广东工业大学

硕士学位论文

进化策略的研究及应用

姓名：陈宝财

申请学位级别：硕士

专业：计算机应用技术

指导教师：吴伟民

20100501

摘 要

进化策略是一种模拟自然界进化规律以解决参数优化问题的典型的进化算法，作为一种新型的优化技术，弥补了传统优化技术的不足。在科学研究、生产实践中许多复杂的计算问题都可以转化为函数优化问题，进化策略在解决这类问题时表现出了比传统的优化算法更好的性能，成为研究热点。

进化策略由于自身固有的缺陷，存在着收敛速度较慢、容易早熟的问题。本文从种群划分和变异策略两个方面对进化策略进行改进，提出了一种基于双种群的改进进化策略(MES)算法。将种群划分为规模较小的精英子群和规模较大的普通子群。精英子群用于存放种群中最优秀的个体，普通子群用于存放种群中的普通个体。对不同的子群采用不同的变异策略，精英子群采用递减的高斯变异算子，普通子群采用柯西变异算子，实现种群在解空间具有较好的全局搜索能力的同时在局部具有尽可能精细的局部搜索能力。通过对算法进行理论分析说明其正确性。对典型的测试函数应用该算法进行模拟进化实验，说明对于中低维函数(30 维以下)，MES 算法具有良好的性能，对于高维复杂问题，MES 算法性能不佳。

针对 MES 算法在处理高维复杂问题时存在易陷入局部极值点和收敛能力欠佳的不足，本文借鉴协同进化的思想，在 MES 算法的基础上，提出了一种基于合作型协同进化的改进进化策略(MESCC)算法，适用于求解高维可分解问题。该算法将目标问题分解为一系列相关的子问题，将所有个体划分为一系列团队，每个团队负责处理一个子问题。不同的团队之间采用协作操作。团队的进化和团队间的合作是交替进行的，直到进化得到目标问题的最优解。通过典型的高维测试函数进行模拟进化实验，表明求解高维可分解问题时 MESCC 算法具有良好的性能。

最后，本文将 MES 算法应用到 K-means 聚类算法当中。提出一种基于改进进化策略（MES 算法）的 K-means 聚类(KAMES)算法，利用了 MES 算法良好的全局搜索能力和 K-means 聚类算法良好的局部搜索能力，提高收敛速度，达到较好的聚类效果。最后通过鸢尾花卉数据集对该算法的性能进行测试，说明该算法具有良好的聚类性能。

关键词：进化策略；MES 算法；协同进化；K-means 算法；双种群

ABSTRACT

Evolution strategies is a typical evolutionary algorithm that simulates the natural laws of evolution to solve parameter optimization problems.As a new optimization technology, evolution strategy make up the deficiency of traditional optimization techniques. But evolution strategy has problems of premature convergence and slow convergence speed.This article improves evolution strategies from two aspects, population division and mutation strategy.And a modified evolutionary strategies (MES) algorithm based on bi-group is proposed. In the new algorithm, the group is divided into two sub-groups, general sub-group and elite sub-group. The size of general sub-group is larger than the size of elite sub-group, in which the best individuals are stored. Evolution of the two sub-groups is parallel performed with different mutation strategies respectively, and then the group can not only explore the solution space separately, but also search the local part in detail. Performance of this algorithm is analyzed in theory. Experimental results demonstrate that for functions whose dimensions is less than 30,the MES algorithm is more efficient to improve convergence speed and avoid premature convergence than classical evolution strategies.But for high dimensional complex issues, the MES algorithm has poor performance.

Aimed at the problems of premature convergence and slow convergence speed of MES algorithm for high dimensional complex issues, basing on the MES algorithm, a modified evolutionary strategy with cooperative coevolution (MESCC) algorithm is proposed, which is suitable for solving high dimensional complex issues. The algorithm divides problem into a series of related sub-issues, each team is responsible for dealing with a sub-problem. Team evolution and cooperation among team alternating until the evolution obtained with the optimal solution. Experimental results demonstrate that MESCC algorithm has good performance for high dimensional complex issues.

The MES algorithm is applied to the K-means algorithm, a K-means algorithm based on modified evolutionary strategies (KAMES) is proposed. KAMES algorithm uses the global search capability of the MES algorithm and the local search capability of the

K-means algorithm to improve the convergence rate and achieve better clustering effect. Experimental results demonstrate that KAMES algorithm has good performance for Iris data.

Keywords: Evolutionary Strategies, MES Algorithm, Coevolution, K-means Algorithm, Bi-group

第一章 绪 论

1.1 进化算法概述

人们在科学研究、生产实践中面对着大量的复杂的计算问题，由于这些问题所表现出来的高度非线性、不连续、不可微、不可导、复杂度非常高，传统的优化算法，比如基于导数的方法，在处理这些问题上遇到了极大的困难，进化算法的出现为解决这类问题提供了新的思路，注入了新的动力，开辟了一条崭新的道路。存在早熟和收敛速度慢是进化算法需要解决的主要问题[1]。

进化算法(Evolutionary Algorithms, EA)是模拟自然界生物进化过程与机制的一类自组织、自适应人工智能技术，是一种宏观意义下的仿生优化算法。通常认为进化算法由四个部分组成[2-4]:

(1)Holland 提出的遗传算法(Genetic Algorithm, GA)。

(2)Rechenber 和 Schwefel 提出的进化策略(Evolutionary Strategies, ES)。

(3)Fogel 提出的进化规划(Evolutionary Programming, EP)。

(4)Koza 提出的遗传规划(Genetic Programming, GP)。

进化算法作为一个新兴的交叉学科已经成为了人工智能领域中的一个重要分支[5]。

1.1.1 进化算法的发展

进化算法可以追溯到二十世纪三十年代。1932 年，Cannon 通过仿真生物进化过程进行机器学习的研究[6]。1958 年, Bremermann 开始了模拟生物进化的优化计算试验，首先将进化思想用于计算机解决数值优化问题[7]。

二十世纪五十年代到七十年代是现代进化算法的形成和发展的时期。Holland, Fogel, Rochenberg 和 Schwefel 等人分别为遗传算法、进化规划和进化策略的建立做出了开创性的贡献。

60 年代初，美国 Michigan 大学的 J.H.Holland 教授和他的学生在从事如何建立机器学习的研究时提出系统本身和外部环境相互协调的遗传算法[4,8]。1967 年 Holland 的学生 J.D.Bagley 在其博士论文中首次提出了"遗传算法"一词[9]。1975 年，

Holland 教授的专著《Adaptationin Natural and Artificial Systems》出版，全面地介绍了遗传算法，提出了“模式定理（SchemaTheorem）”，为遗传算法奠定了理论基础[10,11]。

60 年代中期美国的 L.J.Fogel 在人工智能研究中，为有限状态机的演化而提出一种随机优化方法，称之为进化规划[12]。1966 年，Fogel 的著作《Artificial intelligence through simulated evolution》正式出版，系统阐述了进化规划的基本原理[13]。

60 年代初德国柏林工业大学的 I.Rechenberg 和 H.P.Schwefel 在研究风洞流体力学问题时，借鉴自然突变和自然选择的生物进化思想，提出了一种典型的优化算法，称之为进化策略[14,15]，并成功应用于其他实值优化问题。

90 年代初美国斯坦福大学的 John R.Koza 提出了遗传规划[16-18]。1992 年，他的专著《Genetic Programming : On the Programming of Computer by Means of Natural Seleciont》出版，全面介绍了遗传规划的原理及应用实例[17]。1994 年，他的另一部专著《Genetic Programming II: Automatic Discovery of Reusable Programs》出版，提出自动定义函数的概念，在遗传规划中引入子程序的新技术[18]。

遗传算法、进化规划和进化策略都是在六七十年代发展起来，并在各自的领域获得成功。虽然这三种算法都借鉴了自然界的生物进化思想，面对数值优化问题也极其相似，但是在早期发展过程中，彼此独立发展，直到二十世纪九十年代才有了相互的交流，从而形成进化计算[19]。随着一系列关于进化计算的专业学术刊物如《进化计算》(Evolutionary Computation,MIT Press),《关于进化计算电学学会学报》(IEEE Transaction on Evolutionary Computation, IEEE)等相继创刊和国际进化计算、遗传算法基础会议等学术活动的举行，有力地推动了进化计算的研究和发展，使进化计算成为 90 年代以来关于人工智能领域的热点课题之一[1,3]。

1.1.2 进化算法的特点

进化算法具有较强的鲁棒性，能适应于不同的环境、不同的问题领域，并且在多数情况下都可以得到满意解或最优解。从实质而言，进化算法是一种具有自适应调节功能的随机化搜索方法，从随机生成的初始解出发，通过不断迭代，借助复制、交换（重组）、突变等遗传操作，逐步逼近所研究问题的最优解[3,20]。与其他搜索技术相比，进化算法具有以下特点[3,11,20]：

1.有指导搜索。进化算法进行搜索时用到的是目标函数的信息而不必采用目标函数的导数信息或其它与待求解问题有关的特殊领域的知识。

2.自适应搜索。在进行搜索时用到的是自适应的变换规则，而不是确定性的规则。

3.并行搜索。进化算法是从一组点集开始搜索，而不是从某个单一的初始点开始搜索。

4.具有较强的鲁棒性。即在存在噪声的情况下，对同一问题的进化算法的多次求解中可以得到相似的结果。

5.通用性强。进化算法可以用来解决各种类型的优化、搜索与学习等问题，能够运用在结构化优化、复杂问题的优化、复杂系统分析等领域。

1.2 进化策略的演变

进化策略是一类模拟自然界进化规律以解决参数优化问题的典型的进化算法。它是由德国科学家 Rechenberg，Schwefel 和 Bienert 于十九世纪六十年代提出的[21]。当初主要是用于处理流体动力学问题等一类实值参数优化问题。Rechenberg 借鉴生物进化的思想，将生物变异、重组、选择等应用到参数设计中[22]。Schwefel 首先将 Rechenberg 的算法付诸实践，取得了良好的效果[19]。Bienert 将进化思想应用到具有工业意义的实际问题上，将理论与实践相结合[19]。

进化策略经历了不同的演变过程[4,20]：早期的进化策略仅仅使用一个个体，进化操作只有变异一种，这是最简单的进化策略，称为(1+1)-ES 或二元进化策略。随后，Rechenberg 又提出了多群体下的进化策略算法(μ +1)-ES，这种策略不仅含有变异算子，而且增加了重组算子。1975 年，Schewel 首先提出了(μ+λ)-ES，随后又提出了(μ,λ)-ES，这两种策略都采用重组、突变、选择三种算子，在较前两种策略在性能方面有了很大提高，在进化策略领域占有重要的地位。

1.2.1 (1+1)-ES

早期的进化策略基于单个个体的，仅含一个变异算子，每次迭代只有一个旧的个体(父代个体)产生一个新的个体(子代个体)，选择只发生在这两个体。二元进化策略可以描述如下：

1．初始化种群：在问题的可行解空间中随机生成 1 个个体，构成第 1 代群体

的父代。

2．变异：父代个体变异产生 1 个子代，种群数为 2。新生成的后代个体在基因上与父代个体不同。这种差异具有随机性。

3．选择：父代个体与子代个体的基因不同，因此对环境的适应性也不同。计算 2 个个体的适应值。将适应度欠佳的个体舍弃，适应度较佳的个体作为下一代的父代个体。如果终止条件满足，则算法结束。否则回到步骤(2)。

最初的变异是阶梯式的变量调整，将一个满足二项分布的变异作用在初始个体上，这样得到一个新的个体。1965，Schwefel 研究发现进化过程会出现停滞现象， 当进化过程进入到某一个极值点时，需要较大的变异才能跳出局部最优，而二项分布很难得到这种大粒度的变异，导致了高斯变异算子的产生[1]。

进化策略中的个体采用十进制实数表示，变异算子表示为：

$$Z^{t+1} = Z^t + \sigma * N(0,1) \qquad (1.1)$$

其中，$Z = (z_1, z_2, \ldots, z_n)$，每个分量 z_i，i=1,2,…,n 是各自独立的随机数。Z^t 表示第 t 代个体的数值。$\sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$ 是高斯分布随机变量的标准差，又称变异步长、策略参数。$N(0,1)$表示服从正态分布的随机数，其均值为 0，标准差为 1。

进化策略的新个体 Z^{t+1} 是在旧个体 Z^t 的基础上添加一个独立的随机变量 $\sigma * N(0,1)$。当新个体的适应度优于旧个体，则用新个体代替旧个体，否则舍弃。经过变异产生的新个体与旧个体差别不大，这与生物进化的基本状况相吻合：微观进化通常比宏观进化更频繁。

为了控制收敛速度，提高搜索效率，Rechenberg 用两个基本目标函数——超球模型(hypersphere)和走廊模型(rectangular corridor model)来研究(1+1)-ES 的收敛速度，提出了著名的 1/5 成功定律[23]。它奠定了进化策略的理论基础[24]。一般地，在 5 次变异中有一次变异对目标函数的值有改进或提高，则达到最佳的收敛速度，亦即当成功变异的概率约为 1/5 时相应的变异强度是较优的。用 ϕ 表示变异次数中成功变异的比率。Rechenberg 认为，当 ϕ 应为 1/5，如果 ϕ 大于 1/5，适当加大变异步长 σ，如果 ϕ 小于 1/5，适当减小 σ。

$$\sigma^{t+1} = \begin{cases} C_d \times \sigma^t & \text{若 } \phi < 1.5 \\ C_i \times \sigma^t & \text{若 } \phi > 1.5 \\ \sigma^t & \text{若 } \phi = 1.5 \end{cases} \qquad (1.2)$$

其中，ϕ 是最近 w 代中成功变异的次数与总变异次数 w 之比，通常 w>10。

C_d为小于 1 的系数，C_i为大于 1 的系数。Schwefel 在实际运算中建议，C_d= 0.82，C_i = 1.22 = 1 / 0.82。

(1+1)-ES 仅仅用一个个体，变异是通过用独立的随机变量来改变或修正就个体，提高个体性能，它既不涉及重组，也不包括自适应。

1.2.2 (μ+1)-ES

由(1+1)-ES 策略的原理可知，它没有体现出群体的作用，仅仅是单个个体在进化，因而具有一定的局限性。1975 年，Rochenberg 提出了多群体下的进化策略算法(μ +1)-ES。在这种进化策略中，父代有 μ (μ >1)个个体，又引进了重组(Recombination)算子,使父代个体可以重组出新的个体。在执行重组操作时，从 μ 个父代个体中随机选择两个个体：

$$(Z^1,\sigma^1)=((z_1^1,z_2^1,...,z_n^1),(\sigma_1^1,\sigma_2^1,...,\sigma_n^1))$$

$$(Z^2,\sigma^2)=((z_1^2,z_2^2,...,z_n^2),(\sigma_1^2,\sigma_2^2,...,\sigma_n^2))$$

再将这两个个体重组出新的个体：

$$(Z,\sigma)=((z_1^{t_1},z_2^{t_2},...,z_n^{t_n}),(\sigma_1^{t_1},\sigma_2^{t_2},...,\sigma_n^{t_n}))$$

其中 t_i=1 或 2，i = 1, 2, ..., n。

对产生的新个体进行变异操作，变异的方式和σ 的调整方式与二元进化策略相同。突变后的新个体与 μ 个父代个体中最差的一个进行比较，若优于父代最差个体，则用新个体代替最差个体，否则淘汰新个体。

(μ +1)-ES 和(1+1)-ES 都只产生一个新的个体。(μ +1)-ES 比(1+1)-ES 有了明显的改进，它的特点如下：

(1)体现了群体的概念，包含了 μ 个个体。

(2)引入重组算子，它相当于遗传算法中的交换算子，可以从父代继续信息生成新的个体。

尽管(μ +1)-ES 由于由于变异算子缺乏自适应机制而显得先天不足等原因没有得到广泛使用，但这是第一个基于种群的进化策略，常用于多处理机下的异步并行计算，并引入了交叉算子，为促进进化策略的发展奠定了基础[1]。

1.2.3 (μ , λ)-ES 和(μ + λ)-ES

与其他经典的优化方法一样，算法内部参数对 ES 的性能有很大影响。鉴于(μ

+1)-ES 存在减小变异步长的内在趋势，Schwefel 在 1975 年提出了(μ+λ)-ES，随后又提出了(μ,λ)-ES[25]。(μ+λ)-ES 和(μ,λ)-ES 都由 μ 个个体组成父代群体，通过重组和变异操作产生 λ 个子代个体。这两种进化策略的差别在于下一代群体的组成上。(μ+λ)-ES 是从 μ 个父代个体和新生成的 λ 个子代个体中(共 $\mu+\lambda$ 个个体)选择出最优的 μ 个个体作为下一代群体，属于精英保留型。而(μ,λ)-ES 仅仅是从新生成的 λ 个子代个体中选择出最优的 μ 个个体作为下一代群体，这时要求 $\lambda>\mu$。(μ,λ)-ES 只是从子代中进行选择，每一个体的生命周期仅限于一代，无论其父代个体多么优秀，都被“抛弃“。

(μ+λ)-ES 和(μ,λ)-ES 都是采用了变异、重组、选择算子，其中重组算子类似于(μ+1)-ES，而变异算子有了新的发展，标准差 σ 采用自适应调整，而不是固定常数，也不是使用 1/5 成功法则的确定型。标准差 σ 的调整方式如下：

$$\sigma_i' = \sigma_i \cdot \exp(\tau_1 \cdot N(0,1) + \tau_2 \cdot N_i(0,1)) \qquad (1.3)$$

其中，N(0,1)表示具有期望值为 0，标准方差为 1 的正态分布随机变量。τ_1 是标准向量 σ 的整体步长参数，τ_2 是 σ 的每个分量 σ_i 的步长参数。

(μ+λ)-ES 和(μ,λ)-ES 算法可以描述如下：

1．初始化种群：在问题的可行解空间中随机生成 μ 个个体，构成第 1 代群体的父代。

2. 重组和变异：每个父代个体经过重组和变异产生 λ/μ 个子代，总共生成 λ 个子代个体。新生成的后代个体在基因上与父代个体不同。这种差异具有随机性。

3．选择：在 λ 个个体中((μ,λ)-ES 或者 μ+λ 个个体中((μ+λ)-ES)选择适应度较强的 μ 个个体成为下一代的父辈个体。如果终止条件满足，则算法结束。否则回到步骤(2)。

