文章编号：1000-8608(2011)02-0205-05

综合作业调度问题基于可行域的遗传算法

王林平*[1,2]， 贾振元[1]， 王福吉[1]， 赵国凯[1]

（1.大连理工大学 精密与特种加工教育部重点实验室，辽宁 大连 116024；
2.烟台大学 机电汽车工程学院，山东 烟台 264005）

摘要：综合作业调度问题(CJSSP)同时考虑了加工约束与装配约束，成为约束严调度问题，而现有求解JSSP的遗传算法不能保证约束严问题染色体的可行性.描述了基本CJSSP，在定义的装配体分类和可操作基因串两个关键概念的基础上，设计出了基于可行域的交叉和变异方法.采用改编自JSSP的基准问题和实际调度问题进行了算法测试，并与规则算法的结果进行了比较，表明设计的遗传算法是可行和有效的.

关键词：综合作业调度问题；可行域；遗传算法；装配体分类；可操作基因串
中图分类号：TP301；F406.2 **文献标志码**：A

0 引 言

传统的生产调度问题研究大多针对零件加工过程，而对产品装配调度问题涉及较少，并且通常把加工和装配作为两个独立阶段分开进行研究.有限的装配调度研究也侧重于装配线平衡问题，很少考虑装配工序约束，对于日益重要的多品种小批量产品(job-shop)装配问题的研究很少[1~4].为此，本文研究加工与装配作业全过程调度问题，称之为综合作业调度问题或全作业调度问题(complete job shop scheduling problem，CJSSP).CJSSP是一种有实际应用背景的生产调度问题[5~7]，作者为企业开发生产调度管理系统时，也遇到了此类需求.CJSSP的研究不仅有利于缩短生产周期，而且有助于保持生产的同步性和齐套性[4]，可消除MRP(物料需求计划)不能满足生产装配约束的缺陷.

另一方面，包含装配约束的已有作业调度研究一般以JSSP(job shop scheduling problem)的推广或个体实例的形式零星见于文献中，采用规则方法研究的相对较多[7~10]，而以makespan为目标的研究很少.虽然有一些采用分析方法或近似算法进行研究的成果，但多属于单机或小规模调度问题[4,11].装配约束的增加，致使CJSSP与JSSP有了本质区别，求解JSSP的遗传算法不再适用，原来可保证解的可行性的编码及其遗传操作方法均已失效.仅检索到一篇应用智能优化方法研究装配作业调度问题的文献：利用遗传算法求解最小成本问题[12]，因不能保证交叉和变异后染色体的可行性，增加了复杂费时的修补环节，势必会影响进化过程的可控性.本文在介绍CJSSP概念及其背景基础上，定义装配体分类和可操作基因串两个概念，设计基于可行域的操作算子，求解以makespan为目标的3个调度实例和构造的10个基准问题.本文是作者对CJSSP问题的深入研究结果：不同于文献[13]，不再采用对不可行染色体进行转换和在不可行域进行遗传进化的方法，而是完全在可行域进行遗传进化，以解决求解CJSSP问题的约束处理难题，在短时间内求出FT10C的最优解.

1 CJSSP概念及其描述

基本CJSSP可定义为p个产品，每个产品有不相等的n个装配体，每个零件有不相等的o个工序，在m台机器上加工，已知p个产品的结构、各工序的制造时间和各装配体在各机器上的制造

收稿日期：2008-10-08； **修回日期**：2010-12-17.
基金项目："八六三"国家高技术研究发展计划资助项目(2003AA414022).
作者简介：王林平*(1963-)，男，博士，E-mail：qutinwlp@yahoo.com.cn；贾振元(1963-)，男，博士，教授，博士生导师，E-mail：jzyxy@dlut.edu.cn.

次序约束，要求确定与产品结构和工艺条件相容的各机器上所有装配体的制造开始时间或完成时间或制造次序，使制造性能达到最优. 其中装配体(constituent)不加区分地表示组成产品的所有物料，包括零件、部件和产品. 图 1 为由 10 个装配体组成的产品结构，多个产品时各自按从上到下、自左至右对装配体进行标识.

