

华中科技大学

硕士学位论文

基于遗传算法的生产调度优化方法研究

姓名：肖力

申请学位级别：硕士

专业：物理电子学

指导教师：杨坤涛

20060427

摘要

遗传算法是一种模拟生物进化过程的随机搜索算法，其自组织、自适应、自学习和种群进化能力使其适合于大规模复杂优化问题。它将问题的求解表示成“染色体”的适者生存过程，通过种群的一代代不断进化，包括复制、交叉和变异等操作，最终收敛到“最适应环境”的个体，从而求得问题的最优解或者满意解。随着计算机技术的发展，遗传算法越来越受到人们的重视，并在机器学习、模式识别、神经网络、优化控制、组合优化等领域得到了成功的应用。

生产调度问题几乎在现实环境中，特别是在工业工程领域无所不在。许多制造工业提出的调度问题从本质上讲非常复杂，难以用传统优化方法求解。因此，调度问题成为遗传算法领域里的一个热门话题。原因是该问题表现出约束组合优化问题的所有特征，并且成为测试新算法思想的范例。

本文第一章介绍了生产调度理论的产生、发展、分类和已有的解决调度问题的方法。第二章介绍了遗传算法的基本原理，描述了遗传算法的一般流程，并给出了标准遗传算法的参数及基本操作的设计方法，并进一步指出了和传统优化方法相比遗传算法具有的独特优点，并提出了几种改进的遗传算法。第三章和第四章分别讨论了 Job Shop 调度问题和 Flow Shop 调度问题的基本框架、标准遗传算法的设计，并分别提出了两种改进的遗传算法：嫁接共生遗传算法和佳点集遗传算法来提高了生产调度的效率，并克服了标准遗传算法容易早熟收敛的缺点。第五章对本文提出的几种算法进行了数值仿真实验。

由于遗传算法理论分析上的困难，以及生产调度问题的复杂性，我们采用仿真的方法来验证了相关算法的有效性。通过和已有结果进行对比，验证了我们提出的算法在解的质量上、运行效率上都有了一定程度的提高。

关键词：遗传算法 生产调度 嫁接共生遗传算法 佳点集 仿真

ABSTRACT

The genetic algorithms is a random search algorithms, which simulates the processes of biological evolution the ability of self-learning, self-organization, self-adaptation and population evolution adapt itself to cosmically complex problem. It indicates the exploratory methods of a problem into the chromosome's survival of the fittest processes. Through the population's evolution of the ages, including the replication, the intersection and the variation. They converge at a unit, which can adjust themselves to the environment perfectly, therefore get the optimum solution or starvation solution of a problem. As the development of computer technology, genetic algorithms is taken more seriously, and is applied successfully in the fields of machine learning, pattern recognition, neural network, optimizing control and combination optimization.

Production scheduling problem is ubiquitous in the actual circumstances, especially in the field of engineering and industry. The scheduling problem, in its nature, is very complex, and it is hard to get the resolution with traditional optimization methods. Therefore the scheduling problem becomes a pet topic in the field of GA. Because the problem put up all the features which restrict optimization problem, and turn into the example to test the new arithmetic ideas.

The first chapter introduced the production scheduling theory's emergency, development, and classification, and also introduced the solution to scheduling problem. The second chapter introduced the basic principle of GA, and described the general flow of GA. It also showed the parameters and the methods of basic operation and pointed out the unique advantages. Contrasting with the traditional optimization methods. Then raised several improved genetic algorithms. The third and forth chapters, respectively, discussed the basic framework and the design of standard GA, and pointed out two improved GA. The grafting -symbiotic GA and good-point sets GA improved the efficiency production scheduling and overcame the shortcoming of standard GA. The fifth chapter discussed the numerical simulation experiments according to the algorithms mentioned.

Because of the difficulty of the GA's technical analysis, and the complexities features of production scheduling problem, verified the validity of relating algorithms with simulation


华中科技大学硕士学位论文

methods .Contrasting with the existing results ,it can prove the algorithms which we put up has improved the quality and operational efficiency to a certain extent.

Key Words: genetic algorithms Production Scheduling
grafting- symbiosis genetic algorithm good point-set Simulation

独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：


日期：2006年4月28日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于 ☐ 保密， ☐ 在_____年解密后适用本授权书。
☒ 不保密。

（请在以上方框内打“√”）

学位论文作者签名：

日期：2006年4月28日

指导教师签名：

日期： 年 月 日

1 绪论

1.1 研究的背景及意义

车间生产过程的调度问题,是制造系统运筹技术、管理技术与优化技术发展的核心。有关资料表明,制造过程的 95%的时间消耗在非切削过程中。有效的调度方法与优化技术的研究和应用,已经成为先进制造技术实践的基础和关键,所以对它的研究与应用具有重要的理论和使用价值。调度的目的是针对一项可分解的工作(如产品生产),探讨在一定的约束条件下(如工艺约束),为每个加工对象确定具体的加工路径、时间、机器和操作等,以获得产品制造时间或成本的最优化。优良的调度策略对于提高生产系统的最优性、提高经济效益,有着极大的作用。由于系统建模方法的多样性,以及问题的侧重点不同,调度方法和研究对象也有着明显的不同。就对象而言,有确定性和随机性调度、离散事件和连续事件调度,静态和动态调度等。就调度方法而论,有 Gantt 图法、构造型方法、动态规划、分支定界法、排队论方法、规则调度方法和仿真方法等。调度的优化指标包括正规性能指标和非正规性能指标,常用的指标有最大完成时间、平均加工时间、平均延迟时间、生产成本和 E/T 指标等。

车间作业调度作为生产管理的最为困难的问题,而且业已证明车间作业调度问题属于 NP 难题,随着机器数和工件数的增加,调度方案呈指数增长。因此,对车间作业调度问题的研究,吸引了国内外许多学者和实际车间作业调度人员的关注。

车间作业调度问题的研究不仅具有较大的理论意义,而且还有相当大的实用价值。这是由于:一方面,车间作业调度问题的研究不仅可以推动相关算法的研究,如模拟退火算法,遗传算法,神经网络,人工智能等,而且还能在此基础上提出新的算法,这为其他领域类似问题的解决提供了条件和手段;另一方面,车间作业调度问题的解决本身具有实际意义,一个好的车间作业调度方案不仅可以降低生产成本,而且可以提高企业产品的准时交货能力,从而增强企业的竞争力。

1.2 生产调度问题的描述

生产调度问题一般可以描述为:针对某项可以分解的工作,在一定的约束条件下,

如何安排其组成部分(操作)所占用的资源、加工时间及先后顺序,以获得产品制造时间或者成本等性能指标最优。

生产调度问题主要集中在车间的计划与调度方面,许多学者作了大量研究,出了不少的研究成果。制造系统的生产调度是针对一项可分解的工作(如产品制造),探讨在尽可能满足约束条件(如交货期、工艺路线、资源情况)的前提下,通过下达生产指令,安排其组成部分(操作)使用哪些资源、其加工时间及加工的先后顺序,以获得产品制造时间或成本的最优化。在理论研究中,生产调度问题常被称为排序问题或资源分配问题^[1, 2]。

生产调度中涉及的资源包括:原料、设备(加工、存贮、运输)、人力、资金、能源等。资源的详细分配受到产品的生产工艺的限制。

影响调度问题的因素很多,正常情况下有:产品的投产期,交货期(完成期),生产能力,加工顺序,加工设备和原料的可用性,批量大小,加工路径,成本限制等,这些都是所谓的约束条件。有些约束条件是必须要满足的,如交货期,生产能力等,而有些达到一定的满意度即可,如生产成本等。这些约束在进行调度时可以作为确定性因素考虑。而对于设备故障,原料供应变化,生产任务变化等非正常情况,都是事先不能预见的,在进行调度时大都作为不确定性因素考虑。

生产调度的性能指标可以是成本最低、库存费用最少(减少流动资金占用)、生产周期最短、生产切换最少、设备利用率最高、三废最少等。车间调度问题的优化目标是评价调度方案优劣的标准,常见的调度指标有:

(1)反映调度成本的指标:调度中发生的费用有:启动成本、换线成本、加工费用、工人加班费用、过期赔偿费用、在线库存费用、调度管理费用等。由于调度净现值指标能综合反映上述费用,所以得到了广泛应用。

(2)反映调度性能的指标 包括:生产周期、平均流动时间、机床利用率、工人利用率等。

(3)反映用户要求的指标 包括:最大拖期时间、平均拖期时间、拖期零件的数量等。

在传统的调度中,一般以平均流通时间最小、制造周期最短、满足交货期为调度目标,而在实际生产中,由于提前完成的产品必须保存到交货期,而拖期产品必须交付违

约金, 因此, 在实际调度中经常考虑提前或者拖后惩罚。

1.2.1 生产调度问题的分类

生产调度系统的分类方法很多, 主要有以下几种:

(1) 根据加工系统的复杂度, 生产调度可分为单机调度、作业车间调度 (Job Shop)、流水车间 (Flow Shop) 调度、多机器并行加工调度。

单机调度是指所有的操作任务都在一台机器上完成, 为此存在任务的优化排队问题, 对于单机调度比较有代表性的请见文^[3, 4]; Job Shop 是最一般的调度类型, 它是指由 m 个不同的机器加工 n 个有特定加工路线 (顺序) 的工件, 不同工件的工序间没有顺序约束, 工序加工不能中断, 它不限制作业操作的加工设备, 并允许每个作业加工具有不同的加工路径。对于 Job Shop 型调度问题的研究, 文献很多, 综述文章可参见 Lawler 等^[5]; Flow Shop 调度假设所有工件都在相同的设备上加工, 并有一致的加工操作和加工顺序。Johnson 早在 1954 年首次发表了关于 Flow Shop 型调度问题的文章^[6], 从此便开始了对流水车间调度问题的广泛研究^[7, 8]; 多机器并行加工调度是指多台机器并行加工工件, 而且并行加工的机器和工件都是类似的。文^[9, 10, 11]等研究了多台并行机的调度; 文献^[11]则首次利用启发式算法对非等同多机调度问题进行了尝试。

实际应用中的调度问题通常是上述几种调度类型的组合。

(2) 根据优化准则, 可以分为基于代价和性能的调度两大类。

代价包括为了实现调度方案所消耗的各种费用和所造成的损失, 如运行费用、运输费用、存储费用和延期交货损失等。性能主要包括设备利用率、最大完成时间、拖延加工任务的百分比等。虽然在理论分析上, 大部分只注意调度的性能, 但在实际生产中, 通常要综合考虑代价和性能两方面因素。

(3) 根据生产环境的特点, 可将调度问题分为确定性调度问题和随机性调度问题。

确定性调度问题是指加工时间和其他参数是已知的、确定的量; 而随机性调度问题的加工时间和有关参数是随机的变量。

(4) 根据作业的加工特点, 可将调度问题分为静态调度和动态调度。

静态调度是指所有待安排加工的工作均处于待加工状态, 因而进行一次调度后、各作业的加工被确定、在以后的加工过程中就不再改变; 动态调度是指作业依次进入待加

工状态、各种作业不断进入系统接受加工、同时完成加工的作业又不断离开,还要考虑作业环境中不断出现的动态扰动、如作业的加工超时、设备的损坏等。因此动态调度要根据系统中作业、设备等的状况,不断地进行调度。

1.2.2 生产调度问题特点

生产调度问题有以下特点^[12]:

(1)复杂性:车间中工件、机器、缓存和搬运系统之间相互影响、相互作用。每个工件要考虑它的加工时间、安装时间和操作顺序等因素,因而相当复杂。调度问题是在等式或不等式约束下求指标的优化,在计算量上往往是具有 *NPHard* 特性,随着问题规模的增大,其计算量急剧增加,使得一些常规的方法无能为力,对于这一点已经证明^[13]。

(2)随机性:在实际的作业车间调度系统中存在很多随机的和不确定的因素,环境是不断变化的,在运行过程中会遇到多种随机干扰,比如工件到达时间的不确定性、作业的加工时间也有一定的随机性,而且生产系统中常出现一些突发偶然事件,如设备的损坏、修复、作业交货期的改变等,故作业车间调度过程是一个动态的随机过程。

(3)约束性:车间调度问题中资源的数量、缓存的容量、工件加工时间以及工件的操作顺序等都是约束。此外还有一些人为的因素,如要求各机器上的负荷要平衡等。

(4)多目标性:实际的车间调度问题是多目标的,而且这些目标之间往往是发生冲突的。调度目标分为三类:基于作业交货期的目标、基于作业完成时间的目标和基于生产成本的目标。

1.3 生产调度问题的优化方法

生产调度问题早已被证明属于 *NP* 难题,随着机器数和工件数的增加,调度方案呈指数增长,怎样才能尽快地得到最优调度方案,这一问题吸引了国内外许多学者和实际生产调度人员的关注,提出了很多的解决方法。下面分别对这些方法进行总结。

1.3.1 解析法

解析算法包括数学规划和随机优化。在数学规划方法中,研究较多的是分枝定界法、割平面法、动态规划以及拉格朗日松弛法。在随机优化领域中,则是应用排队论、批量技术、贮存论和可靠性理论等等。然而,正如前面所述,车间作业调度问题大都属于

NP 难题,会出现“维数灾难”,对其求解会导致组合爆炸。因此,尽管国内外学者对利用解析法求解车间作业调度问题进行了大量的研究^[14,15,16,17],但距离实际应用还有相当大的差距。

1.3.2 规则的调度方法

由于解析法不能有效地求解车间作业调度问题,于是人们转向借助于启发式方法,并主要集中在启发式规则(Heuristic Rule,HR)上,如最短加工时间优先(Shortest Processing Time,SPT)、先来先服务(First Come First Served,FCFS)和交货期最早优先(Earliest Due Date,EDD)等。不同的启发式规则适用于不同的生产环境和车间作业调度问题。由于生产系统的动态性,这就要求对于不同类型问题应用相适宜的启发式规则。应用启发式规则求解车间作业调度问题的主要缺点是求出的解不一定满足所有的约束条件,并且无法判断解的优劣,时常需要专家的介入。同时,由于这些启发式规则是根据调度人员的经验得到的,因此,利用启发式规则求解车间作业调度问题一般不太可能求得最优的调度方案。

调度规则因其易于实现、计算复杂度低等原因,能够用于动态实时调度系统中,许多年来一直受到学者们的广泛研究,并不断涌现出许多新调度规则,随着计算机运算速度的飞速提高,人们希望寻找新的近似调度方法,以合理的计算时间,换得比单纯启发式规则所得到的调度更好的调度。

1.3.3 基于知识的调度方法

近年来受实际需要的推动,基于知识的智能调度系统和方法的研究取得了很大的进展。基于知识的调度方法是用专家系统自动产生调度或辅助人去调度。它是将传统的调度方法与基于知识的调度评价相结合的方法。

在实际工作中,许多调度人员和专家积累了相当多的车间作业调度的知识,基于这一方面的考虑,可以借助专家系统来求解生产调度问题。然而,一方面,由于专家系统适应性较差,它一般适用于相对稳定的系统,但生产调度问题却是高度不确定的,如调度目标经常发生改变,目标之间出现冲突等;另一方面,有些学者还认为调度问题的复杂性已超出了专家的认识能力。同时,调度环境的动态特性使知识库中的知识很快就过时了,在专家系统中存在着知识获取和推理速度两个瓶颈。因此,利用专家系统求解生

产调度问题, 尽管有一些成功的例子, 但也确实存在许多困难。

1.3.4 仿真调度方法

由于大多数车间作业调度问题比较复杂, 不太可能用解析法来进行分析研究, 此时可借助仿真技术来求解车间作业调度问题。一般地说, 基于仿真的方法主要指离散事件系统仿真、petri 网等。仿真方法一般不单纯追求生产调度问题参数之间的数学描述, 侧重于系统运行过程中的逻辑关系描述。在车间作业调度问题中应用仿真技术不仅能对系统进行比较、评价和选择, 分析系统的动态性能, 而且能够选择系统结构和优化系统参数。同时, 管理者也能从系统的角度来观察车间作业调度的有效性, 从而增加产品产出, 减少在制品库存量及库存占用资金, 提高生产率和保证物资的及时传送。然而, 仿真方法也有许多不足之处: 1) 对于复杂的车间作业调度问题, 仿真模型运行时间长, 费用也较高, 很难保证调度的实时性要求; 2) 仿真的结果和精度受到软件技术和编程人员技术水平的限制; 3) 一次仿真运行只是对实际加工过程的一次抽样, 因此, 仿真方法一般不能进行有效的优化, 它给出的结果是一个特解, 而不是一个通解。