(μ+λ)-ES 和(μ,λ)-ES 的特点如下：

(1)父代生成 λ 个子代个体，而不像(μ+1)-ES 和(1+1)-ES 都只产生一个新的个体。

(2)引进了自适应机制。把变异步长作为内在策略参数，而非外部控制，可以在进化的过程中自适应调节变异步长。

(3) (μ+λ)-ES 较好地继续了父代的优良特性，但容易陷入局部最优，适用于参数优化等问题。而(μ,λ)-ES 由于放弃了对上一代的继承，容易跳出局部最优，但收敛速度慢，适用与组合优化问题。

(μ+λ)-ES 和(μ,λ)-ES 在进化策略中占据了重要的地位，并且得到广泛的应用，并取得了巨大的成功。

1.3 进化策略的特点

进化策略是一种模拟自然进化规律的全局随机搜索算法。首先随机生成初始种群，在重组、变异和选择算子的作用下，在迭代过程中逐步改进当前群体中个体的适应值，直到得到所求问题的最优解或满意解为止。

进化策略具有以下特点：

(1)进化策略是最早引入自适应机制的算法[25]。

(2)进化策略构造简单、易于实现[19]。

(3)进化策略具有典型的动力学特征。特别是在 ONE-MAX(1+1)问题上，已经有了肯定的结论[19]。

(4)在进化策略中，目标参数和策略参数都需要编码到染色体中[11]。

(5)变异算子是进化策略的主要算子。在经典进化策略中变异是通过给目标参数上加上个服从正态分布的随机数来实现的。

(6)重组操作是提供两个不同个体之间交换信息的机制。在进化策略中重组算子并不是必需的。

(7)进化策略中的选择方式是完全确定性的。 (1+1)-ES 从 2 个个体中选择出最优的 1 个个体。(μ +1)-ES 和(μ+λ)-ES 从 μ 个父代和 1 或 λ 个子代中选择最优的 μ 个个体作为下一代的父代。而(μ,λ)-ES 是从 λ 个子代个体中选择 μ 个最好的个体组成下一代种群。

1.4 进化策略的研究现状

1.4.1 进化策略的应用

进化策略是一种新型的优化技术，弥补了传统优化技术的不足。最初用于连续函数优化，并表现出较好的性能[1]。现在进化策略在多目标优化[26] 、离散组合优化[27]、数值计算[28]、约束优化[29]、含噪声的优化[30]、模糊控制[31]、并行计算[32]、图像处理[33]和神经网络[34]等方面都得到应用。随之时间的推移，进化策略将会被运用到更多的领域。

1.4.2 进化策略的改进研究

进化策略从(1+1)-ES 演变到 (μ,λ)-ES 和$(\mu+\lambda)$-ES 的演变过程伴随着进化策略的改进过程，除此之外，很多学者从其他方面对 ES 进行了改进研究。Kursawe[35]研究了重组对收敛速度、收敛可靠性的影响。Gruenz[36]用球模型函数试验研究了有σ自适应和无自适应。Matsumura 等[37]通过对离散重组和中间重组作了大量试验，表明对在大多数情况下，目标变量和策略参数同时使用离散重组，可以获得鲁棒性，但重组算子的作用机理严重缺乏理论指导。

变异算子是进化策略的主要算子，对变异算子的改进是 ES 的主要研究内容之一。Kappler[38]和 Yao[39]受快速模拟退火算法的启发，分布用柯西变异算子代替(1+1)-ES 和(μ,λ)-ES 中的高斯变异算子，利用了柯西变异较强的分散能力来获得较好的跳出局部极值点的能力，从而具有较强的全局搜索能力，但收敛速度有所降低。林丹[40]，Gunter[41]通过对高斯变异、柯西变异和均匀变异(uniform mutation)的局部和全局搜索能力进行理论分析，表明高斯变异算子具有较强的局部搜索能力，而均匀变异算子和柯西变异算子具有较强的局部逃逸能力。王云诚等[42]试验了用均匀变异算子来求解单峰函数，研究了其步长控制策略。刘若辰[43]等用基于细胞克隆选择学说的克隆算子和 Cauchy 变异算子改进了进化策略。Change Ming 等[44]研究了在进化策略中使用组合算子的效果，实验结果表明组合算子对单模态函数的优化性能有改进，而对多模态的函数的效果并不显著。王战权等[45]试验使用对数柯西分布代替高斯分布。Lee 等[46]提出通过强化学习实现步长的自适应。Liang 等[47]提出二方式变异法(two-way mutation)，通过试探，调节变异方向和变异步长，取得了良好的效果。王湘中等[48]提出了适用于高维优化问题的改进进化策略，建立了单基因变异与均匀变异相结合、使用精英繁殖、递减型策略参数、小种群规模的$(\mu+\lambda+k)$-ES，该算法特别对于高维优化问题效果显著。边增远[19]提出了一种基于极坐标方向寻优的进化策略算法，该算法能够有效地继承父代的优点，能够得到更快、更优的收敛结果。

在进化策略中，种群规模对 ES 的多样性具有直接的影响。夏慧明等[49]提出了双种群进化策略(Bi-group Evolution Strategies，BES)，将种群划分为规模相同的两个子群，对一子群采用高斯变异算子，对另一子群采用柯西变异算子。Schonemann[50]研究了在动态环境下，种群规模的自适应方法及对种群多样性的影

响。

1.4.3 进化策略的理论研究

与遗传算法相比较，进化策略的理论研究比较薄弱，由于进化策略本身的复制性，理论研究变得很复杂，很多理论成果使用简化的对象模型(比如球型模型)和简化的算法模型，普遍性受到怀疑[1]。这些都增加了理论研究的难度。

Rechenberg 在研究解决多模态适应值曲面问题时，给出了如下公式[51]：

$$\forall \alpha,\beta \qquad \|w-y_\alpha\| \le \|w-y_\beta\| \Leftrightarrow F(y_\alpha) \le F(y_\beta) \tag{1.4}$$

其中，w 为单调最小吸引子，F(.)表示应用值。Rechenberg 认为满足公式(1.4)的问题，就可以通过进化策略的作用搜寻到全局最优解。该结论虽然在实际应用中得到验证，但理论上这一问题尚未得到解决。

进化过程原则(Evolutionary Progress Principle, EPP)是指状态空间中个题的适应值的增加减去适应值的减小构成的进展。EPP 类似于遗传算法中的模式定理，是进化策略的最基本假设，是进化策略理论分析的基础[19]。

Beyer 在进化策略的理论研究方面做出了重要的贡献。Beyer 首先建立进化策略的数学模型，通过微分几何研究分析了非球面适应值空间的性质，研究了(μ，λ)-ES 等多种进化策略的收敛性和算法的动力学性质，建立了较为全面的收敛性理论，其中包括全局收敛性和收敛性的单步测度(one-step measures)[1,19]。对于进化策略中重组算子的作用，Beyer 提出了遗传修复假说(Genetic Repair Hypothesis,GR[51]。重组算子共享了父代的信息，通过交换组合产生新的个体，从而提高算法的收敛速度。Beyer 在 1998 年提出了变异通过重组生成物种原则(Mutation-induced speciation by recombination, MISR)，即种群中的变异是通过全局性的重组操作扩散到整个种群，从而生成新的种群[51]。

Rudolph[52]使用称为“诀窍理论”(martingaletheory)的数学工具证明了(1+λ)-ES 的收敛性。Yin 等[53]用随机逼近(stochasticap proximation)分析了(1,λ)-ES 的收敛性。为了控制收敛速度，提高搜索效率，Rechenberg 用以走廊模型和球模型函数为优化对象，提出了著名的 1/5 成功定律[23]，它奠定了进化策略的理论基础。Eiben 等[54]分析了不同变异步长控制策略下的性能。Kim[55]等应用次序统计学分析了使用柯西变异算子、柯西与高斯组合变异算子的进化算法的局部收敛速度。

1.5 进化策略的流程

进化策略作为一种具有适于高度并行及自组织、自适应、自学习等特征的群体寻优算法，工作流程包括问题的表达、参数初始化、产生初始种群、计算个体适应度、确定终止条件、执行重组、变异、选择等进化操作，再经过反复迭代，逐渐向最优解靠近。进化策略的工作流程如图 1-1 所示。

在图 1-1 中，t 表示进化的代数，k 表示产生的新个体的数目。在第 0 代，初始化群体，随机产生 μ 个个体，再计算每个个体的适应度，然后依次执行重组和变异操作，产生新的个体，接着根据所选择策略，从 λ 个新产生的个体或($\mu+\lambda$)个个体中选择 μ 个个体组成下一代的群体。这样，便完成一代进化。反复执行上述的进化过程，直至满足终止条件。

1.6 进化策略的基本技术

1.6.1 问题的表达

进化策略采用传统的十进制实数表达问题，目标参数和策略参数都需要编码到染色体中。目标参数是指直接涉及适应值计算的参数[1]。策略参数是指应用于进化算法中的控制参数，例如种群规模、变异步长等[1]。进化策略采用实数编码将目标变量直接编码到染色体中。同时它也将策略参数编码到染色体中。进化策略通常有两种表达方式[3]。

(1)二元表达方式。在这种表达方式中，算法中的个体由两部分组成：目标变量 X 和变异步长 σ，每一部分又由 n 个分量组成，即：

$$(X,\sigma)=((x_1,x_2,...,x_n),(\sigma_1,\sigma_2,...,\sigma_n))$$

目标变量 X 和变异步长 σ 的关系如下：

$$\begin{cases}\sigma_i^{t+1}=\sigma_i^t\cdot\exp(\tau_1\cdot N(0,1)+\tau_2\cdot N_i(0,1))\\ x_i^{t+1}=x_i^t+\sigma_i^{t+1}\cdot N_i(0,1)\end{cases}\qquad(1.5)$$

其中，(x_i^t,σ_i^t) 表示第 t 代的个体(父代个体)的第 i 个分量。$(x_i^{t+1},\sigma_i^{t+1})$ 表示第 $t+1$ 代的个体(子代新个体)的第 i 个分量。$N(0,1)$ 是服从高斯分布的随机数。$N_i(0,1)$ 是针对第 i 个分量重新产生一次服从高斯分布的随机数。τ_1 是标准向量 σ 的整体步长参数，常取 1。τ_2 是 σ 的每个分量 σ_i 的步长参数，常取 1。从公式（1.5）可以看出，进化策略的子代新个体是父代个体的基础上增加一个随机数，变异步长

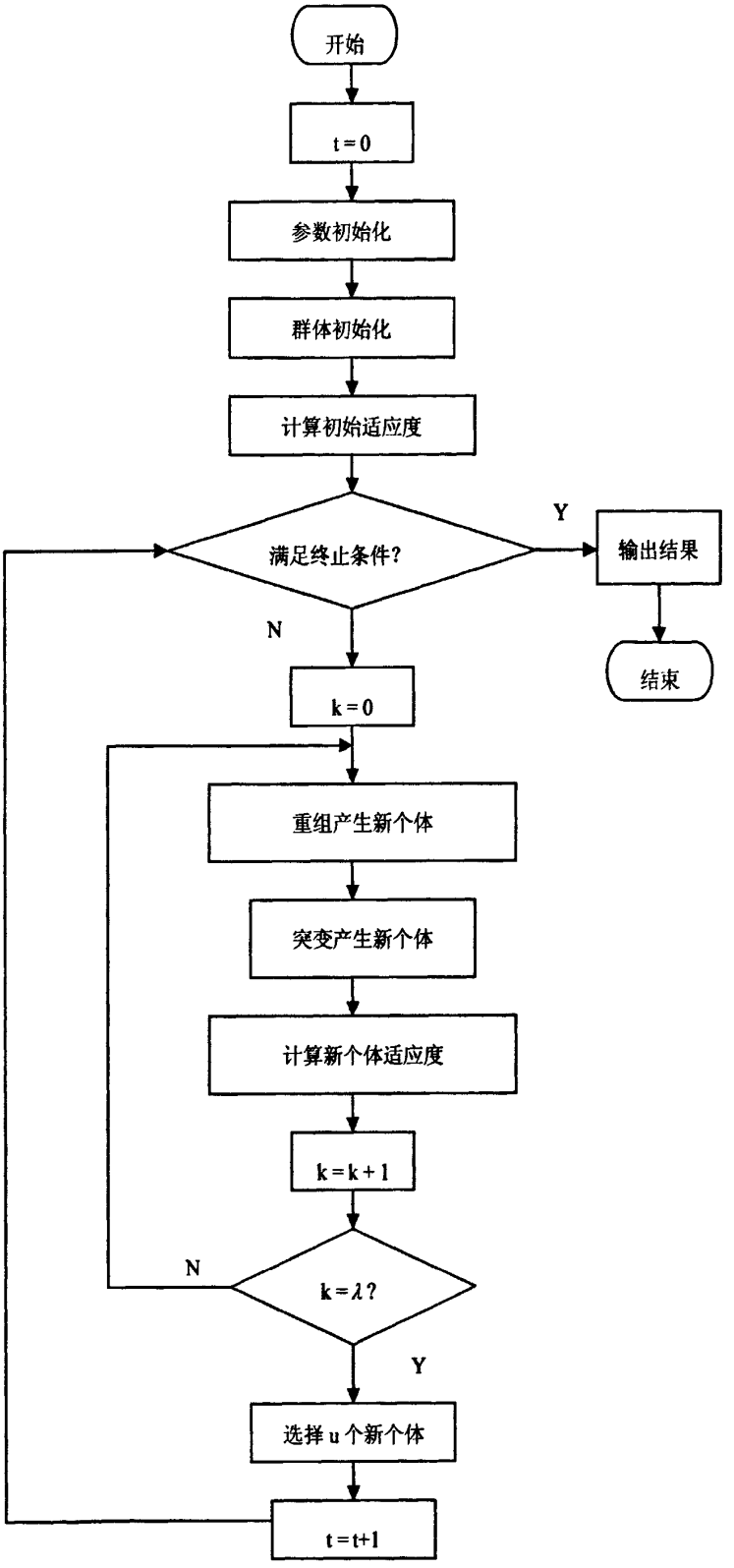

图 1-1 进化策略算法的流程

Figure 1-1 the process of evolutionary strategy algorithm

也在进化过程中自适应调整。由于二元表达方式简单明了，被广泛的运用。

(2)三元表达方式。为了改善收敛速度，提高搜索效率，Schwefel 在常用的二元表达方式的基础上添加一个新的变量，即坐标旋转角度。算法中的个体由三部分组成：目标变量 X、变异步长 σ 和坐标旋转角度 α，即：

$$(X,\sigma,\alpha)=((x_1,x_2,...,x_n),(\sigma_1,\sigma_2,...,\sigma_n),(\alpha_1,\alpha_2,...,\alpha_n))$$

目标变量 X、变异步长 σ 和坐标旋转角度 α 的关系如下：

$$\begin{cases}\sigma_i^{t+1}=\sigma_i^t\cdot\exp(\tau_1\cdot N(0,1)+\tau_2\cdot N_i(0,1))\\ \alpha_j^{t+1}=\alpha_j^t+\beta\cdot N_j(0,1))\\ x_i^{t+1}=x_i^t+w_i\end{cases}\tag{1.6}$$

其中，α_j^t 表示第 t 代的个体(父代个体)的第 i 个分量和第 j 个分量之间坐标的旋转角度。α_j^{t+1} 表示第 $t+1$ 代的个体(子代新个体)的第 i 个分量和第 j 个分量之间坐标的旋转角度。β 是系数，常取 0.0873。w_i 取决于 α_j^{t+1} 和 σ_i^{t+1} 的高斯分布随机数。

1.6.2 初始群体的产生

在(μ，λ)-ES 和(μ+λ)-ES 中，初始种群包含 μ 个个体，每个个体 (X,σ) 又包含 n 个 x_i 和 n 个 σ_i。初始种群是随机产生的，可以从某一个初始点 (X^0,σ^0) 出发，通过多次变异，生成 μ 个初始个体。σ^0 不宜取太大，否则选择力度不过，搜索的步长过大，容易使种群中的个体过于分散。当 σ^0 较小时，仍可通过个体在进化过程中的自适应调整使种群中的个体分散在整个可行解的空间区域内。

1.6.3 适应值计算

进化策略是使用适应值是用来衡量个体优劣。与遗传算法和遗传规划采用二进制数进行编码不同，进化策略采用了十进制实数表达问题，因此进化策略的适应度计算更加简便、直观，更易于执行。

对于约束条件，进化策略是采用重复试凑法进行出来。经过重组、变异操作产生的新个体，如果满足约束条件，则进入候选群体等待选择，如果不满足约束条件，则舍弃该个体，重新产生的新个体。由于进化策略采用实数编码，这种检验比较简单易行[3]。

对于(μ,λ)-ES，是在新生成的 λ 个子代个体中选择 μ 个个体组成新的种群，父代个体不参加选择，直接舍弃。因此，不用计算初始种群的适应值，直接对初始

种群的个体进行重组、突变等操作。对于(μ+λ)-ES，是在新生成的 λ 个子代个体和 μ 个父代个体中选择 μ 个个体组成新的种群，父代要参与选择操作，因此必须计算初始种群的适应值。

1.6.4 重组

进化策略中，除了(1+1)-ES 方式外，其余都引入了重组算子。进化策略的重组算子相当于遗传算法的交换算子。重组操作是提供了两个不同个体直接交换信息的机制。一般认为，在进化过程中，变异是基于一个父代个体，而重组共享了两个父代的信息[1]。进化策略的重组算子主要有以下三中方式[3]：

(1) 离散(直接)重组。先从父代个体中随机选择两个个体

$$\begin{cases}(X^1,\sigma^1)=((x_1^1,x_2^1,...,x_n^1),(\sigma_1^1,\sigma_2^1,...,\sigma_n^1))\\(X^2,\sigma^2)=((x_1^2,x_2^2,...,x_n^2),(\sigma_1^2,\sigma_2^2,...,\sigma_n^2))\end{cases} \tag{1.7}$$

再把两个父代个体的分量进行随机交换，生成子代新个体，如下所示：

$$(X,\sigma)=((x_1^{w_1},x_2^{w_2},...,x_n^{w_n}),(\sigma_1^{y_1},\sigma_2^{y_2},...,\sigma_n^{y_n})) \tag{1.8}$$

