遗传算法及其在多目标优化中的应用研究

摘　要

遗传算法是模拟达尔文的遗传选择和自然淘汰的生物进化过程的一种新的迭代的全局优化搜索算法，已经广泛地应用到组合优化问题求解、自适应控制、规划设计、机器学习和人工生命等领域。由于现实世界中存在的问题往往呈现为多目标属性，而且需要优化的多个目标之间又是相互冲突的，从而多目标遗传算法应运而生，它使得进化群体并行搜寻多个目标，并逐渐找到问题的最优解。

本文在广泛深入地查阅国内外文献的基础上，对遗传算法及其面向多目标优化问题的基础理论和基本方法进行了深入的理论研究和实验分析，主要内容如下：

系统、详尽地介绍了遗传算法的一般流程和基本理论、方法，以及面向多目标优化问题的遗传算法的基本理论和方法。对经典的方法进行了全面的分析和比较，指出其应用范围、不足之处，并在此基础之上提出了改进的算法。

提出了对遗传算法的改进策略，在具体问题中结合相应的特点再作相应的改进，通过 TSP 等算例的验证，表明算法是可行的，同时也提高了算法的效率。

介绍了多目标优化问题的基本概念和实现步骤，探讨了多种采用遗传算法的实现方法并比较了其优缺点，表明了遗传算法用来解决多目标优化问题的有效性。该文以 NSGA-II 为基准，提出了改进的多目标遗传算法。针对多目标 JSSP 问题，比较试验结果表明算法在运行效率与保持群体多样性等方面取得了较好效果。

关键词： 遗传算法；多目标遗传算法；Pareto最优解；多目标优化；
小生镜技术

Genetic Algorithm and Its Application Research in Multi-objective Optimization

Abstract

Genetic Algorithm (GA) is a set of new-global-optimistic repeatedly search algorithm which simulates the process of creature evolution that of Darwinian's genetic selection and natural elimination. It is widely applied to the domain of combinational evolutionary problem seeking, self-adapt controlling, planning devising, machine learning and artificial life etc. However, there are multi-objective attributes in real-world optimization problems that always conflict, so the multi-objective genetic algorithm (MOGA) is put forward. MOGA can deal simultaneously with many objections, and find Pareto-optimal solutions gradually.

Based on extensive and deep review of literature, a thorough analysis and research on many theoretical and application oriented problems is presented. The main contents follow:

The basic theory and application of GAs and GAs for multi-objective optimization are systematically and thoroughly introduced. By analyzing on the classical methods, the thesis points out their special applying areas and shortcomings. Some improved algorithms are introduced.

A kind of general improvement in Genetic Algorithm is presented, combining some concrete features, such as TSP programming. Simulation results show that the improved algorithm is feasible, enhancing the efficiency at the same time.

This thesis presents the concept of the multi-objective optimization methods, analysis its implementation steps and the implementation methods with genetic algorithm, and shows that algorithm is practical. In this paper, we take NSGA-II as a benchmark. It improves simulation results on multi-objective JSSP problems and shows that the improved multi-objective genetic algorithm has ideal effects on the aspects of its speed and diversity.

Keywords: Genetic algorithm, Multi-objective genetic algorithm, Pareto-optimal solutions, Multi-objective optimization, Niche technology

插图清单

表格清单

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 合肥工业大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：张伟利　　签字日期：2007年 6月5 日

学位论文版权使用授权书

本学位论文作者完全了解 合肥工业大学 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 合肥工业大学 可以将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、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：张伟利　　导师签名：

签字日期：2007年 6月 5日　　签字日期：2007 年 6 月 5 日

学位论文作者毕业后去向：

工作单位：　　电话：

通讯地址：　　邮编：

致 谢

值此论文完成之际，我谨向所有关心和帮助过我的老师、同学、朋友以及家人致以最真诚的谢意。

首先，我要深深地感谢我的导师倪志伟教授。本人在学习、论文写作以及平时的生活中，自始至终得到了倪老师的悉心指导。无论从课程学习、论文选题，还是到资料收集、论文修改定稿，无不渗透着倪老师的智慧和心血。倪老师渊博的知识、严谨的治学作风以及富于创新的学术思想，让我在学业上受益匪浅，同时也培养了我踏踏实实研究学问的态度，这些都将使我终身受益，并激励我不断前进。

在此，真诚感谢管理学院的全体老师，他们的教诲为本文的研究提供了理论基础，并创造了许多必要条件和学习机会。

感谢智能管理研究所的所有同学，正是通过与你们的互相交流、互相帮助，我才得以不断提高。衷心地祝各位前程似锦！

此外，特别感谢我的室友王小红、徐莹在本文撰写期间所提供的帮助，以及在生活中给予我的照顾。

同时，研究生阶段学习和生活中，我得到了许多朋友的关心、帮助和支持，在此表示衷心感谢！

感谢各位评审专家在百忙之中抽出时间对论文进行了仔细的评阅！

借此机会，感谢我的父母家人，是他们二十多年来的呵护、关心、支持和鼓励，使我得以顺利完成学业。感谢他们给我健康的身体、上进的思想！

第一章 绪论

1.1 论文选题的背景

遗传算法[1]是借鉴生物的自然选择和遗传进化机制而开发出的一种全局优化自适应概率搜索算法。与传统的启发式优化搜索算法相比，遗传算法的主要本质特征在于群体搜索策略和简单的遗传算子。群体搜索使遗传算法得以突破邻域搜索的限制，可以实现整个解空间上的分布式信息探索、采集和继承；遗传算子仅仅利用适应值度量作为运算指标进行随机操作，降低了一般启发式算法在搜索过程中对人机交互的依赖。

几乎每个重要的现实中的决策问题都要在考虑不同约束的同时处理若干相互冲突的目标。最优化处理的是在一堆可能的选择中搜索对于某些目标来说是最优解的问题。如果存在的目标超过一个并需要同时处理，就成为多目标优化问题。由于多目标优化问题的广泛存在性与求解的困难性，该问题一直是富有吸引力和挑战性的。

自 20 世纪 60 年代以来多目标优化问题就受到研究人员越来越多的关注。在多目标优化问题中，多个目标函数需要同时进行优化。由于目标之间的无法比较和矛盾等现象，导致不一定存在所有目标上都是最优的解。某个解可能在一个目标上是最优的但在另一个上是最差的。因此，多目标问题通常存在一个解的集合，它们之间不能简单地进行比较好坏。对这种解来说，不可能使得在任何目标函数上的改进不损害至少一个其他目标函数，这种解称作非支配解和 Pareto 最优解。

九十年代开始的进化计算为求解多目标优化问题提供了有力的工具。遗传算法[2]作为一种超启发式算法，可以灵活地将传统方法结合进其主框架，因此可以利用遗传算法和传统方法两方面的优势来建立对问题更有效的求解方法。遗传算法内在的特征说明了为何遗传搜索适合用于多目标优化问题。遗传算法的基本特征是通过在代与代之间维持由潜在解组成的种群来实现多向性和全局搜索，带有潜在解的种群因此能够一代一代维持下来。在遗传多目标优化中，这种从种群到种群的方法在搜索 Pareto 解时是高效的。在具体问题上，遗传算法与多目标优化问题的结合中最关键的是如何在种群中通过多个目标来评价个体的好坏，即如何根据多个目标来确定个体的适应值。

在过去的几年中，将遗传算法应用于多目标优化问题成为研究热点，这种算法通常称作进化多目标优化或遗传多目标优化。本文主要研究用于解决多目标优化问题的遗传算法，同时将遗传算法和多目标优化引入 TSP 和 Job-shop 调度问题中，重点研究遗传算法的优化作用，利用遗传算法和多目标优化技术增强优化智能。

1.2 遗传算法的发展及研究现状

1.2.1 遗传算法的发展过程

早在20世纪50年代和60年代，就有计算机学者开始研究“人工进化系统”，将进化的思想发展成为许多工程问题的优化工具。早期的研究形成了遗传算法的雏形[3,4,5]，大多数系统都遵循“适者生存”的仿自然法则。60年代初期，柏林工业大学的 I. Rechenberg 和 H. P. Schwefel 在风洞实验中利用了随机变异的思想，并且在对这种方法研究的基础上，形成了进化计算的另一个分支——进化策略(Evolutionary Strategy, ES)。L. J. Fogel 等人在设计有限态自动机(Finite State Machine, FSM)时提出了进化规划(Evolutionary Programming, EP)。20世纪60年代中期，美国 Michigan 大学的 Holland 教授在 A. S. Fraser 和 H. J. Bremermann 等人工作的基础上，提出了位串编码技术。随后，Holland 将该算法用于自然和人工系统的自适应行为的研究中，并于1975年发表了《自然系统和人工系统中的适应问题》(“Adaptation in Natural and Artificial Systems”)一书[6]，该书系统地阐述了遗传算法的基本理论和方法，为其奠定了数学基础，提出了对遗传算法的理论研究和发展极为重要的模式理论(Schemata Theory)。该理论首次确认了结构重组遗传操作对于获得隐并行性的重要性。该书的出版标志着遗传算法的诞生。Holland 等人在以后的应用研究中，将遗传算法推广到优化以及机器学习等问题的应用中，遗传算法的通用编码技术和简单有效的遗传操作为其成功地广泛应用奠定了基础。

Holland 在这本著作中指出，在自然界和人们生活中存在着一类优化问题，这类优化问题具有如下特点:①搜索空间足够复杂，而且搜索开始时关于问题的知识的不确定性很大，即对问题本身的了解很少；②只有通过不断试探解获得新的信息，才能减少这种不确定性；③在获得新信息的同时，还要利用这些新信息，从而使试探解的平均性能随着获取信息量的增加而改善。

求解这类问题时面临两个难题:要获得有关问题的新信息，就要实验以前没有实验过的搜索空间的点；然而要提高试探解的平均性能，就要重复地实验那些已经实验过的，且性能超过平均值的个体。也就是说，如果要使获取信息的速度最大，则所获取的信息无法利用；反之，如果要最大限度地利用已经获取的信息，则很难再去探测新的信息。

Holland 将这类优化问题称为“适应性问题”(Problem of Adaptation)，因为这与生物种群在适应环境时面临的状况相似:生物对环境的适应程度也很难用函数来表示，只能用生物种群中的每一个个体的体验来获得，进化的目标是使生物种群(而不是某个生物个体)适应环境的能力最大。

由于这类问题的目标函数是未知的或没有解析的数学表达式，因而传统的基于导数或解析的优化方法是不可用的；从理论上说，穷举法可以找到全局最

优点，且实现起来简单，但由于这类问题的搜索空间太大，利用这种方法求解时，会因求解时间太长而无法忍受，有时甚至无法完成搜索。这是因为，穷举法不能利用以前的实验结果来缩小搜索空间，即搜索的顺序不受以前测试结果的影响。

因而求解此类问题的算法应该注意以下几点：

(1) 不能简单地认为只要能产生适应环境的结构就好，还要保证求解问题的时间处于合理的范围；

(2) 应该在保持穷举法的通用性的基础上提高效率；

(3) 由于不了解环境，在算法中要有存储和利用实验结果，从中获取信息的手段。

由于适应性问题与生物在进化过程中遇到的问题类似，因此自然界的进化过程就为解决适应性问题提供了思路。分析自然界的进化现象可以看到，进化过程关键是通过群体搜索、编码表示、遗传操作来找到适合环境的最优解，并最终解决了两难问题。这三者缺一不可。没有群体就无法引入生存竞争，无法选出优良基因；不用编码表示就无法存储进化所得的成果，无法将某些优良特性与基因模式联系起来，无法利用适应度来改善搜索；不用遗传操作就无法根据个体基因的适应度，搜索更适合环境的个体。

因而，Holland[6]提出了模仿自然界的进化机制来解决适应性问题的优化算法——遗传算法。由于他提出的遗传算法简单实用，因而已普遍被接受采纳。一般就认为这标志了遗传算法的诞生。

鉴于 Genetic Algorithm 的特点与优点，继 Holland 之后，越来越多的学者开始致力于 GA 的研究与应用，主要是探讨 GA 的应用和在应用时遇到的问题。其中有 De Jong[7]对 GA 各种策略的性能和机理进行的大量而细致的实验和分析，在一定程度上是 De Jong 的工作使人们开始看到了 GA 的应用价值。在 Holland 的书中还给出了一个自适应的规则学习系统(Classifier System)，Goldberg[8]将它成功地应用于天然气管道系统的控制规则的优化问题上，这是 GA 应用的一个著名例子。Bethke 的博士论文提出了用 Walsh 函数来研究 GA 的方法；Albert 大学的 Brindle 在博士论文[9]中对选择策略进行了研究。这一时期的研究成果主要是回答了 GA 到底有何意义，有何价值。鉴于他们的研究工作，很多人的注意力逐渐转向了 GA。1985 年召开了第一届 GA 的国际会议，以后每隔一年举行一次。

自 80 年代中期以来，是 GA 的蓬勃发展期，研究者对遗传算法的应用研究不断扩大和深入。有用于 TSP 问题[10,11]、调度问题[12]、键盘构造问题、多关节机械手轨迹规划问题[13]、工程中的设计辅助问题、机器学习问题[14]、囚徒困境(Prisoner's Dilemma)问题、模式分类问题，神经网的基因综合，神经网的辅助设计、程序自动生成——遗传编码。此外，遗传算法还在预测控制等问题中

都有运用研究。目前 GA 的研究已构成与神经网络，模糊控制并驾齐驱的一大领域。

1.2.2 遗传算法的国内外研究现状

GA 的研究热潮有其自身的原因，首先它思想独特，与传统的一些优化算法有很大不同；其次就是 GA 非常一般化，实现起来较简单，什么问题都可以套用，好象成了一把万能钥匙；第三就是它虽然操作简单，但人们对它的性能却了解得不透，这就给研究者们留下了广阔的探索空间。

人们目前主要研究 GA 的以下三个方面[15,16,17]：

(1) GA 的搜索。当进化到一定程度后，种群中的个体适应性往往会变得非常接近，这样，个体间的选择概率就会变得非常接近，此时的选择过程就变成了一种接近纯随机的抽样过程。另外，群体规模往往不可能取得很大，因此而产生的一些涨落误差会使实际抽样结果与预期结果有差距，这就可能使一些好的或比较好的个体落选。针对这样的一些问题，研究者们提出了不同的选择策略和复制策略，还有一类改进是对种群中的适应度的分布进行变换，例如进行线性尺度变换(Linear Scaling)，截取变换(Sigma Truncation)，窗口变换(Window Scaling)，基于分享函数(Sharing Functions)的变换等。对 GA 的行为有实质影响的改进研究主要在这两方面，此外还有与问题有关的 GA 的变种。

(2) GA 的收敛性研究。主要结果是 GA 不会收敛到全局最优解，对于保留精英(Elitism Selection)的选择策略，GA 能收敛到全局最优。尽管这一结论说明了改进的 GA 最终能收敛到全局最优解，但收敛到最优解的时间会很长。

(3) GA 的性能研究，包括与其它算法的比较研究。到目前为止，还没有什么肯定的结论。它与同类算法相比，例如爬山法，模拟退火算法，Tabu 算法等，往往是问题不一样结论也不一样，甚至对同一问题，不同人的研究结果也不一样。目前，GA 界的研究者们多用人为结构的问题，如皇家大道函数(Royal Road function)，来研究 GA 的表现，以期找出具有什么特性的问题适合或不适合 GA。还有人用构造的 Walsh 多项式函数来研究 GA 的行为，以及其它人为构造的函数。GA 的优越性是建立在积木块假设的基础上，但是有些函数不满足这个假设，叫做欺骗性函数(Deceptive Functions)，有关这个问题的论题包括什么样的函数具有欺骗性，欺骗现象的普遍性怎样等[18,19]。

