柔性 Flow-Shop 调度的遗传算法优化

周辉仁[1],唐万生[1],魏颖辉[2]
ZHOU Hui-ren[1],TANG Wan-sheng[1],WEI Ying-hui[2]

1.天津大学 电气与自动化工程学院,天津 300072
2.辽宁科技学院 管理系,辽宁 本溪 117022
1.School of Electrical Engineering & Automation,Tianjin University,Tianjin 300072,China
2.Department of Management,Liaoning Institute of Science and Technology,Benxi,Liaoning 117022,China

ZHOU Hui-ren,TANG Wan-sheng,WEI Ying-hui.Optimize flexible flow-shop scheduling using genetic algorithm.Computer Engineering and Applications,2009,45(30):224-226.

Abstract: Flexible Flow-shop Scheduling Problem (FFSP) is expansion of general flow-shop scheduling problem.It is more complex than general flow-shop scheduling problem because there are parallel machines on some operations.In order to efficiently solve this problem,a new method solving flexible flow-shop scheduling problem based on genetic algorithm is proposed.A new improved encoding and decoding with matrix method for the flexible flow-shop scheduling problem are proposed.These operators can easily keep the feasibility of solution.Finally,an example of production scheduling problem for metalworking workshop in a car engine plant is simulated.Through comparison,the results show the effectiveness of the algorithm.
Key words: flexible flow-shop scheduling;genetic algorithm;encoding method;decoding with matrix form

摘 要: 柔性 Flow-shop 调度问题(Flexible Flow-shop Scheduling Problem,FFSP)是一般 Flow-shop 调度问题的推广,由于在某些工序上存在并行机器,所以比一般的 Flow-shop 调度问题更复杂。为了有效地解决柔性 Flow-shop 调度问题,用遗传算法求解,给出了一种改进的编码方法,能够保证个体的合法性;并根据编码方法提出了矩阵解码方法。最后以某汽车发动机厂金加工车间的生产调度实例进行仿真,通过比较表明了算法的有效性。
关键词: 柔性 Flow-shop 调度;遗传算法;编码方法;矩阵解码
DOI: 10.3778/j.issn.1002-8331.2009.30.066 **文章编号:** 1002-8331(2009)30-0224-03 **文献标识码:** A **中图分类号:** TP278

1 引言

自从 Johnson 1954 年发表第一篇关于 Flow-shop 调度问题(Flow-shop Scheduling Problem,FSP)的文章[1]以来,许多学者对纯 FSP 作了深入地研究。但还没有一个求解最优解的简明算法,遗传算法已经被成功地应用于求解 FSP[2-4]。

柔性 Flow-shop 调度问题(Flexible Flow-shop Scheduling Problem,FFSP)也称为混合 Flow-shop 调度问题,是一般 FSP的推广,但更复杂。其特征是在某些工序上存在并行机器,因此非常具有代表性,相当普遍地存在于化工钢铁制药等流程工业中被称为柔性流水线。

柔性 Flow-shop 调度问题可描述为:需要加工多个工件,所有工件的加工路线都相同,都需要依次通过几道工序,在所有工序中至少有一个工序存在着多台并行机器。需要解决的问题是确定并行机器的分配情况以及同一台机器上工件的加工排序,目标是最小化最大流程时间。

对于柔性 Flow-shop 调度问题已经提出分支定界算法、启发式算法等算法,但这些算法都只能解决规模较小的生产调度问题。随着智能优化算法的发展,遗传算法等应用于柔性 Flow-shop 调度问题[5-9]。

为了有效地解决柔性 Flow-shop 调度问题,提出了一种改进的染色体编码方法,并提出一种基于矩阵的解码方法,即找到一个最优调度,使完工时间最小。仿真结果证明,在处理柔性 Flow-shop 调度问题时,提出的编码方法能达到满意的效果,而且鲁棒性好,并行效率高,具有实际应用的价值。

2 遗传算法设计

将遗传算法应用于柔性 Flow-shop 调度问题中的关键是采用有效的编码和解码方式以及适当的交叉、变异操作。遗传算法对种群重复地进行选择、交叉、变异等基本遗传操作,不断产生出比父代更适应环境的新一代种群,直到满足要求条件为止。