其中，w_i=1 或 2，y_i=1 或 2，i=1,2,...,n。新个体的分量是从两个父代个体中随机选取的。

(2)中值重组。同样先从父代个体中随机选择两个个体如式（1.7），再把两个父代各个分量相加起来，然后取平均值，构成新的子代个体，如下所示：

$$\begin{aligned}(X,\sigma)=&((x_1^1+x_1^2)/2,(x_2^1+x_2^2)/2,...,(x_n^1+x_n^2)/2),\\&((\sigma_1^1+\sigma_1^2)/2,(\sigma_2^1+\sigma_2^2)/2,...,(\sigma_n^1+\sigma_n^2)/2))\end{aligned} \tag{1.9}$$

从公式（1.9）可以看出，新生成的个体的各个分量兼容了两个父代个体的信息，而离散重组中，新生成的个体的各个分量只含有一个父代个体的信息。

(3)混杂(全局)重组。与上述的两种重组方式不同，混杂重组是先从父代个体中随机选择一个固定的个体，然后针对子代个体的每个分量再从父代个体中随机第二个个体与固定的个体进行重组。在这种重组方式下，第二个父代个体是经常变换的。公式(1.10)给出了混杂重组目标变量 X 的表示形式。

$$x_i^{'}=\begin{cases}x_{S,i}\ or\ x_{T,i}\\(x_{S,i}+x_{T,i})/2\\x_{S,i}+\theta(x_{T,i}-x_{S,i})\end{cases} \tag{1.10}$$

其中，S 和 T 代表选择的两个父代个体。$x_i^{'}$ 代表新个体的目标变量的第 i 个

分量。$x_{S,i}$代表固定的父代个体的目标变量的第 i 个分量。$x_{T,i}$代表随机选择的另一个父代个体的目标变量的第 i 个分量。θ在[0,1]区间内，为一均匀变异的随机变量。

将上述的三种方法排列组合，进化策略共有以下 7 种重组方式[20]：

1. 无重组。
2. 直接重组。
3. 混杂离散重组。
4. 算术平均重组。
5. 混杂平均重组。
6. 全局算术平均重组。
7. 全局混杂平均重组。

可以表示为：

$$x_i^{'}=\begin{cases} x_{S,i} \\ x_{S,i}\ or\ x_{T,i} \\ x_{S,i}\ or\ x_{W_i,i} \\ x_{S,i}+(x_{T,i}-x_{S,i})/2 \\ x_{S,i}+(x_{W_i,i}-x_{S,i})/2 \\ x_{S,i}+\theta(x_{T,i}-x_{S,i}) \\ x_{S,i}+\eta_i(x_{W_i,i}-x_{S,i}) \end{cases} \tag{1.11}$$

其中，S 和 T 表示从群体中随机选择的两个父代个体，并且 $S\neq T$。W_i 表示针对子代的每一个分量重新随机选择的父代个体。θ 和 η_i 均表示[0,1]区间均匀分布的随机数。η_i 还表示是子代目标变量的第 i 个分量对应的随机数。

Schwefel 建议对目标变量 X 宜采用离散重组，对变异步长宜采用中值或混杂平均重组。Aemeyer 等[56]认为重组操作可以有效提高进化策略的性能，但是出于方便和效率方面的考虑，许多应用中均不采用重组算子。

1.6.5 变异

在进化策略中，变异算子是主要算子，占有很重要的地位。一般情况下，进化算子的设计应遵循以下的基本原则[19]：

(1) 可行性，即在有限的时间内是可以实现的。

(2) 可度量，每一步变异的长度是可以度量的。

(3) 变异的分布满足最大熵原则。

(4) 对称性要求，即变异的随机变量的数学期望值为零。

进化策略的变异是给旧个体加上一个服从正态分布的随机变量来实现的。进化策略目标变量和变异步长的变异如公式（1.6）所示。τ_1、τ_2和β是给定的外部参数，Schwefel 建议可按如下设置[25]：

$$\tau_1 \propto (\sqrt{2\sqrt{n}})^{-1}, \quad \tau_2 \propto (\sqrt{2n})^{-1}, \quad \beta \approx 0.0873 \tag{1.12}$$

其中，$\propto$表示正比，n代表个体中所含分量的数目，τ_1和τ_2分布代表全局步长系数和局部步长系数，用于对随机量 *N(0,1)*进行缩放。系数β涉及旋转角度，建议取 5 度，转换为弧度就是$\pi \times 5/180 \approx 0.0873$。

如果取消旋转因子，进化策略的表达式由三元组变为两元组，目标变量和变异步长的变异公式转变为公式（1.5）。

近年来，一些学者，如 Kappler[38]，Yao[39]，提出了用柯西变异算子代替进化策略中的高斯变异算子。一维柯西分布的概率密度函数集中在原点附近，其定义为：

$$f_t(x) = \frac{1}{\pi}\frac{t}{x^2 + t^2}, \quad -\infty < x < \infty \tag{1.13}$$

其中，t>0 为比例系数，相应的分布函数定义为：

$$\mathrm{F}_t(x) = \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x}{t}) \tag{1.14}$$

柯西分布的概率密度函数类似于高斯分布的概率密度函数，其主要差异主要表现在：柯西分布在垂直方向上略小于高斯分布，而柯西分布在水平方向上越靠近水平轴，变得越缓慢，柯西分布可以看作是无限的。因此，采用柯西分布进行变异，使个体的变换范围更大，更容易跳出局部极值点。图 1-2 给给出了高斯分布和柯西分布的关系。

1.6.6 选择

每种进化算法都需要面向目标的选择算子，用以指导搜索朝向目标参数空间的期望区域，使进化具有方向性。进化策略的选择算子相当于遗传算子的复制算子，都体现了达尔文的“物竞天择、适者生存”的思想。进化策略的选择是一个

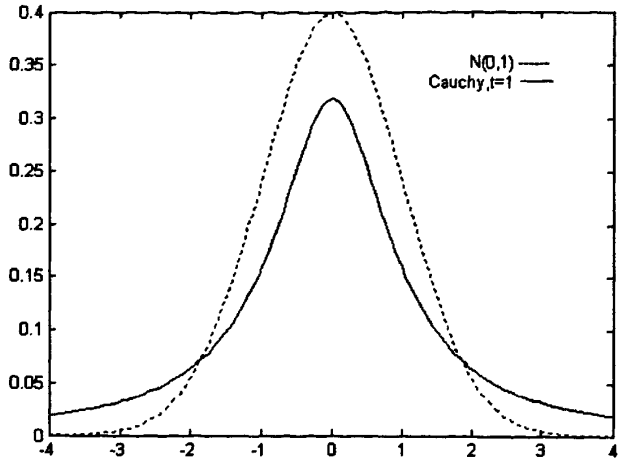

图 1-2 柯西分布和高斯分布比较

Figure 1-2 Comparison between Cauchy distribution and Gaussian distribution

确定的过程，根据适应度将最好的个体保留下来，而劣质的个体完全被淘汰，以便使优良个体产生更优良的后代。在群体中，只有 μ 个最优的个体能成为下一代的父代个体，这种选择方式也称为截断(truncation)或育种(breeding)选择，可以表示为：

$$P^{t+1} = \{a_{1:w}, a_{2:w}, ..., a_{\mu:w}\} \tag{1.15}$$

其中，$a_{m:w}$ 表示从 w 个个体中选择适应度排名为 m 的最佳个体。

在进化策略中，为了保证能够收敛到全局最优解，通常使用(μ+λ)-ES 模型和(μ,λ)-ES 模型。这两种不同的模型所采用的选择方式也是不同的。

(μ+λ)-ES 模型考虑了父代个体，是从 μ 个父代个体和新生成的 λ 个子代个体中中确定地选择最优的 μ 个个体组成下一代群体。在此，μ 和 λ 的取值没有限制[1]。(μ,λ)-ES 的选择方式保证了最优个体存活，使群体的进化过程呈单调上升趋势，是一中精英保留型的选择方式。(μ,λ)-ES 较好地继承了父代的优良特性，收敛性好 ，但容易陷入局部最优解[19]。

(μ,λ)-ES 模型只从新生成的 λ 个子代个体中确定地选择最优的 μ 个个体组成下一代群体。每个个体只存活一代，随即被新个体代替。显然必须 $\mu<\lambda$，才能实现选择功能。一种极端的情况，当 $\mu=\lambda$ 时，将无选择功能，因为新生成的个体全部被选为下一代的父代个体。此时，选择不能提供有关的搜索信息，种群的进化处于一种盲目的状态。为了控制群体的多样性和选择的力度，在(μ,λ)-ES 中，μ/λ 的比值是一个重要的参数，它会对算法的收敛速度产生影响[3]。μ 取值太小，种群太

单调，进化速度缓慢，例如 μ=1 的极端情况，种群无多样性可言。μ 取值太大，则计算量过大。通常，μ 取 15 或更多一些。Schwefel[25]认为比率 $\mu/\lambda \approx 1/7$ 是最优的。

1.6.7 终止

进化策略作为一个算法，应该具有终止条件。迭代的结束通常以算法不能再找到更优解为准则，而在现实中，在每次迭代循环中都需要指明是否终止。进化策略经过多次迭代进化，算法逐渐收敛，判别进化策略算法是否终止通常有四种方法[3]：

(1)规定最大的迭代次数 N。一旦进化策略算法的迭代次数达到 N，不管是否已经找到最优解或满意解，都停止操作，输出结果。这样避免了算法执行太多迭代，有效地限制了算法的运行时间。

(2)规定最小偏差 ξ。进化策略是用适应度对个体的质量进行衡量。对于适应度目标已知的进化策略，一旦群体中的最优个体的适应度与适应度目标的差值小于允许的最小偏差 ξ 时，令算法终止。这种终止条件可以表述为：

$$|f_{\max} - f^*| \le \xi \tag{1.16}$$

其中，f^* 表示已知的适应度目标，$f_{\max}$ 表示每代最优个体的适应度。

(3)观察适应度的变化趋势。在进化策略算法的初期，群体的最优个体的适应度以及群体的平均适应度都较小，以后随着重组、变异、选择等操作，种群不断进化，适应值增加。到了进化策略算法的后期，这种增加已趋于缓和或停止，一旦这种增加停止并且持续了一代的代数，例如连续进化了 50 代，种群的最优个体的适应度都没有变化，则终止算法。

(4)最优个体和最差个体的最小偏差 ξ。针对进化策略，Schwefel 提出了用群体中的最优个体和最差个体的适应度的比较来判断算法是否终止。一旦最优个体的适应度与最差个体的适应度的差值小于给定的最小偏差 ξ 时，令算法终止。这种终止条件可以表述为：

$$|f_{\max} - f_{\min}| \le \xi \tag{1.17}$$

其中，$f_{\max}$ 代表种群中最优个体的适应度，$f_{\min}$ 种群中最差个体的适应度。

公式（2.13）也可以写成相对差的形式，即：

$$\left|\frac{f_{max}-f_{min}}{f_{max}}\right| \leq \xi \tag{1.18}$$

1.7 进化策略与遗传算法的比较

进化策略和遗传算法是进化算法的两个分支，它们存在着很多相同的地方，同时也存在着明显的区别。

1.7.1 相同

进化策略和遗传算法存在着很多相同点[3]：

(1)两者都是借鉴了生物进化的思想，体现了生物进化过程中的“物竞天择、优胜劣汰”的原则。从一组随机生成的初始解出发，通过变异、交换、选择等进化操作，反复迭代，逐步逼近最优解。

(2)两种算法都采用了群体的概念。虽然早期的进化策略是基于单个个体的，但发展到(μ,λ)-ES 或$(\mu+\lambda)$-ES 后都是基于多个个体，从一组点集开始搜索，体现了并行算法的特点。

(3)两种算法都是渐进式搜索算法，都要经过多次的迭代，使新的一代的结果优于上一代，最终找到最优解，或满意解，这是一个渐进寻优的过程。

(4)两者都是自适应搜索。在进化过程中，仅仅借助变异、交换、选择等进化操作，而无需添加任何额外的作用，就能使种群的质量不断提升，具有自动适应环境的能力。

(5)两者都是有指导搜索。不同于盲目搜索和穷举搜索，进化策略和遗传算法都是有指导地搜索，在适应度的驱动下，使得算法逐步逼近最优解。

(6)两者都是全局式寻优。进化策略和遗传算法都采用了并行搜索，而且通过借助变异和重组产生新的个体，不断扩大搜索范围，因而容易找到全局最优解而不是局部最优解。

1.7.2 差别

进化策略和遗传算法同样存在着很多不同点[3]：

(1)表达方式上的差别。进化策略主要用于求解函数优化问题，因此，群体中的个体采用十进制的实数表达问题。遗传算法可以广泛用于搜索、优化与机器

学习等问题领域，更适合处理离散问题，其群体中的个体采用二进制编码表达问题。进化策略源于函数优化处理，是一种类似于爬上问题的优化方法。遗传算法起源于自适应搜索，强调全方位探查。

(2)变异(突变)算子的差别。变异算子是进化策略的主要算子，每个新个体都经历变异。进化策略中的变异是在父代个体的基础上添加一个高斯分布的随机数，从而产生新的个体。遗传算法中的突变是对旧个体的某个字符进行取反运算。在遗传算法中，仅仅是某一部分的个体的个别字符产生突变。

(3)重组(交换)算子的差别。进化策略中的重组算子，不但可以继承父代个体的部分信息，而且可以通过中值计算产生新的信息。遗传算法中交换算子，只能交换父代的信息，而不能产生新的信息。交换算子是遗传算法的主要算子，而在进化策略中重组算子可以使用也可以不使用。

(4)选择算子的差别。在更新群体过程中，进化策略通过同一代或父子两代按竞争方式进行选择。从 λ 个或($\mu+\lambda$)个个体中选择出最优的 μ 个个体组成新的的群体，选择的方法是确定的。遗传算法主要按基于适应度的比例进行选择，优秀的个体被选择的机率高，但劣质个体也是有机会被选中。

(5)执行顺序的差别。进化策略的执行顺序是先执行重组，再执行变异，最后执行选择。重组和变异都是对同一个父代个体依次进行。遗传算法的执行顺序是先执行选择和复制，再执行交换，最后才是突变操作。交换和突变操作不一定发生在同一个父代个体。

(6)参数的差别。进化策略的变异步长是在进化过程中不断地自适应调整的。遗传算法的复制概率，交换概率及突变概率这些参数在进化过程中原则上是固定的。

1.7.3 相互借鉴

进化策略和遗传算法是进化算法的两个分支，它们在早期是独立发展的，并在各自的领域获得成功，在长期的实践中它们又相互借鉴，相互影响和融合，不断完善。

遗传算法在表达方式上除了采用传统的二进制编码外，已经开始采用和进化策略一样的十进制的实数来表达问题。

早期的进化策略(1+1)-ES 只有变异算子，从(μ+1)-ES 才开始才引入了重组算

子，这是借鉴了遗传算法中的交换算子。而遗传算子中的交换算子也借鉴了进化策略的重组算子，有人建议交换算子采用类似于进化策略的广义中值重组，产生新的信息。

进化策略的参数的自适应调整的特征也开始被遗传算法所使用。遗传算法中的群体规模、突变概率，可以随时间而变化。

1.8 论文的组织

本文的内容组织如下：

第一章概述了进化算法的发展和特点，对进化策略的演变、特点、流程、研究现状、基本技术等进行详细的介绍，对进化策略和遗传算法进行比较。

第二章对进化策略算法进行改进，提出一种基于双种群的改进进化策略算法（MES）。分析了 MES 算法的变异策略和种群划分依据，从理论上分析算法的正确性。对八个典型的函数应用该算法和其他三种进化策略算法进行模拟进化实验，并对结果进行比较和分析，说明对于中低维(30 维以下)函数优化问题，MES 算法具有良好的性能，对于高维复杂问题，MES 算法性能不佳。最后通过两种典型的情况对 MES 算法的收敛性进行分析。

第三章提出了一种适合求解高维可分解问题的进化策略算法，即基于合作型协同进化的改进进化策略算法（MESCC）。首先分析了进化策略算法在处理高维复杂问题时存在的不足，再对协同进化进行介绍，在此基础上，结合 MES 算法，提出了 MESCC 算法，并从问题分解、团队协作和算法流程三个方面就行分析说明。对五个典型的高维函数应用 MESCC 算法和其他四种进化策略算法进行模拟进化实验，并对结果进行比较和分析，说明对于高维可分解问题，MESCC 算法具有良好的性能。

第四章将 MES 算法应用到 K-means 聚类算法当中，提出了提出了一种基于改进进化策略的 K-means 聚类算法(KAMES)。首先分析了 K-means 聚类算法存在的不足，在此基础上，结合 MES 算法的优点，提出了 KAMES，从编码方式、种群初始化、适应度函数设计、K-means 操作、主要流程等方面进行分析说明。最后通过鸢尾花卉数据集对 KAMES 算法的性能进行测试。

第二章 基于双种群的改进进化策略算法

针对进化策略算法收敛速度较慢、容易早熟的问题，本章提出一种基于双种群的改进进化策略算法。将种群划分为规模较小的精英子群和规模较大的普通子群。精英子群用于存放种群中最优秀的个体，普通子群用于存放种群中的普通个体。对不同的子群采用不同的变异策略，实现种群在解空间具有良好的全局搜索能力的同时具有尽可能精细的局部搜索能力。通过对算法进行理论分析说明其正确性。对几个典型的函数应用该算法进行模拟进化实验，取得良好的效果。

2.1 算法提出的背景

现实中存在的许多工程和科学问题都可以通过建立数学模型从而转化为函数优化问题。进化策略具有独特的群体搜索策略，而且具有全局性、自适应性和稳健性等特点，在解决这类问题时表现出了比传统的优化算法更好的性能。但进化策略由于自身固有的缺陷，存在着收敛速度较慢、容易早熟的问题。其原因是算法存在着两个相互矛盾的因素："种群多样性"和"选择压力"。要提高进化策略算法的性能，就必须在保持种群多样性和提高选择压力之间维持某种平衡。