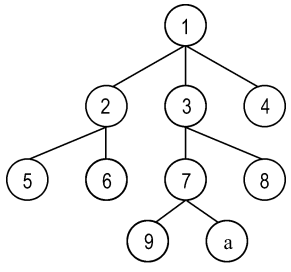


图 1 典型的产品结构

Fig. 1 An example of typical product structure

JSSP 的解空间随着机器数、工件数和调度时间的增加呈指数规律增长，算法所需存储空间和求解时间都急剧增加[14]. 对 CJSSP 来说，产品结构也影响解空间的大小. 假定这 10 个装配体与著名的 FT10 问题数据完全相同[15]，可称之为 FT10C，采用基于工序的编码方案时，其搜索空间与 FT10 相同，约为 2.357×10^{92}，但由于装配约束的作用，其可行解空间很小，大约为 $(C_{20}^{10}\times C_{40}^{10})\times C_{20}^{10}\times1\times C_{60}^{10}\times C_{90}^{30}\approx1.47\times10^{54}$，远比 FT10 的可行域 4.0×10^{65} 要小.

按照加工与组装工序之间共用设备的情况，可将 CJSSP 分为部分共用机器 CJSSP 和无共用机器 CJSSP，后者是前者的简化，JSSP 和装配作业生产调度都是其特例. 本文考虑了最复杂的全共用机器的情况，算法及其结论适用于任何简化模型. 与 JSSP 相似，同样地，也可以对基本 CJSSP 问题做一系列扩展，如考虑工序柔性等[14,15].

2 遗传算法设计

完全随机的初始种群产生方法不能保证染色体的可行性，而应用以可派工集为基础的随机方法，产生的种群多样性较差. 因此，采取了将完全随机的不可行染色体转换为可行染色体的方法，种群多样性得以大大提高. 用轮盘赌进行遗传选择，使用精英保留策略，适应函数 $f(i)=\dfrac{C_{\max}-makespan(i)}{C_{\max}-C_{\min}}+\varepsilon$，其中 ε 是很小的正数，$C_{\max}$ 和 $C_{\min}$ 分别是种群的最大和最小 *makespan*. 以下主要介绍遗传算法的编码方法、约束处理，以及遗传操作算子设计.

2.1 编码方法与约束处理策略

算法设计的核心是处理工艺约束引起的不可行染色体问题. 策略之一是采用修补法，即将不可行染色体转换为可行染色体，缺点是转换算法开销较大，设计速度快和质量高的转换算法是其瓶颈. 策略之二是采用算子法，即纯粹靠合适的交叉和变异算子来保证操作后染色体的可行性，但在现有紧约束下，设计这样的算子难度非常大. 考虑到求解 JSSP 问题时常用基于工序的编码方法，能保证 JSSP 染色体的可行性，而且具有半 Lamarkian 性，因此本文采用分治策略：采用基于工序的实数等长编码满足先序约束(即加工工序约束)，通过操作算子设计满足装配约束.

2.2 装配体分类与可操作基因串概念

尽管在装配约束下只需考虑满足装配约束要求，保证染色体可行性和减少时间开销也不易实现. 交叉和变异操作必然改变基因之间的次序，而装配约束就体现在这种次序中，所以不能在保证染色体可行性的前提下，针对整个染色体使用常规的交叉和变异操作.

受前述不可行染色体转换思想的启示，若能在染色体中找到合适的子染色体，进行交叉与变异操作后就可能满足装配约束要求，即设计保证染色体的可行性. 引入两个关键概念装配体分类和可操作基因串，在装配体分类的基础上即可确定合适的子染色体.

定义 1(装配体分类) 以其子装配体的类型(也即装配约束类型)将装配体分为 3 种类型：

0 型装配体 叶装配体(即零件装配体). 因其无装配约束，亦称为自由装配体.

1 型装配体 子项全部是 0 型装配体.

2 型装配体 子项至少有一个 1 型装配体(即子项是部件装配体).

图 1 产品结构中，4、5、6、8、9、a 为 0 型装配体，也是上文中组成初始可派工集的装配体. 2、7 为 1 型装配体，1、3 为 2 型装配体.

定义 2(可操作基因串) 正常交叉或变异时不破坏染色体可行性的基因序列. 根据装配体分类可以有 3 种可操作基因串：0 型装配体基因序列(位于染色体最左端至左数第一个 1 型装配体基因之间，连续基因串)、1 型装配体的子装配体

兄弟基因序列(一般不连续)；2 型装配体的子装配体兄弟基因序列(位于其所有孙项 1 型或孙项 2 型装配体右侧，至该装配体第一个基因之间，一般不连续).