1.3 多目标遗传算法的发展及研究现状

1.3.1 多目标遗传算法的发展过程

多目标优化问题一直是科学和工程研究领域的一个难题和热点问题，不仅因为许多工程问题本身就是一个多目标优化问题，而且在多目标优化领域内还

存在着许多尚未解决的难题。遗传算法应用于单目标问题之后的20多年以后，多目标遗传算法逐渐成为研究热点。十几年来出现了许多基于进化方法的多目标优化技术。

国际上一般认为多目标最优化问题最早是由法国经济学家 V.Pareto 在1896年提出的。1967年，Rosenberg在其博士论文中提出了在模拟单细胞有机物的化学遗传特性中采用多属性研究方法，在Rosenberg的研究中虽然在他的研究中最终只应用了单一属性方法，正是他的研究开创了这个领域的研究。直到1985年，Scaffer 提出了第一个基于向量评估的多目标遗传算法（VEGA），从而开创了用遗传算法求解多目标规划的先河。真正引起演化界重视的是1990年后，Fonseca 和 Fleming 提出基于排序选择的多目标遗传算法（MOGA），Srinivas 和 Deb[20]提出非劣分层遗传算法（NSGA），Horn 和 Nafpliotis 也提出了小组决胜遗传算法（NPGA），Hom 等人提出的基于小生镜(Niche)技术[21]的"小生镜 Pareto 遗传算法"。Goldberg 等人提出了一种 Pareto 排序算法，这些算法已成为遗传多目标优化算法研究的基石。在此基础上，人们又提出可以在多目标算法中引入最优保存策略和约束处理策略，并提出了实现这些策略的不同的多目标进化算法。

遗传算法的操作需要适应度值的信息，那么，对于多目标优化问题来说最直接的方法就是采用加、乘或其他算子将各个目标函数整合成一个单目标优化问题。但是，这种方法存在很多问题：首先我们必须提供目标函数空间的精确的标量信息，以避免其中的一个目标函数支配其他目标函数。这就意味着必须了解每一个目标函数的行为，而其计算量十分巨大，通常都是不可能实现的。显然，如果这种整合方法可以实现的话，那么它将是最简单、有效的处理多目标问题的方法。因为它不需要和决策者做任何交互就可以得出最优解或至少是近优解。整合成的目标函数称为和函数，早期的多目标优化中有很多这方面的工作，比如线性加权法、变权重加权法等。线性加权法为每个目标函数采用不同的权重因子，对所有的目标函数进行整合。这种方法是第一种用来求多目标优化问题的非劣解的方法，它是受 Kuhn 和 Tucker 的数值优化方法的启发而演变来的。线性加权法简单、有效、实用，对它的研究和应用有很多。1991 年 Syswerda 和 Palmucci 就使用权重法和遗传算法相结合来解决资源计划问题。

目标规划法中决策者必须确定理想中的每个目标函数所能达到的目标值，然后将这些值作为附加的约束整合进问题中，对其进行优化就是最小化理想的目标值和目标函数值之间的绝对偏差。此法适用于目标函数是线性的或者是部分线性的情况，对于非线性的情况它可能不太适用。1992年 Wienke et al.使用目标规划法和遗传算法结合来解决核物理学中的问题。

1993年 Wilson 和 Macleod 采用目标满意法和遗传算法来设计滤波器。1997年 Quagliarella 和 Vicini 共同采用杂合遗传算法解决多目标优化问题。早期

的多目标遗传算法一般都是遗传算法和一些经典的多目标优化技术的结合的产物，他们普遍效率不高、局限性大、鲁棒性差。之后出现了基于多种群的 VEGA、基于字典序的方法、基于博弈论的方法等等，这些方法的特点是实用性差、解的精度不高、不能保证求得最优解。

1.3.2 多目标遗传算法的研究现状

目前多目标遗传算法的研究热点是采用 Pareto 机制[22]的多目标优化技术，包括:MOGA, NSGA, NPGA, SPEA 等等。

MOGA 由 Fonseca 和 Fleming 于 1993 年提出，主要思想是个体排序的序号由当前种群中支配它的个体的数量来决定。MOGA 在多目标优化领域得到了广泛的应用。Chen Tan 和 Li 于 1997 年成功地将其应用于 ULTIC 控制器的优化问题中，得到了一组满意的时域和频域参数；Chipperfield 和 Fleming 于 1995 年将其应用于喷气发动机的多变量控制系统优化问题中； Obayashi 于 1997 年应用 Pareto 排序、表现型共享和 best-N(从 N 个父代个体和 N 个子代个体中选择 N 个最佳个体最为下一代种群)选择来设计空气动力学中的压缩机叶片形状；Rodrguez Vdzquez et al.将 MOGA 和遗传规划(genetic programming)结合，形成了 MOGP (Multiple Objective Genetic Programming)算法；Todd 和 Sen 于 1997 年采用改进的 moo 来解决大规模组合优化问题——集装箱布局(containership layouts)问题等等。

非支配排序遗传算法——NSGA 是由 Srinivas 和 Deb[23]于 1993 年提出的。算法是基于对个体的几层分级实现的。NSGA 在现实问题的求解中得到了广泛的应用，Periaux et al.于 1997 年采用 NSGA 最小化空气动力学中的反射器的散射;Vedarajan et al.于 1997 年采用 NSGA 进行投资利润的最优化;Michielssen 和 Weile 于 1995 年采用 NSGA 设计电磁系统，并取得了较好的解。

Horn和Nafpliotis于1993年提出了一种基于Pareto支配的联赛选择方法——NPGA，该方法中联赛规模不局限于两个个体，而是由种群中的多个个体参与来决定支配关系(一般在10个左右)。当两个个体之间没有明显的支配或被支配关系时，通过适应度共享来决定联赛选择的结果。Belegunduet al.于1994年应用NPGA设计多层复合陶瓷问题。SPEA 由Eckart Zitzler于1999年提出，在他的论文中解决了计算任务在不同体系结构的系统上的调度问题，并和其他算法的解进行了详尽的比较。

当前，遗传多目标优化算法作为一个优化工具在解决智能决策问题研究中越来越显示出它的强大生命力,在生产系统中对生产计划调度集成系统的优化，在金融证券和经济预测系统中的模型优化，在决策支持系统中解决模型结构选择和模型实例确定的问题等领域都取得了较显著的成果。

1.4 论文主要内容与结构

论文的整体架构图见图 1.1。

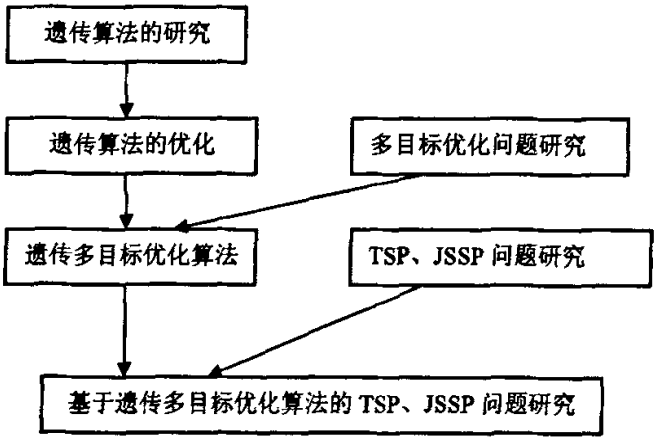

图 1.1 论文结构图

全文共分五章，各章主要内容分述如下：

第一章是绪论。说明了论文的选题背景、依据，对遗传算法和多目标优化遗传算法的国内外研究现状进行了概述，给出全文的整体架构图和各章的研究内容。

第二章是遗传算法的基本原理和方法。首先对遗传算法的基本框架进行了概述，简述了遗传算法的编码原理，初始种群的生成，适应度函数原理，参数设定的方法，遗传操作，模式定理及积木块假说。最后介绍了遗传算法的欺骗问题，隐含并行性问题和收敛性分析。

第三章是遗传算法的若干改进及其应用研究。首先在基本遗传算法的基础上，针对其缺点，介绍了在遗传算法的使用范围，遗传算法的参数、编码，选择、交叉、变异等遗传操作的若干改进。随后成功地将遗传算法应用到 TSP 问题中，提出一种改进的遗传算法的优化方案，进行仿真研究，给出实验测试结果。

第四章是遗传多目标优化的应用研究。首先对多目标遗传算法的基本理论进行了概述。随后，介绍了多目标优化和遗传算法的融合、发展状况及其分类。针对多目标 JSSP 问题的求解需求，使用改进的多目标遗传算法对 JSSP 问题进行求解。利用国际案例库中的数据对算法进行测试，实验结果证明算法达到了良好的效果。

第五章是总结与展望。首先阐述总结遗传算法操作的相关改进。接着对遗传算法的研究方向和未来趋势进行介绍。最后对本文所作工作进行了展望。

第二章 遗传算法的基本原理和方法

遗传算法[1,2,3]是模拟生物在自然环境中的遗传和进化过程而形成的一种自适应全局优化概率搜索算法。它最早由美国密执安大学的Holland教授提出，起源于20世纪60年代对自然和人工自适应的研究。70年代De Jong基于遗传算法的思想在计算机上进行了大量的纯数值函数优化计算实验。在一系列研究工作的基础上，80年代由Goldberg进行归纳总结，形成了遗传算法的基本框架。

2.1 遗传算法的基本框架

遗传算法是一种群体型操作，该操作以群体中的所有个体为对象。选择(selection)，交叉(crossover)和变异(mutation)是遗传算法的三个主要操作算子。遗传算法包括6个基本要素：参数编码、初始种群的设定、适应度函数的设计、遗传算子的设计、停止运行准则及结果的编码、控制参数设定(包括种群规模、交叉概率、变异概率，停止运行准则的参数等)。

2.1.1 遗传算法的运算步骤

步骤1：初始化，设置进化代数计数器t_0，设置最大进化代数max_*gen*，种群规模N，随机产生N个个体作为初始种群M_0；

步骤2：个体评价，计算种群M_t中各个个体的适应度；

步骤3：选择操作，将选择算子作用于种群M_t；

步骤4：交叉操作，将交叉算子作用于种群M_t；

步骤5：变异操作，将变异算子作用于种群M_t；种群M_t经过选择、交叉、变异操作后得到下一代种群M_{t+1}；

步骤6：终止条件判断，若$t+1=$max_*gen*，则算法停；否则置$t=t+1$，转步骤2。

2.1.2 遗传算法的基本框架

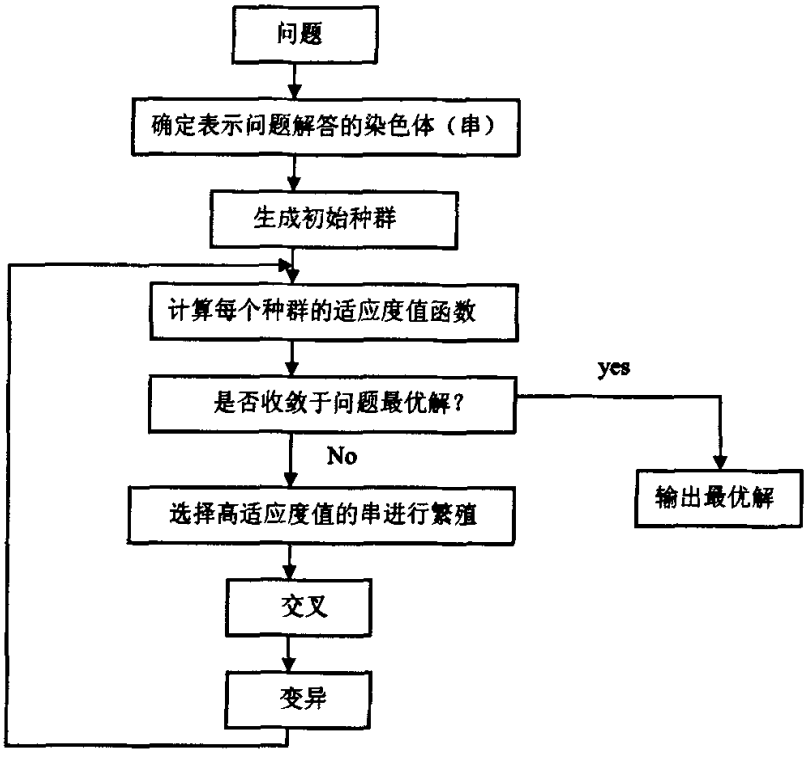

图 2.1 遗传算法的基本流程图

2.2 编码

把问题空间中的参数或解转化成遗传空间中的染色体或个体，称作编码(encoding)，相反过程称为译码(decoding)。由于遗传算法通常不能直接处理解空间的数据，所以必须通过编码将它们表示成遗传空间的基因型串结构数据。编码规则有：

(1) 完备性(completeness)：问题空间中的所有点都成为编码后空间中的点；

(2) 健全性(soundness)：编码后空间的点能对应原问题空间的所有点；

(3) 非冗余性(non-redundancy)：编码后空间的点与原问题空间的点一一对应；

(4) 有意义积木块编码规则：应使用能易于产生与所求问题相关的底阶，短定义长度模式的编码方案。

对于实际应用问题，必须对编码方法、交叉操作方法、变异操作方法、解码方法等统一考虑，以寻求到一种对问题的描述最为方便、遗传算法效率最高的编码方案。编码方法有很多，如：二进制编码、格雷码编码、实数编码、符号编码、多参数级联编码、多参数交叉编码。对不同问题，应选用合适的编码方法。

2.3 初始种群的生成

一定数量的个体组成了种群(population)，种群中个体的数目称为种群规模(population size)。由于遗传算法的种群型操作需要，所以必须为遗传操作准备一个由若干初始解组成的初始种群。初始种群的每个个体都是通过随机方法产生，它也称为进化的初始代。种群规模是遗传算法的控制参数之一，其选取对遗传算法效能的发挥有影响。一般种群规模在几十到几百之间取值，问题越难，维数越高，种群规模越大。

初始种群的设定可采取以下策略：

(1) 设法把握最优解在整个问题空间的分布范围，然后在此分布范围内均匀设定初始种群。

(2) 先随机生成一定数目的个体，然后从中选出最好的个体加入到种群中，不断重复这一过程，直到达到种群规模。

2.4 适应度函数

各个体对环境的适应程度叫适应度(fitness)。遗传算法在进化搜索过程中一般不需要其它外部信息，仅用适应度来评估个体或解的优劣，并作为以后遗传操作的依据。

(1) 评价个体适应度的一般过程是：对个体编码串进行解码处理后，可得到个体的表现型，由个体的表现型可计算出对应个体的目标函数值，根据最优化问题的类型，由目标函数值按一定的转换规则求出个体的适应度。

(2) 适应度尺度变换：在遗传算法中，种群的进化过程是以种群中各个体的适应度为依据，通过一个迭代过程，不断地寻求出适应度较大的个体，最终就可得到问题的最优解或近似最优解。在遗传算法运行的不同阶段还需要对个体的适应度进行适当的扩大或缩小，这就是适应度尺度变换。在进化的初期，为避免未成熟收敛，应缩小个体间的差异；在进化的最后阶段，为了加快收敛，应放大个体间的差异。常用的尺度变换有三种:线性尺度变换、乘幂尺度变换、指数尺度变换。

2.5 参数设定

遗传算法中需要选择的参数主要有串长l，种群规模n，交叉概率P_c，变异概率P_m等，这些参数对遗传算法的性能影响比较大。在简单遗传算法(SGA)或标准遗传算法(CGA)中，这些参数是不变的。然而，许多学者意识到这些参数需要随着遗传算法的进程而自适应变化，这种有组织性能的遗传算法具有更高的鲁棒性、全局最优性和效率，但会增加算法的复杂性和计算量等。还有一些参数的改进方法，比如，模糊控制参数、模拟退火方法、基于均匀设计的参数设

定等。

2.6 遗传操作

遗传操作是模拟生物基因遗传的操作。在遗传算法中，通过编码组成初始群体后，遗传操作的任务就是对群体的个体按照他们对环境适应的程度（适应度评估）施加一定的操作，从而实现优胜劣汰的进化过程。从优化搜索的角度而言，遗传操作可使问题的解，一代又一代地优化，并逼近最优解。遗传算法的遗传操作包括以下三个基本遗传算子（genetic operator）：1、选择(selection)，2、交叉(crossover)，3、变异(mutation)。这三个遗传算子有如下特点：

(1) 这三个遗传算子的操作都是在随机扰动情况下进行的。换句话说，遗传操作是随机化操作，因此，群体中的个体向最优解迁移的规则是随机的。需要强调的是，这种随机化操作和传统的随机搜索方法是有区别的。遗传操作进行的是高效有向的搜索而不是如一般随机搜索方法所进行的无向搜索。

(2) 遗传操作的效果和上述三个遗传算子所取的操作概率、编码方法、群体大小、初始群体以及适应度函数的设定密切相关。

(3) 三个基本遗传算子的操作方法或操作策略随着具体求解问题的不同而异。更具体而言，是和个体的编码方式直接有关。由于目前二值编码仍是最常用的编码方法，所以，以下的对遗传算子的论述是以二值编码为基础的。