1.3.5 领域搜索方法

一般来说, 车间作业调度问题通常具有大量的局部极值点, 往往是不可微的、不连续的、多维的、有约束条件的、高度非线性的 NP 问题, 因此, 精确地求解车间作业调度问题的全局最优解一般是不可能的。

近年来各种局部搜索算法在车间调度中得到了广泛的应用, 这些局部算法在优化机制方面存在一定的差异, 但优化流程却具有较大的相似性, 均是一种“领域搜索”框架。算法从一个(一组)初始解出发, 在算法参数的控制下通过邻域函数产生若干邻域解, 按接受准则(确定性、概率性或混沌方式)更新当前状态, 然后按参数修改准则调节控制参数, 如此重复以上搜索过程直至算法终止准则满足, 最终得到问题的优化解。

邻域搜索技术由于其是随机性和启发式的, 当搜索解空间时, 它们仅对选定的目标函数值的变化做出响应, 因而通用性强, 因此近来受到各领域广泛的关注和应用。尽管这种技术需要的运行时间比 HR 方法长, 但解的质量一般能得到显著提高。这类方法包括模拟退火法(Simulated Annealing, SA)^[18], 禁忌搜索法(Tabu Search, TS)^[19]和遗传算法(Genetic Algorithms, GA)^[20]等。

(1) 模拟退火法(Simulated Annealing)

模拟退火算法(SA)将组合优化问题与统计力学中的热平衡问题类比,它模仿了晶体结晶的冷却过程,在较高温度 T_k 下,系统状态为 S , 能量(即目标函数)为 E , 选择 S 的一个领域 S' , 如果 $E(S') < E(S)$, 则接受 S' 为下一状态, 否则以概率 $e^{-(E(S')-E(S))/T_k}$ 接受 S' 。经过一定次数(称为 Markov 链长)的搜索, 认为系统在此温度下达到平衡, 则降低温度 T_k 再进行搜索, 直到满足结束条件。

由于模拟退火法能以一定的概率接受差的能量值, 因而有可能跳出局部极小, 但它的收敛速度较慢, 很难用于实时动态调度环境。

(2) 禁忌搜索法(Tabu Search)

对于复杂的组合优化问题, 禁忌搜索也是一种通过领域搜索以获取最优解的方法, 禁忌搜索是一种迭代方法, 它开始于一个初始可行解 S , 然后移动到领域 $N(S)$ 中最好的解 S' , 即 S' 对于目标函数 $R(S)$ 在领域 $N(S)$ 中是最优的。然后, 从新的开始点重复此法。为了避免死循环, 禁忌搜索把最近进行的 T 个移动(T 可固定也可变化)放在一个称作 tabu list 的表中(也称短期记忆), 在目前的迭代中这些移动是被禁止的, 在一定数目的迭代之后它们又被释放出来。这样的 tabu list 是一个循环表, 它被循环地修改, 其长度 T 称作 Tabu size。最后, 还须定义一个停止准则来终止整个算法。由于 tabu list 的限制, 使其在搜索中有可能跳出局部极小。

(3) 遗传算法

遗传算法是一种新的并行优化搜索方法。它是一种模仿生物群体进化过程的一种优化算法, 给定一组初始解作为一个群体, 通过选择、交叉和变异等遗传操作来搜索最优解。

遗传算法的最大优点是通过群体间的相互作用, 保持已经搜索到的信息, 这是基于单次搜索过程的优化方法所无法比拟的。但是, 对于某些问题遗传算法也存在着计算速度较慢的问题。这时通常将遗传算法和其他优化方法结合使用, 有利于改善搜索效率。

1.4 研究课题的关键技术

本文主要是对生产调度问题进行描述、分析和研究, 并应用遗传算法及改进的遗传算法对他们进行了求解。另外对相关的研究工作进行了必要的综述。

(1) 遗传算法的改进

遗传算法虽然具有全局优化和隐含并行性等优点，但也存在一些不足之处，所以设计了改进的遗传算法。

(2) 求解 Job shop 调度问题的遗传算法

Job shop 调度问题是经典强 NP 问题。本文针对 Job Shop 调度问题的特殊性，对遗传算法的编码/解码方式、遗传算子的设计、目标函数等进行了研究，设计了求解 Job Shop 调度问题的标准遗传算法，并在此基础上进行改进，设计了嫁接共生遗传算法。

(3) 基于遗传算法的 Flow Shop 调度问题

Flow Shop 调度问题是一类复杂且极有代表性的流水线调度问题，一般用遗传算法可以成功求解 Flow Shop 问题。本文根据 Flow Shop 调度问题的特点设计了一种佳点集遗传算法，有效地解决了 Flow Shop 调度问题。

2 遗传算法及其实现

遗传算法的术语来源于自然遗传学。1975 年由美国 J.Holland 教授提出的遗传算法 (Genetic Algorithm, 简称为 GA) 是基于自然选择原理、自然遗传机制和自适应搜索 (优化) 的算法。遗传算法启迪于生物学的新达尔文主义 (达尔文的进化论、魏茨曼的物种选择学和孟德尔的基因学说)、模仿物竞天演、优胜劣汰、适者生存的生物遗传和进化论的规律性。

2.1 遗传算法的基本描述

遗传算法是建立在自然选择和群体遗传学基础上的一种非数值计算优化方法。遗传算法利用某种编码技术将问题的解表示成字符串, 并把这样的字符串当作人工染色体或称为个体, 多个个体构成一个群体。随机产生若干个个体构成初始群体, 通过对群体的不断进化, 利用“优胜劣汰”的自然选择机制, 使群体中的个体不断朝着最优解的方向移动, 最终搜索到问题的最优解。个体通过遗传算子的作用生成子代个体。通过定义个体的评价函数, 称为适应度函数来评价个体的优劣。个体的适应度反映个体适应环境的能力, 适应度大的个体生存能力强。按照自然选择的基本原理, 适应度越大的个体被选择用来繁殖后代的机会越大。

60 年代美国 Michigan 大学的 John Holland 在从事如何能建立能学习的机器的研究中注意到学习不仅可以通过单个生物体的适应而且通过一个群的许多代的进化也能发生。受达尔文进化论的启发, 他逐渐认识到, 在机器学习的研究中, 为了获得一个好的学习算法, 仅靠单个策略的建立是不够的, 还有依赖于一个包含许多候选策略的群体的繁殖。它们的思想起源于遗传进化, Holland 就将这个研究领域命名为遗传算法^[21, 22, 23]。

2.2 遗传算法的基本流程

遗传算法是一类随机优化算法, 但它不是简单的随机比较搜索, 而是通过对染色体的评价和对染色体中基因的作用, 有效地利用已有信息来指导搜索有希望改善优化质量的状态。

标准遗传算法的主要步骤如描述下:

- 步骤 1: 随机产生一组初始个体构成的初始群体;
- 步骤 2: 对初始群体迭代地执行下面的步骤 3、4, 直到满足停止准则后转步骤 5;
- 步骤 3: 计算群体中每一个个体的适应度;
- 步骤 4: 应用复制、交叉、变异算子产生下一代群体;
- 步骤 5: 把在任何一代中出现的最好的个体串指定为遗传算法的执行结果, 这个结果可以表示问题的一个解(或近似解)。

标准遗传算法的流程图描述如图 2-1 所示:

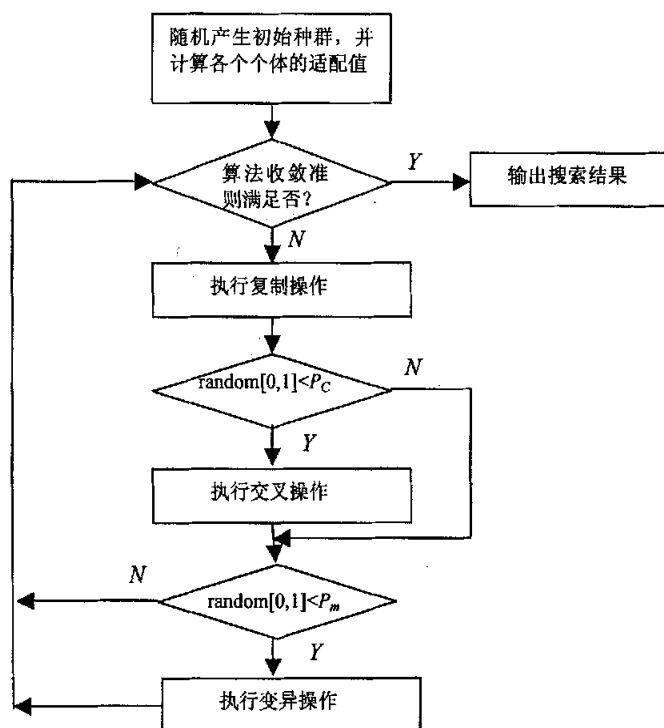


图 2-1 标准遗传算法的基本流程

2.3 遗传算法参数与操作的设计

通常，遗传算法的设计是按以下步骤进行的:

(1) 确定问题的编码方案。由于遗传算法通常不直接作用于问题的解空间，而是利用解的某种编码表示来进行进化，因此选择合理的编码机制对算法质量和效率有很大影响。

(2) 确定适应度函数。由于遗传算法通常基于适应度进行遗传操作，因此合理的适

应度能够将各个体的优劣程度得以体现,并适应算法的进化过程。

(3) 算法参数的选取。通常包括种群数目、交叉概率、变异概率、进化代数等。

(4) 遗传算子的设计。通常包括初始化、选择、交叉、变异和替换操作等。

(5) 确定算法的终止条件。终止准则应根据所求解问题的性质,在优化质量和效率方面作合理均衡或侧重。

下面对关键参数与操作的设计作简单介绍。

2.3.1 编码

遗传算法的寻优过程主要是通过遗传算子对群体中具有某种结构形式的个体进行结构重组处理,从而不断地搜索出群体中个体间的结构相似性,形成并优化积木块以逐步逼近最优解。由于遗传算法不能直接作用于问题参数本身,而是在一定编码机制对应的码空间上进行的,因此必须先把待优化问题用遗传算法可以处理的串结构来处理,这一过程称为编码,编码的选择是遗传算法应用遇到的首要的问题,也是应用成功的关键问题,是影响算法性能与效率的重要因素。

评价编码策略可以参考下面 3 个基本规则^[24]。

(1) 完备性:对问题空间中的任何一个点都可以表示为遗传算法空间中的点,即问题空间的所有可能解都能表示为所设计的基因编码方式。

(2) 健全性:遗传算法空间中的任何一个点都有问题空间中的一个点与之对应,即任何一个基因编码都对应一个可能解。

(3) 非冗余性:遗传算法空间中的点作为问题空间中的点的表示是唯一的。

上述 3 条规则只是一般性的评价准则,它缺乏具体的指导思想,满足上述 3 个规则的编码不一定能有效地提高遗传算法的搜索效率,为此 De Jong 提出了两条较为具体的编码原则^[25]。

(1) 有意义积木块原则:所定编码应易于生成与所求问题相关的短距、低阶的积木块。

(2) 最小字符集原则:所定编码应采用最小字符集以使问题得到自然的描述。

针对函数优化的编码技术主要有二进制编码、十进制编码、实数编码等。

2.3.1.1 二进制编码方法

二进制编码方法是将原问题的解映射为 0、1 组成编码串的遗传空间,此操作称为

二进制编码，结果再通过解码过程还原成其解空间的解，然后再进行适应度的计算。

由于很多数值与非数值优化问题都可以用二进制编码来应用遗传算法，同时表达的模式最多，所以二进制编码方法是遗传算法中最常用的一种编码方法，它具有以下优点：

- (1) 编码、解码操作简单易行。
- (2) 交叉、变异等遗传操作便于实现。
- (3) 符合最小字符集编码原则。
- (4) 便于用模式定理进行分析，因为模式定理就是以二进制为基础的。

2.3.1.2 格雷码编码方法

格雷码可以克服二进制编码在结构特征，以及一些连续函数的优化问题中显现出局部搜索能力差的缺陷。格雷码在连续的两个整数所对应的编码值之间仅有一个码位不相同，其余码位都相同，从而消除了所谓“Hamming 悬崖”，增强了算法的局部搜索能力。

格雷码是将二进制编码通过一个转换而得到的编码。

设有二进制编码串为 $B=b_mb_{m-1}\cdots b_2b_1$ ，其对应的格雷码为 $G=g_mg_{m-1}\cdots g_2g_1$ 。则将二进制编码转换为格雷码的公式为：

$$\begin{cases} g_m = b_m \\ g_i = b_{i+1} \oplus b_i \quad i = m-1, m-2, \dots, 2, 1 \end{cases} \quad (2-1)$$

式中，符号 \oplus 表示异或运算符。

由格雷码转换为二进制编码的转换公式为：

$$\begin{cases} b_m = g_m \\ b_i = b_{i+1} \oplus g_i \quad i = m-1, m-2, \dots, 2, 1 \end{cases} \quad (2-2)$$

2.3.1.3 实数编码

直接采用十进制的实数编码是连续参数优化问题直接的自然描述，不存在编码和解码过程。在实数编码情况下，我们将一个实参数矢量对应一个染色体，一个实数对应一个基因，一个实值对应一个等位基因。使用实数编码存在如下优点：

(1) 基因以实数编码消除了因编码精度不够，使得搜索空间中具有较优适应度的可能解未能表示出来的隐患。

(2) 遗传算法用实数编码的基因，具备了利用连续变量函数具有的渐变性的能力。

渐变性表示变量小的变化所引起的对应函数值的变化也是小的，我们考虑的问题多为连续函数，都具备这种渐变性。遗传算法的这一特征证明对某些优化问题是至关重要的。

(3)采用浮点数编码可以消除“Hamming 悬崖”。

2.3.2 适应度函数

遗传算法中使用适应度来对个体进行评价。适应度较高的个体遗传到下一代的概率就较大；而适应度较低的个体遗传到下一代的概率就较小。度量个体适应度的函数称为适应度函数。

对于简单的最小化问题，通常可以直接将目标函数变换成适应度函数，譬如将个体 X 的适应度 $f(X)$ 定义为 $M-c(X)$ 或 $e^{-ac(X)}$ ，其中 M 为一足够大正数， $c(X)$ 为个体的目标值， $a>0$ 。对于复杂优化问题，往往需要构造合适的评价函数，使其适应遗传算法进行优化。

由于适应度度量意义下的个体差异与目标函数值度量意义下的个体差异有所不同，因此若适应度函数设计不当，将难以体现个体的差异，选择操作的作用就很难体现出来，从而造成早熟收敛等缺点。对适应度进行调节是常用的改进方法，如线性变换和指数变换等，即通过某种变换改变原适应度间的比例关系。

2.3.3 遗传操作

优胜劣汰是设计遗传算法的基本思想，它应在选择、交叉、变异和种群替换等遗传操作中得以体现，并考虑到对算法效率与性能的影响。

2.3.3.1 种群初始化

鉴于保优遗传算法的概率 1 收敛特性，初始种群通常是随机产生的。但考虑到搜索效率和质量，一方面要求尽量使初始种群分散地分布于解空间，另一方面可以采用一些简单方法或规则快速产生一些解作为遗传算法的初始个体。如优化 Flow Shop 问题时，可以用 NEH 方法产生一个遗传算法初始个体，其余个体则在解空间中均匀选取。

2.3.3.2 选择操作

遗传算法中的选择操作就是用来确定如何从父代群体中按某种方法选取哪些个体遗传到下一代群体中的一种遗传运算。

最常用的选择方法是适应度比例选择、最优个体保存选择和锦标赛选择。

(1) 适应度比例选择(fitness proportional model)

适应度比例选择方法是目前遗传算法中最基本也是最常用的选择方法。它也称为轮盘赌或蒙特卡罗(Monte Carlo)选择。在该方法中,各个个体的选择概率和其适应度值成比例。

