基于改进变异算子的遗传算法求解柔性作业车间调度①

A Genetic Algorithm with Modified Mutation Operator for the Flexible Job Shop Scheduling Problem

尹作海　邱洪泽　周万里　(山东大学 计算机科学与技术学院 山东 济南 250101)

摘　要： 柔性作业车间调度问题是对传统车间调度问题的扩充，它更接近于现实的生产调度问题。针对柔性作业车间调度的特点，设计了基于关键工序的变异算子，使变异集中于关键路径，从而提高了变异过程的效率。还采用二向量编码、初始种群定位法和 POX 交叉算子，设计了新的应用于柔性作业车间调度的遗传算法，并通过实验验证了算法的有效性。

关键词： 遗传算法 柔性作业车间调度 关键工序 变异算子

柔性作业车间调度问题(Flexible Job Shop Scheduling Problem, FJSP)是传统作业车间调度问题的扩展，是实际生产中迫切需要解决的一类问题。在传统的作业车间调度问题中，工件的每个工序只能在一台确定的机器上加工。而在柔性作业车间调度中，每个工序可以在多台机器上加工，并且在不同的机器上加工所需的时间不同。柔性作业车间调度问题减少了机器约束，扩大了可行解的搜索范围，增加了问题的难度。

相对传统作业车间调度问题(Job Shop Scheduling Problem, JSP)，FJSP 更为复杂，因为首先要为每个工序分配合理的机器，然后才是工序调度问题。FJSP 是对 JSP 的扩展，由于已经证明 JSP 是 NP-hard 问题[1]，所以 FJSP 也是 NP-hard 问题。

1　柔性作业车间调度问题的描述

柔性作业车间调度问题的描述如下：一个加工系统有 m 台机器，要加工 n 种工件，每个工件 i 包含了 ni 个顺序化的工序($O_{i1},O_{i2},\cdots,O_{ini}$)。FJSP 不再指定工序 O_{ik} 的处理机器，而是允许它从可行机器集 Aik 中任选一台进行处理，工序 O_{ik} 在机器 j 上的处理时间为 $P_{ikj} > 0$。调度目标是为每个工序选择最合适的机器、确定每台机器上各工件工序的最佳加工顺序及开工时间，使系统的某些性能指标达到最优。针对实际生产的需要，这里考虑常用的三种性能指标：所有工件的最后完工时间(makespan)最小、每台机器上最大工作负载最小，和在所有机器上的总工作负载最小。

柔性作业车间调度问题根据资源限制条件的不同，可分为完全柔性作业车间调度和偏柔性作业车间调度两类。在完全柔性车间调度中，每一个工序都可以在任何一台机器上加工；而在偏柔性车间调度中，某些工序将只能选择部分机器。由于可以把偏柔性看做是部分机器上加工时间无穷大的完全柔性车间调度，本文中我们只给出完全柔性的情况。一个包括 3 个工件、4 台机器的完全柔性作业车间调度的加工时间表如表 1 所示。

柔性作业车间调度问题的求解过程包括两个部分：用来选择各工序加工机器的设备分配过程和确定每台机器上工件先后顺序的工序调度过程。

2　遗传算法设计

遗传算法是一种借鉴生物界自然选择和遗传进化机制的高度并行、随机、自适应的群体优化算法。遗

① 收稿时间:2009-01-13

表 1　柔性作业车间调度加工时间表

	M_1	M_2	M_3	M_4
O_{11}	7	6	4	5
O_{12}	4	8	5	6
O_{13}	9	5	4	7
O_{21}	2	5	1	3
O_{22}	4	6	8	4
O_{23}	9	2	2	2
O_{31}	8	5	3	5
O_{32}	3	5	8	3

传算法依据适应度函数，对初始种群中的个体施加遗传操作，模拟生物基因的遗传和变异，实现个体结构重组的迭代处理。在这一过程中，初始种群中的个体一代代得以优化并逐步逼近最优解。