ES 从(1+1)-ES 和(λ+1)-ES 演变到目前最常用的(μ+λ)-ES 和(μ,λ)-ES，统称为 CES(Classical Evolutionary Strategies)[57]。近年来，有学者提出了一些改进 CES 算法性能的方法。 Yao[39]提出了快速进化策略(Fast Evolution Strategies，FES)，用柯西变异算子代替 CES 算法使用的高斯变异算子，利用了柯西变异较强的分散能力来获得较好的跳出局部极值点的能力。一般情况下，FES 算法在求解多极值复杂问题时取得较好的结果，但对于局部极值较少的问题，FES 算法的性能与 CES 算法差别不大[36]。王战权等[45]试验使用对数柯西分布代替高斯分布。Lee 等[46]提出通过强化学习实现步长的自适应。夏慧明等[49]提出了双种群进化策略(Bi-group Evolution Strategies，BES)，将种群划分为规模相同的两个子群，对一子群采用高斯变异算子，对另一子群采用柯西变异算子。BES 算法在求解多极值复杂问题时性能优于 CES 算法，但不如 FES 算法，对于局部极值较少的问题，BES 算法的性能优于 FES 算法，但不如采用高斯变异算子 CES 算法。

基于平衡 ES 算法的局部搜索和全局搜索能力的目的，本文对 BES 算法在种群划分和变异策略两个方面进行改进，提出了一种基于双种群的改进进化策略(Modified Evolutionary Strategies，MES)算法，并将其用于求解函数优化问题。

2.2 MES 算法

在 MES 算法中，将种群划分为两个子群：精英子群 X_S 和普通子群 X_L。精英子群 X_S 规模较小，用于存放种群中最优秀的个体。普通子群 X_L 规模较大，用于存放种群中的普通个体。对精英子群 X_S 使用递减的高斯变异算子，确保子群能够以较高的精度找到子群所在的局部的极值，收敛速度快。对普通子群 X_L 采用柯西变异算子，柯西变异能产生变异的范围比高斯变异更大，又由于普通子群 X_L 规模较大，因而该子群具有良好的全局搜索能力。

2.2.1 变异策略分析

在 MES 算法中，精英子群 X_S 使用递减的高斯变异算子：

$$\sigma_s^{(k)} = \sigma_0 \exp(-\mu k) \tag{2.1}$$

其中，μ 为一常数，其选取与求解的精度有关。k 为当前的进化代数。

高斯分布的概率密度函数为：

$$f_{G(u,\sigma^2)}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{2.2}$$

高斯分布的值落在区域$[u-3\sigma, u+3\sigma]$的概率为：

$$P_{G(u,\sigma^2)}(u-3\sigma < X \le u+3\sigma) = \int_{u-3\sigma}^{u+3\sigma} f_{G(u,\sigma^2)}(x)dx = 0.9974 \tag{2.3}$$

从公式（2.3）可以看出高斯分布的值几乎都落在区域$[u-3\sigma, u+3\sigma]$内。在 MES 算法中，均值$\mu = 0$，而σ是按指数递减，产生的变异几乎都在一个小的范围$[-3\sigma, 3\sigma]$内。因此精英子群具有良好的局部搜索能力，能够以较高的精度找到子群所在的局部的极值。

普通子群 X_L 采用了柯西变异算子。

$$\begin{cases} \sigma_i^{t+1} = \sigma_i^t \cdot \exp(\tau_1 \cdot \eta + \tau_2 \cdot \eta_i) \\ x_i^{t+1} = x_i^t + \sigma_i^{t+1} \cdot \eta_i \end{cases} \tag{2.4}$$

其中，(x_i^t, σ_i^t)表示第 t 代的个体(父代个体)的第 i 个分量。$(x_i^{t+1}, \sigma_i^{t+1})$表示第

$t+1$ 代的个体(子代新个体)的第 i 个分量。η 是服从柯西分布的随机变量。η_i 是针对第 i 个分量重新产生一次服从柯西分布的随机变量。τ_1 是标准向量 σ 的整体步长参数，常取 1。τ_2 是 σ 的每个分量 σ_i 的步长参数，常取 1。

公式（1.13）给出了柯西分布的概率密度函数。图 1-2 给出了高斯分布和柯西分布的关系。高斯分布产生的变异集中在一个区域内，而柯西分布能够产生变异的范围更大[58]。因此普通子群具有良好的全局搜索能力。

2.2.2 种群划分依据

在进化策略算法中，种群的划分会对算法的性能产生影响。在 MES 算法中，将种群划分为一个规模较小的精英子群和一个规模较大的普通子群。

设当前种群大小为 N，精英子群的大小为 N_S，普通子群的大小为 N_L，满足 $N=N_L+N_S$。假设全局最优点为 x^*，收敛精度为 ε，高斯变异算子的均值为 0，标准差为 σ，柯西变异的比例系数 t=1。那么经过一次变异，种群进入到区域 $[x*-\varepsilon,x*+\varepsilon]$ 的概率 P 为：

$$P=N_S P_{G(0,\sigma^2)}(|x-x*|\le\varepsilon)+N_L P_{C(1)}(|x-x*|\le\varepsilon) \tag{2.5}$$

当 $[x*-\varepsilon,x*+\varepsilon]\cap[-3\sigma,3\sigma]=\varnothing$ 时，由于

$$\int_{-\infty}^{-3\delta} f_{G(0,\sigma^2)}(x)dx+\int_{3\delta}^{+\infty} f_{G(0,\sigma^2)}(x)dx\approx 0 \tag{2.6}$$

故

$$\int_{x*-\varepsilon}^{x*+\varepsilon} f_{G(0,\sigma^2)}(x)dx\approx 0 \tag{2.7}$$

因此公式（2.5）可以转化为：

$$P\approx N_L P_{C(1)}(|x-x*|\le\varepsilon)=N_L\int_{x*+\varepsilon}^{x*-\varepsilon}\frac{1}{\pi(1+x^2)}dx \tag{2.8}$$

当 $[x*-\varepsilon,x*+\varepsilon]\cap[-3\sigma,3\sigma]\neq\varnothing$ 时，

$$P=N_s\int_{x*+\varepsilon}^{x*-\varepsilon}\frac{1}{\sqrt{2\pi}\times\sigma}\exp(-\frac{x^2}{2\times\sigma^2})dx+N_L\int_{x*+\varepsilon}^{x*-\varepsilon}\frac{1}{\pi(1+x^2)}dx \tag{2.9}$$

在 MES 算法中，对精英子群使用递减的高斯变异算子，随着进化代数的增加标准差 σ 不断变小，当进化到一定的代数使 σ 取足够小时使得 $N_s\int_{x*+\varepsilon}^{x*-\varepsilon}\frac{1}{\sqrt{2\pi}\times\sigma}\exp(-\frac{x^2}{2\times\sigma^2})dx$ 远远大于 $N_L\int_{x*+\varepsilon}^{x*-\varepsilon}\frac{1}{\pi(1+x^2)}dx$ 时，上式可以转化为：

$$P \approx N_s \int_{x*+\varepsilon}^{x*-\varepsilon} \frac{1}{\sqrt{2\pi}\times\sigma} \exp(-\frac{x^2}{2\times\sigma^2})dx \tag{2.10}$$

为使 P 在$[x*-\varepsilon, x*+\varepsilon] \cap [-3\sigma, 3\sigma] = \varnothing$的情况下能取得较大的值，由公式（2.8）可知，$N_L$应取较大的值，但由$N = N_L + N_S$可知，$N_L$的取值过大时，$N_S$过小，在$\sigma$较小，$[x*-\varepsilon, x*+\varepsilon] \cap [-3\sigma, 3\sigma] \neq \varnothing$的情况下由公式（2.10）可知，$P$的取值较小。经过对不同规模的种群大小以及不同种群比例的反复实验发现N_L / N_S的值在[2，5]时，算法的性能较优。

2.2.3 算法流程

MES算法主要从普通子群X_L中选取出优秀的个体与精英子群X_S中的个体进行比较，把其中相对优秀的个体放到精英子群X_S中。再对精英子群X_S中的个体进行高精度地搜索。算法的流程如下：

1. 参数初始化。设置种群个数 N，最大的进化代数K_{MAX}，选取高斯变异算子标准差σ_0，设置种群的进化代数$k = 0$，设置进化终止条件。

2. 种群初始化。在问题的可行解空间中随机产生 N 个个体作为初始种群X(0)。

3. 终止检验。如果满足进化终止条件，则输出结果，终止计算。否则，转4。

4. 从种群中选取最优秀的S个个体组成精英子群X_S，剩下的L个个体组成普通子群X_L。其中，$2S \le L \le 5S$。

5. 变异产生后代。对精英子群X_S采用下面的方式进行变异：

$$x_s^{(k)}(i)' = x_s^{(k)}(i)' + \sigma_s^{(k)} \tag{2.11}$$

其中，$\sigma_s^{(k)}$为高斯分布随机变量$N(0, \sigma_s^{(k)^2})$的标准差。

$$\sigma_s^{(k)} = \sigma_0 \exp(-\mu k) \tag{2.12}$$

其中，μ为一常数，其选取与求解的精度有关。X_S变异后产生种群$X_s^{'}$。

对普通子群X_L采用柯西变异：

$$x_L^{(k)}(i) = x_L^{(k)}(i) + \sigma'_L(i)\xi_L \tag{2.13}$$

其中ξ_L表示服从柯西分布的随机变量。X_L变异后产生种群X'_L。

6. 选择。$X_s^{'}$，$X_L^{'}$组成临时种群X_T，再从X_T中选择出最优秀的N个个体组成新的种群$X^{(k+1)}$。

7. $k = k + 1$，转3。

2.3 模拟试验

为了分析和验证本文提出的 MES 算法的性能，本文选用了八个标准函数进行分组试验[39]。将这八个函数分为两组，第一组为函数 $f1\sim f5$，是中低维函数（20维以下）。第二组为函数 $f6\sim f8$，是高维函数(维数均为 100)。

1. $f_1=\sum_{i=1}^{N}x_i^2$，球模型函数，定义域为(-5.12,5.12) ^{N}，维数 N=3，最小值为0。其二维函数图形如图 2-1 所示。

2. $f_2=0.5+\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1+0.001(x_1^2+x_2^2)]^2}$，定义域为(-100,100) ^{N}，维数 N=2，最小值为 0。其二维函数图形如图 2-2 所示。

3. $f_3=\sum_{i=1}^{N}(x_i+0.5)^2$，定义域为(-100,100) ^{N}，维数 N=20，求最小值。其二维函数图形如图 2-3 所示。

4. $f_4=\sum_{i=1}^{N}\{x_i^2-10\cos(2\pi x_i)+10\}$，定义域为(-5.12,5.12) ^{N}，维数 N=20，求最小值。其二维函数图形如图 2-4 所示。

5. $f_5=-20\exp\{-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\}-\exp\{\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i^2)\}+20+\exp(1)$，定义域为(-32,32) ^{N}，维数 N=20，求最小值。其二维函数图形如图 2-5 所示。

6. $f_6=-\sum_{i=1}^{N-1}[100(x_i-x_{i+1})^2+(x_i-1)^2]$，定义域为(-5.12,5.12) ^{N}，维数 N=100，求最大值。其二维函数图形如图 2-6 所示。

7. $f_7=\sum_{i=1}^{N}x_i\sin(\sqrt{|x_i|})$，定义域为(-512,512) ^{N}，维数 N=100，求最大值，理论最大值为 41898.288。其二维函数图形如图 2-7 所示。

8. $f_8=-\frac{1}{N}\sum_{i=1}^{N}(x_i^4-16x_i^2+5x_i)$，定义域为(-5,5) ^{N}，维数 N=100。求最大值，理论最大值为 78.33236。其二维函数图形如图 2-8 所示。

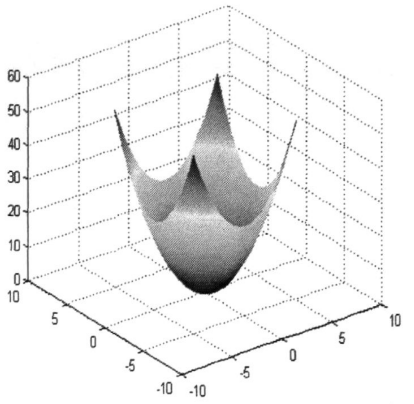

图 2-1　函数 1

Figure 2-1 function 1

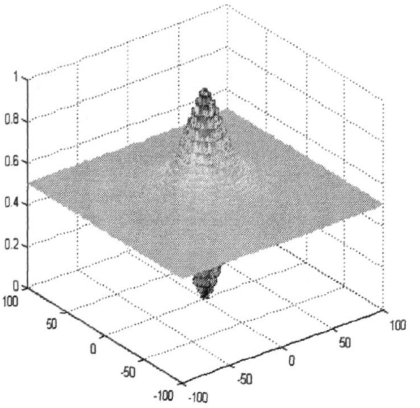

图 2-2　函数 2

Figure 2-2 function 2

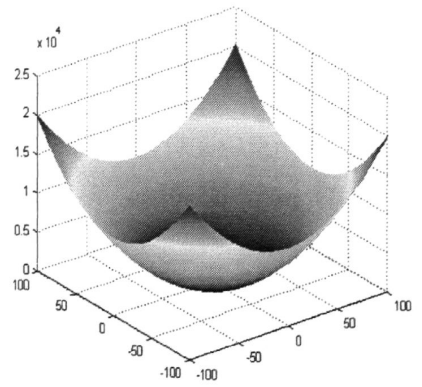

图 2-3　函数 3

Figure 2-3 function3

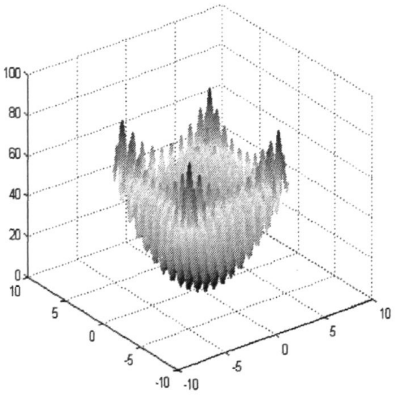

图 2-4　函数 4

Figure 2-4 functions 4

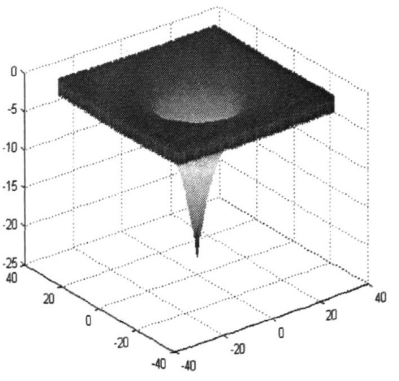

图 2-5　函数 5

Figure 2-5 function 5

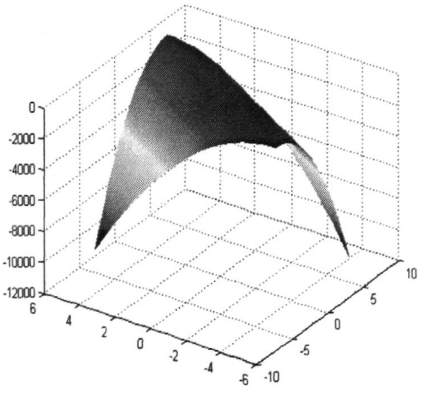

图 2-6　函数 6

Figure 2-6 function 6

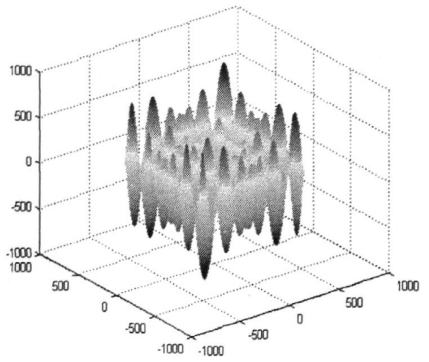

图 2-7　函数 7

Figure 2-7 function 7

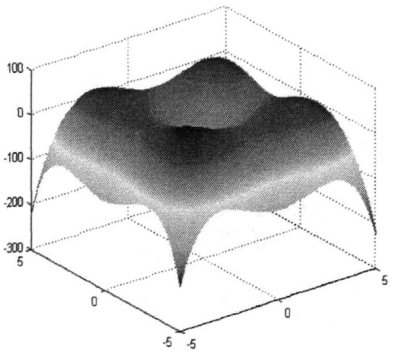

图 2-8　函数 8

Figure 2-8 function 8

2.3.1 实验一

将 MES 算法与 CES 算法、FES 算法、BES 算法在性能方面进行比较。对于函数 $f1 \sim f5$，MES 算法参数的选取如下：变异参数 $\mu = 0.001$，初始标准差 $\sigma_0 = 1$，普通子群 X_L 的规模是精英子群 X_S 规模的 3 倍。CES 算法和 BES 算法中使用的高斯变异的初始标准差 $\sigma_0 = 1$，FES 算法、BES 算法与 MES 算法使用的柯西算子的比例参数 t=1。所有算法的最大的进化代数 $K_{MAX} = 2000$，种群大小 $N = 40$。

每个函数进行测试时均运行 20 次，将收敛到目标值的平均代数作为衡量算法收敛速度的标准。用 20 次运行中收敛的总次数作为衡量算法稳定性的标准。测试结果中改进率若为正值，表示改进后的收敛速度加快，若为负值，则表示改进后的收敛速度变慢。收敛次数成功率的计算公式如下：

$$S = \frac{N_{suc}}{N_{total}} \times 100\% \tag{2.14}$$

其中 N_{suc} 表示收敛的总次数，N_{total} 表示运行的总次数，本实验中为 20。收敛速度改进的计算公式如下：

$$P = \frac{N_{CES}\text{-}M_{ES}}{N_{CES}} \times 100\% \tag{2.15}$$

其中 N_{CES} 表示 CES 算法平均迭代代数，M_{ES} 表示当前选用的算法平均迭代代数。函数 $f1 \sim f5$ 的实验结果如表 2-1 所示。图 2-9、2-10、2-11、2-12、2-13 分别给出了 $f1$、$f2$、$f3$、$f4$、$f5$ 的进化过程。