设群体大小为 n , 其中个体 i 的适应度值为 f_i , 则 i 被选择的概率 P_{si} 为:

$$P_{si} = \frac{f_i}{\sum_{j=1}^n f_j} \quad (2-3)$$

显然,概率 P_{si} 反映了个体 i 的适应度在整个群体中的个体适应度总和中所占的比例。个体适应度值越大,其被选择的概率就越高,反之亦然。

(2) 最优个体保存选择

使用最优保存策略进化模型来进行优胜劣汰操作,即当代群体中适应度最高的个体不参与交叉和变异运算,而是用它来替换本代群体中经过交叉和变异等操作后所产生的适应度最低的个体。

(3) 锦标赛选择(tournament selection model)

首先在父代种群中随机选取 k 个个体(称为锦标赛规模),然后令其中适应度或目标值最好的个体为被选中个体保存到下一代。这一过程反复执行,直到保存到下一代的个体数达到预先设定的数目为止。

显然, k 的大小影响选择性能,一般情况下取 $k=2$ 。

2.3.3.3 交叉操作

遗传算法中起核心作用的是遗传操作中的交叉操作。所谓交叉操作,是指对两个相互配对的染色体按某种方式相互交换其部分基因,从而形成两个新的个体。交叉运算是遗传算法区别于其他进化算法的重要特征,选择运算并没有建立新解,没有创立新的染色体。交叉算子是产生新个体的主要方法。通过交叉操作,遗传算法的搜索能力得以飞跃提高。

在设计交叉操作时应考虑以下几点:

(1) 交叉操作需保证前一代中优秀个体的性状能在下一代新个体中尽可能得到遗传和继承。

(2)交叉操作的设计和编码设计需要协调进行。

二进制编码遗传算法通常采用单点交叉和多点交叉。

(1)单点交叉

其操作为：首先随机确定一个交叉位置，然后对换交叉点后的子串。例如，父串为(1011|001)和(0010|110)，若单点交叉位置为4，则后代个体为(1011110)和(0010001)。

单点交叉的特点是：若邻接基因座之间的关系能提供较好的个体性状和较高的个体适应度的话，该操作破坏这种个体性状和降低个体适应度的可能性最小。

(2)多点交叉

其操作为：首先随机确定多个交叉位置，然后对换相应的子串。若两点交叉，交叉位置为2, 5，父代个体同上，则后代个体为(10|101|01)和(00|110|10)。

一般来讲，多点交叉不经常被采用。这是因为交叉点的增多，个体的结构被破坏的可能性也逐渐增大，有可能破坏一些好的模式，很难有效地保存较好的模式，从而影响遗传算法的性能。

十进制编码遗传算法的交叉操作类似于二进制编码遗传算法。

实数编码遗传算法通常采用双个体算术交叉或多个个体算术交叉。

(1)双个体算术交叉

针对选中的两个个体进行如下交叉，即 $x_1' = \alpha x_1 + (1-\alpha)x_2$ ， $x_2' = \alpha x_2 + (1-\alpha)x_1$ ，其中，随机数 $\alpha \in (0,1)$ ， x_1, x_2 为父代个体， x_1', x_2' 为后代个体。

(2)多个体算术交叉

针对多个选中个体进行如下交叉， $x' = \alpha_1 x_1 + \dots + \alpha_n x_n$ ，其中， x_i 为父代个体， x' 为后代个体，随机数 $\alpha_i \in (0,1)$ ，且 $\sum_{i=1}^n \alpha_i = 1$ 。

2.3.3.4 变异操作

遗传算法中的变异运算，是指将个体染色体编码串中的某些基因座上的基因值用该基因座的其它等位基因来替换，从而产生一个新的个体。变异具有增加群体多样性的效果，它能避免选择、交叉过程过早收敛到局部最优解。

(1) 基本位变异

对于基本遗传算法中用二进制编码符号串所表示的个体,若需要进行变异操作某一基因座上的原有基因值为 0,则变异操作将该基因值变为 1;反之,若原有基因值为 1,则变异操作将该基因值变为 0。

(2) 均匀变异

均匀变异操作是指分别用符合某一范围内均匀分布的随机数,以某一较小的概率来替换个体编码串中各个基因座上的原有基因值。

(3) 边界变异

在进行边界变异操作时,随机的去基因座的一个对应边界基因值之一去替代原有基因值。

2.3.3.5 替换策略

种群替换策略通常采用整体替换或部分替换。整体替换即将新种群完全覆盖原先种群,而部分替换则用部分新个体替换部分旧个体。譬如, (μ, λ) 策略由 μ 个父代个体产生 λ 个后代个体,然后从后代个体中选取 μ 个最好的个体作为下一代种群; $(\mu + \lambda)$ 策略则从所有 μ 个父代个体和 λ 个后代个体中选取最好的 μ 个个体作为下一代种群。

应该说,遗传算法是一种复杂的非线性随机智能优化计算模型,纯粹用数学方法来预测其运算结果很难。目前为兼顾优化质量和效率所采取的设计方法大多是经验法或试探法,该方面理论还有待更深入地研究与完善。

2.3.4 遗传算法运行参数的选择

标准遗传算法有 4 个关键参数,包括种群数目 (Population size, 记作 M)、交叉概率 (crossover rate, 记作 P_c)、变异概率 (mutation rate, 记作 P_m) 和代沟 (generation gap, 记作 G)。这些参数的不同选取会对遗传算法的性能产生很大的影响,要想得到算法执行的最优性能,必须确定最优的参数设置。

2.3.4.1 种群数目 M

种群数目是影响算法最终优化性能和效率的因素之一。通常,种群太小时,不能提供足够的采样点,以致算法性能很差,甚至得不到问题的可行解;种群太大时,尽管可

增加优化信息以阻止早熟收敛的发生,但无疑会增加计算量,从而使收敛时间太长^[26]。

2.3.4.2 交叉概率 P_c

交叉概率用于控制交叉操作的频率。交叉概率较大时,可增强遗传算法开辟新的搜索区域的能力,种群中串的更新很快,但会使高适应度的个体很快被破坏掉;交叉概率太小时,交叉操作很少进行,交叉产生新个体的速度较慢,从而会使搜索停滞不前。

一般建议取值范围在 0.4~0.99 之间。

2.3.4.3 变异概率 P_m

变异概率直接影响到算法的收敛性和最终解的性能。若变异概率取较大的值,会使算法能不断地探索新的解空间,增加模式的多样性,但较大的变异概率会影响算法的收敛性;若取值较小,则变异产生新个体的能力和抑制早熟现象的能力会变差,一般建议变异概率的取值范围是 0.0001~0.1。

2.3.4.4 代沟 G

代沟用于控制每一代种群中个体被替换的比例,即每代有 $M \times (1-G)$ 个父代个体被选中进入下一代种群。 $G=100\%$ 意味着所有个体将被替换; $G=50\%$ 意味着一半种群将被替换。“代沟”方式的选用为遗传算法利用优化过程的历史信息提供了条件,加速了遗传算法的收敛过程。当代沟过小时,可能导致遗传算法过早的不成熟收敛。一般取 0.30~1.00。

2.3.4.5 算法的终止条件

遗传算法的收敛理论说明了遗传算法具有概率 1 收敛的极限性质,然而实际算法通常难以实现理论上的收敛条件。因此,追寻的目标应该是具有一定优化质量的快速搜索。由于实际应用遗传算法时不允许其无停止地进行搜索,同时问题的最优解也通常未知,因此必须设计一些近似收敛准则来终止算法进程。常用方法有^[27]:

(1) 停止规则是给定一个最大的遗传进化代数,当达到此值时,就停止运行,并将当前群体中的最佳个体作为所求问题的最优解输出。一般建议的取值范围为 100~1000。

(2) 当判断出群体已经进化成熟,而且连续若干步不再有明显的进化趋势时,便可终止算法的运行。

尽管遗传算法是一种有效的优化算法，但最佳遗传算法参数是依赖于问题的，要确定它本身就是一个极其复杂的优化问题。

2.4 遗传算法的特点

遗传算法利用生物进化和遗传的思想实现优化，是寻求优化问题的效率和稳定性之间的有机协调，计算方法新颖独特，与传统优化计算算法相比有以下特点：

- (1) 遗传算法是对问题参数的编码(染色体)群进行进化，而不是参数本身。
- (2) 遗传算法的搜索是从问题解的串集开始搜索，而不是从单个解开始。
- (3) 遗传算法使用适应度这一信息进行搜索，而不需导数等其它信息。
- (4) 遗传算法使用的选择、交叉、变异这三个算子都是随机操作，而不是确定规则。
- (5) 遗传算法最善于搜索复杂地区，从中找出期望值高的区域。
- (6) 遗传算法具有可扩展性，易于同别的算法技术混合。

2.5 遗传算法的改进

尽管遗传算法在理论上具有概率 1 的收敛特性，但实际应用时往往出现早熟收敛或收敛缓慢等缺点。因此，设计遗传算法的框架、操作和参数，主要应针对如何设法高效产生或有助于产生优良的个体成员，而这些成员应能够充分表征整个解空间的特性，从而提高算法的搜索效率，避免早熟收敛现象。自从 1975 年 Holland 系统地提出遗传算法完整结构和理论以来，众多学者一直致力于推动遗传算法的发展，对编码方式、控制参数的确定和交叉机理等进行了深入的研究，提出了各种改进的遗传算法。主要的改进遗传算法有：

2.5.1 层次遗传算法

对于一个问题，首先随机生成 $N \times n$ 个样本，然后将它们分成 N 个子种群，每个子种群包括 n 个样本，对每个子种群独立地运行各自的遗传算法，记它们为 GA_i ($i=1,2,\dots,N$)。在每个子种群运行道一定代数后，将 N 个遗传算法的结果种群记录到二维数组 $R[1,\dots,N;1,\dots,n]$ 中，则 $R[i,j]$ ($i=1,2,\dots,N; j=1,2,\dots,n$) 表示 GA_i 的结果种群的第 j 个个体。同时，将 N 个结果种群的平均适应值记录到数组 $A[1,\dots,N]$ 中， $A[i]$ 表示 GA_i 的结果种群平均适应值。该高层遗传算法的操作可分为三步：

步骤 1: 选择, 基于数组 $A[1, \dots, N]$, 即 N 个遗传算法的平均适应度值, 对数组 $R[1, \dots, N; 1, \dots, n]$ 代表的结果种群进行选择操作, 一些结果种群由于它们的平均适应度值高而被复制, 甚至复制多次; 另一些结果种群由于它们的平均适应度值低而被淘汰。

步骤 2: 交叉, 如果 $R[i, 1, \dots, n]$ 和 $R[j, 1, \dots, n]$ 被随机地匹配到一起, 而且从位置 x 进行交叉 ($1 \leq i, j \leq N; 1 \leq x \leq n-1$), 则 $R[i, x+1, \dots, n]$ 和 $R[j, x+1, \dots, n]$ 相互交换相应的部分。这一步骤相当于交换 GA_i 和 GA_j 中结果种群的 $n-x$ 个个体。

步骤 3: 变异, 以很小的概率将少量的随机生成的新个体替换 $R[1, \dots, N; 1, \dots, n]$ 中随机抽取的个体。

至此, 高层遗传算法的第一轮结束。 N 个遗传算法 $GA_i (i=1, 2, \dots, N)$ 可以从相应于新的 $R[1, \dots, N; 1, \dots, n]$ 个种群继续各自的操作。在 N 个 GA_i 各自运行到一定代数后, 再次更新数组 $R[1, \dots, N; 1, \dots, n]$ 和 $A[1, \dots, N]$, 并开始高层遗传算法的第二轮运行。如此继续循环操作, 直至得到满意的结果。

上述改进的遗传算法称为层次遗传算法 (HGA)^[28]。这种改进的遗传算法与并行或分布式遗传算法相比, 上一层的个体交换中不需要人为地控制或交换什么样的个体, 也不需要人为地指定处理器将传送出的个体送往哪一个处理器, 或者从哪一个处理器接受个体。这种改进的遗传算法不但在每个处理器上运行着遗传算法, 同时对各处理器不但生成新种群进行着高一层的运算和控制。

2.5.2 CHC 算法

CHC 算法是 Eshelman 于 1991 年提出的一种改进的遗传算法^[29], 第一个 C 代表跨世代精英选择策略, H 代表异物种重组, 第二个 C 代表大变异。CHC 算法与基本遗传算法不同点在于: 基本遗传算法的遗传操作比较单纯, 简单地实现并行处理; 而 CHC 算法牺牲了这种单纯性, 换取了遗传操作的较好效果, 并强调优良个体的保留。

步骤 1: 选择, 上世代种群与通过新的交叉方法产生的个体种群混合起来, 从中按一定概率选择较优的个体。这一策略称为跨世代精英选择。其明显的特征表现在: 1) 健壮性: 由于这一选择策略, 即使当交叉操作产生较劣个体偏多时, 由于原种群大多数个体残留, 不会引起个体的评价价值降低; 2) 遗传多样性保持: 由于大个体群操作, 可以更好地保持进化过程中的遗传多样性; 3) 排序方法: 克服了比例适应值计算中的尺度变换

问题。

步骤 2: 交叉, CHC 算法使用的重组操作是对均匀交叉的一种改进: 当两个个体位值相异的位数为 m 时, 从中随机地选取 $m/2$ 个位置, 实行父代个体位置的互换。显然, 这样的操作对模式具有很强的破坏性。

步骤 3: 变异, CHC 算法在进化前期不采取变异操作, 当种群进化到一定的收敛时期, 从优秀个体中选择一部分个体进行初始化。初始化的方法是选择一定比例的基因座, 随机地决定它们的位置。这个比例值称为扩散率。

2.5.3 Messy 遗传算法

根据积木块假设, SGA 中定义距长的模式容易受到破坏, 只有从小积木块的模式中才能最终构成最优解, 这对进化模拟非常不利。为了克服这一缺点, Goldberg 等在 1989 年提出了一种变长度染色体遗传算法即 Messy GA^[30]。该算法在不影响模式定义距的情况下, 使优良的模式得以增殖。

在生物进化过程中, 其染色体的长度并不是固定不变的, 而是随着进化过程在慢慢地变化。另一方面, 在遗传算法的实际应用中, 有时为简化描述问题的解, 也需要使用不同长度的编码串。

Messy 遗传算法将常规的遗传算法的染色体编码串中各基因座位置及相应的基因值组成一个二元组 (2-Tuple), 把这个 2-Tuple 按一定的顺序排列起来, 就组成一个变长染色体的一种编码方式, 一般表示为:

$$X^m : (i_1, v_1) (i_2, v_2) \dots (i_k, v_k) \dots (i_n, v_n)$$

上述变长度染色体描述形式中, i_k 是所描述的基因在原常规染色体中的基因座编号, v_k 为对应的基因值。对于所求解的问题, 若使用常规遗传算法时的染色体长度固定为 l , 各基因值取自集合 V , 则有:

$$1 \leq i_k \leq l, \quad k=1, 2, \dots, n$$

$$v_k \in V, \quad k=1, 2, \dots, n$$

Messy 遗传算法由于编码长度可变, 遗传操作算子选择具有特殊性。一般选择算子选用锦标赛选择方法, 不再使用通用的交叉算子, 而代之以切断算子和拼接算子。切断

算子是以某一预先指定的概率，在变长度染色体中随机选择一个基因座，使之成为两个个体的基因型。拼接算子是以某一预先指定的概率，将两个个体的基因型连接在一起，使它们合并成一个个体的基因型。这两个算子的处理过程称为并列阶段。

2.5.4 自适应遗传算法

遗传算法的参数中，交叉概率 P_c 和变异概率 P_m 的选择是影响遗传算法行为和性能的关键所在，直接影响着算法的收敛性。 P_c 越大，新个体产生的速度就越快，但此时，遗传模式被破坏的可能性也越大，会使得具有高适应值的个体结构很快被破坏。但是如果 P_c 过小，会使搜索过程缓慢，以致停滞不前。对于变异概率 P_m ，如果 P_m 过小，不易产生新的个体结构；如果 P_m 取值过大，那么遗传算法就变成了纯粹的随机搜索算法。针对不同的优化问题，需要反复实验来确定 P_c 和 P_m ，并且很难找到适应于每个问题的最佳值。为此，Srinivas 等提出了自适应遗传算法 (AGA) [31]， P_c 和 P_m 能够随适应值自动改变。当种群各个体适应值趋于一致或者趋于局部最优时，使 P_c 和 P_m 增加；而当群体适应值比较分散时，使 P_c 和 P_m 减小。同时，对于适应值高于群体平均适应值的个体，对应于较小的 P_c 和 P_m ，使该解得以保护进入下一代；而低于平均适应值的个体，对应于较高的 P_c 和 P_m ，使该解被淘汰。因此自适应的 P_c 和 P_m 能够提供相对于某个解的最佳 P_c 和 P_m 。自适应遗传算法在保持群体多样性的同时，也保证了遗传算法的收敛性。