下面分别介绍三种遗传算子中比较常用的几种操作方法：

2.6.1 选择算子

选择操作就是用来确定如何从父代种群中按某种方法选取哪些个体遗传到下一代种群的遗传运算。选择操作建立在对个体的适应度进行评价的基础上，其主要目的是为了避免基因缺失，提高全局收敛和计算效率。

常用的选择算子有:比例选择、保留最佳个体选择、期望值选择、排序选择、随机联赛选择。

1.适应度比例方法

适应度比例方法是目前遗传算法中最基本的选择方法。它也叫赌轮或蒙特卡罗选择。在该方法中，各个个体的选择概率和其适应值成比例。

设群体大小为n，其中个体i的适应度值为fi，则i被选择的概率P_{si}为

$$P_{si}= fi\Big/\sum_{i=1}^{M}\mathrm{fi} \qquad (2.1)$$

显然，概率P_{si}反映了个体i的适应度在整个群体的适应度总和中所占的比例。个体适应度越大，其被选择的概率就越高，反之亦然。按式2.1计算出群体中各个个体的选择概率后，就可以决定哪些个体被选出。该思想可用以下的算

法描述：

```
Procedure selection
begin
        i=0
        sum=0
        wheel-pos=random * FSUM
        while  sum<=wheel-pos or i<=PSIZE do
             i=i+1
             sum=sum+Fi
        endwhile
        str-no=i
end
```

这里，FSUM为群体中所有个体适应度和，Fi为第i个个体的适应度，random是在[0，1]区间内产生随机数的随机函数，wheel-pos相当于赌轮环上的位置，str－no是所选个体号。

2.最佳个体保存方法

该方法的思想是把群体中适应度最高的个体不进行配对交叉而直接复制到下一代中。此种选择操作又称复制。De Jong对此方法作了以下定义：

定义2.1 设到时刻t（第t代）时，群体中$a^*(t)$为最佳个体。又设$A(t+1)$为新一代群体，若$A(t+1)$中不存在$a^*(t)$，则把$a^*(t)$作为$A(t+1)$中的第$n+1$个个体（其中，n为群体大小）。

采用此选择方法的优点是，进化过程中某一代的最优解可不被交叉和变异操作所破坏。但是，这也隐含了一种危机，即局部最优个体的遗传基因急速增加而使进化有可能限于局部解。也就是说，该方法的全局搜索能力差，它更适合单峰性质的搜索空间搜索，而不是多峰性质的空间搜索。所以此方法一般都与其他选择方法结合使用。

3.期望值方法

在赌轮选择方法中，当个体数不太多时，依据产生的随机数有可能会出现不正确地反映个体适应度的选择，即存在统计误差。也就是说，适应度高的个体也有可能被淘汰，为克服这种误差，期望值方法用了以下思想。

(1) 计算群体中每个个体在下一代生存的期望数目：

$$M = fi/\overline{f} = fi/\sum fi \ / \ n \qquad (2.2)$$

(2) 若某个体被选中并要参与配对和交叉，则它在下一代中的生存的期望数目减去0.5；若不参与配对和交叉，则该个体的生存期望数目减去1。

(3) 在(2)的两种情况中，若一个个体的期望值小于零时，则该个体不参与选择。

De Jong 曾对比了适应度比例选择法，最佳个体保存法和期望值法用于函数优化中的性能。对比实验表明，采用期望值法的遗传算法离线性和在线性都高于采用另外两种方法的遗传算法性能。

4.排序选择方法

所谓排序选择方法是指在计算每个个体的适应度后，根据适应度大小顺序对群体中个体排序，然后把事先设计好的概率表按序分配给个体，作为各自的选择概率。所有个体按适应度大小排序，而选择概率和适应度无直接关系而仅与序号有关。这种方法的不足之处在于选择概率和序号的关系需事先确定。此外，它和适应度比例方法一样都是基于概率的选择，所以仍有统计误差。

5.联赛选择方法

该方法的操作思想是，从群体中任意选择一定数目的个体(称为联赛规模)，其中适应度最高的个体保存到下一代。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。联赛规模一般取2。

以上介绍的是目前常用的几种选择方法。每种方法对于遗传算法的在线和离线性能的影响均各不相同。在具体使用时，应根据问题求解特点采用较合适的方法或者把它们结合使用。

2.6.2 交叉算子

交叉操作是遗传算法中最主要的遗传操作，交叉算子的设计和实现与所研究的问题密切相关，一般要求即不要太多破坏个体编码串中表示优良性状的优良模式，又要能够有效地产生一些较好的新个体模式，它的设计要和个体编码设计统一考虑。

常用的交叉算子有:单点交叉、双点交叉、多点交叉、一致交叉、均匀交叉、算术交叉。

(1) 单点交叉

单点交叉又叫一点交叉。具体操作是：在个体串中随机设定一个交叉点，实行交叉时，该点前或后的两个个体的部分结构进行互换，并生成两个新个体。

(2) 双点交叉

双点交叉的操作和单点交叉类似，只是设置2个交叉点(依然是随机设定)。一个双点交叉的例子表示如下：

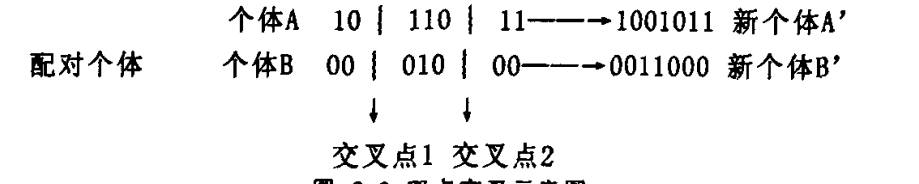

图 2.2 双点交叉示意图

由此可见，2个交叉点分别设定在第二基因座和第三基因座以及第五基因座

和第六基因座之间。A，B两个体在这两个交叉点之间的码串相互交换，分别生成新个体A'和B'。对于双点交叉而言，若染色体长为n，则可能有$(n-2)\times(n-3)$种交叉点的设置。

(3) 多点交叉

多点交叉是前述两种交叉的推广，有时又被称作为广义交叉。若用参数cp来表示交叉点数，则当$cp=1$时，广义交叉就等效于单点交叉。当cp是偶数时，根据交叉互换规律，染色体可以当作基因环来处理。

一般来讲，多点交叉较少采用，因为它影响遗传算法的在线和离线性能。事实上，若染色体长为n，则对多点交叉而言共有$(\frac{n}{cp})$种交叉设置随机挑选。当cp较大时，每种交叉点被随机选中的可能性很小，因此能被保存的部分结构也就很少。这有些类似于随机搜索的行为。也就是说，多点交叉不能有效地保存重要的模式。

(4) 一致交叉

所谓一致交叉是指通过设定屏蔽字来决定新个体的基因继承两个旧个体中哪个个体的对应基因。一致交叉的操作过程表示如下：当屏蔽字中的位为0时，新个体A'继承旧个体A中对应的基因，当屏蔽字位为1时，新个体A'继承旧个体B中对应的基因，由此生成一个完整的新个体A'。反之，可生成新个体B'。显然，一致交叉包括在多点交叉范围内。

(5) 其它交叉方法

针对各个问题的不同，可以提出各种交叉方法。其他交叉方法还有：二维交叉，Messay GA中的交叉，树结构交叉以及TSP问题的部分匹配交叉（PMX），顺序交叉（OX）和周期交叉（CX）等。

2.6.3 变异算子

变异运算是指将个体染色体编码串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，从而形成了一个新个体。交叉操作是产生新个体的主要方法，它决定了遗传算法的全局搜索能力；而变异操作只是产生新个体的辅助方法，它决定遗传算法的局部搜索能力，维持种群的多样性，防止出现早熟现象。

遗传算法通过交叉和变异这一对相互配合而又相互竞争的操作而使其具备兼顾全局和局部的均衡搜索能力。所谓相互配合，是指当群体在进化中陷于搜索空间中某个超平面而仅依靠交叉不能摆脱时，通过变异操作可有助于这种摆脱。所谓相互竞争，是指当通过交叉形成所期望的积木块时，变异操作有可能破坏这些积木块。因此，如何有效地配合使用交叉和变异操作，是目前遗传算法的一个重要研究内容。

常用的变异算子有：基本位变异、均匀变异、边界变异、非均匀变异、高斯变异。

(1) 基本变异算子

基本变异算子是指对群体中的个体码串随机挑选一个或多个基因座并对这些基因座的基因值做变动（以变异概率 Pm 做变动），{0, 1}二值码串中的基本变异操作如下：

个体 A　1 0 1 1 0 1 1——→1 1 1 0 0 1 1 个体 A'

变异基因座

图 2.3 基本变异算子示意图

(2) 均匀变异算子

均匀变异适用于实数编码的染色体串。假设有个体 $x=(x_1,x_2,\ldots, x_k;\ldots; x_q)$。随机选择第 k ($k\in(1,q)$)个基因进行变异所得的子代个体为：

$x=(x_1,x_2,\ldots, x'_k;\ldots; x_q)$，其中：$x'_k\in$[left(k), right(k)]，left(k)和 right(k)分别为基因的取值范围的上、下界。

(3) 逆转算子

逆转算子是变异算子的一种特殊形式。它的基本操作内容是，在个体码串中随机挑选二个逆转点，然后将两个逆转点间的基因值以逆转概率 P_j 逆向排序。

{0, 1}二值码串的逆转操作如下：

逆转

个体 A　10 11010 00 ——→ 10 01011 00　　个体 A'

逆转点

图 2.4 逆转变异算子示意图

由此可见，通过逆转操作，个体中从基因座 3 到基因座 7 之间的基因排列得到逆转，即从 11010 序列变成了 01011 序列。这一逆转操作可以等效为一种变异操作，但是逆转操作的真正目的并不在变异（否则仅用变异操作就行了）而在实现一种重新排序操作。

(4) 自适应变异算子

该算子与基本变异算子的操作内容类似，唯一不同的是交叉概率 P_m 不是固定不变而是随群体中个体的多样性程度而自适应调整。一般是根据交叉所得两个新个体的海明距离进行变化。海明距离越小，P_m 越大，反之 P_m 越小。

2.7 模式定理

模式定理是遗传算法的理论基础，它描述了遗传算法的搜索策略。

2.7.1 模式的定义

模式是描述字符串集的模板，该字符串集中的串的某些位置上存在相似性。

不失一般性，我们考虑二值字符集{0，1}，由此可以产生通常的0，1字符串。现在我们增加一个符号“*”，称作“无关符”或通配符，即“*”既可以被当作“0”，也可以被当作“1”。这样，二值字符集{0，1}就扩展成三值字符集{0，1，*}，由此可以产生诸如0110、0*11**、**01*0等字符集。

定义 2.2 基于三值字符集{0，1，*}所产生的能描述具有某些结构相似性的0、1字符串集的字符串称作模式。

定义 2.3 模式是指在某些基因座上具有相同基因的染色体集合。记作 H∈{0，1，*}[1]。

定义 2.4 模式 H 中确定位置的个数称作模式的该模式的模式阶，记作 O（H）。

比如模式 011*1*的阶数为4，而模式 0*****为 1。显然，一个模式的阶数越高，其样本数就越少，因而确定性越高。

但是，模式阶并不能反映模式的所有性质。以后将看到，即使具有同阶的模式，在遗传操作下，也会有不同的性质。对于模式“01***”和“0***1”它们的阶相同，但是，模式“0***1”要跨越更长的距离，更容易被破坏。所以为此，我们再引入定义距概念：

定义2.5 模式H中第一个确定位置和最后一个确定位置之间的距离称作该模式的定义距，记作δ（H）。

比如模式 011*1*的定义距为4，而模式 0*****的定义距为0。

2.7.2 模式定理

定理 2.1 （模式定理）在遗传算子选择、交叉和变异的作用下，具有低阶、短定义距以及平均适应度高于群体平均值的模式在子代中将得以指数级增长。

假设给定时间步 t，一个特定的模式 H 有 m 个代表串包含在种群 $A(t)$ 中，记为 $m=m(H,t)$。在再生阶段，每个串根据它的适应值进行复制，一个串 A_i 的再生概率为：

$$P_i = fi \Big/ \sum_{i=1}^{n} \text{fi} \tag{2.3}$$

当采用非重叠的 n 个串的种群代替种群 $A(t)$ 时，可以得到下式：

$$m(H,t+1)=m(H,t)*n*\frac{f(H)}{\sum_{j=1}^{n} fj} \tag{2.4}$$

其中 $A(t)$ 是在时间步 t 时表示模式 H 的串的平均适应度，整个种群的平均适应度可记为：

$$\bar{f}=\frac{\sum_{j=1}^{n} fj}{n} \tag{2.5}$$

故在基本遗传算法的结构条件下，若遗传操作只选择转移到下一代的话，则下式成立：

$$m(H,t+1)=m(H,t)*\frac{f(H)}{f} \tag{2.6}$$

假设从 t=0 开始，某一特定模式适应度值保持在种群平均适应度值以上的一个 $c\bar{f}$，c 为一常数，则模式的选择生长方程变为：

$$m(H,t+1)=m(H,t)*\frac{\bar{f}+c\bar{f}}{\bar{f}}=(1+c)*m(H,t)=(1+c)^{t}*m(H,0) \tag{2.7}$$

由于交叉本身是以随机方式进行的，即以概率 P_c 进行交叉，因此对于模式 H 的生存概率可以计算如下：

$$P_s\geq 1-P_c*\frac{\partial(H)}{l-1} \tag{2.8}$$

同时考虑选择、交叉操作对模式的影响，由于选择和交叉操作是不相关的，可以得到子代模式的估计：

$$m(H,t+1)\geq m(H,t)*\frac{f(H)}{\bar{f}}[1-P_c*\frac{\partial(H)}{l-1}] \tag{2.9}$$

当模式 H 中 $O(H)$ 个确定位都存活时，这时模式 H 被保留下来，存活概率为：

$$(1-P_m)^{O(H)}\approx 1-O(H)*P_m \qquad (P_{m<<1}) \tag{2.10}$$

因此，在考虑选择、交叉和变异操作的作用下，一个特定模式在下一代中期望出现的数目可以近似的表示为：

$$m(H,t+1)\geq m(H,t)*\frac{f(H)}{\bar{f}}[1-P_c*\frac{\partial(H)}{l-1}-O(H)*P_m] \tag{2.11}$$

式中：

$m(H,t+1)$——表示在 $t+1$ 代种群中存在模式 H 的个体数目；

$m(H,t)$——表示在 t 代群体中存在模式 H 的个体数目；

$f(H)$——表示在 t 代群体中包含模式 H 的个体平均适应度；

$\bar{f}$——表示在 t 代群体中所用个体的平均适应度；

L——表示个体的长度；

P_c——表示交叉概率；

P_m——表示变异概率。

综上所述 ，可以得出模式定理(schema theory)：

在遗传算子选择、交叉、变异的作用下，具有低阶、短定义距、平均适应度高于种群平均适应度的模式在子代中呈指数增长。由于交叉、变异的作用，并非所有的模式都是有效模式，但在一个规模为 N 的种群中，有 $O(N^3)$ 个模式

是有效的，即每一代GA处理N个个体的同时，获得了对N^3个模式的并行处理，并且无须额外的存储量，这一性质称为内在并行性。也就是GA在群体中搜索的过程可看作是隐含的模式抽样过程。然后，通过遗传算子的作用将短的低阶信息组合起来，形成高阶模式，直至搜索到最优解。

2.7.3 模式定理的拓广和深入

Radelife 把模式分析进行了一般化，提出了完整的Forma分析理论。在解决积木块假设是否成立的问题上，主要研究遗传算法的骗函数，主要方法是Walsh变换。目前尚未有人对隐含并行性提出改进办法，只是加以主观默认。

2.8 积木块假设

由于遗传算法的求解过程并不是在搜索空间中逐一地测试各个基因的枚举组合，而是通过一些较好的模式，像塔积木一样，将他们拼接在一起，从而逐渐地构造出适应度越来越高的编码串。

Holland 指出，遗传算法中的交叉操作可以将不同的低阶模式组合成高阶模式，从而大大缩小后代的搜索范围。他认为，交叉是 GA 探索未知空间的主要操作，而变异只是一种保持种群多样性的“背景”操作。所以，应将交叉概率取较大的值，而将变异概率取很小的值。Goldberg 给出了积木块的定义。

定义 2.6 具有低阶、短定义距以及高适应度的模式称作基因块或积木块(Building Block)。

模式定理说明了具有某种结构特征的模式在遗传进化过程中其样本数将按指数级增长，这种模式就是具有低阶、短的定义长度，且平均适应度高于群体平均适应度的模式。这种类型的模式被称为积木块。