2.1 编码

用遗传算法解决传统的作业车间调度问题时一般使用基于工序的编码方法[2]。但是柔性作业车间调度问题不仅要确定工序的加工顺序，还需为每个工序选择一台合适的机器，仅采用基于工序的编码方法不能得到问题的解。因此，对于柔性作业车间调度问题，遗传算法的编码由两部分组成，第一部分为基于工序的编码，用来确定工序的加工先后顺序；第二部分为基于机器分配的编码，用来表示每道工序的加工机器。融合这两种编码方法，即可得到柔性作业车间调度问题的一个可行解。

工序编码	1	2	2	1	2	1	3	3
机器分配编码	3	4	4	1	3	2	3	1

图 1　柔性作业车间调度的染色体

图 1 为表 1 中柔性车间调度的一个染色体，其长度等于工序总数。每个工件的工序都用相应的工件号表示，根据它们在染色体中出现的顺序加以解释。如图中工序编码部分第一次出现的“1”表示工件 1 的第一个工序，第二次出现的“1”表示工件 1 的第二个工序，整个染色体所对应的工序序列为(O_{11}, O_{21}, O_{22}, O_{12}, O_{23}, O_{13}, O_{31}, O_{32})。在机器分配编码中的每个基因则表示对应工序的加工机器。

2.2 初始种群

初始种群采用 Kacem 等人提出的初始种群定位法[3]生成，该方法在为工序分配机器的同时还考虑了机器的负载情况。初始种群定位法在每个工序的可行机器集 Aik 中寻找具有最短加工时间的机器，并将其做为该工序的加工机器。然后将这个最短加工时间加到时间表中同一列的其它项上，即对机器负载进行更新。表 2 给出了对表 1 中 FJSP 进行设备分配的过程和结果，其中黑体表示负载的更新。

表 2　运用定位法进行设备分配

	M	M	M	M		M	M	M	M
O_{11}	7	6	**4**	5	.	7	6	**4**	5
O_{12}	**4**	8	9	6		**4**	8	5	6
O_{13}	13	5	8	7		9	**5**	4	7
O_{21}	6	5	5	3	....	2	5	1	**3**
O_{22}	8	6	12	4		4	6	8	**4**
O_{23}	13	2	6	2		9	2	**2**	2
O_{31}	12	5	7	5	...	8	5	**5**	5
O_{32}	7	5	12	3		**3**	5	8	3

由于以上方法得到的结果跟加工时间表中工件和机器的排列顺序有关，我们可以通过改变表中工件和机器的排列顺序，得到不同的分配结果。

设备分配完成后，还要决定机器上工件的调度顺序。只要调度满足同一工件中不同工序间先后顺序的约束，就都是可行的，即 $O_{i,j+1}$ 不可以早于 $O_{i,j}$。这里我们采用了 Pezzella 等人[4]使用的混合策略，其由 3 种规则组成：

(1) 随机选择一个工件(Random)；

(2) 优先选择余下加工时间最长的工件(MWR)；

(3) 优先选择余下操作数最多的工件(MOR)。

其中，20％的初始种群使用规则(1)生成，使用规则(2)和规则(3)生成的初始种群各占 40%。

2.3 交叉

交叉操作是将种群中两个个体随机地交换某些基因，产生新的基因组合，期望将有益的基因组合在一起。染色体中两部分基因串的交叉分别进行，其中第一部分基于工序编码的基因串采用 Shi 提出的 POX 交叉算子[5]，第二部分基于机器分配编码的基因串采用两点交叉法。

2.3.1 工序编码基因串的交叉

这部分的过程为：随机划分工件集{1,2,3,...,n}为两个非空的子集 J_1 和 J_2；复制 Parent1 包含在 J_1 的工件到 Children1，Parent2 包含在 J_1 的工件到 Children2，保留它们的位置；复制 Parent2 包含在

J_2 的工件到 Children1，Parent1 包含在 J_2 的工件到 Children2，保留它们的顺序。

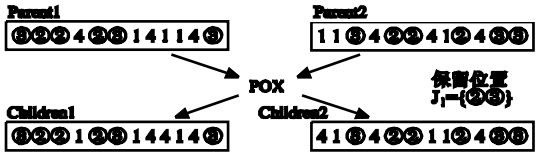

图 2　POX 交叉

图 2 说明了两个父代执行交叉操作的过程。可以看出经过 POX 交叉，保留了工件{2，3}在机器上的位置，使子代继承父代每台机器上工序的次序，并且 POX 操作生成的全部都是可行调度。