2.1 个体编码

文献[7-8]中提出了一种编码矩阵,该文的编码方法是在此基础上进行改进,改进后的编码矩阵意义更为直观,不需要再

基金项目: 辽宁省教育厅科研课题资助(No.20060439)。
作者简介: 周辉仁(1972-),男,博士,博士后在站,CCF 会员,主要研究领域为工业工程、智能优化理论与方法、决策理论与方法;唐万生(1962-),男,教授,博士生导师,主要研究领域为复杂系统建模与控制等;魏颖辉(1970-),女,博士,副教授,主要研究领域为企业管理、人力资源。
收稿日期: 2008-11-24 **修回日期:** 2009-02-10

在染色体编码中加分隔符，从而使染色体编码的长度相应缩短，在染色体交叉、变异操作运算过程中直接对整条染色体进行，而不需要专门分段进行操作。

这里，FFSP 的编码方法同样巧妙地利用了矩阵的元素和位置信息表示 FFSP 的工序之间的约束关系，使得产生的每个染色体对应一个可行的调度而且在进行遗传操作时也不会产生非法解。

假设要加工 N 个工件，每个工件都要依次经过 S 个加工工序，每个工序的并行机数为 $M_i(i=1,\cdots,S)$。首先构造一个 $S\times N$ 维的 FFSP 的编码矩阵：

$$\boldsymbol{A}_{S\times N}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{S1} & a_{S2} & \cdots & a_{SN} \end{bmatrix} \tag{1}$$

其中，编码矩阵第 1 行的元素 a_{1j} 为区间 $(1,M_1+1)$ 上的一个实数，表示工件 j 的第 1 个工序在第 $\mathrm{Int}(a_{1j})$ 台并行机上加工，函数 $\mathrm{Int}(x)$ 表示对实数 x 取整；编码矩阵第 $m(m>1)$ 行的元素 a_{mj} 为区间 $(1+\sum_{i=1}^{m-1}M_i,1+\sum_{i=1}^{m}M_i)$ 上的一个实数，表示工件 j 的第 m 个工序在第 $\mathrm{Int}(a_{mj})$ 台并行机上加工。显然，可能会出现 $\mathrm{Int}(a_{ij})=\mathrm{Int}(a_{ik})$，$j\neq k$，这表明多个工件在同一台机器上加工同一个工序。这时，假如是第一个工序($i=1$)，则按照 a_{1j} 的升序来加工工件。假如不是第一道工序($i>1$)，则根据每个工件的前一个工序的完成时间来确定其加工顺序，前一个工序先完成的先加工。假如完成时间相同，则也按照 a_{ij} 的升序来加工。

根据上述编码矩阵(1)可以确定染色体。染色体由 S 个小段组成，每个小段包括 N 个基因。即由编码矩阵的每一行组成一个小段，表示不同的工序。因此染色体的长度为 $S\times N$。染色体可表示为：

$$\mathrm{Chrom}_i=[a_{11},a_{12},\cdots,a_{1N},a_{21},a_{22},\cdots,a_{2N},\cdots,a_{S1},a_{S2},\cdots,a_{SN}] \tag{2}$$

由于该文在矩阵(1)中对元素的取值范围进行了改进，这与文献[7-8]是不同的，染色体的长度比文献[7-8]也减少了($N-1$)个基因。

如表 1 所示，有 3 个工件、3 道工序、各工序的并行机器数分别为 3、2、2 的柔性 Flow-shop 调度问题。

表 1　工件在各个机器上的加工时间

工件	工序 1			工序 2		工序 3	
	机器 1	机器 2	机器 3	机器 4	机器 5	机器 6	机器 7
1	2	2	3	4	5	6	7
2	6	5	4	3	4	4	7
3	3	5	4	6	5	3	2

假设产生的编码矩阵如下所示：

$$\boldsymbol{A}=\begin{bmatrix} 2.1 & 2.4 & 1.9 \\ 4.6 & 5.1 & 5.3 \\ 6.1 & 7.4 & 6.2 \end{bmatrix} \tag{3}$$

2.2　个体解码

在柔性 Flow-shop 调度问题中，解码是一项非常重要的工作，提出一种矩阵形式的解码方法。这里，以表 1 所示的柔性 Flow-shop 调度问题为例，对编码矩阵 $\boldsymbol{A}$ 进行解码。

对编码矩阵 $\boldsymbol{A}$ 的各个元素分别取整，得到如下矩阵：

$$\boldsymbol{B}=\begin{bmatrix} 2 & 2 & 1 \\ 4 & 5 & 5 \\ 6 & 7 & 6 \end{bmatrix} \tag{4}$$