之所以称之为积木块，是由于遗传算法的求解过程并不是在搜索空间中逐一地测试各个基因的枚举组合，而是通过一些较好的模式，像搭积木一样、将它们拼接在一起，从而逐渐地构造出适应度越来越高的个体编码串。

模式定理说明了积木块的样本数呈指数级增长，亦即说明了用遗传算法寻求最优样本的可能性，但它并未指明遗传算法一定能够寻求到最优样本。而积木块假设却说明了遗传算法的这种能力。

定理2.2 积木块假设(Building block hypothesis)：个体的基因块通过选择、交叉、变异等遗传算子的作用，能够相互拼接在一起，形成适应度更高的个体编码串。

积木块假设说明了用遗传算法求解各类问题的基本思想，即通过基因块之间的相互拼接能够产生出问题更好的解。基于模式定理和积木块假设，就使得我们能够在很多应用问题中广泛地使用遗传算法的思想。

需要说明的是，虽然积木块假设并未得到完整而严密的数学证明，但大量

的应用实践说明其有效性。

2.9 遗传算法欺骗问题

提到积木块假设，这里就必须说明一下遗传算法欺骗问题[23,24]（GA Deceptive problem）。应用实践表明，存在着一类用遗传算法难以求解的问题，这类称为“GA-难”的问题往往不满足积木块假设，即由基因块之间的拼接，往往会欺骗遗传算法，使其进化过程偏离最优解。

各种研究结果表明，属于“GA-难”的问题一般包含有孤立的最优点，即在这个最优点周围是一些较差的点，从而使得遗传算法较难通过基因之间的相互拼接而达到这个最优点的模式。实际上，目前也尚无解决这类问题的较好的方法或策略。所幸的是，现实所遇到的各种应用问题中，很少有这种奇怪的性质。

2.10 隐含并行性

在遗传算法的运行过程中，每代都处理了N个个体，但由于一个个体编码串中隐含有多种不同的模式，所以算法实质上却并行处理了更多的模式，所处理的模式总数与群体规模N的立方成正比。

2.11 遗传算法的收敛性分析

到目前为止，还没有一套完整的理论可以准确、全面地阐述一般遗传算法的收敛性，从而对其在大量应用中所表现出的全局优化能力作出理论解释；也还没有找到一个恰当的度盈与论证方法精确刻画遗传算法在不同实现下的收敛速度，从而对遗传算法的各种改进作出统一、公正的评判。为了保证遗传算法的收敛速度，有两个参数非常重要：一个参数是过程进入满意解后下一步脱离满意解集的可能性；另外一个是过程进入满意解时下一步仍不能进入满意解的可能性。这两个参数的匹配构成了遗传算法收敛的一般理论，用这种方法研究收敛性变为纯概率的方法。已有的遗传算法收敛性结果可大致分为四类：

一是马氏链型：将遗传算法的种群迭代序列视为一个有限的马尔可夫链来加以研究。主要运用种群马氏链转移概率矩阵的某些一般性，分析遗传算法的极限行为，但由于所采用的有限状态马氏链理论本身的限制，该模型只能用于描述通常的二进制遗传算法，另外，转移概率的具体形式很难表达。因而，对于较大规模的遗传算法，只能借助于遍历性考察而得Q粗糙的结论。

二是Vose模型：其思想是用两个矩阵算子分别刻画比例选择与组合算子(杂交算子与变异算子的复合)，通过研究这两个算子不动点的存在性与稳定性来刻画遗传算法的渐进行为。在无限种群假设下，Vose模型可精确刻画遗传算法，但解释实际有限种群遗传算法行为的能力相对差一些，而且该模型只适用于简

单遗传算法，对变异、杂交概率随时间变化的情形不易处理。

三是公里化模型：徐宗本等发展了一个既可用于分析时齐又可用于分析非时齐遗传算法的公里化模型，其思想是：通过公里化描述遗传算法的选择算子与重组算子，并利用所引起的参量分析遗传算法的收敛性。

四是连续(积分算子)模型：这是对采用实数编码的连续变量遗传算法的收敛性分析，但分析连续遗传算法的框架与方法均不完善，存在着这样或那样的有限或不足。没有一个好的方法可用于准确描述连续遗传算法的动态行为，并在不强加任何严格的或人为条件下给出相关的收敛性结果。

第三章　遗传算法的若干改进及其应用研究

遗传算法本质上是一种求解的超策略。当采用遗传算法来求解给定的问题时，有必要对其主要组成部分（比如编码方式、重组算子、适应值分配、选择、约束处理等等）进行改进使其获得对给定问题更有效的实现。

遗传算法是一种“弱”方法，它“弱”在对所求的问题要求很少，不需要知道问题的一些数学特性就可以求得所需的解。然而用遗传算法求解问题时，往往面临着如何确定遗传算法的有关参数的问题，如交叉率、变异率、群体容量、代距和迭代次数等，它们的选择基本上没有理论指导。Holland教授提出的简单遗传算法不仅激励人们继续提出很多修正的遗传算法，而且还提供了对遗传算法进行理论分析的基础。然而对于求解实际问题，简单遗传算法的搜索能力是有限的，并不是所有问题都适合使用二进制编码，适应值成比例选择不一定总是最好的方法，简单遗传算子不一定是最有效或最合适的算子。更进一步来说，简单遗传算法忽略了许多生物学中具有潜在用途的概念。Holland教授建议在遗传算法中使用其中的部分概念，但直到最近还没有系统地加以研究。下面，分别介绍遗传算法的若干改进及其应用研究。

3.1 GA的使用范围

文献[25]中已有很多使用遗传算法解决实际问题的成功例子，但也有部分实例说明遗传算法的性能较差。给定一个实际问题，如何判断遗传算法就是一个值得使用的好方法呢?目前还没有严格的答案。许多学者都凭直觉认为，当研究的问题具有如下特征时，使用遗传算法不失为一种好方法:搜索空间较大、非光滑、多峰函数，适应度函数具有躁声或并不需要全局最优解，即只要能快速发现满意解就足够了。然而，这些直觉并不能严格地预测遗传算法何时比其它方法有效，遗传算法的性能在很大程度上依赖于如下因素：如问题解的编码方法、遗传算子、参数设置等。下面对这些因素予以进一步分析。

3.2 GA的编码

就任何搜索和学习方法而言，问题解的编码方法是遗传算法搜索成功的关键因素。遗传算法的大部分应用是采用固定长度、固定次序的位串来表示问题的候选解。然而近年来已有许多人采用了其它种类的改进的编码方法。

3.2.1 二进制编码

二进制编码(即位串)是最常用的编码方法，这主要是历史原因造成的。Holland教授及其学生的早期研究，集中在二进制编码方面，遗传算法的应用也

趋向于采用这种方法。目前有关遗传算法的大部分理论都是基于固定长度、固定次序的二进制编码来假定的。也有许多理论被扩展为采用非二进制编码的方法，但这种扩展并不如基于二进制的方法完善。此外二进制编码还有一些变种，如Gray编码，二倍体等，对某些问题的有关参数设置已经有人做了大量实验，并提出了一些参数指标。尽管具有上 述优点，对某些问题来说，二进制编码仍是不自然和难以使用的(如神经网络权重表示等)，因此它们倾向采用任意排列(Ordering)次序。

3.2.2 多字符及实数编码

对某些问题，最自然的表示方法是采用多字符或实数编码来表示染色体。例如Kian的多字符表示图形产生语法，Montana和Davis的用实数表示神经网络权重等。Holland教授的模板统计参数指出，遗传算法在多字符编码中，将比二进制编码具有较差的性能。这种提法已经引起某些人的质疑，Janikow[26]等人通过实验比较说明，多字符和实数编码的性能要比二进制好，但其性能依赖于所研究的问题以及所采用遗传算法的具体细节。到目前为止，还没有预测哪一种编码方法效果最好的原则。

3.2.3 树编码

树编码模式有若干优点，其中包括搜索空间是开放的。在搜索过程中树的增长(生长)是自由的,这就阻止了形成更具有结构化和层次性的问题解:而且由于生成的树比较大，因而理解和简化它非常困难。在遗传规划领域，学者们才刚刚开始从事树编码的有用性以及与其它编码进行比较的实验研究。同样，把遗传算法理论扩展到树编码的尝试也还处于初期阶段。

对某一问题，如何确定其正确的编码，具有丰富经验的Lawrence Davis学者极力主张，所采用的编码对问题来讲应该是最自然的，并据此设计出能够处理该编码的遗传算法。在比较完善地形成遗传算法和编码理论以前，这也许是最好的原则。目前有些学者正在尝试采用不同的编码来探索他们的遗传算法，而最令人鼓舞的作法是让编码本身适应环境，以便让遗传算法更好地利用它。

3.2.4 自适应编码

事先选择固定编码对潜在的遗传算法用户来说是一个难题。事实上，提出最合适的编码同解决问题本身一样困难。因此，许多用户认为既然要用遗传算法来解决问题,为什么不让它同时调整编码呢?这就是自适应编码的方法。其中，树编码也是一种自适应编码，因为染色体的长度是随着环境的变化而增长或缩短的。

除了上述介绍的自适应方法外，还有一种方法，它不是进化染色体，而是进化交叉。在这种方法中除了正常的编码串外，还有另一个称为交叉模板的串，在模板串中若某位是1，则表示交叉可在此位发生；若为0，则此位不发生交叉。在这种编码中，变异不但作用在染色体上，也作用在模板串上，但只有染色体是用来确定适应值的。遗传算子不但能寻找满意解，它同时还能进化好的交叉模板。Schaefer[27]等人发现这种方法在某些问题上的性能优于简单遗传算法。

3.3 选择的方法

选择是应用遗传算法解决问题要作的第2个决策。选择染色体的目的是为了强调群体中适应性强的个体，并希望其后代也能具有较强的适应性。选择强度应该适中，选择强度太大将会使局部最优个体在群体中占优势，从而使群体的多样性减少，影响群体进化；反之，选择强度太小，将极大地减缓群体的进化过程。

下面介绍几种改进选择方法。

3.3.1 成比例选择

经典遗传算法采用与个体适应值成比例的选择方法，选择个体的数目等于个体适应值除以群体平均适应值。采用比例选择模式时，少数适应值高的个体及其后代将会很快在群体中占据统治地位，这就阻止了遗传算法做进一步搜索，从而形成了成熟前收敛。换句话说，比例选择以牺牲搜索其它空间为代价，过分强调搜索高适应值的个体。当群体中的这些个体非常相似时，就无法进行选择以寻找更好的个体，进化过程就停滞不前。因此进化的速度依赖于群体间适应值的区别。

3.3.2 ∂选择

为改进选择功能，遗传算法学者又提出了几种分割方法(Scaling Method)以使遗传算法对成熟前收敛不太敏感。选择方法就是其中一种(其它方法如线性分割等)。这种选择模式使选择压力保持相对稳定，而不是依赖于群体中适应值的变化。在这种选择模式下，个体(染色体)的期望值是其适应值、群体平均适应值以及群体标准差的函数。

在迭代的初始阶段，适应值的标准偏差比较大，适应值高的个体不会是平均值以上标准差的若干倍；同样，在迭代后期群体接近收敛，标准偏差相对较小，适应值高的个体作用明显，从而使进化继续进行。

3.3.3 杰出者选择

它是由De Jong[7]于1975年提出，其主要思想是强迫遗传算法在每次迭代中保留少数的最好个体。如果不采取这种措施，在迭代中，这些个体可能会由于误差或交叉、变异的影响而丢失。许多学者发现，这种方法极大地改进了遗传算法的性能。

3.3.4 伯茨曼选择

∂选择方式在迭代过程中，保持选择压力相对不变。但实际上，在搜索过程中，不同的时间需要有不同的选择压力。例如，在迭代的初期，最好使各种个体都具有相近的机会来繁殖后代，并使选择能够维持群体较多的多样性；随着迭代的发展，逐步加强选择压力以使适应值高的个体有较多的机会。

伯茨曼选择就可以实现上述设想，在这个选择中，连续变化的温度根据事先安排控制着选择的速率。开始迭代时，温度较高而选择压力较低。在以后的迭代过程中，温度慢慢降低而使选择压力逐步加强，从而使遗传算法的搜索空间逐渐缩小到搜索空间的最好部分，并同时维持群体合理的多样性。采用伯茨曼选择方法，随着T的降低，适应值高和适应值低的个体期望值间的区别就会增加。一般情况下，在搜索过程中，希望温度根据事先安排而缓慢变化。

3.3.5 次序选择

次序选择(Rank Selection)是一种可供选择的方法，该方法的目的也是为了防止搜索过程快速收敛。群体中的个体根据其适应值进行排序，个体的期望值(被选中的次数)取决于它的次序(Rank)而不是其绝对适应值。当适应值变化较大时，排序避免了超级个体在群体中繁殖过多的后代，从而减轻了选择压力；而当适应值变化较小时，它也能保持选择压力。次序为i和i十1的个体，不管其绝对适应值是高还是低，它们的个体期望值的比率都是相同的。

Baker提出的线性排序方法如下：群体中的个体根据其适应值的大小以升序方式排队(从1到群体容量 p)，用户选择具有次序为 p 的个体的期望值 Max，其中 $Max \geqslant 0$。在时刻 t 每一个个体的期望值用以下公式来计算：

$$ExpVal(i,t) = Min + (Max - Min)\frac{rank(i,t)-1}{p-1} \qquad (3.1)$$

式(3.1)中：Min 表示次序为1的个体期望值。

由于 $Max \geqslant 0$，$\sum ExpVal(i,t)$ = 群体容量 p，这就要求 $1 \leqslant Max \leqslant 2$，$Min=2-Max$。

在迭代过程中，首先对每一代群体中的个体进行排序，并根据式(3.1)计算出每一个个体的期望值。

次序选择可能有缺点，因为减轻选择压力意味着，遗传算法在某些情况下寻找高适应值个体的速度会减慢。然而在大多数情况下，采用次序选择保持群体多样性会比比例选择产生较好的搜索结果。

3.3.6 竞争选择

比例选择在具体实现时要经过两个步骤，首先计算群体适应值，其次计算每一个个体的条件期望值；次序选择需要保存排序的群体，它们都是潜在的浪费机时的步骤。竞争选择(Tournament Selection)就其选择压力而言，与次序选择相似，但从计算能力方面来看，竞争选择更有效，更适合于并行计算。其计算过程如下：首先在群体中随机选择两个个体，然后选择一个随机数：$(0 \le r \le 1)$，如果$r < k$ （k是一个参数，例如$k=0.7$），则选择其中较好的个体，否则选择较差的个体，然后把它们放回到群体中，以使其在以后可重新被选择。

3.3.7 稳定状态选择

大多数遗传算法的群体是逐代被全部替换的，即每一代新群体的个体完全是由其父代群体经遗传操作形成的。在某些模式中，例如杰出者模式，其父代群体中的某些个体，被保存到新的群体中。每一代群体中，新产生的个体数与群体数的比值称为代距。在稳定状态选择中，每一代只有一小部分的最差个体被替换，替换的个体是交叉、变异所产生的最好个体。

3.4 其它遗传算子

实现遗传算法的第3个决策是决定选用什么样的遗传算子，这在很大程度上取决于编码策略。以下主要介绍位串编码的交叉和变异算子[29]。

3.4.1 交叉

遗传算法的主要特点就是使用交叉算子。单点交叉算子是最简单的一种算子，其交叉点是随机确定的，配对个体相互交换交叉点后面的部分，从而形成两个新个体，交叉的基本思想是重组不同个体的结构单元。单点交叉存在一些不足，首先它不能组合所有可能的模板，例如，一般情况下它不能组合模板11*****1和****11**以形成模板11**11*1。其次，定义长度长的模板在单点交叉下容易被破坏。Eshelman[28]等人称其为位置倾向性(Positional Bias)，即交叉组合破坏的模板在很大程度上取决于它在染色体位串上的位置。单点交叉假定定义长度短的、低次的模板是位串功能结构单元，但人们事先并不知道哪些位的次序将把这些有联系的位组合在一起，因此出现了倒位算子。Eshelman等人还指出没有任何办法可以把功能上有联系的位放到一个串上，因为某一个

特殊位可能对几个模板来说都是关键的。他进一步指出，单点交叉保留短模板完整的倾向可导致在模板中保存无关的位，即这些位并不是所希望保留的模板的一部分。此外，单点交叉总是包含父辈个体位串的终端部分。