仍然以图 1 所示 FT10C 调度问题为例，在染色体 aa9aa999895955a96949a6a8aa498a7687857-8878867756446577644456657756835433432322 2-22323322433211111111111 中，0 型装配体基因序列为 aa9aa999895955a96949a6a8aa498a. 2 个 1 型装配体 2、7 中，2 的子装配体兄弟基因序列为 55566656566565665565；2 个 2 型装配体 1、3 中，3 的孙项装配体为 9 和 10，3 的子装配体兄弟基因序列为 78787887887777778. 3 种可操作基因串中，只有 1 型基因串是等长而且基因相同的，其余两种可能长度不等，也可能长度虽相等但基因有差别.

2.3　交叉与变异算子设计

可操作基因串的长度相对于整个染色体很短，一般情况下其基因已不再连续，而且一个装配体的所有工序基因相同，采用常规的操作算子效率较低. 将可操作基因串看作积木块，采用交换交叉方法，即交换两个染色体中的可操作基因串，以维持子代染色体中的积木块. 从整个染色体来看，可操作基因串的交换属于染色体部分调度交换交叉. 基于同样的考虑，变异采用全逆序操作，以保持较强的搜索能力.

需要指出的是，不等长或非同因染色体交叉时，可操作基因串中未参与实际交叉的所有基因，保持其染色体中的基因座不变. 而参与交叉或变异的所有基因，则按交叉或变异后的基因次序依次填入原来父基因串占有的基因座. 这种保位措施，有利于保持 Lamarkian 进化特性和增强对遗传进化的可控性.

交叉操作如图 2 所示：(1) 随机选择装配体类型及其装配体，在两个父代染色体中确定各自的可操作基因串，假设分别为 P_1 = 78787887887777778，P_2 = 87787787. (2) 比较两基因串的基因种类及其个数，找出公共基因. (3) 随机选择并标记出可参与交叉(公共基因)的基因座. (4) 交换已标记的子基因串，从左至右依次占有原来标记的基因座，C_1、C_2 即得到的子代基因串. (5) 按父代基因串占有的基因座，将子代基因串中的基因从左至右逐个插入原父代染色体，即得到子代染色体.

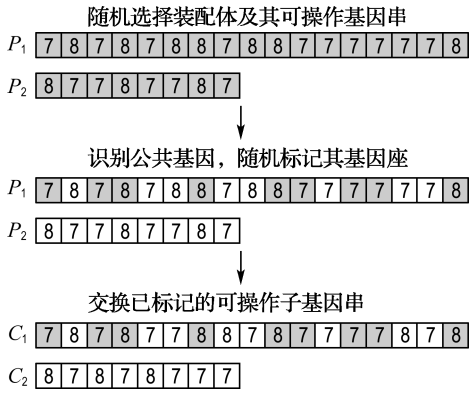


图 2　基于可操作基因串的交换交叉

Fig. 2　The swap crossover based on the operable gene string

3　计算结果及其分析

目前尚未发现有 CJSSP 基准测试问题方面的研究工作，相关文献也很少有详细的问题说明. 模型与算法测试使用了两类问题. 第一类是在 JSSP Benchmark 基础上构造的 CJSSP 基准问题，规模为 10×10，其产品数据结构如图 1 所示，其余数据与 JSSP 相同(见文献[14]附录 1)，产品的 1～a 号节点分别对应 0～9 号工件. 为了与相应的 JSSP 问题区别开，问题代号后加了标识 C. 第二类是文献[8、9]的实例. 算法代码全部用 VS2005 C＃实现，机器为 Pentium(R) CPU，主频 3.06 GHz，内存 512 MB. 遗传参数影响算法性能，但考虑到寻求最优遗传参数的代价以及算法的实用性，研究中未采用基于大量实验数据的复杂分析方法，而是以少量实验数据和一般遗传算法应用经验确定遗传参数. 遗传种群为 100，终止代数为 30，交叉概率 0.85，变异概率 0.2.