从表 2-1 和图 2-9、2-10、2-11、2-12、2-13 可以看出 MES 算法的性能明显优

表 2-1 CES, FES, BES and MES 算法的实验结果（函数 $f1 \sim f5$）

Table 2-1 Comparison among CES, FES, BES and MES on functions $f1 \sim f5$

函数	算法	总收敛次数	收敛次数成功率(%)	平均迭代代数	收敛速度改进率（%）
f1	CES	20	100	549	0.0
	FES	20	100	611.75	-11.4
	BES	20	100	558.1	-1.6
	MES	20	100	228.45	58.4
f2	CES	0	0	2000	0.0
	FES	14	70	1017.35	49.1
	BES	8	40	1436.4	28.2
	MES	17	85	530.5	73.5
f3	CES	20	100	1851.6	0.0
	FES	20	100	1850.5	0.6
	BES	20	100	1663.95	10.1
	MES	20	100	512.2	72.3
f4	CES	2	10	1930.7	0.0
	FES	13	65	1444	25.2
	BES	10	50	1532.45	20.6
	MES	17	85	1186.2	38.6
f5	CES	20	100	1442.15	0.0
	FES	20	100	1640.4	-13.7
	BES	20	100	1374.3	4.7
	MES	20	100	648.35	55.0

于其他算法，在收敛速度上大部分函数都有 50%以上的提高(除了 f4 的 38.6%外)，稳定性也有显著提高，对 f1、f3、f5，20 次运行都达到了收敛的要求，对 f2，f4 也有 17 次收敛到了全局最优解。这主要是因为：

(1)MES 算法的普通种群使用了柯西变异算子，而且普通种群的规模较大，因而 MES 算法具有良好的局部逃逸能力，当遇到像 f3 这一类高原函数问题[60]，或象 f2、f4、f5 这类局部极值点随维数增加而呈指数增加的多峰函数时，能有效地

从局部值点的邻域跳出。从图 2-10 可以看出，FES 的逃逸能力最强，很快就进入了全局最优解的邻域，这是因为在 FES 中所有个体都使用了柯西变异算子。MES 算法，在几十代的时候陷入到局部最优，经过四百多代的进化，成功逃离局部最优，进入全局最优解的邻域。BES 算法在化过程中有两次（在几十代和五百代左右）陷入到局部最优。而 CES 的局部逃逸能力最差，在陷入到局部最优点后就一直无法逃离。

(2)MES 算法的精英种群使用了递减的高斯变异算子，并且精英种群中保存的是当前最优秀的个体，因此 MES 算法具有精细的局部搜索能力。当个体进入全局最优解的邻域时，递减的高斯变异算子，确保子群能够以较高的精度找到子群所在的局部的极值。而其他几种算法，局部搜索能力比 MES 算法差，产生的变异过大使得个体与全局最优解的距离没办法达到精度的要求，所以需要进化的代数明显增多。从图 2-9 和图 2-11 可以看出，几种算法在进化之初，收敛速度都比较快，在 100 代左右时都进入全局最优解的邻域。主要的差别在于进化的中后期，MES 由于能够产生足够小的变异的概率较高，很快就得到满足精度要求的解。而其他几种算法，产生足够小的变异的概率较低，大的变异使得与全局最优解的距离没办法达到精度的要求，所以需要进化的代数明显增多。

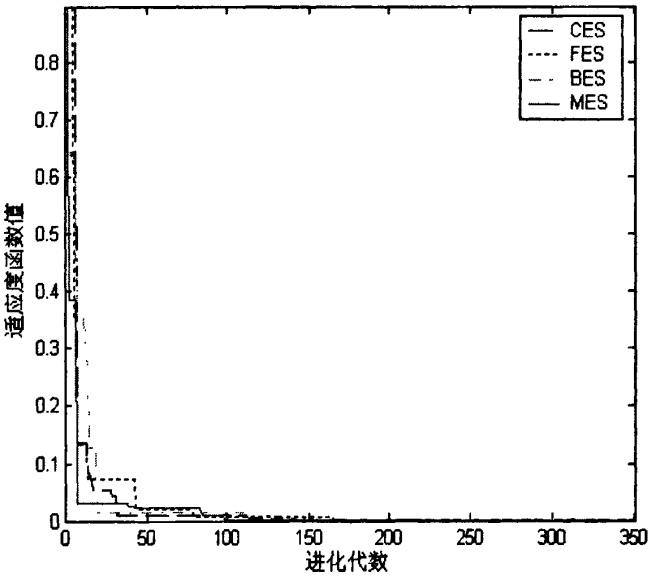

图 2-9 函数 f1 的进化过程

Figure 2-9 evolution of the function f1

从图 2-12 和图 2-13 可以看出，在进化之初（前一百代左右），几种算法的性

能都差不多，之后 MES 由于具有良好的逃逸能力和精细的局部搜索能力，表现出了良好的性能，收敛速度明细优于其他几种算法。

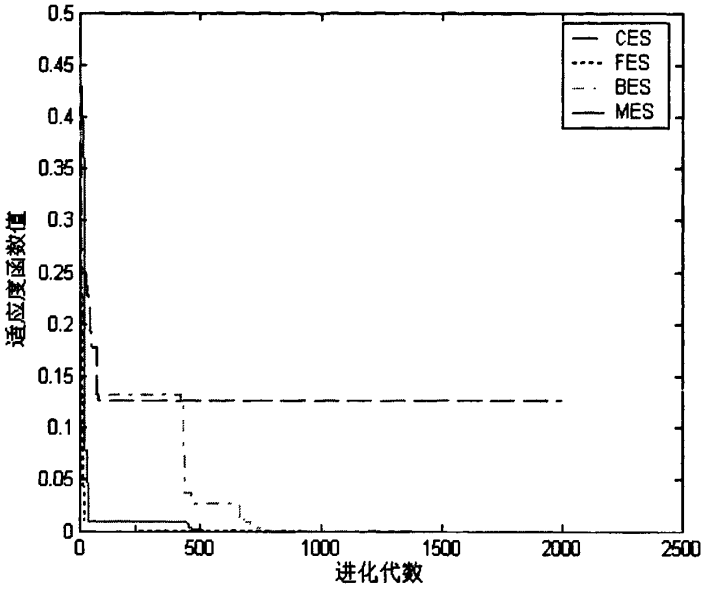

图 2-10 函数 f2 的进化过程

Figure 2-10 evolution of the function f2

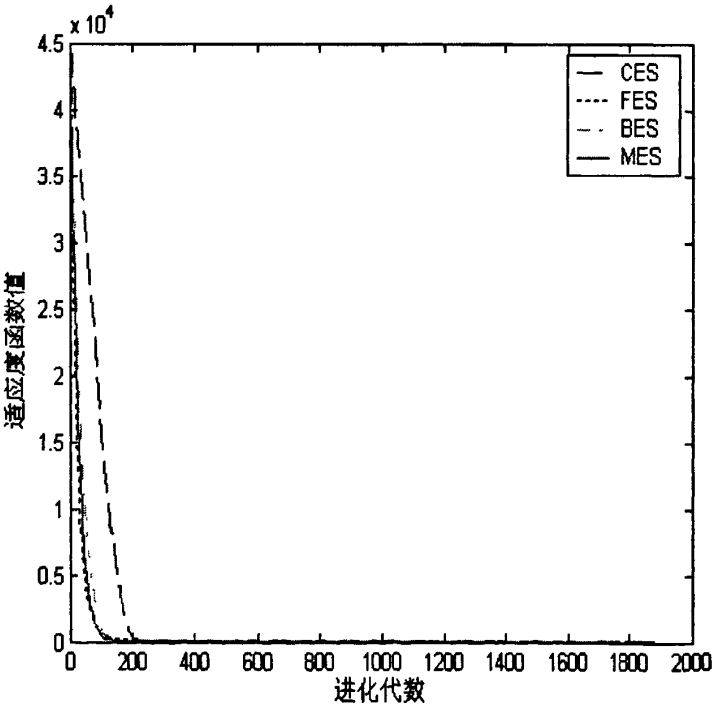

图 2-11 函数 f3 的进化过程

Figure 2-11 evolution of the function f3

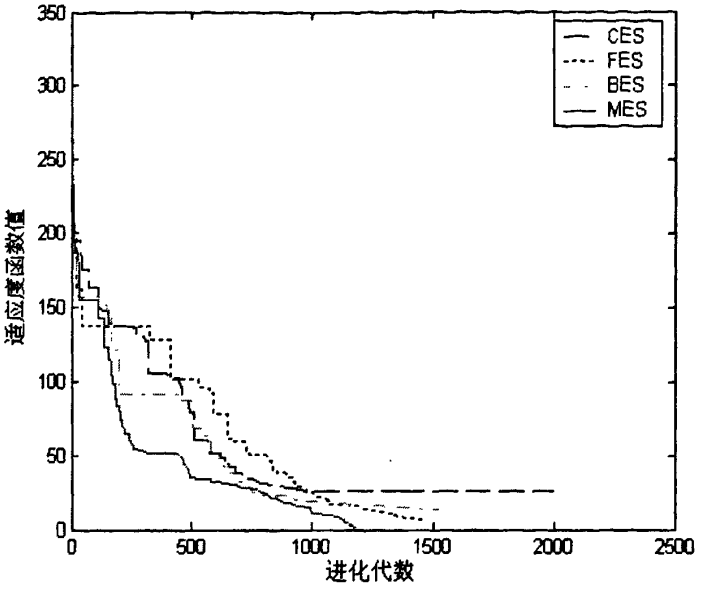

图 2-12 函数 f4 的进化过程

Figure 2-12 evolution of the function f4

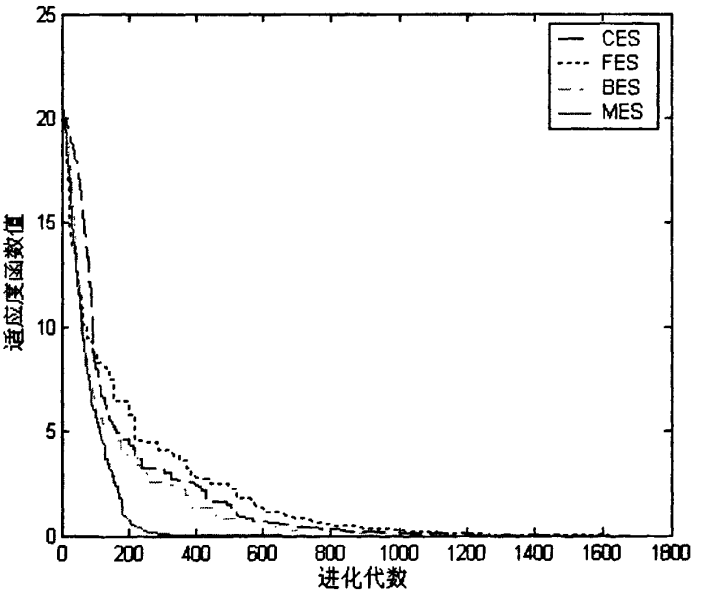

图 2-13 函数 f5 的进化过程

Figure 2-13 evolution of the function f5

2.3.2 实验二

对于 $f6 \sim f8$ 由于维数较高(N=100)，计算的复杂性增加，收敛精度设置为 $|f - f_{\max}| < 0.000001$，算法需要搜索到与最优解距离非常近的距离才能收敛，

$K_{MAX}=10000$，$N=1000$，其他参数与实验一相同。

每个函数进行测试时均运行 20 次，将收敛到目标值的平均代数作为衡量算法收敛速度的标准。用 20 次运行中收敛的总次数作为衡量算法稳定性的标准。最优个体是将每次运行得到的。函数 $f6\sim f8$ 的实验结果如表 2-2 所示。

表 2-2 CES, FES, BES and MES 算法的实验结果（函数 $f6\sim f8$）

Table 2-2 Comparison among CES, FES, BES and MES on functions $f6\sim f8$

函数	算法	总收敛次数	平均迭代代数	理论上的最大值	最优个体函数值的平均值
f6	CES	0	10000	0	-24.8714
	FES	0	10000		-82.8956
	BES	0	10000		-61.897
	MES	0	10000		-75.6026
f7	CES	0	10000	41898.288	25151.4
	FES	0	10000		35834.7
	BES	0	10000		36913.1
	MES	0	10000		36809.3
f8	CES	0	10000	78.33236	71.3288
	FES	0	10000		77.2663
	BES	0	10000		77.549
	MES	0	10000		77.3605

从表 2-2 可以看出对于高维函数 f6，f7 和 f8，不管是 CES、FES、BES 还是 MES 算法，性能都不佳，经过 10000 代的进化都没有求出全局最优解。这主要是因为当问题规模比较大(N>=100)时，计算的复杂度大大增加。其次，不管是高斯变异还是柯西变异，变异方向都具有随机性，往往使变异的效果互相抵消，很难得到进化了的后代个体，导致在求解大规模问题时的收敛速度很低，甚至停顿。例如个体 X 在第 i 维产生有效的变异，个体的适应度提高了，但在第 j 维产生的变异使得个体的适应度降低，两者相互抵消。当维数比较大时，这种问题常常发生。

总之，MES 算法不适合求解高维复杂问题，本文将在四章借鉴协同进化的思想，在 MES 算法的基础上，提出一种新的适用于求解高维复杂问题的进化策略算法。

2.4 收敛分析

王向军等[59]提出了两类不收敛的情况。第一类是指由于种群陷入局部极值不能逃离。第二类是指种群中的某些个体虽然进入了全局最优解的邻域但由于变异过大与全局最优解的距离始终没达到精度的要求。

将 MES 算法与 CES 算法在两种情况下的收敛结果进行对比。

情况一：假设某一时刻整个种群的个体都进入局部极值 x3 的一个较小的邻域内，且未有个体进入全局最优解所在的区域[x1，x2]。参数的选取如下：初始标准差$\sigma_0 = 0.1$，种群大小 N = 40，x1 = 3.5，x2 = 4，x3 = 5。

在 CES 算法中，种群经过一次变异进入到区域[x1，x2]的概率为[59]：

$$P_1 = 40 \times P_{G(0,0.1^2)}(-1.5 < X \le -1) = 3.05 \times 10^{-22} \qquad (2.16)$$

因此要经过至少3.2×10^{21}代后，才有一个个体进入到进入到区域[x1，x2]。由于这个代数太大，种群陷入第一类不收敛[59]。

情况二：假设某一时刻整个种群的一个个体进入全局最优解 x4 附近的区域[x5，x6]。 其中，x4 = 0，x5 = 0.0001，x6 = 0.0002.

在经典的进化策略算法中，该个体经过一次变异进入收敛精度对应的区域[x4，x5]的概率为[59]：

$$P_2 = P_{G(0,0.1^2)}(-0.0001 < X \le 0) \approx 4 \times 10^{-4} \qquad (2.17)$$

因此大约需要经过 2500 代才有个体满足精度的要求。当设定最大进化代数小于 2500 代时，种群陷入第二类不收敛[59]。

在 MES 算法中，假设普通子群的规模是精英子群的 3 倍。对于情况一，MES 算法对普通子群采用柯西变异，普通子群中的每个个体进入到区域[x1，x2]的概率为：

$$P_{C(1)}(-1.5 < X \le -1) = \int_{1.5}^{1} f_{C(1)}(x)dx = \int_{1.5}^{1} \frac{1}{\pi(1+x^2)}dx \approx 0.063 \qquad (2.18)$$

整个种群进入到区域[x1，x2]的概率为：

$$P_1 = 30 \times P_{C(1)} + 10 \times P_{G(0,01^2)} \geq 30 \times P_{C(1)} = 1.89 \qquad (2.19)$$

因此经过一次变异大概就有两个个体进入到区域[x1，x2]。

对于情况二，当精英子群中的一个个体进入区域[x5，x6]，MES 算法对精英子群采用指数递减的高斯变异算子标准差σ_s，只要 μ 选取合适，总可以使$\sigma_s \approx 0.0001$。该个体经过一次变异进入收敛精度对应的区域[x4，x5]的概率为：

$$P_{G(0,0.0001^2)}(-0.0001 < X \leq 0) = \int_{-0.0001}^{0} f_{G(0,0.0001^2)}(x)dx \approx 0.3413 \qquad (2.20)$$

因此当有三个个体进入到该区域，经过一次变异，大概就有一个个体可以满足收敛精度的要求。

正是由于 MES 算法将种群划分为不同规模的两个子群，对精英子群使用递减的高斯变异算子，对普通子群使用柯西变异算子，保证了整个种群在解空间具有尽可能分散的全局搜索能力的同时在局部具有尽可能精细的局部搜索能力。

2.5 本章小结

本章首先首先详细介绍了进化策略的基本技术，从进化策略问题的表达、初始种群的产生、适应值计算、重组、变异、选择、终止等方面进行论述。针对进化策略收敛迅速，但存在后期收敛能力欠佳，易陷入局部极值点的弱点，从平衡进化策略的局部搜索能力和全局搜索能力出发，提出了一种基于双种群的改进进化策略算法(MES)。

其次对 MES 算法的变异策略和种群的划分进行分析介绍，通过对算法进行理论分析说明其正确性。

再通过典型的测试函数对 MES 算法的性能进行测试，并对结果进行分析。对于中低维函数(N<30)，MES 算法具有良好的性能。对于高维复杂问题，MES 性能不佳。本文将在四章提出一种新的适用于求解高维复杂问题的进化策略算法。

最后通过两种典型的情况对 MES 算法的收敛性进行分析。

第三章 适用于高维可分解问题的改进进化策略算法

提高收敛速度和全局收敛能力仍然是进化策略的研究热点，特别是对于高维复杂问题(维数大于 100)，然而大部分对进化策略的分析和实验结果都是针对维数不高（30 维以下）的问题。

进化策略采用了达尔文的适者生存、强调竞争的进化思想，然而在自然界中，生物之间不仅存在竞争也存在协作。协同进化算法在进化算法的基础上，考虑了种群与环境之间、种群与种群之间在进化过程中的协调。

在 MES 算法的基础上结合协同进化的思想，本文提出了一种新的求解高维可分解问题的算法，即基于合作型协同进化的改进进化策略算法(Modified Evolutionary Strategies with Cooperative Coevolution，MESCC)算法。

3.1 协同进化算法简介

协同进化(Coevolution)一词最早是由 Ehrlich 和 Raven 在讨论植物和植物昆虫相互之间的进化影响时提出的[61]。

协同进化算法(Coevolution Algorithm，CEA)是针对遗传算法的不足而提出的。首先遗传算法在求解复杂问题时，由于搜索复杂量度很高，求解缓慢，容易出现早熟现象，而且在后期搜索中效率较低。此外，遗传算法只是采用基于个体自身适应度的进化模式，适应度是进化的引导方向，而没有考虑群体进化的环境和个体之间复杂联系对个体进化的影响。对于进化策略，存在同样的问题。