为了减少位置倾向性及位串终端的影响，许多学者采用2点交叉算子。在交叉操作中，仅把配对串中的交叉点内的部分进行交换。2点交叉不易破坏定义长度长的模板，而且比单点交叉可以组合更多的模板；此外，它还不必变换位串的终端部分。然而2点交叉对有些模板也不能进行组合。

遗传算法学者还对不同的交叉点进行了实验。一些研究人员深信参数化均匀交叉的优越性，在这个交叉操作中，位串中的每一位都以概率交换内容。该交叉算子没有位置倾向性，父辈中包含在不同位置的任何模板都有潜在可能被组合到子代个体中。然而正是由于没有位置倾向性这一特点，阻止了自适应等位基因值，因为该算子能够极大地破坏任何模板。

以上介绍了几种交叉算子，在实际应用中究竟应该选用哪一个算子更好？这还没有简单的答案。一个交叉算子的成功与否取决于适应度函数编码方式及其它因素，充分理解它们之间的相互作用仍是一个有待于进一步探讨的问题。尽管有许多文献量化了各种交叉算子的各个方面(如位置倾向性、破坏性、每次迭代产生的模板数量等)，但是都没有给出确定的指导原则，即什么时候采用何种交叉算子较好。此外，还有学者对不同类型的交叉算子的有效性进行了实验研究，但这些实验都依赖于测试函数，不同的研究有时候产生了互相冲突的结果。一般来说，很难找到通用的答案。

3.4.2 变异

一般认为，交叉是遗传算法具有搜索能力的主要算子，位于第一位。而变异是保证位串中任一位的等位基因不会永久不变，位于第二位。遗传算法中的变异算子与其它进化算法中的变异算子不同，在其它进化算法中，变异算子是搜索的主要算子。

然而，随着对遗传算法解决复杂问题的理解，变异算子的作用正逐步被遗传算法的学者所接受。已有学者对交叉和变异算子的搜索能力进行了比较研究。有的研究结果表明交叉是强壮的新模板的构造者，而有的研究认为变异的作用在传统遗传算法中被低估了。然而重要的不是在交叉和变异之间进行选择，而是在交叉、变异和选择之间找到平衡，正确的平衡还依赖于适应度函数和编码等问题。

此外，交叉和变异的有效性随迭代过程而变化，因此找到平衡的最吸引人的前景是寻找一种在搜索过程中能够使遗传算法自我调整交叉率和变异率的方法。

3.4.3 其它算子及匹配策略

尽管大多数遗传算法的文献都采用了交叉和变异算子，也有一些研究人员在探索采用其它算子，如倒位、重复算子以及其它可以用来保持群体多样性的算子，例如De Jong的Crowding算子，该方法是用新产生的个体去取代群体中与其最相似的个体，这样可以防止在群体中同时存在许多相似的个体。Goldberg和Richard采用适应值分享函数(Fitness Sharing Function)也达到了上述目的。其思想大致为：每个个体的适应值由于其它群体个体的存在而减少，其减少量是两个个体相似性的增函数。因此，与其它若干个体相似的个体受到惩罚，而不相同的个体受到奖励。在某种情况下，这可诱导出合适的物种(Speciation)，允许群体中的个体收敛到几个峰值而不是同一个峰值。提高群体多样性的另一种方法是配对受限。例如只有非常相似的个体才允许配对，这将倾向于形成不同的个体。与此相反，还有文献介绍不允许相似个体配对，其目的不是形成物种而是尽可能地保持群体的多样性。Holland和Booker还建议采用配对标记(Mating Tag)，该标记是个体的一部分，用它来识别潜在的配偶。只有具有配对标记的个体才允许配对。原则上，这些标记与个体的其它部分一起自适应地实现适当的配对限制。

3.5 GA的参数

实现遗传算法要做的最后一个决策是如何设置各种参数，如群体大小、交叉率、变异率等。这些参数相互影响，且具有非线性特征，因此不可能分别进行优化。有关参数设置及参数自适应的文献很多，但都还没有结论性的结果。

De Jong系统地研究了参数变化对遗传算法在线及离线性能的影响。所谓在线性能，指所有个体到时刻t时的平均适应值，而离线性能，是指到时刻t时，遗传算法各代最好性能的平均值。实验结果表明，当群体容量为50-100，单点交叉率为0.6左右，变异率为0.001时，遗传算法的性能较好。尽管有时并不清楚遗传算法用这些参数设置在解决其它问题时的运行性能如何，但它们仍在遗传算法的应用领域里被广泛应用。

遗传算法可以作为优化程序来解决实际问题，当然也可用来优化遗传算法的参数设置问题。Grefenstettel[29]通过实验研究了采用遗传算法优化遗传算法参数的问题。他采用元遗传算法优化参数并得到一组在线性能较优的参数：群体容量为30，交叉率Pc=0.95，变异率Pm=0.01，代距G=1，采用杰出者策略。这些参数对De Jong的参数进行了改进，且其在线性能优于De Jong的参数设置。这是很有趣的实验，鉴于测试函数的特殊性，还不清楚他推荐的参数的通用性。而且已有其他人证明，许多适应函数在采用这些参数设置时并不是最优的。

Schaffer等花费了一年多的CPU时间，系统地测试了范围广泛的参数组合。

参数集合的性能是遗传算法采用这些参数时，解决一组数值优化问题的在线性能。测试结果发现:群体大小、交叉率和变异率独立于他们所测试的问题。而且与Grefenstette的结果类似：群体容量为20-30，交叉率为0.75-0.95，变异率为0.005-0.01。

尽管Grefenstette,Schaffer等人根据其实验找到了一些适合于他们测试函数的最好的参数集合，鉴于问题的多样性、编码及性能准则，这些参数设置未必能形成经验(准则)。况且优化的群体容量、交叉率及变异率是随着迭代过程而改变的。因此，许多人认识到最吸引人的方法是使参数值在搜索过程中实时自我调整。

一种自适应遗传算法参数的方法是根据算子对较优个体的贡献程度，对其赋予一个适应值，若一个算子直接或间接产生了很多的个体，则其获得较高的适应值。在开始时，每个算子都具有相同的初始适应值。在给定时间内，每一个操作算子的适应值是其以前的适应值以及其产生的个体而获学分总和的函数。原则上说，动态改变算子的适应值应该跟上与其在不同搜索阶段的算子的实际有效性，以使遗传算法在不同时间以合适的速率来使用它们(算子)。

自适应方法优化参数存在一个问题，即参数自适应速率如何与遗传算法群体自适应速率匹配?参数设置的反馈来源于群体适应度函数值的优劣,但这种信息很难快速地传送到参数设置，以使其停留在与群体当前状态相应的位置。关于衡量不同自适应速率以及它们与不同参数自适应是如何较好配合的问题，应该是一个很重要且值得研究的问题，从而使自适应方法的效果更好。

3.6 求解 TSP 问题的改进遗传算法

3.6.1 TSP 问题

TSP（Travelling Salesman Problem）问题是一个典型的易于描述却难以处理的NP－Hard问题[30]。有效解决TSP问题十分重要，它成为比较各种启发式搜索或优化算法的数值实验性质的一个间接标准。TSP问题描述：设有n个城市的集合 $city=[C_1,C_2,\ldots,C_n]$，对于城市 $C_i,C_j \in city$，从 C_i 到 C_j 的距离记为 $d_{ij} \in R_+$，这里假设 $d_{ij}=d_{ji}$，即考虑对称TSP问题，TSP问题的解就是在集合city中找一个不重复的全排列 $C_{i1},C_{i2},\ldots C_{in}$，使其距离E：

$$E=\sum_{i}^{n} d_{i,i+1} \text{ 最短，其中 } Ci_{n+1}=Ci_1。 \qquad (3.2)$$

当城市数目N较大的时候，会产生所谓的“组合爆炸”问题。如当N=50时，使用每秒计算一亿次的巨型计算机用穷举搜索法计算，所需时间为 $5*10^{48}$ 年。

由于还没有在较快的时间内找到有用解的完美算法，人们将大量精力投入到快速产生近似最优解。

这里，采用本文提出的改进的遗传算法：通过搜索附近最临近的节点，解

决了传统GA中“搜索”能力和对问题解空间覆盖能力之间的相互制约特性，从而既能够搜索广阔的解空间，又有很好的“搜索”性能。

3.6.2 改进的选择算子

为了避免轮盘赌选择法的缺点，并保留它的优点[31]，通过总结并给出一种改进选择算子的方案，具体操作如下：

(1) 在当前种群中，找到适应度最高的个体（多数情况下，不止一个），强制地把他们放入到下一代种群和当前种群的第一类基因库。

(2) 在当前种群中，找到距离适应度最高的个体最远的个体（也可能不止一个），强制地把它们放入当前种群的第二类基因库。

(3) 对当前种群实行轮盘赌选择法，选出的个体放入下一代种群。

算法的过程描述如下：

Begin

Step1:利用适应度计算函数，计算当前群体G中个体的适应度；

Step2:构造G的第一类基因库G_h；

Step3:构造G的第二类基因库G_l；

Step4:构造G的基因库$G_t = G_h \cup G_l$；

Step5:把基因库G_t中的第一类G_h直接放入下一代群体G；

Step6:在群体G/G_h中使用轮盘赌法选择个体，把结果放入下一代群体G；

End

算法过程流程图如图3.1所示：

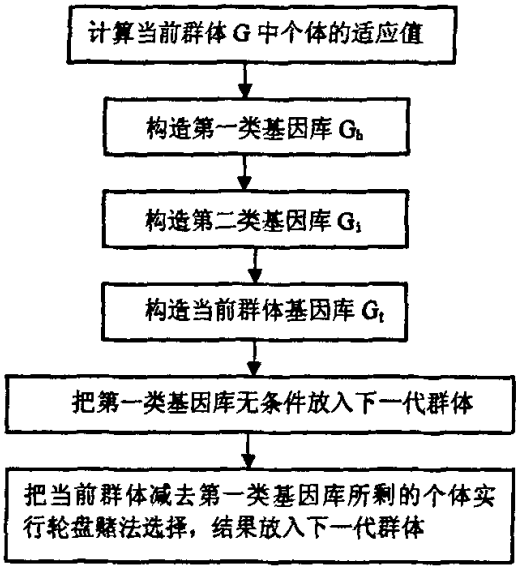

图 3.1 改进选择算子算法流程图

3.6.3 算法设计

求TSP问题的遗传算法的编码有很多种，本文采用路径表示编码，路径表示可能是周游路径的最自然的表示方法，将染色体定义为TSP的一条解路中的城市号序列。

假设有n个城市C_1，C_2，…，C_n，假定都从C_1出发，最后回到C_1，则任意一条染色体可用图3.2的数串来表示，其中N_i=1，2，3，…，n。染色体中的单元称为结点，连在一起的n个结点称为片断。这样表示含义明确，求解距离比较方便。

1 2 3 4 5 6 7 8 9 10 11 12 … N

图 3.2 TSP 问题的路径编码

染色体的适应值：$f(x)=\sum_{i=1}^{n} d_{i,i+1}$。其中从$C_i$到$C_j$的距离记为$d_{ij} \in R_+$。

选择操作：采用上述提出的改进的选择算子。

变异操作：是对一条染色体中的某些城市位置用刚刚随机产生的城市进行插入。

下面以十个城市为例，介绍交叉算子和启发算子：

City_1(3639,1315)　City_2(4177,2244)　City_3(3712,1399)

City_4(3569,1438)　City_5(3757,1187)　City_6(3493,1696)

City_7(3904,1289)　City_8(3488,1535)　City_9(3791,1339)

City_10(3506,1221)

交叉操作过程如下所示：

首先选定两个父代$parent_1$和$parent_2$作为交叉父代；

数组位置 1 2 3 4 5 6 7 8 9 10

$parent_1$：8 6 7 9 3 2 10 1 5 4

$parent_2$：7 2 3 8 10 4 1 5 9 6

产生两个随机数k和i指明$parent_1$和$parent_2$从哪里开始进行交叉；

如：k=4对应着从$parent_1$的城市9，i=4对应着从$parent_2$的城市8。

以$parent_1$的城市9作为条件，从$parent_2$的第一个城市9相同为止。经过查找，找到$parent_2$的第九个城市str[9]=9和$parent_1$的第4个城市str[4]=9相同即$parent_1$.str[4]= $parent_2$.str[9]=9。

以刚产生的随机数i=4为新个体的开始标志：

（1）新个体i=4之前的位置（i=3）用$parent_1$的第4个城市str[4]=9开始之后的城市来填充：

即新个体str[3]=$parent_1$.str[4]=9

str[2]=$parent_1$.str[5]=3

str[1]=$parent_1$.str[6]=2

（2）新个体i=4开始以后的位置用$parent_2$的第9个城市str[9]=9之前的城市

来填充：

即新个体str[4]=$parent_2$.str[8]=5

str[5]=$parent_2$.str[7]=1

str[6]=$parent_2$.str[6]=4

重复执行上述两个步骤，直到后继产生的新个体和前面刚产生的新个体重复为止。

后代str[3]=9；str[4]=5；str[2]=1；str[1]=2；str[6]=4；str[10]=10。

进行如上操作时，可能存在的问题就是新个体产生不完全，如对于新个体str[7]，str[8]，str[9]就不能经过上述方法处理后得到，因此必须补充未完成的新个体，因为用随机产生城市来作为补充，所以这相当于对刚产生的新个体进行了一次变异操作。在接下去的操作中，就不再对新个体进行单独的变异操作。

随机选择填补后辈没有继承下来的节点：例如后代str[9]=6；str[8]=7；str[7]=8交叉后产生的完整后代如下：2 3 9 5 1 4 8 7 6 10。

经过上述交叉操作的同时进行变异操作之后，会发现产生的新个体继承了两个父母中相邻的边，这样有赖于保存好的个体，产生更好的个体。

启发邻接矩阵算子的设计：

受Dijkstra算法的启发，提出一个启发邻接矩阵算子，构思如下：

针对TSP问题而言，把每一个城市的附近城市按邻接程度的大小排列好放在下面的二维数组里，命名为neihgbor[][]。在本事例中，因为只选取了十个城市作示例，所以只保存每一个城市的五个附近城市。表3.1是neihgbor[10][5]数组具体值，最左边是城市号，右边依次是跟当前城市临近且按临近程度从大到小的排列的。

启发算子的目的就是要使每一个城市的后继城市都在它的“邻接表”中即neihgbor[10][5]数组中，改进的过程如下所示：

表 3.1 neihgbor[10][5]数组具体值

City	Neighbor				
1	3	4	9	10	5
2	6	3	9	8	7
3	9	1	4	5	7
4	8	1	3	10	9
5	9	1	7	3	10
6	8	4	3	1	9
7	9	5	3	1	4
8	4	6	3	1	10
9	3	7	1	5	4
10	1	4	5	3	9

(1) 首先找到想要对它进行改进的个体，比如我们将要改进的个体：6 2 9

5 1 4 10 7 3 8。

(2) 随机选出一个开始节点，例如：start=5。它的后继节点为：start' s next=1。

本来的二维“邻接表”neihgbor[10][5]数组中，离城市5最近的城市应该neihgbor[5][1]即城市9才好，但是从路径中可以看出来，城市9已经在城市5前面成为其他城市的后继城市了，如果neihgbor[5][2]也被用了，那么只能依次类推，找到邻近点，即城市1。若neihgbor[5][1]，neihgbor[5][2]，neihgbor[5][3]，neihgbor[5][4]，neihgbor[5][5]都在前面被用过了，那么采用下面的方法。

比如：以结点7为例，结点7就是这样的情况的后继结点9、5、3、1、4已经用在前面的结点位置上了，这时去查找还没有用过的结点，如找到结点2还没有用过，再看结点7是不是结点2的邻近结点，经验证结点7是结点2的临近结点，即neihgbor[2][5]＝7，所以把结点2作为结点7的后继结点，再以结点2开始往后改进。一般来说，出现这样的情况很少，除非保存的临近结点个数很少才会经常出现这种情况。

(3) 出现以下情况时就要进行调整两个城市的位置，即：

dist[n1][n2]+dist[n3][n4]> dist[n1][n3]+dist[n2][n4]时，要把n2,n3的位置进行对调：

6 2 9 5 1 4 10 7 3 8

n_1 n_2 n_3 n_4

如：当前要改进的结点是n_1＝1，n_1的后继结点是n_2＝4，n_3是n_1的邻接表中的结点n＝3，即neihgbor[1][1]＝3，n_4是n_3的后继结点n_4＝8。