表 1 中前 10 行数据为第一类问题计算结果，后 3 行为第二类 3 个实例的计算结果，其中遗传算法结果是 5 次实验的平均值. 在以 makespan 为目标的规则算法中，*SPT*(最短工时工序优先)是求解 JSSP 的性能仅次于 *MWKR*(作业剩余操作加工时间最长)的调度规则. *MTWRP*(最长剩余时间路径工序优先)和 *MTORP*(最多剩余工序路径工序优先)，可认为是为了适应 CJSSP 而对 *MWKR* 作了扩展，本质上分别与关键路线规则(CP)和最多后继规则(LNS)相同[1].

可以看到，FT10C、Orb2C 和 Orb5C 的遗传算法结果与一个或两个规则的相同，均得到了最优解，这 3 个问题相对比较容易. 其余 10 个问题

都是遗传算法优于规则算法.虽然在1 s内遗传算法没有求出Orb3C和Orb4C问题的最优解,但也已经非常接近,考虑到最优值出现的概率,说明这两个问题比较难解.

文献中3个实例的结果分别为225、260、255,对比表1中对应3个问题的结果,文献中的规则算法明显优越于表1中的3个常见规则算法,原因是文献中的规则采用了一些高级的优化调整技术.当然,如表1数据所示,遗传算法在1 s内求出了全部13个问题中的11个最优解,更优于文献中的规则.从最优值的频率和种群平均值上可以看出,文献中3个实例的种群均已收敛,图3为遗传算法求解表1中文献[9]②的结果.

从上述两类问题的实验结果看出,遗传算法有很好的最优解搜索能力,也有满意的速度和收敛特性.

表1 两类问题的几种算法结果比较

Tab.1 The comparison among several algorithms for the two kinds of problems

序号	问题	最优解	*SPT*	*MTORP*	*MTWRP*	遗传算法			
						最优值	最优值频率/%	种群平均值	计算时间/s
1	FT10C	1 985	2 234	1 985	1 985	1 985	22.4	2 027.8	1
2	La16C	2 199	2 570	2 251	2 251	2 199	29.6	2 218.3	1
3	Orb1C	2 152	2 773	2 253	2 209	2 152	4.0	2 209.5	1
4	Orb2C	2 180	2 284	2 180	2 180	2 180	74.4	2 189.8	1
5	Orb3C	2 136	2 571	2 273	2 240	2 140	0.4	2 271.8	1
6	Orb4C	2 071	2 720	2 421	2 122	2 173	0.2	2 172.2	1
7	Orb5C	1 958	2 322	2 090	1 958	1 958	45.6	1 998.1	1
8	Orb6C	2 200	2 689	2 266	2 328	2 200	5.6	2 279.4	1
9	Orb7C	1 001	1 157	1 007	1 006	1 001	48.2	1 005.1	1
10	Orb8C	2 023	2 231	2 059	2 059	2 023	58.0	2 054.6	1
11	文献[8]	195	255	200	210	195	98.8	195.1	<1
12	文献[9]①	245	325	265	270	245	95.6	245.3	<1
13	文献[9]②	250	325	295	265	250	99.2	150.1	<1

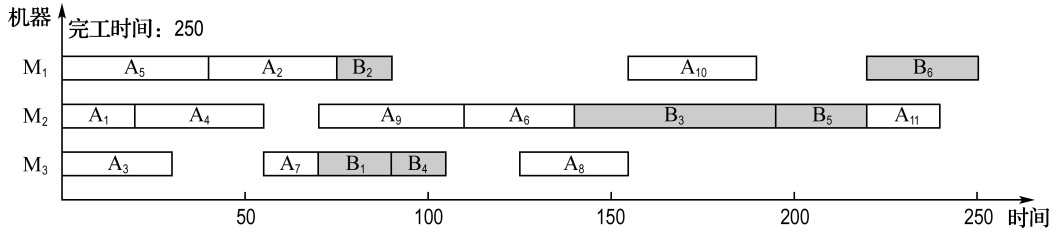


图3 文献[9]②问题的遗传算法结果

Fig.3 The result of genetic algorithm for the problem of the Lit. [9]②

4 结 语

CJSSP有重要理论和实际应用价值.遗传算法在解决调度问题时较少应用领域知识,更适合于解决实际生产中的问题.本文在建立CJSSP数学模型的基础上,通过深入分析染色体的基因关系特性,提出了一种基于装配体分类的可行基因串构造方法,设计出一种在可行域进行遗传操作的算法,能同时满足加工工序约束与装配工序约束.将设计的遗传算法应用于从JSSP Benchmark扩展而来的基准问题和实际调度问题,均得到了较好的效果,这对其他约束组合优化问题也有参考价值.