根据编码各工序上的并行机编号规则，由矩阵 $\boldsymbol{B}$ 可得到各工件与机器的对应关系：

工件 1 的 3 个工序分别在机器[2　4　6]上加工；

工件 2 的 3 个工序分别在机器[2　5　7]上加工；

工件 3 的 3 个工序分别在机器[1　5　6]上加工。

得到机器的分配情况后，根据前面的规则进行确定同一机器上加工工件的先后顺序为：

机器 1：仅是工件 3 的第 1 个工序；

机器 2：工件 1 的第 1 个工序，工件 2 的第 1 个工序；

机器 3：闲置；

机器 4：工件 1 的第 2 个工序；

机器 5：工件 2 的第 2 个工序；工件 3 的第 2 个工序；

机器 6：工件 1 的第 3 个工序；工件 3 的第 3 个工序；

机器 7：工件 2 的第 3 个工序。

在计算机编程时，算法流程如下：

(1)根据表 1 生成工件在各个机器上的加工时间矩阵 $\boldsymbol{T}$。其中，矩阵的列分别表示工件，行分别表示机器。前 3 行表示相应工件的第 1 个工序；第 4~5 行表示相应工件的第 2 个工序；第 6~7 行表示相应工件的第 3 个工序。

$$\boldsymbol{T}=\begin{bmatrix} 2 & 6 & 3 \\ 2 & 5 & 5 \\ 3 & 4 & 4 \\ 4 & 3 & 6 \\ 5 & 4 & 5 \\ 7 & 4 & 3 \\ 6 & 7 & 2 \end{bmatrix} \tag{5}$$

(2)根据式(4)所示的矩阵 $\boldsymbol{B}$ 生成工序选取的机器矩阵 $\boldsymbol{S}$。其中，元素为“1”表示相应工序选取该机器，元素为“0”表示相应工序未选取该机器。

$$\boldsymbol{S}=\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{6}$$

在矩阵 $\boldsymbol{S}$ 中，前 3 行表示：对于第 1 个工序，工件 1 选取机器 2、工件 2 选取机器 2、工件 3 选取机器 1；第 4~5 行表示：对于第 2 个工序，工件 1 选取机器 4、工件 2 选取机器 5、工件 3 选取机器 5；第 6~7 行表示：对于第 3 个工序，工件 1 选取机器 6、工件 2 选取机器 7、工件 3 选取机器 6。

(3)将式(5)所示的矩阵 $\boldsymbol{T}$ 和式(6)所示的矩阵 $\boldsymbol{S}$ 点乘得到矩阵 $\boldsymbol{TT}$

$$\boldsymbol{TT}=\begin{bmatrix} 0 & 0 & 3 \\ 2 & 5 & 0 \\ 0 & 0 & 0 \\ 4 & 0 & 0 \\ 0 & 4 & 5 \\ 7 & 0 & 3 \\ 0 & 7 & 0 \end{bmatrix} \tag{7}$$

矩阵表示了相应工件的工序选择的机器及其在该机器上的加工时间。

(4)根据式(3)所示矩阵 $\boldsymbol{A}$ 的含义和式(7)所示矩阵 $\boldsymbol{TT}$ 生成相应工序在所选机器上的完工时间矩阵 $\boldsymbol{ZT}$。

$$\boldsymbol{ZT}=\begin{bmatrix}0&0&3\\2&7&0\\0&0&0\\6&0&0\\0&11&16\\13&0&19\\0&18&0\end{bmatrix} \tag{8}$$

由于矩阵中元素不为 0 的数值表示的是相应工件每个工序在所选机器上的最早完成时间,所以矩阵中元素最大的数即为该条染色体所表示的柔性 Flow-shop 调度问题的最大完成时间:C_i=19。

由于柔性 Flow-shop 调度问题是求解最小化最大完成时间,因此将所有染色体按上述步骤解码后所求得数值的最小数值为该次迭代的最小化的最大完成时间, 染色体通过不断选择、交叉和变异操作,最终求得最优调度。

2.3 群体规模选择

合适的群体规模对遗传算法的收敛具有重要意义。群体太小难以求得满意的结果,群体太大则计算复杂。根据经验,群体规模一般取 10~160。

2.4 适值函数

遗传算法进行选择操作时会出现欺骗问题[10-11]:在遗传进化的初期,通常会产生一些超常的个体,若按照比例选择法,这些异常个体因竞争力太突出而控制了选择过程,影响算法的全局优化性能;在遗传进化的后期,即算法接近收敛时,由于种群中个体适应度差异较小时,继续优化的潜能降低,可能获得某个局部最优解。适值函数设计不当有可能造成这种问题的出现。

