEH 算法的思想进行有限迭代, 最后取各指标计

算结果中最优的解作为最终结果．基于上述的一些考虑,我们设计了几个新的排序指标及相应算法．

在介绍新的算法之前,这里还要先介绍另外一个相关算法,Rajendran 算法[10]．该算法在 1993 年由 Rajendran 提出,它用于求解 Flow Shop 问题的工件在系统的最小停留时间(min total flow time)．该算法基于NEH 算法有限迭代的思想,定义了一个新的排序指标,它使用工件的加权时间和来排序,并且在插入工件时限制了插入的位置范围,从而降低了算法的计算时间．它与NEH 算法的差别在于:

* 对每个工件 j 计算它的加权时间和 T_j: $T_j = \sum_{i=1}^{m} [(m - i + 1) t_{ij}]$

* 插入位置不再是 k 个,而是 ρ 个位置: $\mathrm{int}[k/2] \leq \rho \leq k$

要注意该算法是用来求解 Flow Shop 问题的工件在系统的最小停留时间(min total flow time)的．

3 组合指标算法

本文提出的新的排序指标的定义及相应算法分别借鉴了 Rajendran 算法和NEH 算法的思想．具体算法如下:

1) 对每个工件,计算它的加权加工时间和 T_j^r(排序指标): ($j \in [1, n]$)

$$T_j^1 = \sum_{i=1}^{m} [(m - i + 1) t_{ij}] \qquad \text{RW 1}$$

$$T_j^2 = \sum_{i=1}^{m} [(m - i + 2) t_{ij}] \qquad \text{WYS1}$$

$$T_j^3 = \sum_{i=1}^{m} [(m^2 - (i - 1)^2) t_{ij}] \qquad \text{WYS2}$$

$$T_j^4 = \sum_{i=1}^{m} [(m - i + 1) \cdot i \cdot t_{ij}] \qquad \text{WYS3}$$

$$T_j^5 = \sum_{i=1}^{m} [(m + i - 1) t_{ij}] \qquad \text{WYS4}$$

2) 将工件按 T_j^r 递减顺序排列,得到序列 P;

3) 选出在 P 序列中位于第 1,第 2 位的两个工件,分别计算这两个工件的两种排序下的 $F_{\max}$,取其 $F_{\max}$小的排列,将这两个工件的相对位置固定下来,放入序列 Q 中;令 $k = 3$;

4) 从 P 序列中选出第 k 位的工件,将其插入到 Q 序列中的 k 个位置中,找出其中 $F_{\max}$最小的排列顺序,放入序列 Q 中;

5) 若 $k = n$,转步骤 6),否则,令 $k = k + 1$,转步骤 4);

6) 序列 Q 即为近优排列．

为叙述方便,以下称每个指标所对应的算法分别为 RW 1,WYS1,WYS2,WYS3,WYS4．注意 RW 1 算法中使用的指标同 Rajendran 算法,但在算法实现中,Rajendran 算法的插入位置限制在 ρ 个位置($\mathrm{int}[k/2] \leq \rho \leq k$),而 RW 1 算法仍采用NEH 算法的思想按 k 个位置插入计算．这是因为在数据实验中发现按 ρ 个插入位置的办法来求解最小加工周期得到的结果很差,Rajendran 算法毕竟是用来求解工件在系统的最小停留时间(min total flow time)的,是否在该方面结果较好作者没有进行这方面的实验．

算法 RW 1,WYS1,WYS2 中指标的设计目的是加强前面机器上的工件加工时间的影响作用,算法 WYS3 指标的设计目的是加强中间机器上的工件加工时间的影响作用,WYS4 中指标的设计目的是加强后面机器上的工件加工时间的影响作用．这样各有侧重,综合起来应能得到更好的结果．

4 实验结果

我们使用了 110 个标准的测试数据进行了算法实验,这些数据来自 Taillard[11],也可从网上获得．编程使用ANSIC,在 Pentium Ⅱ 266 上运行．这里以NEH 算法得到的结果作为比较标准．

从实验的结果来看, 各子算法均具有较好的计算结果, 在 Talliard 的 110 例实验数据中, 算法 RW 1 有 41 例结果好于 NEH 算法, 占 37. 3%; WYS1 有 41 例结果好于 NEH 算法, 占 37. 3%; WYS2 有 47 例结果好于 NEH 算法, 占 42. 7%; WYS3 有 46 例结果好于 NEH 算法, 占 41. 8%; WYS4 有 48 例结果好于 NEH 算法, 占 43. 6%; 而取这些算法的并集得到的最终结果, 有 82 例结果好于 NEH 算法, 占 74. 5%.

这个结果证实了设计指标时的思路, 分别设计不同侧重点的指标, 然后取最优. 可得到相当令人满意的结果.