在自适应遗传算法中， P_c 和 P_m 按如下公式进行自适应调整：

$$p_c = \begin{cases} \frac{k_1(f_{\max} - f')}{f_{\max} - f_{\text{avg}}}, & f \geq f_{\text{avg}} \\ k_2, & f < f_{\text{avg}} \end{cases} \quad (2-4)$$

$$p_m = \begin{cases} \frac{k_3(f_{\max} - f')}{f_{\max} - f_{\text{avg}}}, & f \geq f_{\text{avg}} \\ k_4, & f < f_{\text{avg}} \end{cases} \quad (2-5)$$

式中， f_{\max} 为群体中最大的适应值， f_{avg} 为每代群体中的平均适应值， f' 为要交叉的两个个体中较大的适应值， f 为要操作的个体的适应值。这里，只有设定 k_1, k_2, k_3, k_4 取区间 (0,1) 中的值，就可以自适应调整了。

另外，还可以采用其它的自适应调整规律，作进一步的改进，从而提高群体中表现

优良的个体的变异率和交叉率。

2.5.5 小生境遗传算法

小生境技术是将每一代个体划分为若干类，每个类中选出若干适应值较大的个体作为一个类的优秀代表组成一个种群，再在该种群以及不同种群之间通过杂交、变异产生新一代个体群，同时采用预选择机制或排挤机制或分享机制完成选择操作^[32]。

Cavichio 在 1970 年提出了基于选择机制的选择策略。其基本做法是：当新产生的子代个体的适应值超过其父代个体的适应值时，所产生的子代个体才能代替其父代个体而遗传到下一代群体中，否则父代个体仍保留在下一代群体中。由于子代个体和父代个体之间编码结构的相似性，所以替换掉的只是一些编码结构相似的个体，故它能够有效地维持群体的多样性，并造就小生境的进化环境。De Jong 在 1975 年提出了基于排挤机制的选择策略。其基本思想源于在一个有限的生存空间，各种不同的生物为了延续生存，它们之间必须相互竞争各种有限的资源。因此，在算法中设置一个排挤因子 CF ，由群体中随机抽取的 $1/CF$ 个个体组成排挤成员，然后依据新生成的个体与排挤成员的相似性来排挤一些与排挤成员相似的个体量。随着排挤过程的进行，群体中的个体逐渐被分类，从而形成一个个小的生存环境，并维持了群体的多样性。1987 年 Goldberg 等提出了基于共享的选择策略。其基本做法是：通过个体之间相似程度的共享函数来调整群体中各个个体的适应值，从而在群体的进化过程中，算法能够依据这个调整后的新适应值来进行选择操作，以维护群体的多样性，创造出小生境的进化环境。

基于这种小生境技术的遗传算法 (NGA)，可以更好地保持解的多样性，同时具有很高的全局寻优能力和收敛速度，特别适合于复杂多峰函数的优化问题。

2.5.6 并行遗传算法

目前，并行遗传算法的实现方案大致可分为三类^[33, 34]：

(1) 全局型——主从式模型：并行系统分为一个主处理器和若干个从处理器。主处理器监控整个染色体种群，并基于全局统计执行选择操作。各个从处理器接受来自主处理器的个体进行重组交叉和变异，产生新一代个体，并计算适应值，再把结果传给主处理器。

(2) 独立型——粗粒度模型：将种群分成若干子群并分配给各自对应的处理器，每

个处理器不仅独立计算适应值,而且独立进行选择、交叉和变异操作,还要定期相互传递适应值最好的个体,从而加快满足终止条件的要求。

粗粒度模型也称为孤岛模型,基于粗粒度模型的遗传算法也称为分布式遗传算法,是目前应用最为广泛的一种并行遗传算法。

(3)分散型——细粒度模型:为种群中的每一个个体分配一个处理器,每个处理器进行适应值的计算,而选择、交叉和变异操作仅在与之相邻的一个处理器之间相互传递的个体中进行。

细粒度模型也称为领域模型,适合于连接机、阵列机和 SIMD 系统。目前, Manderick 和 Spiessens 基于二维网络拓扑结构开发了一种大规模并行遗传算法,用于研究同类群体大小及染色体长度对于算法性能的影响。Muhlenbein 提出了一种异步通信模式并将它应用到复杂组合问题上。Fosak 将群体中个体映射到一个连接机的处理单元,并指出这种方法对网络图设计的有效性。

迁移是并行遗传算法引入的一个新的算子,是指在进化过程中子群体间交换个体的过程。一般的迁移方法是将于群体中最好的个体发给其他的子群体,通过迁移可以加快较好的个体在群体中的传播,提高收敛速度和解的精度。因此,迁移算子的采用使并行遗传算法更适用于全局寻优,并且计算量较小。以基于粗粒度模型的并行遗传算法为例,其迁移策略可分为两种:1)一传多:每个处理器对应若干个相邻的处理器,每个处理器产生新一代个体后都将自己最好的一个个体传送给其所有相邻的处理器,每个处理器产生新一代个体后将这些个体与自己的个体同时考虑,淘汰适应值差的个体;2)一传一:考虑到染色体的多样性,每个处理器都将自己最好的个体传给相邻的一个处理器,同时增加两个参数,一是 send rate 决定处理器之间通信的频率,二是 send best 决定每次传送给最好个体的数目。

一般地,个体迁移的选择方法可以有两种:1)均匀随机调选个体作为迁移对象;2)按适应值挑选个体作为迁移对象。对于子群体之间的个体迁移结构也有多种可能:1)迁移发生在所有子种群,即发生在完全的网络拓扑;2)迁移发生在环状拓扑;3)迁移发生在领集拓扑。

最一般化的迁移模型是完全网络拓扑,个体在众多的子群体之间相互迁移,移民均

分布在大种群中。最基本的迁移模型是环状拓扑模型,个体迁移发生在相邻的子种群上。而在邻集迁移模型中,迁移只发生在较近的邻集中。

2.5.7 混合遗传算法

混合遗传算法是融合最速下降法、模拟退火算法等很强的局部搜索能力和启发式算法的高效率的思想,构成的一种新的算法^[35]。目前,混合遗传算法实现方法体现在两个方面:1)引入局部搜索过程;2)增加编码变换操作过程。在构成混合遗传算法时,De Jong提出了三个基本原则:1)尽量采用原有算法的编码;2)利用原有算法全局搜索的优点;3)改进遗传算子。

综合最速下降法和遗传算法在最优问题上各自的优势,可产生适合于连续可微函数全局优化问题的混合算法。在这种混合算法中,为加速遗传算法在全局优化问题上的收敛性,发挥传统数值优化算法在计算速度与计算精度上的优势,在遗传算法中嵌入一个最速下降算子。混合遗传算法中的遗传算子——交叉算子、变异算子和选择算子的作用是宏观搜索,处理的是大范围搜索问题,而最速下降算子中的线性搜索过程的作用是极值局部搜索,即微观搜索,处理的是小范围解决和搜索加速问题。

遗传算法在运行早期个体差异较大,当采用经典的轮盘赌方式选择,后代产生个数与父代适应值大小成正比,因此在早期容易使个别好的个体的后代充斥整个种群,造成早熟。在遗传算法后期,适应值趋向一致,优秀的个体在产生后代时,优势不明显,从而使整个种群进化停滞不前。因此,对适应值适当地拉伸是必要的。Paul L. Stoffa 借鉴模拟退火思想,提出了模拟退火遗传算法(SAGA)。该算法采用如下的适应值拉伸方法:

$$f_i = \frac{e^{f_i/T}}{\sum_{i=1}^M e^{f_i/T}} \quad (2-6)$$

$$T = T_0 (0.99^{g-1}) \quad (2-7)$$

式中, f_i 为第 i 个个体的适应值, M 为种群大小, g 为遗传代数, T 为温度, T_0 为初始温度。这样,在温度高时(遗传算法的前期),适应值相近的个体产生的后代概率相近;而当温度不断下降后,拉伸作用加强,使适应值相近的个体适应值差异放大,从而使得优秀的个体优势更明显。

2.6 小结

遗传算法在求解复杂的组合优化问题中表现出非常明显的优势，但也受到诸如遗传算子的选择和种群规模、变异概率、交叉概率等参数确定问题的影响，为此，本章对遗传算子的选择、变异、交叉方法作了详细的说明，同时介绍了一些有关遗传算法的基本理论，编制了遗传算法流程图，给出了几种改进的遗传算法，为生产调度遗传算法的应用奠定了基础。

3 基于 Job Shop 调度问题的遗传算法

3.1 引言

作业车间调度问题的本质是建立工序在每台机器上的排列,该排列可以满足用于最小化生产时间的先后约束。如果能够得到这样的排列,就可以很轻松地通过依赖于问题的过程得到问题的解。

基于遗传算法的作业车间调度研究最重要的工作之一就是如何进行遗传算法编码,由于作业车间调度问题具有离散、动态、多机、多变量耦合等多种属性,所以作业车间调度遗传算法编码具有一定的难度,一直成为遗传算法直接应用于作业车间调度的瓶颈口。由于受工序加工路线的约束,作业车间调度问题不像旅行商问题和流水车间调度问题那样容易确定一个自然表达。到目前为止,还没有一个用系统不等式来表达先后约束的好方法。因此,不便于应用惩罚法来处理这些类型的约束。将遗传算法应用于作业车间调度问题的一般方法为:

- (1)采用遗传算法来进化适当的排列;
- (2)采用启发式方法来根据排列构造对应的解。

3.2 Job Shop 调度问题的描述

一般地,Job Shop 调度问题可描述为 n 个加工顺序不同的工件要在 $m(>2)$ 台机器上完成加工,已知:

- (1)工件集 $P=\{p_1, p_2, \dots, p_n\}$, p_i 为第 i 个工件, $i=1,2,\dots,n$;
- (2)机器集 $M=\{m_1, m_2, \dots, m_m\}$, m_j 为第 j 台机器, $j=1,2,\dots,m$;
- (3)工序序列集 $OP=\{op_1, op_2, \dots, op_m\}$, $op_i=\{op_{i1}, op_{i2}, \dots, op_{im}\}$ 为工件 p_i 的工序序列。 op_{ik} 为第 i 个工件在第 k 道工序使用的机器号。 $op_{ik}=0$ 表示工件 p_i 在第 k 道工序不加工, $k=1,2,\dots,m$;
- (4)每个工件使用每台机器的时间矩阵 T , $t_{ij} \in T$ 为第 i 个工件 p_i 使用第 j 个机器的时间。当 $t_{ij}=0$ 时表示工件 p_i 不使用机器 j 。

而且问题满足下列条件:

- (1)每个工件使用每台机器不多于 1 次;

(2) 每个工件利用每台机器的顺序可以不同, 即可以有 $op_i \neq op_d$ ($i \neq d$);

(3) 任何工件没有抢先加工的优先权, 服从任何生产顺序;

(4) 工件加工过程中没有新工件加入, 也不临时取消工件的加工;

求关于所有工件 p_i 的排序 S_1, S_2, \dots, S_n , 使总生产费用最小。即

$$MS = \min f(S_1, S_2, \dots, S_n) \quad (3-1)$$

其中, S_q 为排序序列中位序为 q 的工件序号, $q=1, 2, \dots, n$ 。

通常我们要求取的是最小化最大完工时间问题, 即:

$$\min f(S_1, S_2, \dots, S_n) = \min \{c_i\}, \quad i=1, 2, \dots, n. \quad (3-2)$$

其中, c_i 表示工件 i 的完工时间。

对于以上描述的调度问题, 全部工件的加工排序方式有 $(\sum_{i=1}^n op_i)! \cdot n!$ 种, 但是 Job Shop 是加工路线完全给定的加工系统, 其合法的调度总数为:

$$\left(\frac{1}{op_1!} * \frac{1}{op_2!} * \dots * \frac{1}{op_n!}\right) * (\sum_{i=1}^n op_i)! \quad (3-3)$$

调度的任务就是从这么多个排序中选择最优的或是较优的一种调度, 大规模生产过程中工件的加工调度总数将是非常大的。

Job Shop 调度问题的求解远复杂于旅行商 (TSP) 问题, 其原因可归纳如下:

(1) 调度解的编码复杂且多样化, 算法的搜索操作多样化。

(2) 解空间容量巨大, n 个工件、 m 台机器的问题包含 $(n!)^m$ 种排列。

(3) 存在工艺技术约束条件的限制, 必须考虑解的可行性。

(4) 调度指标的计算相对算法的搜索比较费时, 将搜索状态解码为有效调度的操作复杂且多样化。

(5) 优化超曲面缺少结构信息, 通常复杂且存在多个分布无规则甚至彼此相邻的极小, 对算法的优化造成很大困难。

3.3 求解 Job Shop 调度问题的标准遗传算法设计

3.3.1 编码/解码方式

基于遗传算法的 Job Shop 调度研究最重要的工作之一就是如何进行遗传算法编码,

由于 Job Shop 调度问题具有离散、动态、多机、多变量耦合等多种属性,所以 Job Shop 调度问题的遗传算法编码具有一定的难度,一直成为遗传算法直接应用于 Job Shop 调度的瓶颈。由于受工序加工路线的约束,作业车间调度问题不像旅行商问题和流水车间调度问题那样容易确定一个自然表达。

遗传算法的编码技术必须考虑“染色体”的合法性、可行性、有效性以及对问题解空间表征的完全性。一个好的编码方案应该是编码空间到可行解空间的一一映射。鉴于 Job Shop 调度问题的组合特性以及工艺约束性,染色体的 Lamarkian 特性、解码的复杂性、编码存储量的需求是设计遗传算法编码通常要考虑的问题^[36]。

(1) 染色体的 Lamarkian 特性

该特性考虑在所设计的编码技术下,染色体的优点(merit)是否可通过一般的遗传操作传到后代种群。如果后代通过遗传操作可以有效继承父代的优点,则称所采用的编码具有 Lamarkian 特性;如果后代不能够从父代继承任何有用信息,则称所采用的编码不具有 Lamarkian 特性;如果后代所继承的片断只有部分与父代相同,则称所采用的编码仅具有半 Lamarkian 特性。鉴于遗传算法的优化机制,通常我们希望所设计的编码具有 Lamarkian 特性。

(2) 解码的复杂性

解码是基因型到表现型的转换,也就是将染色体串变换为问题的解。就解码复杂性而言,将不需要解码的相应编码归为 0 类复杂性;通过简单映射关系实现编码的相应编码归为 1 类复杂性;通过使用简单启发式方法才能实现的解码的相应编码归为 2 类复杂性;只有通过复杂启发式规则才能实现解码的相应编码归为 3 类复杂性。

而遗传算法执行过程中所需的内存主要取决于染色体的编码长度。对于 n 个工件 m 台机器问题,如果定义染色体的标准长度为 $n \times m$ 个基因,作业车间调度问题的编码可以分为三类:编码长度小于标准长度;编码长度大于标准长度;编码长度等于标准长度。

(3) 存储量需求

就 n 个工件 m 台机器的 Job Shop 调度而言,通常用编码长来反映编码对存储量的需要。称 $n \times m$ 为遗传算法染色体的标准长度,即操作的总数量。基于此,可将编码分为三类:其一,码长等于标准长度;其二,码长大于标准长度;其三,码长小于标准长

度。

在 Job Shop 生产得到问题的编码方案中，主要有以下几种^[37]：

3.3.1.1 基于操作的编码(operation-based representation)