由于优化目标为最小化最大完成时间,因此令目标函数作指数变换得到适值函数:

$$f=\alpha\exp(-\beta*C_i) \tag{9}$$

其中,α,β 为正实数,C_i 为最大完成时间。

2.5 选择

选择是用来确定重组或交叉个体,以及备选个体将产生多少个子代个体。选择的第一步是计算适值,采用按比例的适应度分配,是利用比例于各个个体适应度的概率决定其子孙的遗留可能性。若有 M 个个体,其中某个个体 i,其适值为 f_i,则其被选择的概率表示为:

$$P_i=\frac{f_i}{\sum_{k=1}^{M}f_k} \tag{10}$$

然后对各个染色体计算它们的累积概率,如第 k 个个体的累积概率为:

$$q_k=\sum_{i=1}^{k}p_i \tag{11}$$

第二步用轮盘赌选择法进行选择。为了选择交配个体,需要进行多轮选择,每一轮产生一个[0,1]均匀随机数,将该随机数作为选择指针来确定备选个体。

2.6 交叉与变异

交叉在遗传操作中起核心作用,交叉概率较大可增强遗传算法开辟新搜索空间的能力,但性能好的基因串遭到破坏的可能性较大,算法收敛速度降低且不稳定;若交叉概率较小,则遗传算法搜索可能陷入迟钝状态。由于这里采用实值编码,为保证交叉后产生新的参数值,并开辟出新的搜索空间,这里的交叉操作采用离散重组方式[12],即在个体间交换变量值,对每个变量,贡献给子代变量值的父代是随机的以相同的概率挑选的。

变异在遗传操作中属于辅助性的搜索操作,主要目的是维持群体的多样性。这里采用偏置变异,以一定的概率给变异位基因加一个从偏置区域中随机选取的数值,并保证变异后的基因在自己的取值范围内。

3 实例仿真和比较

3.1 实例

为便于比较,采用文献[7]中的汽车发动机厂金加工车间调度实例。

某汽车发动机厂金加工车间要加工 12 个工件, 每个工件都有车、刨、磨 3 个工序,现有 3 台车床,2 台刨床,4 台磨床,每台机床的加工能力不同,具体加工时间如表 2 所示。

表 2 工件在各个机器上的加工时间

工件	工序 1			工序 2		工序 3			
	机 1	机 2	机 3	机 4	机 5	机 6	机 7	机 8	机 9
1	2	2	3	4	5	2	3	2	3
2	4	5	4	3	4	3	4	5	4
3	6	5	4	4	2	3	4	2	5
4	4	3	4	6	5	3	6	5	8
5	4	5	3	3	1	3	4	6	5
6	6	5	4	2	3	4	3	9	5
7	5	2	4	4	6	3	4	3	5
8	3	5	4	7	5	3	3	6	4
9	2	5	4	1	2	7	8	6	5
10	3	6	4	3	4	4	8	6	7
11	5	2	4	3	5	6	7	6	5
12	6	5	4	5	4	3	4	7	5

在仿真中,交叉概率 P_c=0.75,变异概率 P_m=0.1,种群规模为 100,最大迭代次数为 100。分别随机仿真 10 次,运行结果如表 3 所示,所求最小化的最大完成时间的最优结果为 26。

表 3 10 次运行结果

次数	1	2	3	4	5	6	7	8	9	10
优化值	30	27	26	27	29	27	26	27	26	28

其中,得到最好的其中一条染色体是

[1.247 3 2.956 0 1.732 3 2.993 4 1.488 7 3.807 5 2.201 6 3.874 1 1.509 3 3.436 4 2.609 7 3.338 5 5.168 0 4.345 6 4.704 2 5.913 6 5.342 5 4.523 6 4.197 1 5.990 0 4.250 2 4.457 0 4.220 9 5.882 3 9.139 9 8.838 9 6.867 8 7.472 4 7.302 5 7.624 0 6.703 7 9.797 1 8.451 5 6.716 4 8.138 1 9.483 2]。

相应的甘特图如图 1 所示,图 1 中符号第 1 个数字表示工件,第 2 个数字表示工序。

3.2 比较

在实例中,该文中的方法求得的最优调度最小完成时间为 26,而文献[7-8]中所求得的结果为 29。该文方法优于文献[7-8]

(下转 233 页)