图 1　Taillard 110 算例组合指标算法的最优结果与 NEH 比较

图 1 给出的是这 110 个标准算例的组合指标算法的最优结果与 NEH 算法的结果相对偏差图. 图中百分比计算公式为:

$$\frac{(WYS - NEH)}{NEH} * 100\%$$

图 1 中 X 轴为 ta001 到 ta110 算例, Y 轴为二者偏差百分比, 负方向为组合指标算法好. 从图中可以看出, 在那些较 NEH 算法差的算例中, 组合指标算法与 NEH 算法的偏差也很小, 计算结果依然可认为是很好的(相对于其他的启发式算法).

为了测试组合指标算法的性能, 作者又进行了大量的实验. 表一中给出的是另一组实验结果. 该组实验针对不同的规模, 均生成 120 个算例, 工件的加工时间随机产生, 在[1, 99]区间内均匀分布. 所用的问题生成程序取自 Taillard[11], 随机数种子值不变, 仅修改了机器数和工件数. 然后对产生的 120 个算例分别用 NEH 算法、组合指标算法进行计算. 表 1 中的百分比是指在 120 个算例中组合指标算法结果比 NEH 算法结果好的算例的百分比.

表 1　大量测试数据中组合指标算法占优百分比

Prob. Size ($n \times m$)	Prob. Num.	RW 1	WYS1	WYS2	WYS3	WYS4	Min(*)
20×5	120	60%	60. 8%	66. 7%	67. 5%	65. 8%	92. 5%
20×10	120	52. 5%	52. 5%	58. 3%	47. 5%	51. 7%	88. 3%
20×20	120	50%	51. 7%	53. 3%	46. 7%	50%	85. 8%
50×20	120	40. 8%	40%	41. 7%	51. 7%	50%	77. 5%
50×30	120	41. 7%	43. 3%	47. 5%	45%	50. 8%	80%
50×50	120	37. 5%	40. 8%	50%	50%	43. 3%	79. 2%
100×20	120	41. 7%	46. 7%	47. 5%	39. 2%	44. 2%	79. 2%
100×50	120	42. 5%	45%	44. 2%	46. 7%	56. 7%	74. 2%
100×100	120	34. 2%	34. 2%	35. 8%	46. 7%	45%	79. 2%
200×20	120	28. 3%	25. 8%	34. 2%	22. 5%	34. 2%	62. 5%
200×50	120	32. 5%	31. 7%	35. 8%	32. 5%	42. 5%	69. 2%

注: min(*)是指取各算法中的最小值, 即 min(RW 1, WYS1, WYS2, WYS3, WYS4)

从表 1 的结果看, 各个子算法的性能也都是不错的, 在问题的规模较小时, 各个计算结果都有 40%以上比NEH 算法好 .然而随着问题规模的增大, 各子算法的优势有明显的下降, 整体组合算法的优势也有所降低(但相对于NEH 算法的偏差依然并不大) .

5 算法分析

从数据实验结果来看, 组合指标算法具有相当好的结果, 在大规模情况下也能有约 70% 的解好于NEH 算法 . 本文中共使用了 5 个子指标, 在进一步研究, 应用中, 可适当选取若干个子指标(当然也可以将NEH 算法包括进来), 以期得到更好的解(子指标的选取原则见本节后面的论述) .

从算法的时间复杂度来看, 由于每个子算法的复杂度等同于NEH 算法, 所以整个组合指标算法的时间复杂度为NEH 算法的 r 倍(r 为选取的子指标的个数) . Taillard[2]已经证明,NEH 算法复杂度可降低为 $O(n^2m)$, 故组合指标算法的复杂度为 $O(r \cdot n^2m)$.

本文中给出了 5 个子指标 .这些子指标的选取并不是随意的, 而是有一定目的性的 .前面已经提到, 设计算法时希望各个子指标各有侧重, 从而组合出更好的结果 .NEH 算法的成功提示我们, 工件的总加工时间对加工周期影响颇大, 而NEH 算法中, 工件 j 的总加工时间 $T_j = \sum_{i=1}^{m} t_{ij}$ 的计算, 实际上相当于 t_{ij}的系数为 1, 即平等看待每一道工序 . 如果我们调整这个系数, 使其按照某种预定规律变化, 从而或是加强前面工序的作用, 或是加强后面工序的作用 . 本文设计的几个指标正是基于这样的思想, 参见图 2 . 图 2 中给出的是WYS1～WYS4 算法中指标系数的形态图 .WYS1 指标系数对应形态为一斜率为负的直线, 它表明要放大前面工序所占的比重, 且为线性关系;WYS2 指标系数对应形态为一向下的半抛物线, 它的作用同样是要放大前面工序所占的比重, 但放大比率不同(非线性的);WYS3 指标系数对应形态为一向下的抛物线, 它的作用是要放大中间工序所占的比重, 前后工序以对称比率放大(非线性的);WYS4 指标系数对应形态为一斜率为正的直线, 它表明要放大后面工序所占的比重(线性放大) .

文中所给的指标 RW 1 和WYS1 的系数是同斜率的直线, 按照前面的分析, 这样的指标产生的结果应该差别很小, 因为其对工件的总加工时间的影响是等比例的 . 实验结果也证实了这一点, 从表 1 中可看到, RW 1 和WYS1 算法的结果相差无几 . 文中同时给出这两指标主要是为了做对比, 这说明在选取指标系数时不应该选择同族曲线 .