基本思想是将所有任务的操作进行编码，不同的任务用不同的编码表示，而一项任务的所有操作在染色体中则用相同的编码表示，其解码原则是将染色体上的基因按照从左到右的顺序解释为相应任务的操作，具有相同编码的基因按照其在整个染色体中的位置解释为任务相应顺序的操作。这是一种直接对调度方案进行编码的方式。显然，这种解码过程可产生活动调度。

该编码方式的特点可归纳为：半 Lamarkian 性；1 类解码复杂性；任意基因串的置换排列均能表示可行调度（但为保证后代的可行性，遗传操作需特殊设计）； $n \times m$ 标准长度。

3.3.1.2 基于工件的编码(job-based representation)

由包含 n 个工件的列表组成，每个调度根据工件的顺序来构造。对于一个给定的工件顺序，列表中第一个工件的所有工序将被首先调度，然后考虑列表中第一个工件的工序，依次进行，直到工件的所有工序排完。加工中的第一道工序要求的加工时间应为相应机器最可能得到的加工时间。

将每个染色体用 n 个代表工件的基因组成，是所有工件的一个排列。解码过程是：先加工第 1 号工件的所有操作，然后依次以最早允许加工时间加工后面各工件的所有操作。

基本思想是只对任务进行编码，然后按照编码在染色体中的顺序来以相应的优先级别安排任务。这也是一种直接编码方式。

该编码方式的特点可归纳为：Lamarkian 性；1 类解码复杂性；任意工件的置换均能表示可行调度，但仅能表征部分解空间，不能保证全局最优解的存在性；编码长度小于标准长度。

3.3.1.3 基于先后表的编码(preference list-based representation)

每个染色体用分别对应于 m 台不同机器的 m 个子串构成，各子串是一个长度为 n 的符号串，用于表示该机器上加工工件的一个排列顺序。

该编码方式的特点可归纳为：半 Lamarkian 性；2 类解码复杂性；各状态均能表示

可行调度，但难以用遗传操作实行进化；码长为标准长度。

3.3.1.4 基于工件对关系的编码

Nakano 等利用二元矩阵表示调度，矩阵由相应机器上工件对的优先关系确定。

该编码方式的特点可归纳为：半 Lamarkian 性；1 类解码复杂性；存在较大的冗余；必须考虑合法性；码长大于标准长度。这种编码目前应用较少，也是所有编码方式中最复杂的一种。

3.3.1.5 基于优先规则的编码(priority rule-based representation)

将每个染色体用一个长度 $n \times m$ 的优先分配规则序列构成，每个基因即为一种优先调度规则。算法的优化结果是一个满意的规则序列，然后依次用这些优先规则产生或修改调度方案。

3.3.1.6 基于完成时间的编码(completion time-based representation)

这是一种最直观的方法，它是由 Yamada 和 Nakano 提出了一种基于完成时间的编码方法，一个染色体被编码为各操作完成时间的有序表。

该编码方式的特点可归纳为：不具有 Lamarkian 性；0 类解码复杂性，即无需解码过程；必须考虑状态的合法性，且需要设计特殊的遗传算子；标准长度。

3.3.2 适应度函数

目标函数值区别于适应度值，目标函数值越小适应度值越大，所以需将目标函数值进行转化，从而计算出一代所有显性染色体的适应度值。

一般情况下，可取目标函数的倒数为适应度函数。

3.3.3 选择算子

选择算子采用了轮盘赌方法，在解码过程中得到每个染色体个体的目标函数值，其中每个个体被选中的概率为：

$$P_i = \frac{f_i}{\sum_{i=1}^M f_i} \quad (3-4)$$

其中， f_i 为第 i 个个体的适应度值。

3.3.4 交叉算子

鉴于 Job Shop 调度问题的特殊性, 遗传算法必须结合所采用的编码技术设计相应的交叉操作。考虑到置换编码成为目前遗传算法解决调度问题的主流, 在此仅介绍若干针对置换编码的交叉操作^[38]。

3.3.4.1 部分映射交叉 (partially mapped crossover, PMX)

由 Goldberg 和 Lingle 提出。该方法可以看成是两点交叉的变形, 它通过引入一种特殊的修补过程来处理可能出现的非法染色体的情况, 该方法的主要步骤为:

步骤 1: 在染色体串上随机选择两个交叉点。由两个交叉点定义的子串称为映射片断 (mapping sections)。

步骤 2: 在两个父代间交换子串从而产生原型后代。

步骤 3: 确定两个映射片断间的映射关系。

步骤 4: 采用映射关系使后代染色体合法化。即对于交叉点外的基因, 若它不与换过来的映射片段冲突则保留, 若冲突则通过部分映射来确定直到没有冲突的基因, 从而获得后代个体。

例如: 两个父代个体为 $p_1=(264|7358|91)$, $p_2=(452|1876|93)$, 若交叉位置为 3, 7。则片段(7358)和(1876)将交换。对于 p_1 的剩余基因, 由于 2 不与(1876)冲突则直接填入, 6 存在冲突, 6 的映射基因为 8 仍存在冲突, 8 的映射基因为 3 不存在冲突, 则将 3 填入后代个体 q_1 的相应位置。依此类推, 得到后代个体 $q_1=(234|1876|95)$, 类似地可得到 $q_2=(412|7358|96)$ 。PMX 算子一定程度上满足 Holland 模式定理的基本性质, 子串能够继承父串的有效模式。

3.3.4.2 次序交叉(order crossover, OX)

次序交叉由 Davis 提出。该方法可以看成是 PMX 的一种变形, 它采用了不同的修补过程。OX 的主要步骤如下:

步骤 1: 随机从一个父代中选择一条子串。

步骤 2: 通过将子串复制到与其父代相对应的位置上产生原型后代。

步骤 3: 将第二个父代中所有的子串中已有的符号全部删除掉, 余下的顺序包含了原型后代所需的符号。

步骤 4: 根据从左到右的顺序将符号放入原型后代中余下的空位中, 这样就产生了一个后代。

例如, 若父代个体和交叉点同上, 则片段(7358)和(1876)将交换, 从第 2 交叉位置起 p_1 删除(7358)后剩余(92435), 然后将其填入 q_1 就得到后代个体为 $q_1=(435|1876|92)$, 相应地可得到个体 $q_2=(216|7358|94)$ 。

3.3.4.3 基于位置的交叉 (position-based crossover, PX)

基于位置的交叉由 Syswerda 提出。该方法基本上是针对字母排列编码并结合修改过程的均匀交叉, 它与 OX 类似, 该方法首先产生一个随机的掩码, 然后根据掩码交换父代中对应的基因。交叉掩码就是与染色体长度相同的二进制串。掩码中每位的奇偶性决定了后代中该位从哪个父代所对应的位继承。由于均匀交叉对于字母排列编码将产生非法解, 基于位置的交叉采用了一种修补过程来解决非法现象。基于位置的交叉的主要步骤为:

步骤 1: 从一个父代中随机选择位置的集合。

步骤 2: 将父代中这些位置的符号复制到原型子代中对应的位置上。

步骤 3: 在第二个父代中删除已经被选择的符号。余下的顺序包含了原型子代需要的符号。

步骤 4: 根据顺序从左到右将符号放入原型子代余下的空位中, 这样就产生了一个子代。

例如, 若父代个体同前, 假设随机选取的位置点为 2, 3, 6, 8, 则后代为 $q_1=(652437891)$, $q_2=(264185793)$ 。

3.3.4.4 基于次序交叉 (order-based crossover)

基于次序的交叉也是 Syswerda 提出的。该方法与基于位置的交叉有一点不同, 它将一个父代中选出的位置上的符号次序加在另一个父代对应的位置上。

3.3.4.5 子顺序交换交叉 (subsequence exchange crossover, SEC)

Kobayashi, OnoYamamura 针对 TSP 类似思想的影响提出该方法采用作业顺序矩阵作为编码方式。该方法是采用作业顺序矩阵作为编码方式。对于 n 个作业 m 个机器的问

题, 编码是 $m \times n$ 矩阵, 其中每一行代表一台机器上所有操作的序列或工件的排列(此排列为 PLS), 子顺序定义为一个作业的集合, 两个父代中的这些作业都在一台机器上连续进行加工, 但加工的次序不一定相同。该方法的基本步骤为:

步骤 1: 确定父代之间每台机器的子顺序。

步骤 2: 在父代之间交换这些子顺序来生成后代。

3.3.4.6 基于作业的次序交叉 (job-based order crossover)

该方法也是为作业顺序的矩阵编码方式设计的交叉。该方法的基本步骤为:

步骤 1: 从父代中确定作业的集合, 每台机器对应一个集合。

步骤 2: 对于每台机器, 从第一个父代中将选出的作业复制到第一个子代中对应的位置上。用相同的方法处理第二个子代。

步骤 3: 根据第二个父代中未选中作业从左到右的顺序填充到第一个子代中未确定的作业。用相同的方法处理第二个子代。

3.3.5 变异算子

在过去的 10 年里提出了若干针对排列表示的变异算子, 包括反转变异、插入变异、位移变异、互反变异和移动变异。

(1) 反转变异(inversion mutation): 在染色体中随机寻找两个位置, 然后将这两个位置之间的子串进行反转。

(2) 插入变异(insertion mutation): 随机选择一个基因并将其插入染色体中一个随机的位置。位移变异随机选择子串, 然后将其插入一个随机的位置。插入变异可以看作位移变异的特例, 此时子串仅包含一个基因。

(3) 互反变异(reciprocal exchange mutation): 随机选择两个位置, 然后交换这些位置上的基因。

(4) 移动变异(Shift mutation): 随机选择一个基因, 然后将其随机向左或向右移动到一个新的位置上。

当然, 变异操作的设计也远远不限于这些, 任何能够做到改变个体的邻域操作都可认为是变异操作, 并可加以尝试。另外, 也可以将各种不同的变异操作结合使用, 以丰富变异的方式, 或者说丰富后代的探索方式。

3.4 Job Shop 调度问题的改进的遗传算法求解

3.4.1 嫁接遗传算法描述

以上给出的标准遗传算法在求解大规模的 Job Shop 调度问题时存在两大局限——进化速度过慢和过早收敛（早熟）。为了克服这一问题，受植物学嫁接思想的启发提出一种改进的算法——嫁接遗传算法（Grafted Genetic Algorithm, GGA）。该算法通过引入嫁接种群可以同时实现加快进化和抗早熟的目的，交叉概率矩阵和多交叉算子的采用则是为了提高抗早熟的效率和解的质量。

为了加快进化速度并且防止早熟，嫁接遗传算法的初始种群有两个：一个称之为待进化种群，其地位等同于标准遗传算法中的初始种群，待进化种群的初始染色体均为随机生成，整体适应度比较低；另一个种群是新引入的嫁接种群，其初始染色体是根据启发式规则生成的，因此，染色体的适应度远远高于待进化种群中的染色体的适应度。引入嫁接种群的目的是通过两个种群间染色体的交叉操作来加快待进化种群的进化速度。为了保持嫁接种群的多样性及提高抗早熟的能力，嫁接种群中的染色体一定要保持惟一性。

待进化种群的进化和标准遗传算法的种群进化可采用相同的策略，嫁接种群进化时则要保证染色体的惟一性，因此只有当新产生的子代染色体优于嫁接种群中最差的染色体并且该子代染色体在嫁接种群中不存在时，该子代染色体才可以淘汰嫁接种群中的最差染色体，进而实现嫁接种群的进化。

嫁接遗传算法的交叉是在两个种群中的染色体之间进行的，即分别从嫁接种群和待进化种群中选择一个染色体作为母体进行交叉。这样可以把嫁接种群中的较优染色体的有用信息直接嫁接给待进化种群中的染色体。通过嫁接种群的指导，待进化种群可以明显地并且在整个进化过程中始终加快进化速度。

同时，本算法中的双种群设计对于提高算法抗早熟方面的效率同样有效。在标准遗传算法中，种群的进化是通过选择机制来体现的。当种群繁殖时，适应度较大的染色体进入新一代种群的概率较大，这些较优秀的染色体便在新一代种群中占据了数目上的优势。当新一代的染色体实施交叉操作时，较优染色体之间的近亲结合容易遗失群体多样性的特征，从而失去进化的意义，出现了所谓的早熟现象。而嫁接遗传算法的交叉是在

两个种群中选择母体，嫁接种群中的染色体具有惟一性和多样性，使得选择两个相似或相同染色体作为母体进行交叉的概率明显降低，从而在很大程度上防止了早熟。

据此，我们可以了解到，正是种群中各个染色体之间的不平衡进化导致了早熟。而不平衡进化的原因之一则在于交叉概率为一个固定的值，从而使得数目占优的较优染色体被进化的概率增大。因此，对不同的染色体以不同的概率进行交叉，即较优染色体以较小的概率实施交叉操作，放慢其进化速度；而较差染色体则以较大的概率进行交叉，加快其进化速度，便可保持种群的均衡进化和多样性，从而有效地提高抗早熟能力。这种思想可以通过交叉概率矩阵来实现。首先确定一个和标准遗传算法中一样的交叉概率 P_c ，再确定一个交叉概率的最小值 P_{cmin} 和一个最大值 P_{cmax} ，设 M 是待进化种群中染色体的个数， N 是进化代数，则在交叉概率矩阵 $C=[C_{i,j}]_{M \times N}$ 中， $C_{1,1}=P_{cmin}$ ， $C_{M,1}=P_{cmax}$ ， $C_{1,N}=C_{M,N}=P_c$ ， $C_{i,j}$ 的取值使得矩阵 C 中所有的元素在行向及列向上都构成严格的等差数列。每一代中的染色体按适值从高到低排序，则第 j 代种群中适值排在第 i 位的染色体以概率 $C_{i,j}$ 实施交叉操作。

这样，在同一代种群中，较优的染色体的交叉概率始终小于较差染色体的交叉概率。在进化初期，染色体的适值差别较大，较优染色体与较差染色体的交叉概率的差别也较大；随着进化代数的增加，染色体的交叉概率的差别逐步缩小；到进化的后期，所有的染色体的交叉概率逐步收敛于一个固定值 P_c 。

嫁接遗传算法的另一个改进之处在于可以有多个交叉算子对染色体进行交叉操作。不同的交叉算子表示不同的生态环境，故在交叉过程中可以改变搜索范围和搜索过程，有发现新解的可能。交叉算子的选择规则可以采用轮询法、随机法和重复周期法等。一般来说，进化初期宜采用轮询法或随机法来扩大搜索范围，进化后期宜采用重复周期法来保证收敛的效率。这虽然给设计编码方案和交叉算子带来了困难，但它的优越性却是明显的。采用惟一的交叉算子很容易使种群中的染色体同化，进而出现早熟。在多种群遗传算法中，各个种群均采用相同的交叉算子，为了提高抗早熟的能力，不同的种群采用不同的交叉概率，但其值仍然是常数。本文则是采用不同的交叉算子，根据交叉概率矩阵进行变概率交叉，可使算法搜索新解的能力大大加强，不但可以提高解的精度，还可提高抗早熟能力。

3.4.2 嫁接遗传算法在车间作业调度问题中的应用

该算法从执行过程来看分为两个阶段，即嫁接阶段和共生阶段。嫁接阶段的主要功能是实现种群的快速进化；共生阶段的主要功能则是进一步提高算法的求解精度。算法在实现进化速度和提高解的精度时，还要防止陷入局部最优解。

3.4.2.1 编码

由于采用多个交叉算子对染色体操作，所以需选择一种能同时满足两种交叉算子的编码方案。因此，本文采用基于工件和基于机器的两种编码方式。

(1) 基于机器的表达法

染色体编码为机器的序列，根据序列用瓶颈移动启发式构造调度，即把机器一个一个地排列，每次在没有被调度的机器中识别一个瓶颈机器。当调度一个新的机器时，以前的调度需要重新局部优化，瓶颈识别过程和重新局部优化是基于反复求解一个调度问题来进行的。

瓶颈移动启发式基于瓶颈机器优先的思想，机器瓶颈性能的不同度量方法将得到不同的瓶颈机器序列，用瓶颈移动启发式得到的调度性能依赖于瓶颈机器的顺序。

(2) 基于工件的表达法

这种表达法由包含 n 个工件的列表组成，每个调度根据工件的顺序来构造。对于一个给定的工件顺序，列表中第一个工件的所有工序将被首先调度，然后考虑列表中第二个工件的工序。加工中的工件第一道工序要求的加工时间应为相应机器最可能得到的加工时间，然后考虑第二道工序，如此进行，直到工件的所有工序排完，这个过程以列表中每个工件的适当顺序重复进行。工件的任意序列对应一个可行的调度。