协同进化算法借鉴了自然界中的协同进化机制，在进化算法的基础上，考虑了种群与环境之间、种群与种群之间在进化过程中的协调。协同进化算法通过形成和维持稳定的多样化子种群，在问题空间的不同区域中并行地进行有效的搜索，从而克服遗传飘移的均匀收敛趋势，实现求解高维复杂问题[62]。

目前协同进化算法大致可以分为两种[62]：合作型协同进化算法(Cooperative Coevolution Algorithm)和竞争型协同进化算法(Competitive Coevolution Algorithm)。

3.1.1 合作型协同进化算法

合作型协同进化算法是由 Poter 等人提出的。其基本思想是首先将目标问题的解向量分解为多个子元素，然后每个子元素通过一个物种种群进行进化，直到问题得到求解。每个物种种群内某个个体的适应度通过该个体和来自其他物种种群个体组合而成的解向量来评价[63]。

通常，对于复杂系统的优化问题可以采用分而治之的思想进行求解。首先将系统划分为一系列子系统，再对各个子系统分别进行优化，然后从整体上协调整个系统。子系统的优化和整体协调往往需要交替进行，直到找到系统的最优解。合作型协同进化算法正是采用了这种思想。

图 3-1 给出了合作型协同进化算法模型。合作型协同进化算法借鉴了生物种群间相互合作的思想。首先将待求解问题分解为一系列相互联系的子问题，每个问题对应一个物种，物种由一系列个体组成。然后重复以下两步操作，直到求得问题的最优解或满意解：

(1) 对每个物种种群进行进化。

(2) 从每个物种种群中选取一个个体，合并成新的系统，并对系统进行评估。

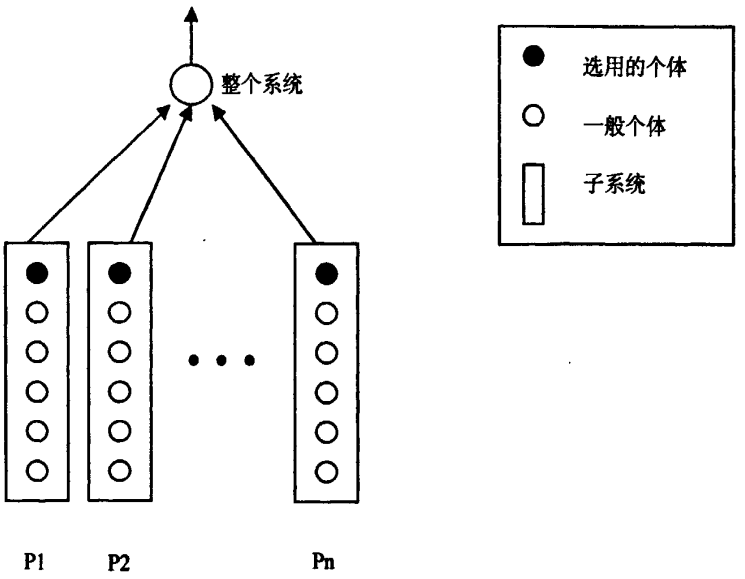

图 3-1 合作型协同进化模型

Figure 3-1 Model of Cooperative Coevolution

在合作型协同进化算法中，每个物种种群的进化是独立的，即每个个体只与

本物种种群的个体发生重组、变异等操作。而物种种群之间是相互合作的，通过信息交换获得个体的适应度，从而使得物种种群间互相指导进化，最终使整个系统的优化问题得到解决。

合作型协同进化算法的关键在于对目标问题进行合理的分解，并设计物种种群间的合作方式，用于计算个体的适应值[62]。物种种群中个体的适应度需要从其他种群中抽取一个代表来组成问题的解。选取的方法主要有三种[62]：

(1)贪心法，即选取物种种群中最好的个体。

(2)保守法，即选择物种种群中最差的个体。

(3)随机法，即随机选择物种种群中的个体。

3.1.2 竞争型协同进化算法

竞争型协同进化算法是指多个物种种群通过适应度的关联同时进化，个体的适应度通过与另一种群的个体直接竞争得到[62]。竞争型协同进化算法鉴了生物种群间竞争协同进化的思想，首先构造两个相互竞争的种群：一个代表问题的解集，另一个代表测试集。图 3-2 给出了竞争型协同进化算法的框架。

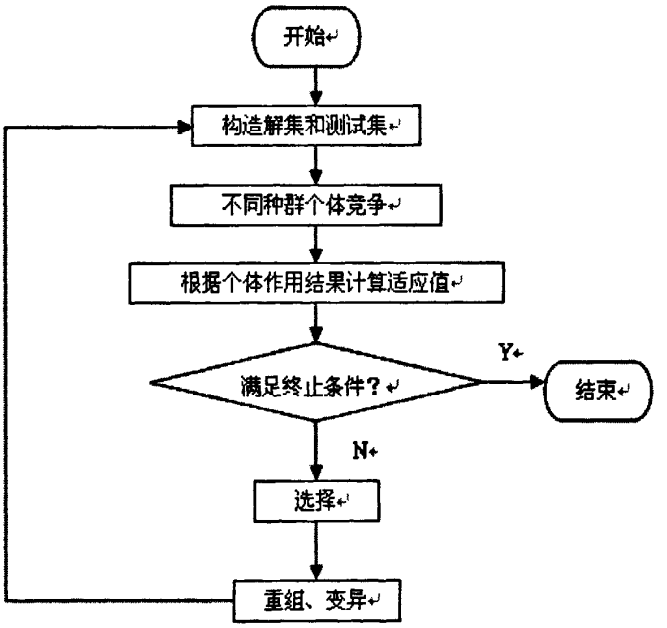

图 3-2 合作型协同进化算法模型

Figure 3-2 Framework of Competitive Coevolution Algorithm

竞争型协同进化算法通过维持种群间的竞争平衡状态，两个种群相互竞争、

相互驱使来提高各自的性能和复杂性，最终得到问题的最优解或满意解。竞争型协同进化算法和合作型协同进化算法的区别在于：

(1) 竞争型协同进化算法采用相对适应度来衡量个体，不需要设计适应度函数。

(2) 竞争型协同进化算法没有对问题进行分解，每个个体都表示一个完整的搜索空间的点。

3.2 MESCC 算法

对进化策略大部分研究和改进都是针对维数不高（30 维以下）的问题。对于高维复杂问题(维数大于 100)的研究相对较少。王湘中等[48]提出了适用于高维优化问题的改进进化策略，建立了单基因变异与均匀变异相结合、使用精英繁殖、递减型策略参数、小种群规模的($\mu + \lambda + k$)-ES，但该算法需要进化时间较长，收敛速度慢。本文借鉴了协同进化的思想，在 MES 算法的基础上，提出一种适合与求解高维复杂问题的 MESCC 算法。

MESCC 算法是在 MES 算法的基础上结合协同进化的思想提出的一种改进的进化策略算法。图 3-3 给出了 MESCC 算法的模型。该算法首先将目标问题分解为 N 个子问题，将所有个体划分为 N 个团队，每个团队负责处理一个子问题，使用 MES 算法进行进化。不同的团队之间采用协作操作，通过团队间的合作求取团队中每个个体的适应值，再从每个团队中选取最优的个体，合并形成当前的最优解。团队的进化和团队间的合作是交替进行的，直到进化得到目标问题的最优解或满意解。各个团队使用 MES 算法求解各自的子问题，将团队划分为一个精英种群 P_S 和一个普通种群 P_L。精英种群 P_S 规模较小，用于存放团队中的最优秀的个体，在变异时使用递减的高斯变异算子。普通种群 P_L 规模较大，用于存放团队中的普通个体，在变异时使用柯西变异算子。由于团队使用了 MES 算法进行进化，因此每个团队在各自的解空间中，都具有良好的全局搜索能力和精细的局部搜索能力。不同的团队可以根据所有处理的子问题的不同，将种群做不同的划分，不同团队的精英种群和普通种群的规模可以相同，也可以不同，这样增加了算法的灵活性。

3.2.1 问题分解

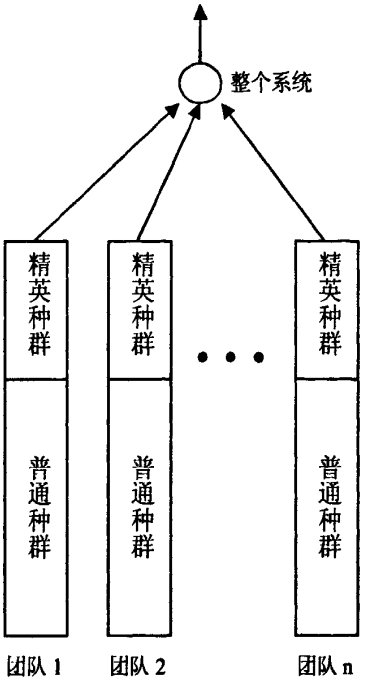

图 3-3 MESCC 进化模型

Figure 3-3 Model of MESCC Evolution

在 MESCC 算法中首先要对目标问题进行合适的分解，分解为一系列相互关联的子问题，每个子问题对应一个团队。分解的方法是灵活多变的，对不同的问题可以采用不用的分解方法，同一个问题也可以有多种的分解方法。例如对与求解球模型函数 $y=\sum_{i=1}^{N}x_i^2$，最自然的一种分解方法是将问题分解为 N 个子问题 $y=x_i^2$，i=1,2,...,3，一个团队处理一个维数的问题。也可以划分为两个子问题，这样每个团队需要处理一个 $N/2$ 维的问题。

对问题的合理分解带来了三方面的好处：

(1) 降低了问题的复杂度，搜索空间大大减小，搜索效率提高，得到解的质量更好。

(2) 个体表示的复杂性降低。个体表示的不是完整解，而仅仅是完整解的一部分。完整解是由每个团队中选取一个个体合并起来得到。

(3) 使得 MESCC 算法不仅在个体级别上实现了并行搜索，也在团队级别上实现并行进化，提高了搜索的效率。

3.2.2 团队协作

在 MESCC 算法中团队合作主要表现在两方面：团队中个体的适应度的计算和当前最优解的求取。

在 MESCC 算法中，由于团队中的每个个体都只是完整解的一部分，通常需要通过团队合作，相互交换信息来求取各自团队中的个体的适应度。例如对于求取函数 $y=\sum_{i=1}^{N-3} x_i^2+x_{i+1}^2+x_{i+2}^2$ 最值的优化问题，如果将问题按照维数划分为 N 个子问题的话，那么团队中每个个体需要借助另外两个相关联团队中的两个个体代表才能够求取。在 MESCC 算法中，代表的选取采用贪心法，即选取团队中最优秀的个体。

同样，由于完整解是由 N 团队中的 N 个个体组合而成的。在求取当前最优解需要团队协作，从每个团队中选取一个代表出来，组成问题的解。这里，代表的选择也是采用贪心法。

3.2.3 算法流程

MESCC 算法借鉴了生物种群间的相互合作的思想，首先把问题分解为几个子问题，每个子问题对于一个团队，团队是使用 MES 算法对子问题进行处理。图 3-4 给出了 MESCC 算法的流程。MESCC 算法的步骤如下：

1. 参数初始化。设置总个体数目 N，最大的进化代数 K_{MAX}，团队中个体的个数 *TeamSize*，选取高斯变异算子标准差 σ_0，设置种群的进化代数 $k=0$，设置进化终止条件。

2. 问题分解。将目标问题分解了 M 个相互联系的子问题，将所有个体划分成 *TeamNum* 个团队 Team，每个问题对一个团队，团队的数目 *TeamNum*= M。

3. 计算团队中每个个体的初始适应值，并从每个团队中抽取一个最优秀的个体用以计算初始的最优解。

4. 终止检验。如果满足进化终止条件，则输出结果，终止计算。否则，转 5。

5. 种群划分。对于第 i 个团队 Team(i)，从中选取最优秀的 Si 个个体组成精英子群 P_S，剩下的 Li 个个体组成普通子群 P_L。其中，$2Si \le Li \le 5Si$。对每个团队都执行这样的操作，不同团队的种群可以采用相同的划分方法，也可以不同。

6. 变异产生后代。在第 i 个团队 Team(i)中，对精英子群 P_S 采用递减的高斯变

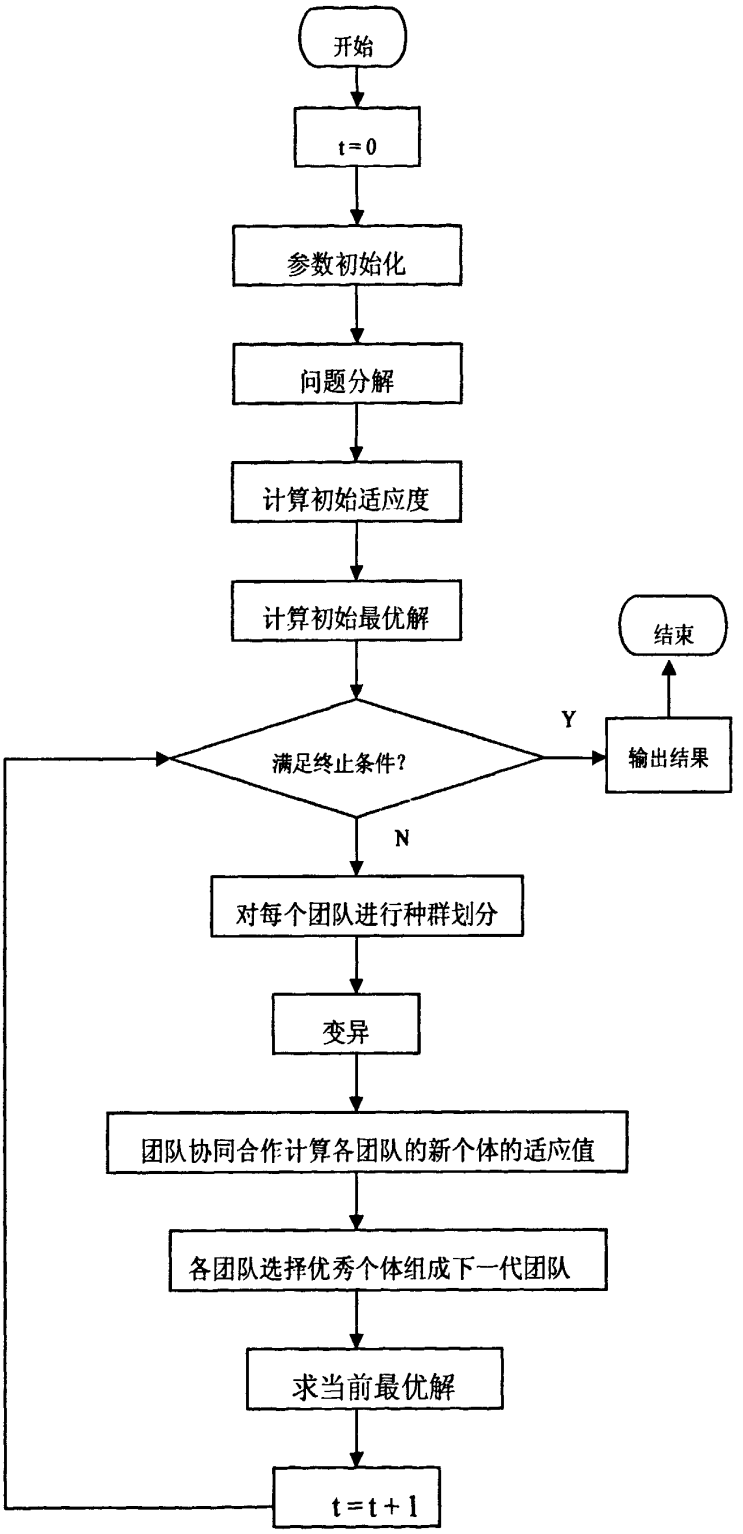

图 3-4 MESCC 算法流程

Figure 3-4 Process of MESCC Algorithm

异算子，P_S变异后产生种群P'_S。对普通子群P_L采用柯西变异，P_L变异后产生种群P'_L。对每个团队都执行这样的操作。

7. 团队协作计算新个体的适应值。对于第 i 个团队 Team(i)新产生的个体，即种群P'_S和种群P'_S中的个体，首先从其他 *TeamNum*-1 个团队中分别选取最优秀的个体，然后利用目标函数的适应值公式计算出每一个新个体的适应值。对每个团队都执行这样的操作。

8. 选择。对于第 i 个团队 Team(i)，种群 P_S，P'_S，P_L，P'_L 组成临时种群 P_T，再从 P_T 中选择出最优秀的 N 个个体组成新的种群 $P^{(k+1)}$。对每个团队都执行这样的操作。

9. 计算当前的最优解。从 *TeamNum* 个团队中分别选取最优秀的一个个体，通过目标函数的适应值公式计算当前的最优解。

10. k = k + 1，转 4。

3.3 模拟实验

为了分析和验证本文提出的 MESCC 算法的性能，本文选用了五个标准函数进行试验[39]：

1. $f_1 = -\sum_{i=1}^{N} x_i^2$，球模型函数，定义域为$(-20,30)^N$，维数 N=100，在点(0,0,...,0)处取得全局最大值 0。其二维函数图形如图 3-5 所示。

2. $f_2 = -\sum_{i=1}^{N-1}[100(x_i - x_{i+1})^2 + (x_i - 1)^2]$，定义域为$(-5.12,5.12)^N$，维数 N=100，在点(1,1,...,1)处取得全局最大值 0。其二维函数图形如图 2-6 所示。

3. $f_3 = \sum_{i=1}^{N} x_i \sin(\sqrt{|x_i|})$，定义域为$(-512,512)^N$，维数 N=100，理论最大值为 418.98288*N。当 N 取 100 时，在点(420.9687, 420.9687,..., 420.9687)处取得全局最大值 41898.288。其二维函数图形如图 2-7 所示。

4. $f_4 = \sum_{i=1}^{N}\{-x_i^2 + 10\cos(2\pi x_i) - 10\}$，定义域为$(-5.12,5.12)^N$，维数 N=100，在点(0,0,...,0)处取得全局最大值 0。其二维函数图形如图 3-6 所示。