因为dist[n1][n2]+dist[n3][n4]>dist[n1][n3]+dist[n2][n4]

dist[1][4]＝141.52384958020326；

dist[3][8]＝262.0534296665472；

dist[1][3]＝111.28791188746656；

dist[4][8]＝126.37246535539299；

dist[1][4]+dist[3][8]＝403.57727934675044；

dist[1][3]+dist[4][8]＝237.66038024285956。

所以要把n_2和n_3位置调换。

经过步骤(3)判断后，最终找到各自的后继结点。结果如下：

1的后继结点是3，即neihgbor[1][1]＝3；2的后继结点是6，即neihgbor[2][1]＝6；3的后继结点是9，即neihgbor[3][1]＝9；4的后继结点是10，即neihgbor[4][4]＝10；5的的后继结点是7，即neihgbor[5][3]＝7后继节点；6的后继结点是8，即neihgbor[6][1]＝8；7的后继结点是2，即neihgbor[9][4]＝5；10的后继结点是1，即neihgbor[10][1]＝1。

对于上面出现的特殊情况：7的后继结点是2，即neihgbor[2][7]＝2，但是结点2却不在结点7的临近结点表中，在步骤二中已经给出解决这类问题的方法。

最后给出已经改进后的结点顺序位置：4 10 1 3 9 5 7 2 6 8。

已经改进后的总长度3091.1445761663044；

改进前的结点顺序位置为：6 2 9 5 1 4 10 7 3 8。

总长度为3605.836545072088。

算法的执行步骤是：

步骤一：初始化随机生成的N个初始群体P（t）个体，t＝0；

步骤二：个体评价。计算群体P（t）中各个体的适应度；

步骤三：进行选择、交叉、变异运算；

步骤四：启发算子运算；

步骤五：终止条件判断。若满足结束条件终止，否则转步骤二。

3.6.4 应用举例：50个城市的TSP问题

50个城市的坐标见表3.2所示。

现在的总长度：550.859874。

表3.2　50个城市的坐标

X	Y	X	Y	X	Y	X	Y	X	Y
10	75	74	98	14	18	15	70	20	20
36	9	91	16	26	80	37	91	39	2
91	78	92	75	49	13	63	54	29	90
54	53	63	73	24	94	46	42	87	46
8	51	95	16	5	92	66	8	62	87
78	51	21	84	66	43	52	28	70	92
73	14	4	23	48	26	61	34	19	74
23	56	41	29	13	33	70	13	32	10
15	40	36	74	11	53	91	55	81	84
26	60	56	15	47	65	36	60	73	90

经过求解后的路径如下：

37 34 19 22 41 1 47 40 20 16 27 7 4 28 8 9 46 30 0 24 23 31 42 15 21 18 39 29 3 33 17 26 35 36 25 32 13 44 45 10 49 48 2 12 38 5 43 14 11 6。

本算例实验，只要3-4秒钟就能很快的得出结果，而且这个结果比较满意，具体结果是550.859874。该结果与561.8531[32]相比，要好的多。

可以看出，本改进算法能够在短时间内得到一个很不错的解，表明遗传算法在解决实际问题时，合适的编码以及恰当的变化算子设计对解决问题是非常重要的，但是在其他方面也作出了一些牺牲，比如在交叉方面，只是继承了父辈中的各自的边，而没有全部考虑双方父母共同的边，因为经调试，如果先找到双方父母中最短的边遗传给后代，这个时间的开销是很大的，而且同时还存

在的问题是会发生丢失，尽管发生这种可能性很小。

3.7 本章小结

自从1975年Holland系统提出遗传算法的完整结构和理论以来，许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、选择和交叉机理等进行了深入的研究，提出了用动态策略和自适应策略以改善遗传算法的性能，提出了各种变形的遗传算法。

总体上，遗传算法的改进途径有下面几个方面：

(1) 改变遗传算法的组成成分或使用技术，如选用优化控制参数、适合问题特定性的编码等；

(2) 采用混合遗传算法；

(3) 采用动态自适应技术，在进化过程中调整算法控制参数和编码粒度；

(4) 采用非标准的遗传操作算子；

(5) 采用并行遗传算法。

本章首先介绍了遗传算法在使用范围、遗传算法的编码、遗传选择的方法、其它遗传算子等方面的改进。在遗传算法的编码方面先介绍了传统的二进制编码，然后给出了多字符与实数编码、树编码、自适应编码等改进的编码方式。选择算子方面，详细介绍了成比例选择、∂选择、杰出者选择、伯茨曼选择、次序选择、竞争选择、稳定状态选择的方法。同时给出了其它遗传算子的改进策略。本章最后给出了解决TSP问题的改进遗传算法，通过实验结果分析，改进的算法有较好的性能。

第四章　遗传多目标优化问题的应用研究

4.1 多目标遗传算法的基本理论

4.1.1 多目标优化问题描述

多目标优化问题也称作向量优化问题、多准则优化问题。多目标优化问题可以描述为：求一个决策变量向量，它满足所有约束并且使得由目标函数组成的向量最优化。而这些组成向量的目标函数通常彼此之间都是互相矛盾的，因此，这里的“优化”意味着求一个解向量使得目标函数向量中的所有目标函数的值满足设计者的要求。

规范化的问题可以描述如下：

求一个决策变量向量 $X^*=[X_1^*,X_2^*,\ldots, X_n^*]^T$，满足m个不等式约束：

$$g_i(x)\geq 0 \qquad i=1,2,\cdots,m \tag{4.1}$$

同时满足P个等式约束：

$$h_j(x)=0 \qquad j=1,2,\cdots,p \tag{4.2}$$

并且使得k个目标函数向量：

$$f(x)=[f_1(x),f_2(x),\ldots, f_k(x)]^T \tag{4.3}$$

最优化。式中 $X=[X_1,X_2,\ldots, X_n]^T$ 为决策变量向量。

对于一个两目标函数而言，就是要在可行域F中确定所有的解集 $X_1^*,X_2^*,\ldots, X_k^*$，使得所有目标函数最优。

4.1.2 Pareto最优的定义

多目标中的最优(Pareto Optimum)，是Vilfredo Pareto在1896年提出的，定义如下：

我们称$X^*\in F$是最优解(即Pareto Optimal Solution)，若 $\forall X\in F$，满足下列条件：

$$\text{要么}\quad f_i(X)=f_i(X^*) \quad i\in I \tag{4.4}$$

要么至少存在一个 $j\in I$，使：

$$f_j(X)>f_j(X^*) \tag{4.5}$$

其中F是满足式（4.1）和式（4.2）的可行解集，即：

$F=\{X\in R_n \mid g_i(x)\geq 0, i=1,2,\ldots, m; h_j(x)=0, j=1,2,\ldots, p\}$

也可以这样来定义Pareto最优解：

若 $X^*\in F$，且不存在 $X\in F$ 使得：

$f_j(X)>f_j(X^*)$，(j=1, 2, …, p)成立，且其中至少一个是严格不等式，则称X^*是多目标问题Pareto最优解。

通常情况下，最优解不只一个，而是一个最优解集(Pareto Optimal

Solutions)。多目标遗传算法的目标是，构造非支配集(non-dominated solutions)，并使非支配集不断逼近最优解集，最终达到最优。

4.2 多目标优化和遗传算法的融合、发展状况及分类

4.2.1 多目标优化问题的产生和发展

最优化处理的是在一堆可能的选择中搜索对于某些问题来说是最优解的问题。如果仅考虑一个目标，就成为单目标优化问题，这种问题在过去50年中已经得到了深入的研究。如果存在的目标超过了一个并需要同时处理，就成为多目标优化问题。多目标优化问题起源于许多实际复杂系统的设计、建模和规划问题，这些系统所在的领域包括工业制造、城市运输、资本预算、水库管理、能量分配等等。几乎每个重要的现实生活中的决策问题都要在考虑不同约束的同时处理若干相互冲突的目标，这些问题大大增加了问题的复杂程度。自20世纪60年代早期以来，多目标优化问题吸引了越来越多不同背景研究人员的注意力。许多学者对多目标优化问题作出了重要贡献。其中Pareto可能是本领域公认的开创者之一。遗传算法作为解决多目标优化问题的新方法受到了相当的关注，这就导致了一类新的研究和应用，称作遗传多目标优化。

国际上一般认为多目标最优化问题最早是由法国经济学家V.Pareto在1896年提出的。1967年，Bagley在他的论文中首次提出了遗传算法这一术语，并讨论了遗传算法在自动博弈中的应用。同年，Rosenberg在其博士论文中曾提出可用基于遗传的搜索算法来求解多目标的优化问题。第一个把遗传算法用于函数优化的是 Hollstien 。1971年他在论文《计算机控制系统中的人工遗传自适应方法》中阐述了遗传算法用于数字反馈控制的方法。直到1985年，Scaffer[33]提出了第一个基于向量评估的多目标遗传算法(VEGA)，从而开创了用遗传算法求解多目标规划的先河。真正引起演化界重视的是1990年后，Fonseca和Fleming提出基于排序选择的多目标遗传算法（MOGA），Srinivas[34]和Deb提出非劣分层遗传算法（NSGA），Horn 和 Nafpliotis也提出了小组决胜遗传算法（NPGA），Hom 等人提出的基于小生镜(Niche)技术的“小生镜Pareto遗传算法”。Goldberg等人提出了一种Pareto排序算法，这些算法已成为遗传多目标优化算法研究的基石。在此基础上，人们又提出可以在多目标算法中引入最优保存策略和约束处理策略，并提出了实现这些策略的不同的多目标进化算法。当前，遗传多目标优化算法[35]作为一个优化工具在解决智能决策问题研究中越来越显示出它的强大生命力，在生产系统中对生产计划调度集成系统的优化，在金融证券和经济预测系统中的模型优化，在决策支持系统中解决模型结构选择和模型实例确定的问题等领域都取得了较显著的成果。

4.2.2 遗传搜索的特征

遗传算法内在的特征说明了为何遗传搜索适合用于多目标优化问题。遗传算法的基本特征是通过在代与代之间维持由潜在解组成的种群来实现多向性和全局搜索。这种从种群到种群的方法在搜索Pareto解时是有用的。

遗传算法不需要许多数学上的必备条件，可以处理所有类型的目标函数和约束。由于算法的进化本质，遗传算法可以在不考虑问题特定内部工作方式的前提下用于搜索解。因此能够用遗传算法求解的复杂问题的数量比能够用传统算法求解的复杂问题的数量大很多。

由于遗传算法作为一种超启发式算法，可以灵活地将传统方法结合进其主框架，以此可以利用遗传算法和传统方法两方面的优势来建立对问题更有效的求解方法。对遗传算法在多目标优化问题中应用不断增长的研究兴趣对数学界提出了巨大的理论和实际上的挑战。

4.2.3 多目标优化与遗传算法的融合、发展

由于遗传算法是对整个群体所进行的进化运算操作，它着眼于个体的集合，而多目标优化问题的Pareto最优解一般也是一个集合，因而可以预计遗传算法是求解多目标优化问题的Pareto最优解集合的一个有效手段。求解多目标优化问题的遗传算法的基本结构与求解单目标优化问题的遗传算法的基本结构相类似。另一方面，在利用遗传算法进行多目标优化问题求解时，需要考虑如何评价Pareto最优解，如何设计适合于多目标优化问题的选择算子、交叉算子、变异算子等问题，所以算法在实现时也有其独特的地方。在算法的实现中，我们可以基于各个子目标函数之间的优化关系进行个体的选择运算；也可以对各个子目标函数进行独立的选择运算；也可以运用小生境技术；还可以把原有的多目标优化问题求解方法与遗传算法相结合构成混合遗传算法。对于具体的应用问题，如何选择哪种方法，仍是取决于对该问题的理解程度及决策人员的偏好。

4.2.4 求解多目标优化问题的遗传算法分类

由于遗传算法是对一个种群并行地进行运算操作，它运行一次能找到多目标优化问题的多个Pareto最优解，因而它是求解多目标优化问题的Pareto最优解几何的一个有效手段。Schaffer首先认识到了这一点，自从他在1984年对多目标遗传算法的开创性工作以来，多目标遗传算法吸引了许多研究者的兴趣，这些研究者都分别提出了自己的算法[36,37,38,39]。概括起来，求解多目标优化问题的遗传算法有以下几种：

1. 权重系数变化法

对于一个多目标优化问题，若给其各个子目标函数$f_i(x)$ $(i= 1,2,\cdots,q)$赋

予不同的权重w_i，（i= 1,2,…,q），其中各w_i的大小代表相应子目标$f_i(x)$在多目标优化问题中的重要程度。则各个子目标函数的线性加权和可表示为：

$$\overline{F}(x)=\sum_{i=1}^{q} wi * fi(x) \qquad (4.6)$$

若以这个线性加权和作为多目标优化问题的评价函数，则多目标优化问题就可以转化为单目标优化问题。权重系数变化法就是在这个评价函数的基础上，对每个个体取不同的权重系数，就可以利用通常的遗传算法来求出多目标优化问题的多个Pareto最优解。

2. 并列选择法

并列选择法的基本思想是：先将群体中的全部个体按子目标函数的数目均等地划分为一些子群体，对每个子群体分配一个子目标函数，各个子目标函数在其相应的子群体中独立地进行选择运算，各自选择出一些适应度较高的个体组成一个新的子群体，然后再将所有这些新生成的子群体合并为一个完整的群体，在这个完整的群体中进行交叉运算和变异运算，从而生成下一代完整的群体，如此这样不断地进行“分割-并列选择-合并”过程，最终可求出多目标优化问题的Pareto最优解。

这种方法很容易产生个别子目标函数的极端最优解，而要找到所有目标函数在某种程度上较好的协调最优解却比较困难。

3. 排序选择法

排序选择法的基本思想是基于“Pareto最优个体”的概念来对群体中的各个体进行排序，依据这个排列次序来进行进化过程中的选择运算，从而使得排在前面的Pareto最优个体将有更多的机会遗传到下一代种群中。如此这样经过一定代数的循环后，最终就可求出多目标优化问题的Pareto最优解。

这里所谓的Pareto最优个体是指群体中这样一个或一些个体，群体中的其它个体都不比它或它们更优越。需要说明的是，在群体进化过程中所产生的Pareto最优个体并不一定就对应于多目标优化问题的Pareto最优解。当然，当遗传算法运行结束时，我们需要取排在前面的几个Pareto最优个体，以它们对应的解来作为多目标优化问题的Pareto最优解。

对群体中的所有个体进行Pareto最优个体排序的算法是：

算法Pareto Individual

(1) 设置初始序号 r←1；

(2) 求出群体中的Pareto最优个体，定义这些个体的序号为r；

(3) 从群体中去掉Pareto最优个体，并更该序号r←r+1；

(4) 转到步(2)，直到处理完群体中的所有个体。

由上述Pareto最优个体排序算法可知，排序选择法仅仅度量了各个个体之间的优越次序，而未度盘各个体的分散程度，所以它易于生成很多相似的Pareto最优解，而难于生成分布较广的Pareto最优解。

4.共享函数法

求解多目标优化问题时，一般希望所得到的解能够尽可能地分散到整个Pareto最优解集合内，而不是集中在其Pareto最优解集合内某一个较小的区域上。为达到这个要求，可以利用小生境遗传算法技术来求解多目标优化问题。这种求解多目标优化问题的方法称为共享函数法，它将共享函数的概念引入求解多目标优化问题的遗传算法中。

在利用通常的遗传算法求解多目标优化问题时，算法并未限制相同个体或类似个体的数量。但当在遗传算法中引入小生境技术后，算法对它们的数量就要加以限制，以便能够产生出种类较多的不同的最优解。对于某一个个体X而言，在它的附近还存在有多少种、多大程度相似的个体，这是可以度量的，这种度量值称之为小生境数(Niche Count)。小生境数有很多种不同的度量计算方法，最流行的小生境技术之一是显式适应性共享：首先种群中的个体被划分成子群，划分的依据是它们基因组的相似程度；接着每个个体的适应性分数都被调整，调整的方法是将自己的适应性分数和自己同属一群的成员共享。这样做就对种群里相似的个体进行了惩罚，因而保持了多样性。适应性共享能够有效地抑制大同小异的基因组的生成，也是使种群保持多样性的极好的方法。