参考文献:

[1] PINEDO M L. Planning and scheduling in manufacturing and services [M] // **Springer Series in Operations Research and Financial Engineering.** Berlin:Springer, 2005

[2] 苏 平,于兆勤. 基于混合遗传算法的混合装配线排序问题研究[J]. 计算机集成制造系统, 2008, **14**(5):1002-1007

[3] DING F Y, ZHU J, SUN H. Comparing two weighted approaches for sequencing mixed-model assembly lines with multiple objectives [J]. **International Journal of Production Economics**, 2006, **102**:108-131

[4] SCHELLHORN H. A note on the first moment of makespan in an assembly shop [J]. **European Journal of Operational Research**, 2007, **180**:963-969

[5] 赵　宁,吴　超,宁汝新,等. 面向配做的柔性生产线调度[J]. 机械工程学报, 2005, **41**(7):181-185

[6] 熊禾根,李建军,孔建益. 考虑工序相关性的动态 JOB SHOP 调度问题启发式算法[J]. 机械工程学报, 2006, **42**(8):50-55

[7] HICKS C, PONGCHAROEN P. Dispatching rules for production scheduling in the capital goods industry [J]. **International Journal of Production Economics**, 2006, **104**:154-163

[8] 谢志强,莫　涛,谭光宇. 非紧密衔接工序动态车间调度算法[J]. 机械工程学报, 2008, **44**(1):156-160

[9] 谢志强,杨　静,杨　光,等. 可动态生成具有优先级工序集的动态 Job-Shop 调度算法[J]. 计算机学报, 2008, **31**(3):502-508

[10] NATARAJAN K, MOHANASUNDARAM K M, BABU B S, *et al*. Performance evaluation of priority dispatching rules in multi-level assembly job shops with jobs having weights for flowtime and tardiness [J]. **International Journal of Advanced Manufacturing Technology**, 2007, **31**:751-761

[11] YOKOYAMA M. Scheduling for two-stage production system with setup and assembly operations [J]. **Computers & Operations Research**, 2004, **31**:2063-2078

[12] PONGCHAROEN P, HICKS C, BRAIDEN P M. The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure [J]. **European Journal of Operation Research**, 2004, **152**:215-225

[13] 王林平,贾振元,王福吉,等. 含装配约束的作业调度问题及遗传解码算法[J]. 大连理工大学学报, 2010, **50**(4):535-539
(WANG Lin-ping, JIA Zhen-yuan, WANG Fu-ji, *et al*. Job-shop scheduling problem incorporating constraint and its solution using genetic decoding approach [J]. **Journal of Dalian University of Technology**, 2010, **50**(4):535-539)

[14] 金　锋,吴　澄. 大规模生产调度问题的研究现状与展望[J]. 计算机集成制造系统, 2006, **12**(2):161-168

[15] 王　凌. 车间调度及其遗传算法[M]. 北京:清华大学出版社, 2003

Solving complete job shop scheduling problem using genetic algorithm in feasible domain

WANG Lin-ping[*1,2], JIA Zhen-yuan[1], WANG Fu-ji[1], ZHAO Guo-kai[1]

(1. Key Laboratory for Precision & Non-traditional Machining Technology of Ministry of Education, Dalian University of Technology, Dalian 116024, China;
2. School of Electromechanical Automobile Engineering, Yantai University, Yantai 264005, China)

Abstract: The complete job shop scheduling problem (CJSSP) considers machining and assembly constraints simultaneously and becomes tight-constraint problem. The existing genetic algorithms developed for JSSP can not maintain chromosome's feasibility of this tight-constrained problem. The basic CJSSP is described. Two key concepts, constituent type and operable gene string, are defined and then applied to building crossover and mutation operation in the feasible domain. The genetic algorithm is tested in both practical instances and problems adapted from JSSP benchmarks. The computation result is compared with that of some heuristics and it is demonstrated that the proposed algorithm is both feasible and efficient.

Key words: complete job shop scheduling problem; feasible domain; genetic algorithm; constituent type; operable gene string