2.3.2 机器分配编码基因串的交叉

在这里我们采取两点交叉的方法，对于选定的父个体，随机选择两个交叉点，将位于交叉点之间的两父个体都有的工序对应的机器进行交换。

2.4 变异

变异操作的目的是改善算法的局部搜索能力，维持种群多样性，同时防止出现早熟现象。FJSP 的变异同交叉一样，也要分为基于工序编码和基于机器分配编码两部分，同时还要保证变异后解的可行性。我们在设计 FJSP 的变异算子时，引入了关键路径的思想。

2.4.1 关键路径

柔性作业车间的可行调度可以通过有向图表示。在有向图从源点到汇点的路径中，长度最长的路径称为关键路径。对于 FJSP，它的最后完工时间(make span)等于关键路径的长度。在关键路径上的所有工序都称为关键工序。任何一个关键工序一旦延迟，该调度的最后完工时间就必然会被推迟。图 3 为图 1 所对应的 Gantt 图，其中有阴影的工序表示关键工序。

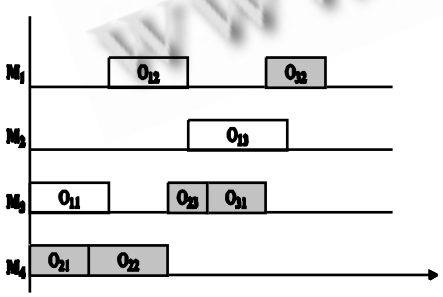

图 3　Gantt 图

在变异过程中，只有当关键路径发生改变时，makespan 才会改变，而对于非关键工序的变异只可能使关键路径更长，不可能得到比父个体更优的解，所以我们给出了针对关键工序的变异操作。当对染色体的工序编码和机器分配编码进行变异时，分别通过改变关键路径上工序的顺序和修改关键工序所处的机器来达到改变关键路径的目的。

2.4.2 工序编码基因串的变异

对工序编码的变异与传统的 PPS 变异过程类似，但是把变异位置的选择由整个染色体缩小到了关键路径上。其过程为：从父个体中随机选择一个关键工序，并且在满足工件内部顺序约束的前提下，将这个工序前插到其紧邻的前一个关键工序之前的某个位置；同时将对应的机器分配编码同步前移。图 3 对应的关键路径为{O_{21}, O_{22}, O_{23}, O_{31}, O_{32}}，图 4 所示为选中关键路径 O_{31} 的情况，其前插位置应在另一个关键工序 O_{23} 之前。

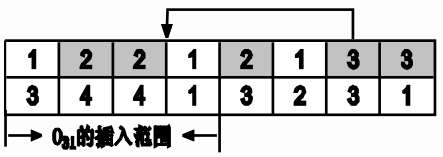

图 4　工序编码的变异过程

2.4.3 机器分配编码基因串的变异

我们给出了两种基于关键工序的机器分配编码变异算子。

第一种变异算子的过程为：首先计算各台机器的工作负载，然后从负载最大的机器上选择一个关键工序，将它重新分配到工作负载最小的机器上。该变异过程，不仅实现了对关键工序的修改，还平衡了机器的负载。图 3 中 M_3 的负载最大为 9，M_2 的负载最小，只有 5。图 5 所示为选择了 M_3 中的关键工序 O_{23} 的情况，通过将 O_{23} 的加工机器由 M_3 修改为 M_2，新个体的 makespan 减为 14，M_3 与 M_2 的负载都变为 7。

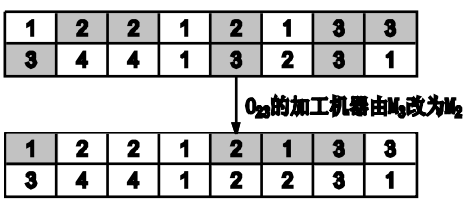

图 5　器分配编码的变异过程

第二种机器分配编码的变异算子不考虑机器的负载情况，只需要选择一个关键工序，然后给它分配一个新

的可行机器。该方法比第一种更能增加种群的多样性。

2.5 应函数和精英保留策略

在进化过程中，使用 F(x) = M − makespan 做为适应度函数，其中 M 为一足够大的正数。同时为保护在进化过程中曾经出现的优秀个体，我们将父代种群和子代种群合并成一个新的种群，根据适应度值的大小，将新种群中前 50 % 的个体加入下一代种群。