在适应值共享遗传算法中定义一个共享函数表示两个个体之间相似性的度量，可记为 $sh(d_{ij})$，其中 d_{ij} 表示个体之间的距离度量，如个体基因型之间的Hamming距离或个体表现型之间的欧式距离：

$$sh(d_{ij})=\begin{cases}1-(\dfrac{d_{ij}}{\sigma})^{a} & d_{ij}<\sigma\\ 0 & d_{ij}\geq\sigma\end{cases} \tag{4.7}$$

其中 σ 为小生境的半径，a 通常为 1。$sh(d_{ij})=1$ 意味这个体 i 与个体 j 是同一个个体或者它们的距离为零。第 i 个个体的小生境 m_i 为：

$$m_i=\sum_{j=1}^{N}sh(d_{ij}) \tag{4.8}$$

其中 N 是种群中所有个体的数目。m_i 的数值等效表示第 i 个个体周围的个体的数量。较大的 m_i 意味这第 i 个个体周围有较多个体。共享后第 i 个个体的适应度为：

$$f_i'=\frac{f_i}{m_i},\ f_i\text{是第}i\text{个个体共享前的适应值} \tag{4.9}$$

如果在一个小生境中包含太多的个体，该小生境中所有个体的共享后适应值将会大大降低，这样已经长大的物种的进一步生长就会受到它们的尺寸的惩罚而得到约束，而相反，小的物种在进化竞赛中则受到奖励，从而使得其它具有较少个体的小生境得以繁衍。 在计算出各个体的小生境数之后，可以使小生境数较小的个体能够有更多的机会被选中遗传到下一代群体中，即相似个体较少的个体能够有更

多的机会被遗传到下一代群体中，这样也就增加了群体的多样性。

5.混合法

前面所介绍的几种求解多目标优化问题的遗传算法各有各的优缺点。例如，并列选择法易于生成单个目标函数的极端最优解，而较难生成一种多个目标在某种程度上都比较满意的折衷解；共享函数法虽然易于生成分布比较广的Pareto最优解集合，但其搜索效率比较低。如果混合使用上述几种求解多目标优化问题的方法，将有可能尽量地克服各自的缺点，而充分地发挥各自的优点。

4.3 求解多目标作业车间调度问题的改进遗传算法

4.3.1 引言

已有许多文献研究解决作业车间调度问题的进化求解方法，但这些方法多数局限于单目标，因此不能满足现实生活中多目标作业车间调度问题的应用需求。本章提出一种改进的遗传算法启发式地搜索近似最优解，同时优化多目标。采用基于工序的编码方式和插入式贪婪解码方法；选择操作采用轮盘赌选择方式的同时集成精英保留策略和小生境技术，保证了解的收敛性和多样性；交叉操作采用改进的POX交叉算子，变异操作采用位移操作变异方法。仿真实验证明，提出的改进遗传算法可以有效解决多目标作业车间调度优化问题。

遗传算法是基于自然界“适者生存”机制的一种全局优化算法。它将问题的可行解编码为由基因组成的染色体，通过模拟染色体群的选择、交叉和变异等操作，不断迭代，最终收敛到高适应度值的染色体，从而求得问题的最优解或满意解。因此遗传算法具有适合求解多目标优化问题的优点。作业车间调度问题（Job-Shop Scheduling Problem，JSSP）即给定一个工件的集合和一个机器的集合。每台机器同一时间最多可以加工一个工件。每个工件包括一系列工序，每个工序需要在特定的机器上不间断地加工事先给定的时间。研究的目的是确定一个调度，将每个工序分配到对应机器的某个时间段，同时最小化完成所有作业所需的最小加工持续时间。

现实环境中的调度问题通常是多目标优化的，和单目标优化不同，多目标优化问题的最优解不是唯一的，而是数量众多的Pareto最优解集合[40]。目前为止，针对多目标作业调度问题的研究还不多，本章提出基于改进遗传算法来搜索更多的可行调度解，这些解是加工周期和总延误时间的近似最优解。算法既考虑了总加工时间的最小化，使加工时间集中紧凑，又保证各工件交工期的总延误最小。

4.3.2 JSSP 问题理论基础

作业车间调度问题[41]是研究多项不同的事务如何分配作为共享资源的机

器，即研究n个加工顺序不同的工件如何在$m(>2)$台机器上加工完成。

(1) 工件集$J=\{J_1,J_2,\ldots, J_n\}$，$J_i$为第$i$个工件，$i=1,2,\ldots, n$。

(2) 机器集$M=\{M_1, M_2,\ldots,M_m\}$，$M_j$为第j号机器，$j=1,2,\ldots, m$。

(3) 工序序列集$OJ_i=\{OJ_{i1},OJ_{i2},\ldots, OJ_{im}\}$为工件$J_i$的工序序列。$OJ_{ik}$为第$i$个工件在第k 道工序使用的机器号。$OJ_{ik}=0$ 表示工件J_i在第k 道工序不加工，$k=1,2,\ldots, m$。

(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J_i使用第j个机器的时间。当$T_{ij}=0$,表示工件J_i不使用机器j。

在该问题中，同时必须满足有如下约束：

(1) 所有机器在t =0时刻都可用，所有工件在t =0时刻都可被加工；

(2) 每个工件的工序加工先后次序是确定的；

(3) 每台机器在任意时刻最多只能加工一个工序，每个工序的加工一旦开始，就不能中途停止。

4.3.3 JSSP多目标优化问题模型

迄今为止，大多数的作业调度问题都是以优化单目标为目的，即在满足相应约束条件的前提下优化加工周期(Makespan)。然而企业的不同部门分别从自己利益出发对车间调度决策寄予不同的期望：销售部门希望更好地满足对客户承诺的交货期；制造部门希望降低成本、提高工作效率；企业高层则希望尽可能地提高现有资源的利用率等。因此本文提出的算法同时优化加工周期和总延误时间，使得它们同时达到最小。

(1) 加工周期Makespan = max[C_i]，其中C_i表示工件i的完工时间。

(2) 工件的总延误时间为$\sum_{i=1}^{n}\max[0,Li]$，其中L_i表示工件i的延误时间。

4.3.4 基于改进遗传算法的多目标优化方法

求解多目标优化问题的常用方法有权重系数变化法、并列选择法、排序选择法和共享函数法。在事先对各目标之间的关系没有任何了解时，基于Pareto最优解的共享函数思想比较符合实际情况。所以本文基于共享函数法，利用改进GA探讨多目标JSSP的优化方法。在改进的GA内，集成了精英保留策略和基于共享函数的小生境技术，同时改进了POX交叉算子[42]。精英保留策略保证了算法的收敛性，小生境技术和改进的POX交叉算子保证了解的多样性。

改进遗传算法的基本步骤如下：

Step1：对优化对象进行基于工序的编码；

Step2：参数初始化：种群规模N，最大代数T，交叉概率P_c，变异概率P_m；

Step3：种群初始化：随机产生N个个体组成初始种群P(t)；

Step4：解码操作，计算适应度值 $P_i=\frac{f_i}{\sum_{i=1}^{n} f_i}$（$f_i$为个体适应值）；

Step5：采用轮盘赌的选择方法，同时集成精英保留策略和基于共享函数的小生境技术，按规定的种群规模选择个体进入下一代；

Step6：以交叉概率 P_c 按改进的POX交叉算子方式对选中的多对个体交叉；

Step7：以变异概率 P_m 按位移变异算子方式对选中的个体变异；

Step8：如果进化代数 $t \geq T$，得到近似Pareto最优解，结束；否则，转到Step4。

4.3.5 遗传编码和解码设计

将遗传算法应用于JSSP问题，首先应解决编码问题。本文采用MITSUO Gen等人提出的基于工序的编码方法，它具有解码和置换染色体后总能得到可行调度的优点[43]，可完全避免不可行解。这种编码方法中每个工件的工序都用相应的工件序号表示，并根据在染色体中出现的次序加以编译。对于一个n个工件在m台机器加工的调度问题，其染色体由 $n*m$ 个基因组成，每个工件序号只能在染色体中出现m次。从左到右扫描染色体，对于第k次出现的工件序号，表示该工件的第k道工序。交叉和变异等遗传算子也作相应的改进以适应染色体的编码形式。

表 4.1　3 ×3 Job - Shop 调度问题

工件	机器顺序			加工时间		
	工序1	工序2	工序3	工序1	工序2	工序3
J1	1	2	3	4	5	6
J2	3	1	2	5	6	4
J3	2	3	1	4	6	5

表 4.2　3 ×3 Job - Shop 调度解

机器号	工件顺序		
M1	1	2	3
M2	1	3	2
M3	2	1	3

例如，表4.1为一个3×3 Job-Shop问题，设它的一个染色体为[2 1 1 3 1 2 3 3 2]，其中1 表示工件J_1，2 和3 意义类似。染色体中的3个1 表示工件J_1的3个工序，此染色体对应的机器分配为[3 1 2 2 3 1 3 1 2]，每台机器上的工件加工顺序（简称机器码）如表4.2。本文解码过程的算法是应用插入式贪婪解码算法[44]。此算法是从第1 道工序开始，按顺序将每道工序插入到对应机器上最早的空闲时段安排加工，以此方式，直到序列上所有工序都安排在最佳可行的地方。以上的解码过程能保证生成主动调度。基于工序编码的染色体与机器码可互相转换，虽然工序编码方法置换染色体后总能得到可行调度，但不

同染色体可能对应同一机器码，即可能对应相同的调度解。本文用机器码可以转化对应的一种主动调度染色体，用于交叉和变异。

4.3.6 改进的选择操作

选择操作用于实现优胜劣汰，这里采用轮盘赌的选择方法。为了保证优秀个体不会因为交叉变异被破坏，采用精英保留策略，使当前群体中适应度最高的Pareto 解个体不参与交叉和变异，而用它(们)来替换本代群体中经过交叉、变异等遗传操作后所产生的适应度最低的个体。为了保证Pareto 解的多样性，每代进化过程中对适应度非常接近的个体采用小生境技术处理。小生境环境的构造直接影响着多样性的质量，这里改进文献[45]提出的小生境环境设计方法。某个个体所在的小生境域的定义是：以该个体为中心，周围各边界的计算方法如式(4.10)。小生境域密度越小的个体被遗传下去的概率越大。

$$\lambda_{lt}=\frac{\text{maxlt}-\text{minlt}}{\sqrt[r]{P}},l=1,2,\ldots,\ r。\tag{4.10}$$

其中，maxlt和minlt分别是进化到t代时第l个目标的最大值和最小值，r是目标个数，p是种群规模大小。仿真过程中，发现该边界太大，淘汰掉了不应该淘汰的最优解，但是如果边界太小，产生的Pareto解很多，也不方便决策者选择，经过多次仿真[46]比较，修正公式如(4.11)：

$$\lambda_{lt}=\frac{\text{maxlt}-\text{minlt}}{2\sqrt[r]{P}},l=1,2,\ldots,\ r。\tag{4.11}$$

4.3.7 改进的 POX 算子

交叉是GA算法中最关键的操作，决定算法的全局搜索能力。为了在JSSP调度问题中应用GA算法，设计交叉算子时最重要的就是保证子代对父代优良特征的继承性、子代的可行性以及保持种群的多样性。张超勇等[42,47]提出了一种基于工序编码的POX (precedence operation crossover)交叉算子，该算子能很好地继承父代的优良特征并且子代总是可行的，但是该算子没有考虑到多代交叉之后能否保持种群多样性的问题。因为POX算子是保留父代染色体中工件在机器上的位置，使子代继承父代每台机器上的工件次序。但是正是由于保留了部分的父代染色体片断，使得在多代交叉之后染色体可能被还原到某一父代，造成种群的退化现象。

因此本章采用一种改进的POX算子，该算子是在原有POX交叉流程的基础上，增加了一步对染色体基因位的移位步骤，使得每代的染色体基因位在交叉之前都不同与以前的任何一代，从而避免了原有POX算子中染色体可能被还原的问题，将保持种群多样性的问题考虑到了算子之中。假设父代$n*m$ 染色体$Parent_1$和$Parent_2$，经过改进的POX算子生成$Child_1$和$Child_2$，改进的POX交叉的具体流

程如下：

(1) 定义2个存放子代染色体的临时数组变量$Child_1[\ ]$，$Child_2[\ ]$。

(2) 随机从父代染色体$Parent_1$和$Parent_2$中选择一个工件号，将所有该工件号的基因按照其在父代染色体的基因位置复制到子代染色体$Child_1[\ ]$和$Child_2[\ ]$相应位置上。

(3) 对子代临时染色体$Child_1[\ ]$和$Child_2[\ ]$按照随机生成的移位量进行向左和向右移位。

(4) 将$Parent_1$中的剩余工件号按照其在染色体中排列顺序复制到$Child_2[\ ]$空余的位置上；将$Parent_2$中的剩余工件号按照其在染色体中排列顺序复制到$Child_1[\ ]$空余的位置上。

4.3.8 变异算子

当交叉操作产生的后代适应值停止进化且没有达到最优值时，变异操作在一定程度上可以改变算法早熟收敛的现象，有利于增加种群的多样性。常用的变异算子有交换（Swap），重置（Position），位移（Shift）。

本文采用了是位移算子。位移算子变换如图4.1，随机选择两个位置作为位移的起点和终点。

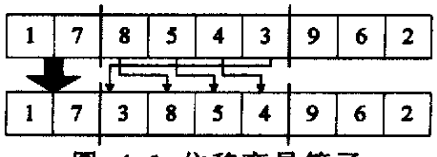

图 4.1 位移变异算子

4.3.9 作业调度问题基准实例和调度结果

为了验证本文设计的IGA 算法的有效性，本文使用Applegate与Cook提出的abz7, abz8, abz9, la21, la24, la25, la27, la29, la38, la40基准问题国际数据库的实验数据，如表4.3所示。分别用标准多目标进化算法(NSGAⅡ)[23]和本文设计的IGA算法进行了仿真实验，并对两算法的性能指标进行了比较，实验调度结果比较如表4.4所示。

两算法选择相同的参数：种群规模为150，染色体为100，交叉概率Pc为0.9，变异概率Pm为0.3。每个基准问题都用不同的初始种群测试30次。得到的每代种群以及非支配解组合在一起形成最终的非支配解。

表 4.3 标准实例问题数据

实例	作业数目	机器数目	最优解
abz7	20	15	654
abz8	20	15	635
abz9	20	15	656
la21	15	10	1040

la24	15	10	935
la25	15	10	977
la27	20	10	1235
la29	20	10	1120
la38	15	15	1184
la40	15	15	1222

表 4.4 实验调度结果比较

实例	算法	加工周期		延误时间	
		计算结果	结果平均值	计算结果	结果平均值
Abz7	IGA	665	697.77	531	676.83
	NSGA II	667	691.19	553	674.28
Abz8	IGA	685	717.78	654	881.32
	NSGA II	686	723.66	757	818.63
Abz9	IGA	694	717.6	1009	1142
	NSGA II	690	720.2	1213	1340
La21	IGA	1046	1085.75	1299	1746
	NSGA II	1046	1089.25	1316	1735.85
La24	IGA	935	1047.5	1249	1408.18
	NSGA II	935	1036.62	1206	1372.14
La25	IGA	977	998.83	1070	1320.5
	NSGA II	977	1016.75	1255	1505.5
La27	IGA	1235	1272.66	3300	4328.12
	NSGA II	1235	1270.25	3436	4540.85
La29	IGA	1156	1191.41	4349	4700.2
	NSGA II	1160	1182.5	4419	4730.5
La38	IGA	1199	1241.88	1342	1519.22
	NSGA II	1196	1261.12	1343	1442.75
La40	IGA	1225	1290.44	573	835.11
	NSGA II	1228	1272.57	535	807.28

4.3.10 算法实验评估和讨论

多目标优化与单目标优化有很多不同之处，对于两个或者多个目标而言，每个目标都对应着不同的最优解。因此，多目标进化算法试图找到所有的均衡面集合。另一个重要的不同点是寻找两个目标的多目标优化搜索到尽可能多样性的Pareto最优解。

由表 4.4 可知,对于规模较大的车间作业调度问题,IGA 算法和标准 NSGA II 均可以得到近似最优解。尤其对于 abz7,abz8 和 la29 问题，IGA 算法具有更好的运行效率，解具有较好的收敛性和多样性。对于 la21, la24, la38 和 la40，本文提出的改进算法得到的解分布更广，因此易于找到更多的极值解。