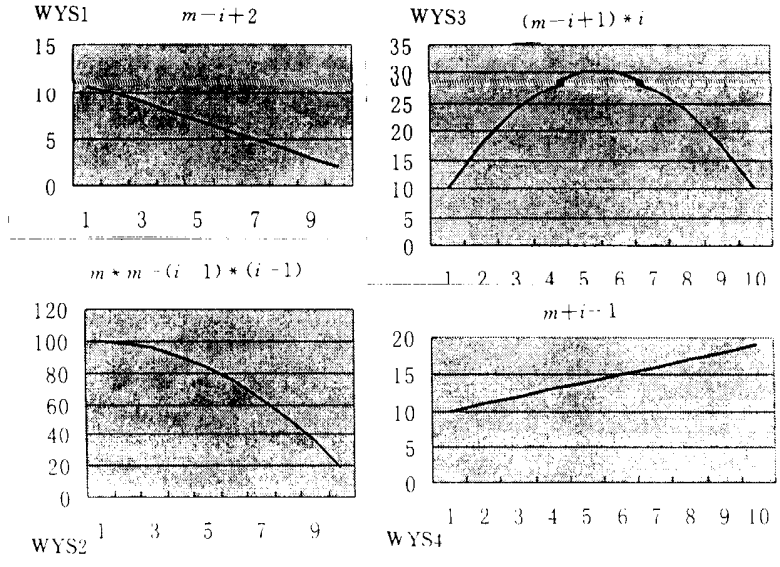

图 2 各指标系数形态对比

既然实验表明指标系数不应选择同族曲线,那么应该选择哪几种类型的曲线呢? 假如选择直线,斜率该如何确定?如果选二次或更高次曲线,其曲线特征应该具备哪些条件?半抛物线与直线哪个效果更好?表一中给出的数据仅是一组实验,对上述问题尚有待进行更广泛的数据实验分析及理论探讨.

6　结论

在解决优化问题的算法中,启发式算法以其简单、快捷且具有相对好的结果而吸引着众多的研究者.本文针对同顺序 Flow Shop 组合优化问题,以求解该问题的最小加工周期为目标,提出了一种新的启发式算法-组合指标法.该算法实际是一种思想,可包含一组共 r 个指标,每个子指标都各有侧重点,然后按照 NEH 算法的基本思路进行计算,最后取各子指标中最优的作为最终解.该算法的复杂度为 $O(r \cdot n^2 m)$.作者在文中选取了 5 个子指标,大量的数据实验证明,该算法具有很好的性能,60%以上的结果都比目前已知的性能最好的 NEH 算法要好,而且相对于 NEH 算法差的解偏差也很小.进一步的研究工作在于寻找一些最佳系数曲线组合,以期用尽可能少的指标获得更好的解.

致谢　感谢北京航空航天大学管理学院的黄海军教授为作者提供了 Sarin 先生寄给他的 SL 算法的 FORTRAN 源程序.

参考文献:

[1] Garey M R, Johnson D S, Sethi R. The Complexity of flow shop and job-shop scheduling[J]. Math. Ops. Res. 1976, 10(1): 117～129.

[2] Taillard E D. Some efficient heuristic methods for the flow shop sequencing problem[J]. European Journal of Operation Research 1990, 47: 65～74.

[3] Michael Pinedo. Scheduling Theory Algorithms and Systems[M], Prentice Hall, 1993.

[4] Campbell H G, Dudek R A, Smith M L. A heuristic algorithm for the n job, m machine sequencing problem[J]. Management Science, 1970, 16(10): B630～637.

[5] Dannenbring D G. An evaluation of flow shop sequencing heuristics[M]. Management Science, 1977, 23(11): 1174～1182.

[6] 陈荣秋. 排序的理论与方法[M]. 武汉: 华中理工大学出版社, 1987.

[7] Nawaz M, Enscore E E, Ham I. A heuristic algorithm for the m-machine, n-job flow shop sequencing problem[J]. OMEGA, International Journal of Management Science, 1983, 11(1): 91～95.

[8] Marino W, Alain H. A new heuristic method for the flow shop sequencing problem[J]. European Journal of Operation Research 1989, 41: 186～193.

[9] 沈英俊　同排序 flow shop 排序问题的启发式算法研究[D], 北航硕士学位论文, 1996

[10] Rajendran C. Heuristic algorithm for scheduling in a flow shop to minimize total flow time[J]. International Journal of Production Economics, 1993, 29: 65～73.

[11] Taillard E D. Benchmarks for basic scheduling problems[J]. European Journal of Operation Research 1993, 64(1): 278～285.

[12] Rajendran C, Zeigler H. An efficient heuristic for scheduling in a flow shop to minimize total weighted flow time of jobs[J]. European Journal of Operation Research 1997, 103: 129～138.

[13] Hoon-Shik Woo, Dong-Soon Yim. A heuristic algorithm for mean flow time objective in flow shop scheduling[J]. Computers Ops. Res. 1998, 25(3): 175～182.