5．$f_5 = 20\exp\{-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}\} + \exp\{\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i^2)\} - 20 - \exp(1)$，定义域为$(-32,32)^N$，维数 N=100，在点(0,0,…,0)处取得全局最大值 0。其二维函数图形如图 3-7 所示。

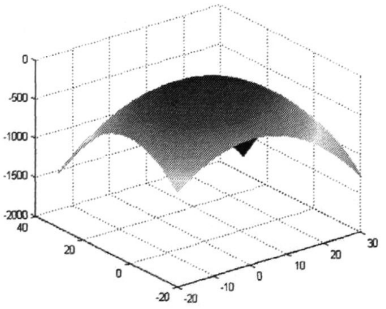

图 3-5　函数 1

Figure 3-5 function1

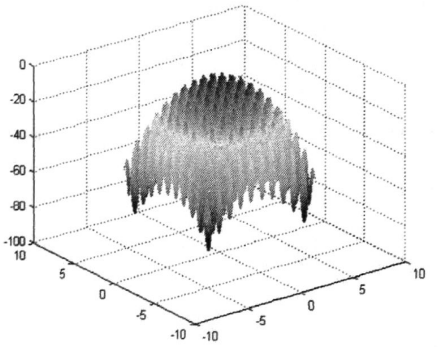

图 3-6　函数 4

Figure 3-6 functions 4

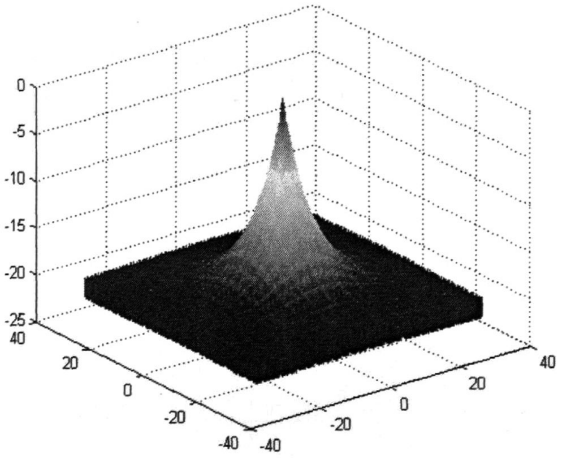

图 3-7　函数 5

Figure 3-7 function5

其中，*f1* 和 *f2* 是单峰函数，*f3*、*f4* 和 *f5* 是多峰函数，其局部极值点的个数随着问题维数的增长而增长，100 维时具有非常多的局部极值，能够很好地测试算法的性能。

3.3.1 实验一

将 MESCC 算法与 MES 算法、CES 算法、FES 算法、BES 算法在性能方面

进行比较。对于 $f1 \sim f5$ 参数的选取如下：对于问题按维数进行划分，团队个数 TeamSize=N，每个团队的大小 $TeamSize$=10。所有算法采用相同的种群规模，即全部个体的数目为 N*10=1000。最大的进化代数 $K_{MAX}=10000$，收敛精度设置为 $|f-f_{\max}|<0.000001$。团队中，变异参数 $\mu=0.001$，初始标准差 $\sigma_0=1$，普通子群 P_L 的规模是精英子群 P_S 规模的 4 倍。

每个函数进行测试时均运行 20 次，将收敛到目标值的平均代数作为衡量算法收敛速度的标准。用 20 次运行中收敛的总次数作为衡量算法稳定性的标准。最优个体是将每次运行得到的。函数 $f1 \sim f5$ 的实验结果如表 3-1 所示。图 3-8、3-9、3-10、3-11、3-12 分别给出了函数 $f1$、$f2$、$f3$、$f4$、$f5$ 的进化过程曲线图。

从表 3-1 可以看出在处理高维复杂问题时，MESCC 算法的性能要大大优于 MES、CES、BES 和 MES。在收敛速度方面，对于函数 $f1$ 和 $f2$，MESCC 算法只需要经过几百代的进化就可以得到满足收敛精度要求($|f-f_{\max}|<0.000001$)的解，对 $f3$、$f4$ 和 $f5$ 这三个在 100 维时具有非常多的局部极值的函数，也只是需要经过一千多代的进化就可以收敛到全局最优点。在收敛稳定性方面，MESCC 算法在每个函数的 20 次运行中，都能够收敛到全局最优点，体现出良好的稳定性。

而其他几种算法除了简单的球模型函数 $f1$ 外，均无法在规定的 10000 代内收敛到全局最优点，都陷入了局部极值点，效果不理想。这主要是因为随着维数的增加，问题的复杂度越来越高，这几种算法中个体的复杂度也随之加大(例如在 100 维中每个体需用一个 100 维的变量来表示)。其次，这几种算法中使用的柯西变异和高斯变异的变异方向都具有随机性，往往使变异的效果互相抵消，很难得到进化了的后代个体，导致在求解大规模问题时的收敛速度很低，甚至停顿。当维数比较大时，这种相互抵消的现象常常发生。

MESCC 算法借鉴了协同进化的思想，首先将问题分割为一系列子问题，每个子问题的复杂性都明显低于原问题。由每个团队来处理一个子问题，团队中个体的复杂性也降低(本实验中每个体只需要一个一维的变量表示)这样搜索空间大大减小，搜索效率提高。在每个团队中使用了 MES 算法，因此在子问题的解空间中，团队具有良好的全局搜索能力和精细的局部搜索能力。此外，由于有一系列团队协同处理问题，使得 MESCC 算法在团队级别和个体级别上实现了并行搜索。

MESCC 算法在求解高维复杂问题时表现出了良好的性能，与 MES、CES、

表 3-1 MESCC,CES, FES, BES and MES 算法的实验结果（函数 $f1 \sim f5$）

Table 3-1 Comparison among MESCC, CES, FES, BES and MES on functions $f1 \sim f5$

函数	算法	总收敛次数	平均迭代代数	理论上的最大值	最优个体函数值的平均值
f1	MESCC	20	525.2	0	$-9.1396*10^{-7}$
	CES	20	2592.45		$-9.8864*10^{-7}$
	FES	20	3234.8		$-9.94462*10^{-7}$
	BES	20	2432.4		$-9.89046*10^{-7}$
	MES	20	2140.3		$-9.79192*10^{-7}$
f2	MESCC	20	728.15	0	$-9.62878*10^{-7}$
	CES	0	10000		-24.8714
	FES	0	10000		-82.8956
	BES	0	10000		-61.897
	MES	0	10000		-75.6026
f3	MESCC	20	1258.45	41898.288	41898.288
	CES	0	10000		25151.4
	FES	0	10000		35834.7
	BES	0	10000		36913.1
	MES	0	10000		36809.3
f4	MESCC	20	1154	0	$-9.987*10^{-7}$
	CES	0	10000		-72.618
	FES	0	10000		-14.7732
	BES	0	10000		-20.9244
	MES	0	10000		-11.9395
f5	MESCC	20	1159.2	0	$-9.83495*10^{-7}$
	CES	0	10000		-0.0270474
	FES	0	10000		-0.0270474
	BES	0	10000		-0.0270474
	MES	0	10000		-0.0270474

BES 和 MES 相比更适合用来处理高维复杂问题。

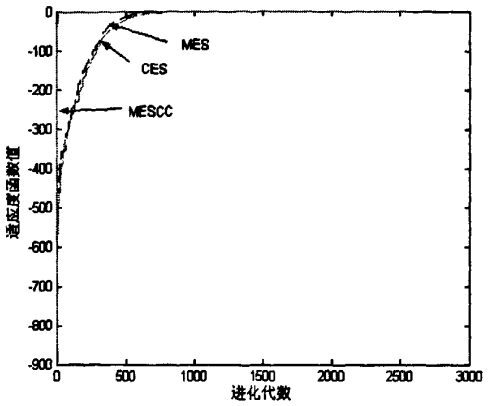

图 3-8 函数 f1 的进化过程

Figure 3-8 evolution of the function f1

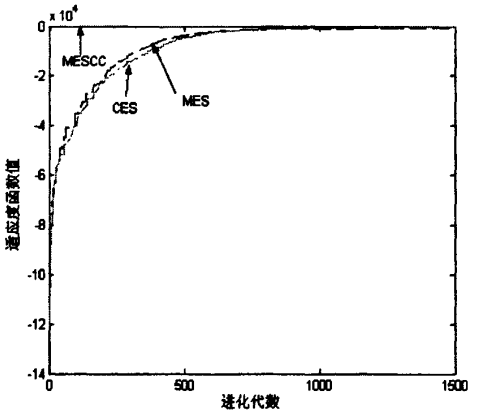

图 3-9 函数 f2 的进化过程

Figure 3-9 evolution of the function f2

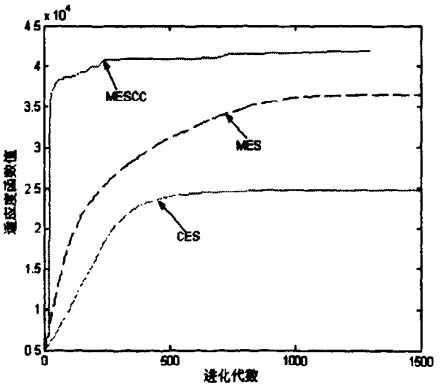

图 3-10 函数 f3 的进化过程

Figure 3-10 evolution of the function f3

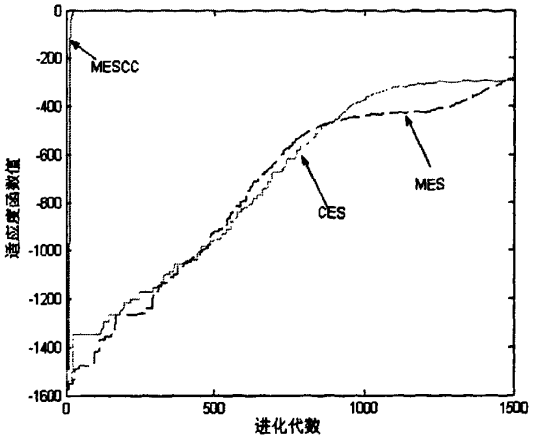

图 3-11 函数 f4 的进化过程

Figure 3-11 evolution of the function f4

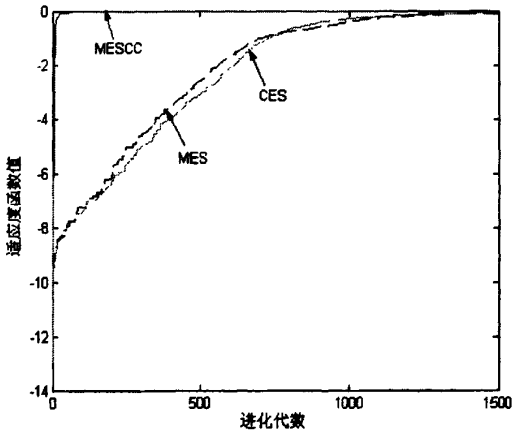

图 3-12 函数 f5 的进化过程

Figure 3-12 evolution of the function f5

从图 3-8、3-9、3-10、3-11、3-12 给出的进化过程曲线可以看出 MESCC 算法在进化初期收敛速度非常快，在几十代之内就可以到达全局最优点的邻域，这主要是因为在 MESCC 算法中，目标问题被分解为子问题，子问题的复杂度降低，易于求解。此外，由于团队中的普通种群使用了柯西变异算子，产生大的变异的概率高，容易跳出局部极值点。在进化后期，MESCC 算法也能够较快地得到收敛精度要求较高的全局最优点。这主要是因为目标问题被分解为子问题后，团队需要搜索空间大大减小，提高了搜索效率。此外，团队中的精英种群存放在团队中最优秀的个体，而且精英种群使用了递减的高斯变异算子，能够产生足够小的变异，实现了精细的局部搜寻。

3.3.2 实验二

试验 MESCC 算法在处理不同维数的高维函数优化问题时的性能，并进行分析比较。对于函数 $f1 \sim f5$ 维数 N 分别取 100，300，700 和 1000。对于问题按维数进行划分，团队个数 *TeamSize=N*，每个团队的大小 *TeamSize*=10。全部个体的数目为 N*10。其他的参数与实验一相同

表 3-2 MESCC,CES, FES, BES and MES 算法的实验结果（函数 $f1 \sim f5$）

Table 3-2 Comparison among MESCC, CES, FES, BES and MES on functions $f1 \sim f5$

函数	维数	总收敛次数	平均迭代代数	理论上的最大值	最优个体函数值的平均值
f1	100	20	525.2	0	$-9.1396*10^{-7}$
	300	20	681.05		$-9.77417*10^{-7}$
	700	20	774.7		$-9.39016*10^{-7}$
	1000	20	791.55		$-9.99597*10^{-7}$
f2	100	20	728.15	0	$-9.62878*10^{-7}$
	300	20	862.1		$-987343.*10^{-7}$
	700	20	955.75		$-9.97969*10^{-7}$
	1000	20	1211.8		$-9.93175*10^{-7}$
f3	100	20	1258.45	41898.288	41898.288
	300	20	1481.68	125694.864	125694.864
	700	20	1671.1	293288.016	293288.016
	1000	20	1915.45	418982.88	418982.88
f4	100	20	1154.1	0	$-9.987*10^{-7}$
	300	20	1256.8		$-9.87327*10^{-7}$
	700	20	1331.55		$-9.95757*10^{-7}$
	1000	20	1388.4		$-9.96313*10^{-7}$
f5	100	20	1159.2	0	$-9.83495*10^{-7}$
	300	20	1237.85		$-9.96029*10^{-7}$
	700	20	1295.75		$-9.95387*10^{-7}$
	1000	20	1361.05		$-9.89867*10^{-7}$

每个函数进行测试时均运行 20 次，将收敛到目标值的平均代数作为衡量算法收敛速度的标准。用 20 次运行中收敛的总次数作为衡量算法稳定性的标准。最优个体是将每次运行得到的。函数 $f1 \sim f5$ 的实验结果如表 3-2 所示。

从表 3-2 的结果可以看出 MESCC 算法在处理维数从 100 到 1000 的高维复杂函数优化问题时表现出了良好的性能。在收敛速度方面，对于 100，300，700 和 1000 维数的优化问题都在 2000 代之内求出符合收敛精度要求的最优解。在收敛稳定性方面，MESCC 算法在每个函数的 20 次运行中，都能够收敛到全局最优点，体现出良好的稳定性。

维数越高，MESCC 算法在求解时需要付出更大的代价，本实验中主要是两方面的代价：第一更长的进化时间，第二.更多的团队。实验中采用按维数划分目标问题的方法，因此 100 维需要 100 个团队，1000 维就需要 1000 个团队。这样 1000 维时每个团队处理的子问题的复杂度和 100 维时的复杂度等同。但是，由于 1000 维时目标问题更加复杂，因此需要更长的时间才能找到问题的最优解或满意解。

3.4 本章小结

本章首先针对进化策略在处理高维复杂问题事存在易陷入局部极值点和收敛能力欠佳的弱点，借鉴协同进化的思想，提出一种新的适合求解高维可分解问题的算法，即基于合作型协同进化的改进进化策略算法(MESCC)。

其次对协同进化算法的背景、思想和内容进行介绍，此外，主要介绍了协同进化算法的两大分类：合作型协同进化算法和竞争型协同进化算法。

再通过典型高维的测试函数，将 MESCC 算法与 MES、CES、FES 和 BES 算法的性能进行比较，并对结果进行分析。对于高维复杂问题，MESCC 算法体现出了良好的性能，在收敛速度和收敛稳定性方面都大大优于其他几种算法。

最后使用 100 维到 1000 维的测试函数，对 MESCC 算法测试，MESCC 算法同样表现出了良好的性能。

第四章 改进进化策略在 K-means 聚类算法中的应用

K-means 聚类算法是一种重要的聚类算法，该算法简单有效、可伸缩性强，被广泛地运用到模式识别、图像和语音数据压缩、径向基神经网络学习算法等领域。K-means 聚类算法采用所谓的爬山法(hill-climbing)来搜索最优解，具有较强的局部搜索能力，但是容易受初始选定的聚类中心的影响而陷入局部最优。进化策略算法是一种模拟自然进化规律的全局随机搜索算法，具有较强的局部逃逸能力，而且对初始值不敏感。本章将 K-means 聚类算法和改进的进化策略算法(MES)相结合，提出一种基于改进进化策略的 K-means 聚类算法(K-means Algorithm Based on Modified Evolutionary Strategies，KAMES)，利用了 MES 算法良好的全局搜索能力和 K-means 聚类算法良好的局部搜索能力，提高收敛速度，达到较好的聚类效果。

4.1 K-means 聚类算法简介

4.1.1 K-means 算法的思想和特点

K-means 聚类算法属于一种基于划分的聚类算法。该算法的主要思想是将 n 个对象划分成 k 份($k \le n$)，每一份代表一个类（簇）。K-means 聚类算法接受输入量 k，从给定的 n 个数据对象的数据集中随机 k 个数据对象作为初始聚类中心。然后计算其他数据对象到聚类中心的距离，根据距离最近原则，将数据对象划分到离它最近的聚类中心所在的类中。再重新计算每个类的平均值，把数据对象划分到最相似的某个类。不断重复这一过程直到准则函数收敛使得平方误差函数最小。

K-means 聚类算法一般采用欧氏距离来衡量数据对象的相似性，目标函数可以定义为：

$$J = \sum_{i}^{k} \sum_{j}^{n} w_{ij} d_{ij} \tag{4.1}$$

其中，n 表示数据对象的个数，k 表示类别（簇）数，欧式距离 $d_{ij} = \left\| x_j - z_i \right\|$，

表示数据对象x_j到类C_i的聚类中心z_i的距离，z_i代表类C_i中所有数据对象的平均值。w_{ij}表示数据对象属于哪个类。

$$w_{ij}=\begin{cases}1, & \text{第j个对象属于第}C_i\text{个类}\\ 0, & \text{第j个对象不属于第}C_i\text{个类}\end{cases} \tag{4.2}$$

K-means 聚类算法具有以下特点：

(1)各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。同一聚类中的对象相似度较高，而不同聚类中的对象相似度较小。

(2)在每次迭代过程中都要对每个数据对象的分类进行检查。如果分类不正确，将该数据对象重新分配。在所有数据对象分配完后，修改聚类中心和目标函数值，然后进入下一次迭代。如果所有数据对象的分类正确，则数据对象不会被重新分配，聚类中心也不会发生变化，这时该算法的聚类结果达到了最优，算法运行结束。