3 实验与分析

为了验证算法的性能，我们使用了 Brandimarte 数据集[6]进行测试，然后与 Pezzella 等人的采用多种策略的遗传算法[4]，Mastrolilli 和 Gambardella 的禁忌算法[7]，以及 Chen 的遗传算法[8]进行了比较。本文中 GA 的主要参数为：种群规模为 2000，最大代数为 500，工序编码交叉概率为 0.4，机器分配编码交叉概率为 0.4，工序编码变异概率为 0.1，第一种机器分配编码变异概率为 0.1，第二种为 0.05，每个实例连续运行 5 次取最优结果。

表 3 给出了实验的结果。其中第二、三列分别表示当前问题的工件数和机器数，第四列为本文给出的 GA 算法的 makespan，后面几列为其它 3 种算法的 makespan 及它们与本文 GA 的相对偏差。

相对偏差 $dev = [(MK_{comp} - MK_{GA})/MK_{comp}] \times 100\%$，

其中 MK_{GA}、MK_{comp} 分别表示本文 GA 算法和进行对比的算法的 makespan。

表中带*号的数据表示多种算法比较后的最优值。

由实验结果可知，我们设计的遗传算法在大部分情况下，都能够达到或接近 M.G.禁忌算法的性能，而明显优于 Chen 给出的算法，与同样采用了初始种群定位法和多种遗传策略的 Pezzella 的算法相比，解的质量也有较大提高。可以证明我们设计的基于关键工序变异的遗传算法是确实有效的。

4 结语

针对柔性作业车间调度问题的特点，本文通过研究关键路径对于染色体进化的影响，设计了基于关键工序的变异算子，使工序编码和机器分配编码的变异过程以改变关键路径为目的。从而使变异过程集中于关键工序，忽略了不能产生期望结果的非关键工序，直接提高了变异算子的效率，优化了算法的整体性能。

表 3 本文 GA 算法与其它算法的性能比较

	n	M	GA	Pezzella		M.G.		Chen	
				C_M	dev	C_M	dev	C_M	dev
Mk01	10	6	*40	*40	0	*40	0	*40	0
Mk02	10	6	*26	*26	0	*26	0	29	10.34
Mk03	15	8	*204	*204	0	*204	0	*204	0
Mk04	15	8	*60	*60	0	*60	0	63	4.76
Mk05	15	4	*173	*173	0	*173	0	181	4.42
Mk06	10	15	59	63	6.35	*58	-1.72	60	1.67
Mk07	20	5	140	*139	-0.72	144	2.78	148	5.41
Mk08	20	10	*523	*523	0	*523	0	*523	0
Mk09	20	10	*307	311	1.29	*307	0	308	0.32
Mk10	20	15	205	212	3.30	*196	-3.54	212	3.30

此外，我们还设计了新的遗传算法，并将其应用于 Brandimarte 测试数据集，取得了较好的结果。

参考文献

1 Garey MR, Johmson DS, Sethi R. The complexity of flow shop and job shop scheduling. Mathematics of Operations Research, 1976,1:117 − 129.

2 Shi GY, Iima H, Sannomiya N. A new encoding scheme for solving job shop problems by geneticalgorithm. 35th IEEE Conference on Deci sion and Control, 4, 4395 − 4400.

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

4 Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem. Computers & Operations Research, 2008:3202 − 3212.

5 Shi GY. A genetic Algorithm Applied to a Classic Job-shop Scheduling Problem. International Journal of Systems Science, 1997,28(1):25 − 32.

6 Brandimarte P. Routing and scheduling in a fexible job shop by tabu search. Annals of Operations Research, 1993,41:157 − 183.

7 Mastrolilli M, Gambardella LM. Effective neighbourhood functions for the fexible job shop problem. Journal of Scheduling, 1996,3:3 − 20.

8 Chen H, Ihlow J, Lehmann C. A genetic algorithm for fexible Job-shop scheduling. IEEE International conference on Robotics and Automation, Detroit, 1999, 1120 − 1125.