3.4.2.2 种群初始化

种群的初始化分为待进化种群的初始化和嫁接种群的初始化。待进化种群的地位等与前面标准遗传算法中的初始种群，其第一代染色体均为随机生成，数目较多。而嫁接种群要满足两个要求：一是初始种群染色体的适应度较高，当它与待进化种群中的染色体进行交叉时，可以提高进化速度；二是内部染色体具有唯一性借以增强抗早熟能力。由于受唯一性和启发式规则数目的限制，嫁接种群的染色体数目较少，一般情况下取待进化种群数目的 20%~30% 即可。针对作业车间调度问题，嫁接种群可根据启发式规则

来生成第一代染色体。启发式规则可以是 SPT(优先选择最短加工时间的工序)、LPT(优先选择最长加工时间的工序)、MWR(优先选择剩余总加工时间最长的工件的工序)、LWR(优先选择剩余总加工时间最小的工件的工序)、MOR(优先选择剩余工序数最多的工件的工序)、LOR(优先选择剩余工序数最小的工件的工序)、FCFS(选择同一机器上工件队列中的第一道工序)及 RANDOM(随机选择工序)等^[37]。若由上述各种一次通过启发式规则得到的解的数目小于嫁接种群所要求的染色体数目,则可利用组合启发式规则产生的解或从待进化种群中选几个较优解来补充。但无论如何,嫁接种群中的染色体一定要保持惟一性。

初始嫁接种群的染色体惟一性可由判断和解决两个步骤来保证。判断是指每产生一个新的染色体就与已有染色体相比较,判断编码是否相同。但为了减小计算量,可先比较其目标函数值是否相同,若不同,则没必要再去比较编码。在目标函数值相同的情况下,则仍需进行编码比较,以判断染色体是否重复。解决是指当存在重复染色体的情况下,采取舍弃当前染色体的措施,继续执行上文的染色体生成方案,直至产生一个满足惟一性的初始嫁接种群。

3.4.2.3 适应度函数和选择策略

染色体的适应度采用逆序法,即将待进化种群中的染色体按其对应的调度的最短完工时间逆序排列,则每个染色体的适应度为该染色体的排列序号。用精选策略作为选择机制。

3.4.2.4 独特的交叉机制

交叉操作是遗传算法的核心,交叉机制的优劣直接决定着算法的效率。嫁接阶段的交叉发生在两个种群间,即分别从两个种群中选择一个染色体作为父代染色体实施交叉操作。这样,嫁接种群中的优秀染色体可以直接把有用信息传递给待进化种群的染色体,实现“以优代劣”的大幅度进化,另外,它还可以有效地防止早熟。由于嫁接种群中的染色体具有惟一性和多样性,使得选择两个相似或相同染色体作为父代进行交叉的概率明显降低,从而在很大程度上防止了由“近亲结合”出现的早熟现象。

由于 Job Shop 调度问题的解空间非常大,如何高效地搜索解空间,寻找较优解,一直是研究的主要目标。本算法采用双交叉算子,不同的交叉算子表示不同的生态环境,

故搜索解时可以探测到解空间的各个位置，大大加强算法搜索新解的能力，并提高解的精度和多样性及抗早熟能力。

在此，选择可行的两种交叉算子分别为：基于工件的交叉和基于机器的交叉。

(1) 基于工件的交叉

基于工件交叉算子的交叉过程为：算法把所有的工件分成两个集合，子代染色体 C_1/C_2 继承 p_1/p_2 中集合 J_1/J_2 内的工件所对应的基因， C_1/C_2 其余的基因位则分别由 p_1/p_2 删除 C_1/C_2 中已确定的基因所剩的基因顺序填充。具体的交叉过程见图 3-1。

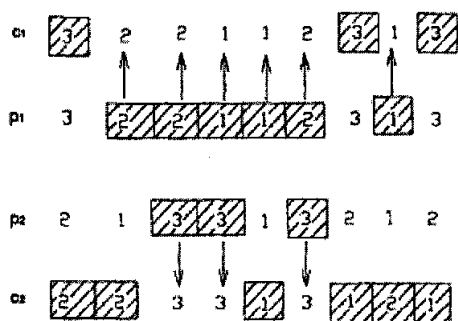


图 3-1 基于工件的交叉 ($J_1=\{1, 2\}$, $J_2=\{3\}$)

(2) 基于机器的交叉

基于机器交叉算子的交叉过程为：算法首先解析出染色体的机器序列，然后把所有的机器随机分成两个集合 M_1 和 M_2 ，子代染色体 C_1/C_2 继承 p_1/p_2 中集合 M_1/M_2 内机器所对应的基因， C_1/C_2 其余的基因位则分别由 p_1/p_2 删除 C_1/C_2 中已确定的基因所剩的基因顺序填充。具体的交叉过程见下图 3-2：

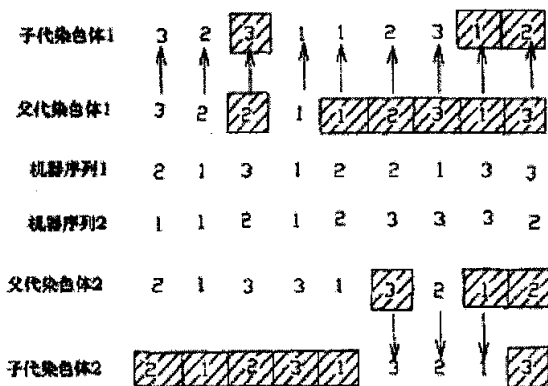


图 3-2 基于机器的交叉 ($M_1=\{1,2\}$, $M_2=\{3\}$)

从交叉过程可以看出,采用上述编码方案和交叉算子生成的染色体均可以得到可行调度。

交叉后产生的子代染色体应同时更新嫁接种群和待进化种群。如该子代染色体优于待进化种群中的最差者,则直接取代;否则,保持待进化种群不变。更新嫁接种群时,则要注意保持染色体的唯一性,如嫁接种群中已存在该子代染色体,则调用变异算子产生新的个体来尝试。当在预定的次数内仍未找到新的染色体时,便取消此次更新嫁接种群操作。

3.4.2.4 变异

变异过程采用邻域搜索变异算子,由于邻域搜索有局部寻优功能,从而弥补了遗传算法全局搜索能力强但局部搜索能力较弱的通病。这种全局搜索和局部搜索相结合的混合式遗传算法虽然增加了计算量,但大大提高了解的质量。

3.4.2.5 共生进化

通过嫁接种群的指导,待进化种群染色体的适应值得到迅速提高。随着进化代数的不断增加,待进化种群染色体的平均适应值会超过嫁接种群染色体的平均适应值,此时的嫁接种群已无法起到指导待进化种群进化方向的作用,算法便进入第二阶段——共生阶段。此时的嫁接种群和待进化种群便转变为两个独立进化、相互合作的共生种群。种群的重新初始化,可以通过把嫁接种群和待进化种群中的所有染色体平均分配给两个共生种群来实现。

两个共生种群的进化是独立的,即一个共生种群以基于工件的交叉算子进行交叉繁殖,另一个共生种群以基于机器的交叉算子进行交叉繁殖。但两个共生种群的进化又是相互促进的,即两个共生种群应该彼此受益于对方,这可通过定期交换两个共生种群中的优秀染色体来实现。

共生进化阶段是一个在嫁接阶段产生优秀个体的基础上继续寻优的过程。当对最优解的精度要求不高时,仅执行嫁接进化即可。共生进化可提高解的精度,不但在于双种群进化,更重要的在于双交叉算子可以扩大搜索领域,找到新解。

4 基于 Flow Shop 调度问题的遗传算法

4.1 引言

流水车间(Flow Shop)调度问题,也被称为同序作业调度问题,是与城市不对称情况下的货郎担问题(TSP)难度相当的同一类型的 NP 完全问题中最困难的问题之一^[39],是一类复杂且极有代表性的流水线生产调度问题的抽象,它无论是在离散制造工业还是在流程工业中都具有广泛的应用,具有一定的代表性。因此对这些企业而言,研究和解决好调度问题,无疑能提高它们的生产效率,从而提高这些企业的竞争力。

尽管相对 Job Shop 调度而言,Flow Shop 调度问题的工艺约束比较简单,但它仍旧是一个非常复杂和困难的组合优化问题,其 NP Hard 特性和强大的工程背景使其一直成为理论界和工程领域研究的热点问题^[40,41]。

4.2 Flow Shop 调度问题的描述

流水车间调度问题一般可以描述为 n 个工件要在 m 台机器上加工,每个工件需要经过 m 道工序,每道工序要求不同的机器。 n 个工件在 m 台机器上加工的顺序相同,工件 i 在机器 m 上的加工时间是给定的,设为 $t_{ij}(i=1,2,\dots,n, j=1,2,\dots,m)$ 。问题的目标是确定 n 个工件在每台机器上的最优加工顺序,使最大流程时间达到最小。

对该问题常常作如下假设:

- (1) 每个工件在机器上的加工顺序是给定的;
- (2) 每台机器同时只能加工一个工件;
- (3) 一个工件不能同时在不同机器上加工;
- (4) 工序不能预定;
- (5) 工序的准备时间与顺序无关,且包含在加工时间中;
- (6) 工件在每台机器上的加工顺序相同,且是确定的。

令 $c(j_i, k)$ 表示工件 j_i 在机器 k 上的加工完成时间, $\{j_1, j_2, \dots, j_n\}$ 表示工件的调度,那么,对于无限中间存储方式, n 个工件、 m 台机器的流水车间调度问题的完工时间可表示为:

$$c(j_1, 1) = t_{j_1 1} \quad (4-1)$$

$$c(j_1, k) = c(j_1, k-1) + t_{j_1 k}, \quad k=2, \dots, m \quad (4-2)$$

$$c(j_i, 1) = c(j_{i-1}, 1) + t_{j_i 1}, \quad i=2, \dots, n \quad (4-3)$$

$$c(j_i, k) = \max\{c(j_{i-1}, k), c(j_i, k-1)\} + t_{j_i k}, \quad i=2, \dots, n, \quad k=2, \dots, m \quad (4-4)$$

最大流程时间为：

$$c_{max} = c(j_n, m) \quad (4-5)$$

调度目标就是确定 j_1, j_2, \dots, j_n , 使得 c_{max} 最小。

4.3 Flow Shop 调度问题的启发式算法

研究表明, 3 台以上机器的 Flow Shop 调度即为 NP 问题, 至今没有一个多项式复杂性的全局优化算法。因此, 为了快速得到问题的优化解, 许多工程领域和调度研究人员设计了若干启发式方法。

启发式算法(Heuristic algorithm)是相对于最优算法提出的, 是在可接受费用内寻找最好的解, 但不一定能保证所得解的可行性和最优性, 甚至在多数情况下, 无法阐述所得解同最优解的近似程度。事实上, 在实际问题中, 最优算法因问题的难度使其计算时间随问题规模的增加以指数速度增加, 这时只能通过启发式算法求得问题的一个可行解。启发式算法可定义: 一个基于直观或者经验构造的算法, 在可接受的花费(指计算时间、占用空间等)下, 给出待解决组合优化问题每一个实例的一个可行解, 该可行解与最优解的偏离程度不一定事先可以预料。下面简要介绍几种启发式算法^[42]。

4.3.1 Johnson 启发式算法

对于以最大流程时间为目标的两台机器流水车间调度问题称为 Johnson 问题, 其最优调度由著名的 Johnson 规则确定。

假设有 n 个工件, 在第一台机器和第二台机器上的加工时间分别为 $t_{i1}, t_{i2}, i=1, 2, \dots, n$, 其最优调度由下列 Johnson 规则确定。

Johnson 规则: 如果 $\min(t_{i1}, t_{i2}) \leq \min(t_{j1}, t_{j2})$, 则将工件 i 排在工件 j 之前。

可以直接利用这个规则构造最优调度, 具体步骤为:

将 P 组工件按它们在第一台机器上加工的时间递增顺序排列, 将 Q 组工件按它们在第二台机器上加工时间递减顺序排列。

将 P 组工件顺序和 Q 组工件顺序连接在一起, 构成的就是生产周期最短的最优产品顺序。

4.3.2 Nawaz-Enscore-Ham (NEH) 算法

Nawaz, Ensore 和 Ham 描述了 Flow Shop 调度问题的两个较好的启发式, 提出了一个插入算法, 他们假定在所有机器上的总加工时间越长的工件比总加工时间越小的工件应该得到越大的优先数。他们不是把原来的 m 台机器问题转化为一个模拟的两台机器问题, 而是通过每一步加入一个新工件, 从而求得最好的局部解, 最后构造工件的加工顺序。

其启发式算法为:

步骤 1: 按工件在机器上的总加工时间递减的顺序排列 n 个工件;

步骤 2: 取前两个工件调度使部分最大流程时间达到极小;

步骤 3: 从 $k=3$ 到 n 把第 k 个工件插入到 k 个可能的位置, 求得最小部分的最大流程时间。

4.3.3 Gupta 启发式方法

该方法首先对每个工件计算参量:

$$S(i) = \frac{C}{\min_{1 \leq j \leq m-1} (t_{ij} + t_{i(j-1)})} \quad (4-6)$$

其中, 若 $t_{im} \leq t_{i1}$, 则 $C=1$, 否则 $C=-1$ 。

然后, 将工件按参量 $S(i)$ 值递增的顺序进行排列, 从而得到一个优化调度。

4.3.4 Rajendran 方法

该方法对 NEH 的工件排序规则作了修改, 它对各工件计算参量:

$$T_n = \sum_{j=1}^m (m-j+1)t_j \quad (4-7)$$

并按递增顺序进行排列。首先选取队列中的第 1 个工件生成的子调度, 然后选取队列中的第 k 个工件尝试插入到已生成子调度的第 l 个位置, 其中 $[k/2] \leq l \leq k$, $k \geq 2$, 进而令其中使得调度指标最小的方案为新的子调度。如此依次将剩余所有工件进行插入, 从

而得到一个优化调度。

这些启发式方法的优点是具有构造调度解的快速性，但调度质量还不是很理想，其中属 NEH 和 Rajendran 方法的性能最好。基于此，优化质量较高的改进型搜索方法得到了重视，如遗传算法。同时，以上启发式算法可以与遗传算法相结合，改进遗传算法的性能。

4.4 Flow Shop 调度问题的基本遗传算法

Flow Shop 调度问题是 Job Shop 调度问题的一个特例，因此许多求解 Job Shop 的遗传操作和遗传算法框架都可以直接或间接地用于求解 Flow Shop 问题。同时，鉴于本质上置换 Flow Shop 问题等同于旅行商问题，因此许多求解 TSP 问题的遗传操作也都可直接或间接用于求解置换 Flow Shop 问题。例如 PMX, OX, LOX 等交叉操作，SWAP, INV, INS 等变异操作。因此，这里不再重复遗传算法的操作和框架的设计。

4.4.1 编码设计

在 Flow Shop 调度问题中，最自然的编码方式是用染色体表示工件的加工顺序，例如，对于有五个工件的 Flow Shop 调度问题，第 k 个染色体为 $V_k[1,2,3,4,5]$ ，表示五个工件的加工顺序为： j_1, j_2, j_3, j_4, j_5 ^[43]。