4.3.11 结论

将遗传算法应用到多目标车间调度问题中，比经典JSSP的单目标调度优化模型更符合车间调度现状。本文提出的改进算法采用轮盘赌选择方式的同时集成精英保留策略和小生境技术，交叉操作采用改进的POX交叉算子，保证了解的收敛性和多样性。仿真实验表明，IGA算法在多目标JSSP问题的应用是有效可行的，算法既考虑了总加工时间的最小化，使加工时间集中紧凑，又保证各工件交工期的总延误最小。而且能够较好地解决一些已有算法中“种群退化”的现象，且具有较快的收敛速度，为求解多目标JSSP问题提供了一种新的方法。目前国内外对该问题的研究较少，很多问题有待进一步研究。

4.4 本章小结

本章首先介绍了多目标优化的基本理论。接着阐述了多目标优化和遗传算法的融合、发展状况及求解多目标优化问题的遗传算法的分类。

目前已经有许多解决作业车间调度问题的进化求解方法，但这些方法多数局限于单目标，因此不能满足现实生活中多目标作业车间调度问题的应用需求。本章最后提出一种改进遗传算法启发式地搜索近似最优解，同时优化多目标。采用基于工序的编码方式和插入式贪婪解码方法；选择操作采用轮盘赌选择方式的同时集成精英保留策略和小生境技术，保证了解的收敛性和多样性；交叉操作采用改进的POX交叉算子，变异操作采用位移操作变异方法。仿真实验证明，提出的改进遗传算法可以有效解决多目标作业车间调度优化问题。

第五章　总结与展望

5.1 遗传算法操作的若干改进的总结

由美国Michigan大学的John Holland提出的遗传算法是一种借鉴生物界自然选择和自然遗传机制，模拟自然进化过程搜索最优解的方法，由无其解决问题以混沌、随机和非线性为典型特征，因而为其它科学技术无法解决或难以解决的问题提供了新的计算模型。

作为一种全局搜索算法，遗传算法的基本框架已经形成，在各种问题的求解个应用中得到了较广泛的应用，然而在应用过程中，也暴露出了其在理论和应用上的一些不足——过早收敛。由于遗传算法中选择及交叉等算子的作用，使得一些优秀的片段基因过早丢失，从而限制了搜索范围，使得搜索只能在局部范围内找到最优解，而不能得到满意的全局最优解。过早收敛在传统的遗传算法中很普遍，而且比较难克服，因此许多学者在关于如何克服过早收敛方面做了许多卓有成效的研究工作[48]。

任何问题的问题空间的编码，只要满足一致类具有稳定性这个充分条件，遗传算法即可收敛到最优解。由于遗传算子是影响遗传算法性能的重要因素，许多研究都集中于遗传算子的改进。吴少岩等以$\{0,1\}^k$空间为讨论对象，通过研究交叉算子同其探索子空间的关系，提出设计良好算子的指导性原则，构造出一种启发式遗传算子，从而大大提高了遗传算法的搜索能力。杨启文等分析了传统变异算子的不足，认为传统的变异算子无法有效地保持同一基因位的多样性，提出用二元变异算子代替传统的变异算子，并讨论了它在克服早熟收敛方面的作用。此外，控制参数主要包括种群规模N、交叉概率P_c、变异概率P_m等。孙艳丰等对自然数编码的遗传算法证明了N的最优值的存在性，给出了N的计算方法。周远军等从控制参数调整出发，提出了多子种群并行进化及自适应参数调整相结合的思想，既保持了进化过程的稳定性，同时又保持了个体的分布多样性，使得GA能开发出更多新的超平面。

李凡等提出了一种基于大变异操作的遗传算法。采用了从种群的不同个体度量种群的多样性和从基因的角度度量种群的多样性这两种种群的度量方法来检测种群的多样性，当检测到种群繁殖很难产生新的个体时，就给出了一个比变异概率大4倍以上的新概率对种群进行一次大变异（称之为大变异操作），以产生新的个体。

5.2 遗传算法发展趋势的展望

GA的研究热潮有其自身的原因，首先它思想独特，与传统的一些优化算法有很大不同；其次就是GA非常一般化，实现起来较简单，什么问题都可以套用，

好象成了一把万能钥匙；第三就是它虽然操作简单，但人们对它的性能却了解得不透，这就给研究者们留下了广阔的探索空间。遗传算法现阶段的研究重点回到了基本理论的开拓和深化以及更通用、有效的操作技术和方法的研究上。下面概述遗传算法现阶段研究课题的几个主要方面：

5.2.1 遗传算法的理论分析

遗传算法自创建以来，在各个领域被广泛应用，但它的理论基础却相当薄弱，目前，只有针对一些特殊的遗传算法进行的理论分析，所以，遗传算法的理论研究无疑对其以后的发展起着非常重要的作用。

5.2.2 优化搜索方向的研究

迄今为止，优化问题的求解仍在遗传算法的研究中占很大比例，并取得了很好的结果。虽然遗传算法比其它传统的搜索方法有更强的鲁棒性，但它更擅长全局搜索，而局部搜索能力并不强。为了改善遗传算法的搜索速度和解的质量，采用与神经网络、模拟退火或专家系统等其它方法相结合的策略正在引起专家们的注意。此外，探明如何能充分发挥遗传算法优越性的问题也是十分有意义的工作。

5.2.3 学习系统的遗传算法的研究

基于遗传算法的机器学习是当前遗传算法研究的一个重要课题。其中，最引人注目的是分类系统的研究。人们期待这种学习系统可以克服专家系统中知识获取的瓶颈问题。因此，基于遗传算法的学习系统当前面临的课题有:①适合遗传算法的高层次知识的表示；②更通用的基于语义的操作；③搜索能力的分析；④更有效地体现和变动与环境相互作用的适应度函数的导入等。

5.2.4 生物进化和遗传算法的研究

随着分子生物学的高速发展，生物的超精密结构和机制正被逐渐探明，遗传算法在吸引这些知识的基础上，其本身从形式到内涵也将受到检验并有可能发生质的飞跃。实际上，从生物学的角度看，现在的遗传算法理论框架基本处于核糖核酸(RNA)的水平，模拟从RNA向DNA(脱氧核糖核酸)的进化过程和机制，从而完善和提高遗传算法的性能是值得注意的研究课题。

5.2.5 遗传算法的并行分布处理

随着遗传算法应用的深入发展，并行分布遗传算法及其实现的研究也变得十分重要。由于遗传算法是群体操作，所以本质上具有很好的并行分布处理特

性，尤其是算法中各个体适应度值的计算可独立进行而彼此间无需任何通信，所以并行处理效率很高。但是，标准的遗传算法除适应度计算外，几乎所有的遗传操作都是建立在全局统计处理的基础上，这意味着在整个进化过程中，这种全局统计处理所引起的通信开销依然是不可忽视的。因此，设计有效的并行遗传算法及相应的实现系统对于遗传算法的理论研究是十分重要而迫切的。

5.2.6 人工生命与遗传算法的研究

近几年来，通过计算机模拟，再现种种生命现象以达到对生命更深刻理解的人工生命的研究正在兴起。尽管目前对人工生命的含义众说不一，但它所涉及的生命现象是很清楚的，其中有生命的起源、自我繁殖、自适应、遗传进化和免疫等。已有不少学者对生命系统的演变、食物链的维持以及免疫系统的进化等用遗传算法作了生动地模拟。但是实现人工生命的手段很多，遗传算法在实现人工生命中的基本地位和能力究竟如何，仍是值得研究的课题。

5.3 全文总结

通过对基本遗传算法的研究，结合现有的一些改进思想的学习，本文对遗传算法提出了在操作算子上的几点可行的改进策略。具体的一些问题的解决中，还要结合特定的问题进行求解，并且通过一些实验和算例验证其是可行的。在对特定的问题的分析中，是否还能寻求更通用的改进策略，有待进一步研究。

在多目标优化问题的研究中，本文介绍了多目标优化算法的基本概念和实现步骤，并探讨了多种运用遗传算法的实现方法并比较了其优缺点，表明了遗传算法解决多目标JSSP优化问题的有效性。在改进的策略中，是否能引入一些新的遗传学的新观点，在实验方针中，能否对多目标的TSP问题，动态的TSP问题等有待进一步的学习和实验研究。

5.4 进一步的工作展望

本文的目的是改进遗传算法，避免“早熟”问题，并对遗传算法在TSP和多目标JSSP问题中的应用作了研究，可是我们的研究结果还只是初步的，还有许多工作可以做。我们将在今后打算继续做以下工作：

(1) 进一步加深对多目标遗传算法的理论分析；

(2) 探讨GA的改进方法，对GA的其他实现方法进行应用，重点研究遗传算法与其他优化方法结合的方法，提高遗传算法的效率；

(3) 就其实质而讲，目前对GA学习人类演化过程还只是形式的，尚未能刻画人类自身的演化过程，更未能刻画神经元思维真实学习过程。因此GA就其模型本身而言需要更加深入的探讨。

参考文献

[1] 王小平，曹立明．遗传算法—理论、应用与软件实现[M]．西安交通大学出版社，2002

[2] 玄光男，程润伟．遗传算法与工程优化[M]．清华大学出版社，2004

[3] 陈国良，王煦法，庄镇泉等．遗传算法及其应用[M]．人民邮电出版社，1996

[4] 史忠植．高级人工智能[M]．科学出版社，1998

[5] 吉根林．遗传算法研究综述[J]．计算机应用与软件，2004，21(2)：69-73

[6] Holland J H. Adaptation in Natural and Artificial Systems [M]. MIT Press, 1975

[7] De Jong. An Analysis of the behavior of a Class of Genetic Adaptive Systems[M]. University of Michigan，1975

[8] Goldberg DE. Genetic Algorithms in search, Optimization, and Machine Learning, Reading[M]. Addison-Wisely，1989

[9] Brindle A. Genetic Algorithms for Function Optimization[M]. Doctoral Dissertation，University of Alberta，1981

[10] Grefenstette, JJ . Genetic Algorithms for the Traveling Salesman Problem[C] . In Proceedings of International Conference on Genetic Algorithms and Their Applications. Lawrence Earlbaum，1985：160-168

[11] Yan Zhen-yu，Kang Li-shan，Chen Yu-ping. A New Multi-Objective Evolutionary Algorithm：Steady Elimination Evolutionary Algorithm[J]. Journal of Wuhan University(Natural Science Edition)，2003，49(1)：33-38

[12] Davis, L. Job Shop Scheduling with Genetic Algorithms[C]. In Proceedings of International Conference on Genetic Algorithms and Their Application. Lawrence Earlbaum，1985：136-140

[13] Lishan S.Kang，Lixin X.Ding. An Orthogonal Multi-objective Evolutionary Algorithm for Multi-objective Optimization Problems with Constraints[J]. Evolutionary Computation. 2004，12(1)：33-36

[14] 杨善林，倪志伟．机器学习与智能决策支持系统[M]．科学出版社，2004

[15] Cartwright HM，Mott GF，Look around：using clues from the data Space to guide genetic algorithm searches[C]. The Fourth ICGA，1991：108-111

[16] 戴晓辉，李敏强，寇纪淞．遗传算法理论研究综述[J]，控制与决策，2000，15(3)：263-268

[17] 孙晓云，蔡远利．利用改进遗传算法的参数估计[J]．自动化技术与应用，2004，23(1)：23-26

[18] Deb K，Goldberg DE. Sufficient Conditions for Deceptive and Easy Binary Functions[J]. Annals of Mathematics and Artificial Intelligence，1994，10(4)：

385- 408

[19] Deb K, Horn J, Goldberg DE. Multimodal deceptive functions[J]. Complex Systems, 1993, 7 (2): 131-153

[20] Deb K, Multi-objective Optimization Using Evolutionary Algorithms [M]. Chichester, U.K. Wiley, 2001

[21] J.Hom, N.Nafploitis, D.E.Goldberg. A niched Pareto genetic algorithm for multi-objective optimization [C]. In Proceedings of the First IEEE Conference on Evolutionary Computation. Z. Michalewiez, Ed. Piscataway, NJ: IEEE Press, 1994: 222-228

[22] Grefenstentte JJ. Optimization of Control Parameters for Genetic Algorithms[J]. IEEE Trans. on Systems, Man, and Cybernetics, 1996, 16(1): 122- 128

[23] Deb, Amritpratap, Sameer Agarwal. A Fastand Elitist Multi-objective Genetic Algorithm : NSGA-II[J]. IEEE transaction on evolutionary Computation. 2002, 6 (8): 116-122

[24] Zitzler E, Deb K, Thielel. Comparison of Multi-objective EA: Empirical Resuls[J]. Evolutionary Computation, 2000, 8(2): 173-195

[25] 贾兆红. 遗传算法及其在知识发现和案例推理中的应用研究[D]. 安徽大学硕士论文, 2003

[26] Janikow CZ, Michalewicz Z. An experimental comparison of binary and floating point representation sin genetic algorithms[C]. In Proceedings of the fourth International Conference on Genetic Algorithms. Morgan Kaufmann, 1991: 31-36

[27] Schafer JD. A Study of Control Parameters Affecting Online Genetic Algorithms for Function Optimization [C]. In Proceeding Performance of the Third International Conference on Genetic Algorithms, 1989: 51-60

[28] Eshelman LJ, Caruana RA, Schafer JD. Bias in the crossover landscape[C]. In Proceedings of the third International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann, 1989: 10-19

[29] Grefenstette JJ. GENESIS : A System for Using Genetic Search Procedures[C]. In Proceedings of the 1984 Conference on Intelligent Systems and Machines, 1984: 161-165

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

[31] 崔俊艺. 改进的遗传算法[D]. 中国科学院数学与系统研究所硕士学位论文, 2000

[32] 鄢烈详. 用队列竞争算法解旅行商问题[J]. 运筹与管理, 1999, 8(3): 24-30

[33] Schafer J D. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms[C]. In Proceedings of lth Congress On Genetic Algorithms and Their Applications. Lawrence Eribaum Associates, 1985: 93-100

[34] N.Srinivas, K.Deb. Multi-objective function optimization using non-dominated sorting genetic algorithm[J]. Evol. Comput, 1995, 2(3): 221-248

[35] T.Ray, K.Tai, C.Seow. An evolutionary algorithm for multi-objective optimization [J]. Eng.Option, 2001, 33(3): 399-424

[36] A.Turkcan, M.S. Akturk. A problem space genetic algorithm in multi-objective optimization[J]. Journal of Intelligent Manufacturing, 2003,14(3): 26-29

[37] M.A.Abido. A novel multi-objective evolutionary algorithm for environmental and economic power dispatch[J]. Electric Power Systems Research, 2003, 65(1) : 44-48

[38] T Hanne . Global Multi-objective Optimization Using Evolutionary Algorithms[J], Journal of Heuristics, 2000, 6(3): 21-24

[39] K.C.Tan, T.H.Lee, E.F.Khor. Evolutionary Algorithms With Dynamic Population Size and Local Exploration for Multi-objective Optimization[J]. IEEE transaction on Evolutionary Computation, 2001, 5(6): 25-28

[40] 赖红松,董品杰,祝国瑞.求解多目标规划问题的Pareto多目标遗传算[J].系统工程, 2003, 21(5): 11-13

[41] Chen Xiong, Kong Qingsheng, Wu Qidi. Hybird algorithm for Job-shop scheduling problem[C]. In Procceding of the 4th Congress on Intelligent Control and Automation, Shanghai: East China Univ.of S&T Press, 2002: 1739 – 1743

[42] 张超勇,饶运清,李培根.求解作业车间调度问题的一种改进遗传算[J].计算机集成制造系统, 2004, 10 (8): 966-970

[43] Mitsou G, Yasuhieo T. Solving Job–shop scheduling problems by genetic algorithm[C]. In Proceedings of the 1995 IEEE International Conference on Systems, Man, and Cybernetics, Vancouver: Institute of Electrical and Electronics Engineers ,1995: 1577 – 1582

[44] Baker K R. Introduction to sequencing and scheduling [M]. NewYork: Wisely, 1974

[45] Hyun CJ, KIM YK. A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines[J]. Computers & Operations Research, 199p8, 25 (7/ 8): 675 – 690

[46] 吴秀丽,孙树栋,余建军.多目标柔性作业车间调度优化研究[J].计算机集成制造系统, 2006, 12 (5): 731-736

[47] 许小栋，李从心．免疫遗传算法在车间作业调度中的应用[J]．东南大学学报，2006, 36 (3)：437-441
[48] 孙艳丰，王众托．遗传算法在优化问题中的应用研究进展[J]．控制与决策，1996，11(4)：425-431