4.4.2 初始种群的产生

遗传算法中初始群体中的个体可以是随机产生的。但是如果采用一些改进算法产生比较好的个体，可以得到更好的结果。因此可采用如下策略设定初始群体：

(1) 根据问题固有知识，设法把握最优解所占空间在整个问题空间中的分布范围，然后，在此分布范围内设定初始群体。

(2) 先随机产生一定数量的个体，然后从中挑选最好的个体加入到初始群体中，这个过程不断循环，直到初始群体中的个体数量达到预先确定的规模。

4.4.3 适应度函数设计

在具体应用中，适应度函数的设计要结合求解问题本身的要求而定，在本例中将目标函数的倒数作为适应度函数。

n 个工件 m 台机器的 Flow Shop 调度问题的染色体 v_k 对应的最大流程时间为：

$$C_{\max} = c(j_n, m) \quad (4-8)$$

则适应度函数取为:

$$1/c_{\max} \quad (4-9)$$

4.4.4 遗传算子

4.4.4.1 选择算子

采用比例选择(轮盘赌)操作,用正比于个体适应值的概率来选择相应的个体,是利用比例于各个个体适应值的概率决定其子代的遗留可能性。

对于某个个体 i , 设其适应度为 f_i , 则其被选取的概率表示为:

$$P_i = f_i / \sum_{i=1}^M f_i \quad (4-10)$$

4.4.4.2 交叉算子

交叉方法是模仿自然生态系统的双亲繁殖机理而获得新个体的方法,它可使父代不同的个体进行部分基因交换组合产生新的优良个体。交叉概率 P_c 较大时可以增强算法开辟搜索区域的能力,但会增加优良子代被破坏的可能性。交叉概率较低时又会使搜索陷入迟钝状态,研究表明, P_c 应取为 0.25 到 1.00 之间。

为了使后代能自动满足优化问题的约束条件。改进方法采用部分匹配交叉(PMX)与顺序交叉(OX)和循环交叉(CX)混合使用的方法来保留双亲染色体中不同方面的特征。达到获得质量较高后代的目的。例如对一部分用 PMX 交叉。而另一部分用 OX 进行交叉;或者先采用 PMX 进行交叉,得到适应度高的后代后再采用 OX 进行交叉。

4.4.4.3 变异操作

这里主要使用互换变异(SWAP)与逆转变异(TNV)相结合的方法。

综上所述,解决 Flow Shop 调度问题的一种标准遗传算法的步骤为:

步骤 1: 编码方式是采用基于工件的编码方法,用染色体表示工件的顺序,如染色体 $V_k[1,2,3,4,5]$,表示五个工件的加工顺序为: j_1, j_2, j_3, j_4, j_5 ;

步骤 2: 遵守工件的工艺约束条件,随机地产生 N 个(N 是偶数)候选解,组成初始种群;

步骤 3: 定义适应度函数为 $f_i=1/C_{max}$, 计算适应度 f_i 。按照轮盘赌规则选择 N 个个体;

步骤 4: 将群体中的各个个体随机搭配成对, 共组成 $N/2$ 对, 以概率 P_c 选择一对个体, 采用了 PMX 交叉算子对该染色体进行交叉操作;

步骤 5: 以概率 P_m 随机选择染色体中的子串, 采用“逆转”方式的变异操作;

步骤 6: 计算出交叉和变异后生成新个体的适应度值, 新个体与父代一起构成新群体;

步骤 7: 判断是否达到预定的迭代次数, 是则结束寻优过程; 否则转向步骤 4。

4.5 Flow Shop 调度问题的改进遗传算法求解

最近十年来, 遗传算法和与其相关的一些混合算法已经广泛地应用于生产调度领域^[44]。因此本节利用数论中的佳点集理论和方法, 设计了佳点集遗传算法^[45], 用来求解 Flow Shop 调度问题, 在此改进的遗传算法中利用数论中关于佳点的优良性质来设计交叉算子, 此交叉算子能使子代保留“父亲”和“母亲”的共同基因段, 且不确定的基因由佳点确定, 这即体现了“遗传”的思想, 又体现了“进化”的思想, 以期得到满意的结果^[56]。

4.5.1 佳点集的基本定义和性质

(1) 设 G_S 是 S 维欧氏空间的单位立方体。即

$$x \in G_S, x=(x_1, x_2, \dots, x_S)$$

其中: $0 \leq x_i \leq 1, i=1, 2, \dots, S$

(2) 设 G_S 中有一点集(n 个点)

$$P_n(k) = \{(x_1^{(n)}(k), x_2^{(n)}(k), \dots, x_S^{(n)}(k)), 1 \leq k \leq n\}$$

其中: $0 \leq x_i^{(n)}(k) \leq 1, 1 \leq i \leq S$

(3) 对任一给定 G_S 中的点 $r=(r_1, r_2, \dots, r_S)$, 令 $N_n(r) = N_n(r_1, r_2, \dots, r_S)$ 表示 $P_n(k)$ 中满足下列不等式组的点的个数:

$$0 \leq x_i^{(n)}(k) \leq r_i \quad i=1, 2, \dots, S$$

$$\varphi(n) = \sup_{r \in G_S} \left| \frac{N_n(r)}{N} - |r| \right|$$

其中 $|r| = r_1 r_2 \cdots r_s$, 则称点集 $P_n(k)$ 有偏差 $\varphi(n)$ 。若对任一 n 均有 $\varphi(n) = o(1)$, 则称 $P_n(k)$ 在 G_S 上是一致分布且偏差为 $\varphi(n)$ 。

(4) 令 $r \in G_S$, 形如 $P_n(k) = \{r_1 \times k, r_2 \times k, \dots, r_s \times k\}$, $k = 1, 2, \dots, n\}$ 的偏差 $\varphi(n)$ 满足 $\varphi(n) = C(r, \varepsilon) n^{-1+\varepsilon}$, 其中 $C(r, \varepsilon)$ 是只与 r, ε (ε 为任意小的正数) 有关的常数, 则称 $P_n(k)$ 为佳点集, r 称为佳点 (good point)。

(5) 取 $r_k = 2 \cos(2\pi k / p)$, $1 \leq k \leq s$, p 是满足 $(p-3)/2 \geq s$ 的最小素数, 则 r 是佳点 (分圆域)。

(6) 定理 1: 令 $P_n(k)$ ($1 \leq k \leq n$) 具有偏差 $\varphi(n)$, $f \in B_S$ (S 维围变函数类), 则

$$\left| \int_{x \in G_S} f(x) dx - \frac{1}{n} \sum_{k=1}^n f(P_n(k)) \right| \leq V(f) \varphi(n) \quad (4-11)$$

其中, $V(f)$ 为 f 的全变差。

(7) 定理 2: 若上式对所有的 $f \in B_S$ 都成立, 则 $P_n(k)$ ($1 \leq k \leq n$) 为有偏差不超过 $\varphi(n)$ 的点集 (该结论可以看成是定理 1 的逆定理)。

(8) 定理 3: 若 $f(x)$ 满足:

$$|f| \leq L, \quad \left| \frac{\partial f}{\partial x_i} \right| \leq L, \quad 1 \leq i \leq s, \quad \left| \frac{\partial^2 f}{\partial x_i \partial x_j} \right| \leq L, \quad 1 \leq i \leq j \leq s, \quad \dots, \quad \left| \frac{\partial^2 f}{\partial x_j \partial x_s} \right| \leq L, \quad \text{用给}$$

定 n 个点的函数值构成的任何加权, 来近似计算函数在 G_S 上的积分, 误差都不能希望比 $O(n^{-1})$ 更小。

需要说明的是:

(1) 这个性质正是上面讨论的点集称为“佳点集”的由来。

(2) 由定理 1、2、3 知, 用佳点集来进行近似积分, 其误差的阶只与 n 有关, 而与空间维数 S 无关, 这为高维的近似计算提供了一个非常优越的算法。

(3) 由以上性质知, 若限定取 n 个点, 用其上函数值的线性组合来近似对应的积分值, 则佳点集方法提供了一个最好的取法。

4.5.2 基于佳点集的交叉算子

通常遗传算法的交叉操作是随机选取两个父代染色体进行交叉。由上节分析可见,对搜索空间了解不多的情况下,为所考虑的空间的佳点提供了最好的选择。下面将利用佳点集的有关信息来构造该问题的交叉算子。

设用以交叉的两个父代染色体为 A_1 和 A_2 (染色体的长度为 l) 不妨设 A_1 和 A_2 的前 t 位不同, 后 $l-t$ 位相同, 取 t 维立方体 G_t 。在 G_t 中取佳点集中一点 $p=(p_1, p_2, \dots, p_t)$, 其中 $p_k=\langle e^k \rangle$, $1 \leq k \leq t$, $\langle a \rangle$ 表示取 a 的小数部分。将 A_1 和 A_2 的对应位置相同的基因不变并从中删除, 这样 A_1 和 A_2 中余下的 t 个基因为 s_1, s_2, \dots, s_t 且 $s_1 \leq s_2 \leq \dots \leq s_t$ 。将 p_1, p_2, \dots, p_t 由小到大排列为 p'_1, p'_2, \dots, p'_t , 则 p'_i 对应 s_i 得到 s_1, s_2, \dots, s_t 的一个排列。将这个排列与 A_1 和 A_2 相同的部分合在一起, 就得到一个子代染色体。

例如, $A_1=(1,3,2,5,4)$, $A_2=(1,4,3,5,2)$, 相同部分为 $(1,*,*,5,*)$, 余下部分: $s=(2,3,4)$ 。取 G_3 , $p=(0.7183, 0.3891, 0.0855)$, 则 $p'=(0.0855, 0.3891, 0.7183)$, 从而得到 s 的一个排列: $(4,3,2)$, 与 A_1 和 A_2 相同的部分合在一起, 就得到一个子代染色体为 $(1,4,3,5,2)$ 。

采用上述交叉算子的遗传算法称为佳点集遗传算法。

4.5.3 佳点集遗传算法设计

4.5.3.1 编码

对于求解有序组合问题的 Flow Shop 调度问题, 其编码采用自然顺序编码形式, 即基于所有工件的一个排列为其染色体编码, 这种表示方法为许多文献所采用。如 $n=6$, 则排列 254613 可以是其一个个体的编码。

4.5.3.2 适应度函数与目标函数

根据第 1 节讨论, 这里目标函数取为 $makespan = \min_{\sigma} f(\sigma)$, σ 为工件的任意一个排列, 所以 σ 也为问题的一个可行解, σ 的适应度函数取 $1/f(\sigma)$ 。

4.5.3.3 初始种群的产生

采用随机的方式产生一定数量的个体作为初始种群, 为了保证种群的多样性, 产生的每一个个体的首位基因代码不同^[46]。

4.5.3.4 选择算子

即实现种群变换操作, 采用典型的轮盘赌选择。

4.5.3.5 交叉算子

采用上面介绍的基于佳点集的交叉算子。

4.5.3.6 变异算子

变异保证了全局的最优性。在本算法中, 变异的方法是: 对于染色体 A , 选取随机数 r_i , 将 r_i+1 位直至最后位上的基因依次向前移 1 位, 而将原 r_i 位对应的基因置于 A 的最后, 如果 $r_i=4$, 那么 $A: 254613$ 变异为 $A: 254136$ 。

4.5.3.7 种群变换

算法中将父代中适应度最大的个体保留到下一代, 然后从父代中经过交叉和变异所得到的 pupsiz (群体中个体数)个个体中取出 $\text{popsiz}-1$ 个个体放到下一代, 从而形成新一代种群。

4.5.3.8 运算结束

取适值最大个体作为问题的最优解, 由此可得到问题的目标值。

4.6 小结

本章首先详细介绍了生产调度问题中的 Flow Shop 调度问题; 介绍了求解 Flow Shop 调度问题的启发式算法; 介绍了求解 Flow Shop 调度问题的遗传算法设计, 详细介绍了遗传算法的编码/解码、交叉算子、变异算子、种群初始化、适应度计算及定标, 并给出了一种改进的遗传算法——佳点集遗传算法。

5 数值仿真实验与结果分析

由于遗传算法理论上的分析比较困难,我们采用实验来说明本文提出的相关遗传算法的性能。

5.1 Job Shop 调度问题的仿真

5.1.1 Job Shop 调度问题的标准遗传算法仿真

下面对一个 10 台机器、10 个工件的 Job Shop 调度问题采用标准遗传算法进行了仿真。该例中,每个工件分别在 10 台机器上各加工一次,每个工件的加工路线与每道工序的加工时间如表 5-1 所示。

表 5-1 10 台机器、10 个工件的调度问题

表 5-1 10 台机器、10 个工件的调度问题

工件	工序与时间									
	1	2	3	4	5	6	7	8	9	10
1	M ₃	M ₁	M ₂	M ₄	M ₆	M ₅	M ₇	M ₈	M ₉	M ₁₀
	5	20	25	35	15	35	30	35	35	20
2	M ₂	M ₃	M ₅	M ₆	M ₇	M ₈	M ₁	M ₄	M ₁₀	M ₉
	40	30	50	40	45	20	30	15	30	35
3	M ₃	M ₄	M ₇	M ₈	M ₂	M ₅	M ₆	M ₁₀	M ₉	M ₁
	20	20	40	45	5	40	15	35	50	10
4	M ₇	M ₈	M ₂	M ₁	M ₃	M ₄	M ₁₀	M ₉	M ₅	M ₆
	30	30	25	25	40	50	35	15	35	50
5	M ₃	M ₇	M ₈	M ₂	M ₅	M ₁₀	M ₉	M ₆	M ₁	M ₄
	40	15	30	20	5	5	35	35	35	30
6	M ₂	M ₄	M ₇	M ₉	M ₁₀	M ₆	M ₈	M ₁	M ₅	M ₃
	10	10	40	50	20	10	35	40	35	30
7	M ₅	M ₆	M ₃	M ₁₀	M ₉	M ₄	M ₇	M ₈	M ₂	M ₁
	35	30	35	30	10	45	45	15	15	45
8	M ₄	M ₉	M ₁₀	M ₆	M ₃	M ₅	M ₁	M ₂	M ₈	M ₇
	20	30	50	20	20	35	30	50	20	30
9	M ₅	M ₁₀	M ₉	M ₆	M ₄	M ₁	M ₂	M ₃	M ₇	M ₈
	10	35	20	30	30	30	45	30	45	50
10	M ₆	M ₂	M ₁	M ₅	M ₈	M ₃	M ₄	M ₇	M ₁₀	M ₉
	40	10	30	35	40	15	35	20	40	20

华中科技大学硕士学位论文

遗传算法的参数设置为：种群规模为 100，最大迭代次数 $N=1000$ ， $P_c=0.3$ ， $P_m=0.1$ 。

该实例的任务在文献^[47]中用神经网络方法调度后，最长加工路径为 465；用本文第 3 章中设计的标准遗传算法调度后，最长加工路径为 560；用本文第 3 章中设计的嫁接共生遗传算法调度后，最长加工路径为 450。

表 5-2 给出的是对于 8 个不同的 10 台机器、10 个工件的问题用标准遗传算法在 1000 代以上取得的最好调度与传统调度方法所得调度的完成时间对比。

表 5-2 几种方法对 10 台机器、10 个工件的作业调度问题的完成时间比较

	GA	RANDOM	MOPNR	MWKR-P	MWKP/P	SPT	MWKR	LWKR
1	64	66	69	83	88	70	67	86
2	71	76	80	84	73	79	83	89
3	64	66	74	74	70	74	75	91
4	73	77	83	87	86	81	84	99
5	83	86	97	98	114	113	105	114
6	83	83	85	96	96	98	84	100
7	89	97	99	114	123	120	105	134
8	87	90	102	116	116	119	97	102

从这些比较可以看出，遗传算法所得结果令人满意。对于 10 台机器、10 个工件的调度问题，遗传算法所得的结果总是最好的。此时相对于所有其他调度方法的平均水平，标准遗传算法总体上将解改进了 17%。同时，可以看出，传统方法的表现随着测试问题的不同有很大的起伏，而遗传算法则在所有测试问题上的表现均相对稳定。

5.1.2 Job Shop 调度问题的嫁接共生遗传算法仿真

为了考察嫁接遗传算法在解决 Job Shop 调度问题时的效率，我们用 MT06, MT10, MT20, LA06, LA16 和 LA21 六个实例对嫁接遗传算法 (GGA)、模拟退火遗传算法 (SAGA) 和混合遗传算法 (HGA) 分别进行了 30 次测试^[48]。在每次测试时为了具有尽可能大的可比较性，我们尽量采用相同的参数。

本算法中的参数取值为：待进化种群规模均为 100，嫁接种群规模为 25， $P_c=0.6$ ， $P_{cmin}=0.2$ ， $P_{cmax}=0.9$ ，邻域搜索变异概率为 1，终止进化代数为 3000。计算结果如表 5-3 所示。

从表 5-3 可以看出，GGA 与 SAGA 在解的精度方面相差不大，但对于收敛到最优

解的问题，GGA 的进化代数明显小于 SAGA。HGA 的进化代数虽然比较小，但其解的精度远远不及 GGA 和 SAGA。SAGA 的解的精度较高的原因在于采用了模拟退火机制，但其进化速度较慢，HGA 由于早熟现象比较严重，故解的精度不高。而 GGA 则在解的精度和进化速度两个方面均取得了较为理想的结果。

表 5-3 GGA、SAGA 和 HGA 的性能比较

问题	m, n	C^{**}	GGA		SAGA		HGA	
			C^*	g_a	C^*	g_a	C^*	g_a
MT06	6, 6	55	55	205	55	363	55	189
LA06	15, 5	926	926	581	926	783	926	552
MT10	10, 10	930	930	2035	930	2877	1014	1533
MT20	20, 5	1165	1205	3000	1197	3000	1482	3000
LA16	10, 10	945	945	828	945	1809	1241	1438
LA21	15, 10	1046	1046	3000	1143	3000	1258	3000

注： m 为工件数； n 为机器数； C^{**} 为目前为止已发现的最优值； C^* 为对应的算法所发现的最优值； g_a 为算法的平均收敛代数；若值为 3000，则表示终止条件为进化到 3000 代。

GGA 由于采用嫁接种群来指导待进化种群，故进化速度较快；而其独特的交叉机理（采用多交叉算子的种群间交叉和基于交叉概率矩阵的变概率交叉）则提高了抗早熟的能力，使解的精度增高。

5.2 Flow Shop 调度问题的仿真

5.2.1 Flow Shop 调度问题的标准遗传算法仿真

以某零部件加工车间为例。设计一个 5 个工件、4 台机器的生产加工调度问题。工件的加工时间如下所示：

$$T(J_i, j) = \begin{bmatrix} 31 & 41 & 25 & 30 \\ 19 & 55 & 3 & 34 \\ 23 & 42 & 27 & 6 \\ 13 & 22 & 14 & 13 \\ 33 & 5 & 57 & 19 \end{bmatrix}$$

其中： J_i 表示第 i 个工件； j 表示第 j 道工序。

为验证算法的有效性，设标准遗传算法的参数为交叉概率 $P_c=0.6$ ，变异概率 $P_m=0.2$ ，种群规模为 20，迭代次数 $N=50$ 。运算结果如表 5-4 所示。

表 5-4 遗传算法运行结果

总运行次数	最好值	最坏值	平均	达到最好解的频率	达到最好解的平均代数
20	213	221	213.95	0.85	12

如果用启发式算法对这个问题进行求解，NEH 方法得到 4-2-5-1-3 解时，需要 213，Palme 方法得到 5-2-4-1-3 解时需要 245，都要比本文第 4 章设计的遗传算法的运行时间要长。因此，仿真结果表明，标准遗传算法求解 Flow Shop 调度问题比用启发式算法得到更满意的调度结果，遗传算法得到优质解的概率较高达到 85%以上，而且求解的收敛速度快，更符合工程上的应用。

5.2.2 Flow Shop 调度问题的佳点集遗传算法仿真

遗传算法中取交叉概率为 0.8，变异概率为 0.2。考虑了算法性能指标：BRE(最好相对误差)，ARE(平均相对误差)，WRE(最坏相对误差)。分别说明如下：

$$ARE = (C_{average} - C^*) / C^* * 100\% \quad (5-1)$$

$$BRE = (C_{best} - C^*) / C^* * 100\% \quad (5-2)$$

$$WRE = (C_{worst} - C^*) / C^* * 100\% \quad (5-3)$$

其中， C^* 为问题的最优解， $C_{average}$ 为算法求得平均解， C_{best} 为算法求得的最优解， C_{worst} 为算法求得的最差解。佳点集遗传算法的有关结果与 Hybrid heuristic 和 Best SGA 的比较结果参考表 5-5，Hybrid heuristic 为文献^[49]中提出的算法。

由表 5-5 中关于 GGA 的算法结果的比较可知，佳点集遗传算法很好地改善了标准遗传算法求解问题的求解质量，提高了通常遗传算法的求解性能，文^[49]讨论的算法 Hybrid heuristic 体现了算法之间的结合运用，显示了求解问题的混合优化策略思想，这也是改善算法性能、提高算法求解质量的重要方法；这里讨论的佳点集遗传算法在求解 Flow Shop 调度问题时显示了相当的优越性。

表 5-5 GGA 与 Hybrid heuristic 和 Best SGA 的结果比较

问题	佳点集遗传算法			Hybrid heuristic			Best SGA
	BRE	ARE	WRE	BRE	ARE	WRE	BRE
Car1	0	0	0	0	0	0	0
Car2	0	0	0	0	0	0	0
Car3	0	0	0	0	0	0	1.19
Car4	0	0	0	0	0	0	0
Rec01	0	0.23	0.36	0	0.14	0.16	2.0
Rec03	0	0.07	0.21	0	0.09	0.18	0.18
Rec05	0	0.38	1.27	0	0.29	1.13	1.13
Rec07	0	1.31	2.3	0	0.69	1.15	1.15
Rec09	0	1.12	1.86	0	0.64	2.41	2.54
Rec13	0.82	2.51	3.47	0.36	1.68	3.06	2.80
Rec15	0.87	3.23	3.89	0.56	1.12	2.00	1.64
Rec19	1.52	3.93	4.21	0.62	1.32	2.25	3.87
Rec25	1.83	4.31	4.74	1.27	2.54	3.98	4.10
Rec31	2.21	4.85	5.31	0.43	1.34	2.50	6.01
Rec37	4.05	5.62	7.93	3.75	4.90	6.18	7.89

6 总结

大多数的调度问题属于 NP 难题，因此，寻找具有多项式复杂性的最优算法几乎是不可能的，但至今仍激发着学者不断的探索。遗传算法作为一类优化方法，由于其在解决组合优化问题方面的优越性而受到各个领域专家、学者的关注，其应用领域已涉及自适应控制、组合优化、模式识别、机器学习、信息处理和人工智能等等。

本文系统研究了遗传算法的基本原理，分析了遗传算法在生产调度中的应用，主要有以下几个方面的工作：

(1) 针对生产作业调度广泛存在于实际的工业生产中，对国内外有关调度问题的研究现状进行了综述，总结了生产作业车间调度问题的已有研究方法，并指出了利用遗传算法解决生产调度问题的优势。

(2) 介绍了遗传算法的基本理论基础、研究的焦点问题以及遗传算法的实现流程、基本参数和操作的设计，并介绍了几种改进的遗传算法。

(3) Job Shop 调度问题是经典强 NP 问题，本文针对 Job Shop 调度问题的特殊性，总结了遗传算法在作业车间调度中的应用方法，给出了应用于生产作业调度的遗传算法的几种常用编码方法的实现技术，研究了常用的交叉方法，讨论了求解 Job Shop 调度问题的标准遗传算法的基本流程，最后设计了一种求解 Job Shop 调度问题的改进遗传算法——嫁接共生遗传算法。

(4) Flow Shop 调度问题是一类复杂且极有代表性的流水车间调度问题，本文设计了求解 Flow Shop 调度问题的标准遗传算法，并针对标准遗传算法容易早熟的缺点，设计了一种混合遗传算法——佳点集遗传算法，有效地提高了调度效率。

(5) 对本文提出的遗传算法相应地进行了仿真。结果表明，本文提出的遗传算法能够有效地提高生产调度的效率，较好地改进了生产调度解的质量。

虽然调度问题是一个非常古老的课题，而且已经取得了很多的成果，但是这些成果与实际的调度问题之间都存在一定的差距，尚不能很好地解决复杂的实际生产调度问题。智能调度是解决复杂的实际调度问题的一个有效手段，是一个较新的研究课题，其理论和方法需要进一步丰富、发展和完善。

(1) 遗传算法中参数值的优化问题。大量的研究已经发现，遗传算法的性能与它的

参数值的选取有很大关系，这些参数与算法的收敛性以及收敛速度之间到底有什么关系，如何统一来考虑这些参数，以便保证算法的全局收敛性，这些问题都有待于进一步研究。

(2) 生产调度问题的新方法研究。从生物进化或自然界的各种现象中获得新的启发，对现有的遗传算法进行改进，或提出新的方法，以便更加有效地求解车间作业调度问题。

致 谢

在论文完成之际，首先感谢导师杨坤涛教授。论文从开题到完成，都离不开杨老师的悉心指导和严格要求。“学高为师，身正为范”的职业精神是我学习的楷模；杨老师渊博的知识、严谨的学风、忘我的工作热情给予我以后工作的影响是我一生所取之不尽，用之不竭的力量源泉。在此，以“桃李不言，下自成溪”这句古语来祝愿您一切顺利。

其次，感谢所有关心过我学习成长的老师。

另外，向养育我成长的父母，时刻关心我的妻子邵晓红及可爱的女儿肖瑶表示谢意，是你们让我在充满爱的天空中自由的飞翔，一步步坚强而自信地实现我人生的目标。

参考文献

- [1] 陈荣秋. 排序理论与方法. 华中理工大学出版社, 1987. 23~32
- [2] 赵传立, 唐恒永. 排序引论. 北京: 科学出版社, 2002. 45~48
- [3] E.L.Lawler. Optimal sequencing of a single machine subject to precedence constraints, Management Sci., 1973, 19: 544~546
- [4] C.L.Monma. Sequencing to minimize the Maximum job cost. Oper Res, 1980 28: 942~951
- [5] E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy Kan, et al. Sequencing and scheduling: Algorithms and Complexity. Operations Research and Management Science, 1986, 4
- [6] Johnson. Optimal two-and-three stage production schedules with setup times included, Naval Research Notes Quarterly, 1954, 1: 61~68
- [7] 韦有双, 杨湘龙, 冯允成. 一种新的求解 Flow shop 问题的启发式算法. 系统工程理论与实践, 2000, 20(9): 41~47
- [8] 王凌, 郑大钟. 模拟退火算法求解 Flow shop 问题的研究. CDC, 1997: 390~394
- [9] 刘民, 吴澄, 蒋新松. 用遗传算法解决并行多机调度问题. 系统工程理论与实践, 1998, 18(1): 14~18
- [10] 刘民, 吴澄. 解决并行多机提前/拖后调度问题的混合遗传算法方法. 自动化学报, 2000, 26(2): 258~262
- [11] 黄德才, 朱艺华, 王万良. 带公共交货期窗口的提前/拖期非等同多机调度问题. 系统工程理论与实践, 2001, 4: 64~69
- [12] Elpidia S T. Agentifying the Process: Task-Based or Robot-Based Decomposition. IEEE Conference on System, Manufacturing and Synbenetics, 1994: 582~587
- [13] Garey, M. R., Johnson, D.S. and Sethi. The complexity of flowshop and jobshop scheduling. Mathematics of operations research, 1976: 117~129
- [14] Luh P B, Hoitomp D J. Scheduling Generation and Reconfiguration for Parallel Machines. IEEE Trans on Robotics and Automation, 1990, 6(6): 687~696
- [15] Hoitomp D J, Luh P B. A Practical Approach for the Job Shop Scheduling Problems. IEEE Trans on Robotics and Automation, 1993, 9(1): 1~13
- [16] Luh P, Hoitomp D. Scheduling of Manufacturing Systems Using the Lagrangian

- Relaxation Technique. IEEE Trans on Automatic Control, 1993, 38: 1066~1079
- [17] Ham I, Hitomi K, Yoshida T. Group Technology-Applications to Production Management. Kluwer-Nijhoff Publishing, 1985, 123~131
- [18] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by Simulated Annealing. Science, 1983, 220~221
- [19] Glover F. Future Paths for Integer Programming and Links to Artificial Intelligence. Comput & Ops Res, 1986, 13(5): 533~549
- [20] Holland J H. Adaptation in Natural and Artificial Systems. Michigan University Press, 1975, 125~128
- [21] 李敏强, 寇纪淞, 林丹等. 遗传算法的基本理论及其应用. 北京: 科学出版社, 2002, 80~86
- [22] 王小平, 曹立明. 遗传算法——理论、应用及软件实现. 西安: 西安交通大学出版社, 2002, 26~30
- [23] 周明, 孙树栋. 遗传算法原理及应用. 北京: 国防工业出版社, 1999, 89~92
- [24] Imed Kacem, Slim Hammadi, Pieer Borne. Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems. IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, 2002, 32(1), 1~13
- [25] GuoYong Shi. A genetic algorithm applied to a classic job shop scheduling problem. International Journal of Systems Science, 1997, 28 (1): 25~32
- [26] 王万良, 姚明海, 吴云高等. 基于遗传算法的混合 Flow shop 调度方法. 系统仿真学报, 2002, 14(7): 863~869
- [27] De Jong K A. An analysis of the behavior of a class of genetic adaptive systems. Ann Arbor: Univ. Microfilms, 1975, 158~166
- [28] Whitley D. Modeling hybrid genetic algorithms. Winter Genetic Algorithms in Engineering and Computer Science Wiley, 1995, 246~249
- [29] Eshelman L J. The CHC adaptive search algorithms. How to have safe search when engaging in nontraditional genetic recombination. Foundation of Genetic Algorithms. Morgan Kaufmann Publishers, 1998: 265~283
- [30] Goldberg D E, Korb B. Messy genetic algorithms: Motivation, analysis in first results.
-

- Complex Systems,1989,3:493~530
- [31] Strnivas M,Patnaik L M. Adaptive probabilities of crossover and mutation in Gas.IEEE Trans.On SMC,1994,24(4):656~667
- [32] 徐金悟,刘纪文.基于小生境技术的遗传算法.模式识别与人工智能,1999,12(1):104~107
- [33] Chen Y W. A parallel genetic algorithm based on the island model for image restoration.Proceedings of the 1996 IEEE Signal Proceedings Society Workshop, 1996. 130~139
- [34] 郭绚,石晓红.并行遗传算法性能分析.航空计算技术,1998,28(3):86~89
- [35] 乔建忠,雷为民等.混合遗传算法研究及其应用.小型微型计算机系统,1998,19(12):14~19
- [36] 王凌.车间调度及其遗传算法.北京:清华大学出版社,2003.56~78
- [37] 玄光男,程润伟.遗传算法与工程优化.北京:清华大学出版社,2004.127~152
- [38] Davis L. Handbook of genetic algorithm. New York Van Nostrand Reinhold, 1991. 78~85
- [39] 房育栋,郝建忠,余英林等.遗传算法及其在 TSP 中的应用.华南理工大学学报(自然科学版),1994,22(3):124~127
- [40] Carlier.Scheduling jobs with release dates and tails on identical machines to minimize makespan. European Joiler.1987,29:280~306
- [41] Y.Cho,S.Sahni.preemptive scheduling of independent jobs with release and due times on open, flow and job shops.Open Res,1981,29:511~522
- [42] 郭怀瑞,张洁,高亮等.启发式 GA 调度算法的研究与应用.华中理工大学学报,2000,28(3):24~26
- [43] Chen, C, Vvempati.An application of genetic algorithms for flow shop problems. European Journal of Operational Research, 1995, 80: 389~396
- [44] 王凌.智能优化算法及其应用.北京:清华大学出版社,2001.79~86
- [45] 张铃,张拔.佳点集遗传算法.计算机学报,2001,24(9):917~920
- [46] 黄宇纯.Flow-shop 调度问题的遗传启发算法.信息与控制,1996,8:212~216
- [47] 张长水,阎平凡.解 Job Shop 调度问题的神经网络方案.自动化学报,1995,21(6):
-

706~712

- [48] Ling W,Dazhong Z.An effective hybrid optimization strategy for job shop scheduling problems.Computers&Operations Research, 2001, 28(6): 585~596
- [49] Wang L,Zhong D Z.An Effective Hybrid Heuristic for Flow Shop Scheduling.Int Adv Manuf Technol, 2003, 21: 38~44

附录 1 攻读学位期间发表论文目录

- [1] 肖力. 基于遗传算法的 Flow Shop 调度问题求解. 井冈山学院学报, 2006, 3 (已录用)

基于遗传算法的生产调度优化方法研究

作者：[肖力](#)

学位授予单位：[华中科技大学](#)

本文链接：http://d.g.wanfangdata.com.cn/Thesis_J006373.aspx