4.1.2 K-means 算法的流程

K-means 聚类算法的主要流程如下：

(1)从给定的 n 个数据对象$\{x_1,x_2....x_n\}$中，随机挑选出 K 个点$c_1,c_2....c_k$，作为 K 个类的初始聚类中心。

(2)计算其他的每个数据对象到各个聚类中心的距离$D(x_i,c_j)$,i=1,2,3,..., n, j = 1, 2, 3...k。$D(x_i,c_j)$表示数据对象x_i到聚类 C_j的中心的距离。如果满足：

$$D(x_i,c_k)=\min\{D(x_i,c_j),i=1,2...n,j=1,2...k\} \tag{4.3}$$

将数据对象x_i划分到类C_k中。

(3)根据类中的点重新计算每个类的聚类中心$c_1^{'},c_2^{'}..c_k^{'}$,计算公式为：

$$c_i^{'}=\frac{1}{n_i}\sum_{x_m\in C_i}x_m\text{，i=1,2...k} \tag{4.4}$$

其中n_i为类C_i中点的个数。再用公式（4.1）计算此时的目标函数值J_1。

(4)将数据对象重新分配，并计算新的目标函数值。假设 x_i 属于类 n，如果$\|x_i-z_m\|^2<\|x_i-z_n\|^2$（其中 z_m和 z_n分别是类 m 和类 n 的聚类中心），将数据对象 x_i分配到类 m 中，再重新计算分配后的目标函数值 J_2。

(5)如果$|J_1-J_2|<\varepsilon$(其中 ε 为迭代终止条件)则算法运行结束，输出结果。否则

返回（2）继续执行。

为防止步骤(5)的迭代终止条件不能满足而陷入到无限循环的状态，通常会给出一个固定的最大迭代次数。

4.2 基于改进进化策略的 K-means 聚类算法

基于改进进化策略的 K-means 聚类算法(KAMES)，将 MES 算法良好的全局搜索能力和 K-means 聚类算法良好的局部搜索能力相结合，提高了算法的局部搜索能力和收敛速度，达到较好的聚类效果。

4.2.1 编码方式

K-means 聚类算法的作用是在 N 维空间中的确定 K 个聚类中心，将 n 个数据对象划分到合适的类中。MES 算法采用传统的十进制实数表达问题，目标参数和策略参数都需要编码到染色体中。KAMES 算法采用基于聚类中心的十进制实数编码方式，将聚类中心作为个体的十进制实数编码，每个个体由两部分组成：一个 $N*K$ 的实数序列 S 和变异步长 σ。对于 N 维空间中，基于聚类中心的个体的编码结构如下：

$$(S,\sigma)=((x_{11},x_{12},...,x_{1n},x_{21},x_{22},...,x_{2n},....,x_{k1},x_{k2},...x_{kn}),(\sigma_1,\sigma_2,...,\sigma_k))$$

实数序列 S 中的第一个 N 位表示 N 维空间中第一个聚类中心，第二个 N 位表示 N 维空间中第二个聚类中心，如此类推，第 K 个 N 位表示 N 维空间中第 K 个聚类中心。这种编码方式简单直观，方便计算。

4.2.2 种群初始化

种群初始化就是随机产生初始种群。首先从待分类的数据对象中随机挑选 K 个个体，再将这 K 个个体按照基于聚类中心的十进制实数编码方式进行编码。然后重复进行 P_{size} 次上述操作，生成种群大小为 P_{size} 的初始种群。

再将初始种群划分为一个精英子群 X_S 和一个普通子群 X_L。精英子群 X_S 规模较小，用于存放种群中最优秀的个体。普通子群 X_L 规模较大，用于存放种群中的普通个体。对精英子群 X_S 使用递减的高斯变异算子，对普通子群 X_L 采用柯西变异算子。

4.2.3 适应度函数设计

MES算法中适应度函数用来计算个体的适应值，适应值越高，个体越优秀。K-means聚类算法属于一种基于划分的聚类算法，就是要找到一种划分使得目标函数J的值取得最小。J值越小表示该聚类划分的质量更高。

为了方便技术，基于MES算法的K-means聚类算法借助K-means聚类算法的目标函数J来构造适应度函数：

$$f=\frac{1}{1+J} \tag{4.5}$$

由上述可以看出，聚类划分的质量越高，目标函数的取值J越小，适应度函数f的取值越大。聚类划分的质量越差，目标函数J的取值越大，适应度函数f的取值越小。

4.2.4 K-means操作

Krishma等[64]在1999年提出了K-means操作的概念。K-means操作是K-means算法中的一个阶段。具体操作如下：

(1)根据给定的分类矩阵U，根据公式（4.4）计算新的聚类中心。

(2)将每个数据对象重新分配到距离最近的类中，形成新的分类矩阵U'。

在KAMES算法中，在每一代执行完进化操作之后，对种群中的每个个体执行K-means优化操作，优化后的种群作为下一代的群体进行下一次的迭代进化。这样不仅可以进一步提高MES算法的局部搜索能力，同时还可以提高收敛速度。

4.2.5 算法的流程

KAMES算法的基本思想主要有以下两部分：

(1)从给定的聚类的数据对象中产生出初始种群，并对种群执行进化操作。

(2)对执行完进化操作后的种群执行K-means操作。

以上两部分不断循环执行，直到得出满足要求的聚类效果或算法的迭代次数超过了指定的最大迭代次数。

图4-1给出了KAMES算法的流程图。KAMES算法的步骤如下：

1. 参数初始化。设置数据对象样本个数N，聚类数K，种群大小P_{size}，最大的迭代代数K_{MAX}，选取高斯变异算子标准差σ_0，设置种群的进化代数$k=0$，设置迭代终止条件。

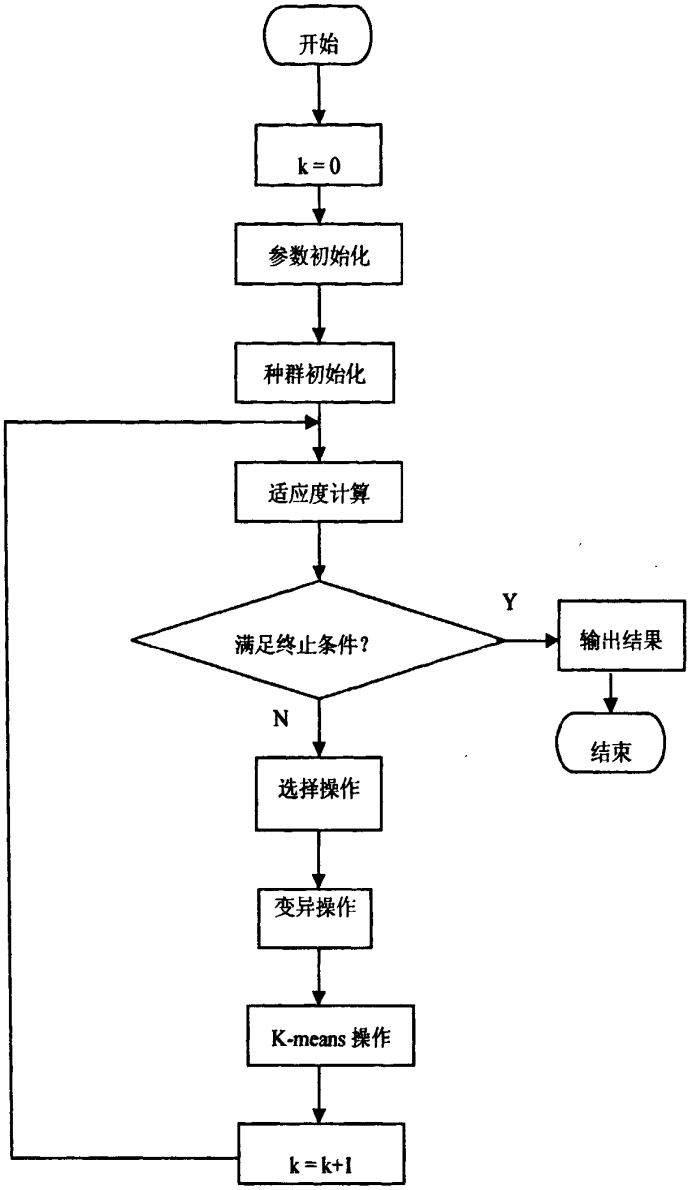

图 4-1 KAMES 算法流程

Figure 4-1 Process of KAMES Algorithm

2. 种群初始化。在 N 个数据对象样本中随机挑选 K 个点作为聚类中心，并进行编码，后重复 P_{size} 次上述操作，生成初始种群。再将初始种群划分为一个精英子群 X_S 和一个普通子群 X_L。

3. 适应度计算。计算种群中每个个体的适应度。

4. 终止检验。如果满足进化终止条件，则输出结果，终止计算。否则，转 5。

5. 根据适应度，执行选择操作。

6. 变异产生后代。对种群中的每个个体使用高斯变异算子。

7. 对新产生的种群中的每个个体执行 K-means 操作。

8. 产生新的种群。$k=k+1$，转 3。

4.3 实验

1920 年左右，植物学家收集了鸢尾花卉数据集，包含了 150 个样本，都属于鸢尾属下的三个亚属，分别是山鸢尾（Iris setosa）、变色鸢尾（Iris versicolor）和维吉尼亚鸢尾（Iris virginica），每种 50 个样本。四个特征被用作样本的定量分析，它们分别是花萼(Sepal)和花瓣(Petal)的长度和宽度。鸢尾花卉数据集的分布如图 4-2 所示。

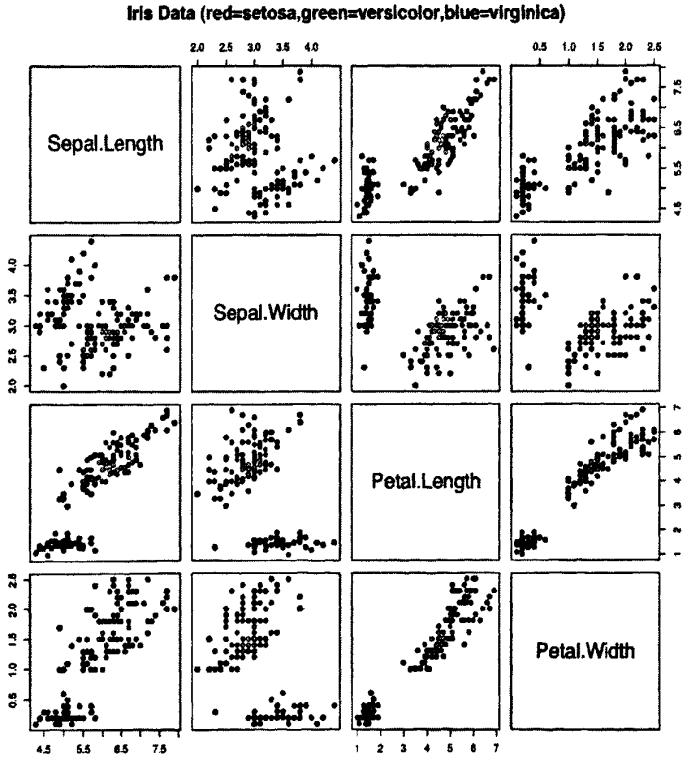

图 4-2 鸢尾花卉数据集的分布图

Figure 4-2 Iris data distribution

将 KAMES 算法与 K-means 算法在性能方面进行比较。对 KAMES 算法，参数的选取如下：种群的大小 P_{size}=40，聚类数 K=3，最大的迭代代数 K_{MAX}=200，选取高斯变异算子标准差 σ_0=3，普通子群 X_L 的规模是精英子群 X_S 规模的 3 倍。。两种算法通过运行 20 次来比较算法的性能。实验结果如表 4-1 所示。图 4-3 给出

了两种算法的进化过程（收敛曲线）。

表 4-1 K-means 算法与 KAMES 算法的聚类性能比较

Table 4-1 the performance comparision between K-means algorithms and KAMES

聚类算法	聚类正确率（%）	目标函数值 J_{min}
K-means	65.8	145.1574
KAMES	86.4	78.9406

由图 4-2 可以看出，鸢尾花卉数据集的每组数据的空间分类不是特别明显。从表 4-1 可以看出，运用 K-means 算法的聚类效果不是很好，正确率只达到了 65.8%，而使用 KAMES 算法正确率提高到了 86.4%。从图 4-3 可以看出，KAMES 算法的收敛效果明显优于 K-means 算法，在每一次的迭代过程中，KAMES 算法的目标函数值都比 K-means 算法小。KAMES 算法与 K-means 算法相比，能够达到更好的聚类效果。

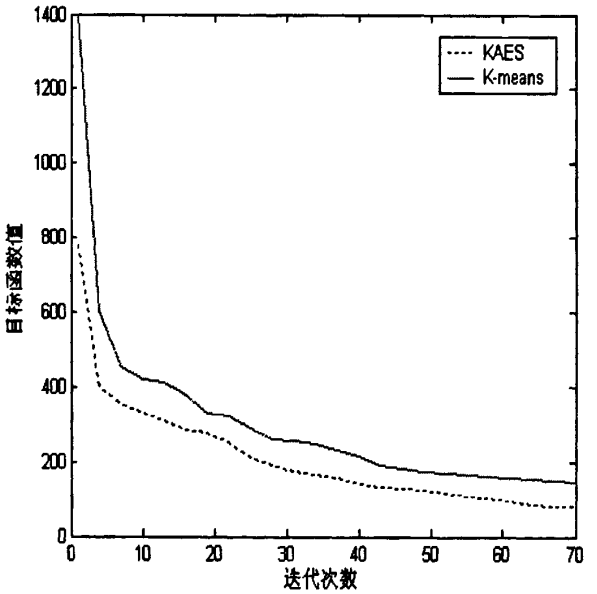

图 4-3 K-means 算法与 KAMES 算法的进化过程

Figure 4-3 evolution of K-means algorithms and KAMES

4.4 本章小结

本章首先介绍了 K-means 聚类算法的思想、特点和主要流程，分析了 K-means 聚类算法的优点和不足。在此基础上，结合 MES 算法的优点，提出了一种基于

MES算法的K-means聚类算法(KAMES)。

其次对KAMES算法的编码方式、种群初始化、适应度函数设计、K-means操作、主要流程等方面进行介绍。

最后通过鸢尾花卉数据集对KAMES算法的性能进行测试。与K-means算法相比，KAMES算法具有更好的聚类性能。

结 论

在科学研究、生产实践中许多复杂的计算问题都可以转化为函数优化问题，进化策略在解决这类问题时表现出了比传统的优化算法更好的性能，成为研究热点。本文针对进化策略收敛迅速，但存在后期收敛能力欠佳，易陷入局部极值点的弱点，提出了一种基于双种群的改进进化策略算法（MES）。通过理论分析和实验表明：MES 具有良好的局部逃逸能力和精细的局部搜索能力，适用于求解中低维(30 维以下)函数优化问题，但对于高维复杂问题，MES 性能不佳。

针对 MES 算法在处理高维复杂问题时表现出的不足，本文在 MES 算法的基础上结合协同进化的思想提出的一种适用于高维可分解问题的算法，基于合作型协同进化的改进进化策略算法(MESCC)。通过几个从 100 到 1000 维的典型高维的测试函数，将 MESCC 与几种典型的进化策略算法性能进行比较，说明对于高维可分解的复杂问题，MESCC 算法具有良好的性能。

最后，将 MES 算法应用到 K-means 聚类算法当中，提出一种基于改进进化策略的 K-means 聚类算法（KAMES）。通过理论分析和实验表明：KAMES 算法比 K-means 算法具有更好的聚类性能。

在以后的工作中，将对以下几个方面进一步研究：首先，进一步提高 MES 算法收敛速度和稳定性，后续研究考虑引入多种群和重组算法，采用新的变异策略。其次，对高维不可分解的问题就行研究，提出求解该类问题的算法。最后，将改进的进化策略应用到更多的实际问题当中。

参考文献

[1]王湘中. 进化策略的变异算子与仿真平台研究[D].中南大学博士学位论文. 2005.

[2]D.B.Fogel. Evolutionary computation: toward a new philosophy of machine intelligence [J]. IEEE Neural Networks Council. New York: IEEE Press, 1995.

[3] K.Miettinen, P.Neittaanmaki, and J.Periaux.Evolutionary algorithms in engineering and computer science: Recent advances in genetic algorithms, evolution strategies, evolutionary programming, genetic programming[C]. New York: John Wiley & Sons. June 1999.

[4]云庆夏. 进化算法[M]. 北京:冶金工业出版社, 2000:148-180.

[5]余有明等. 进化计算的理论与算法[J], 计算机应用研究. 2005.09.

[6]T.Back, U.Hammel and H.P.Schwefel. Evolutionary computation: comments on the history and current state[J]. IEEE Trans On Evolutionary Computation. 1997.1(1):3- 17.

[7]H.J.Bremermann.Optimization through evolution and recombination Self organizing Systems[M].M.C.Yovits, GTJacobi, C.D.Goldstine, eds.Startan Books, Washington D.C. 1962:93~106.

[8]J.H.Holland. Concerning efficient adaptive systems [M]. In M.C.Yovits, eds, Self Organizing Systems, Spartan Books, Washington, D.C.1962.

[9]J.D.Bagley, The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms[J]. Dissertation Abstracts International. 1967, 28(12), 5106B (Univ. Microfilms No.68-7556).

[10]J.H.Holland, Adaptation in Nature and Artificial System.Cambridge[M], MA: MIT Press, 1975.

[11]阎岭.进化策略学习、收敛和逃逸能力的研究及应用[D]. 浙江大学博士学位论文.2005.

[12]L.J.Fogel. On the organization of intellect. Doctoral Dissertation [M], University of California, LosAngeles, 1964

[13]L.J.Fogel, A.J.Owens, and M.T.Walsh, Artificial Intelligence Through Simulated Evolution [M]. New York:Wiley, 1966.

[14]SCHWEFEL H P, BACK T.Evolution strategies I: Tariants and their computational implementation[C]// Proceedings of the Genetic Algorithms in Engineering and Computer Science. New York: Wiley, 1995:111-126.

[15]SCHWEFEL H P, BACK T. Evolution strategies II: Theoretical aspects[C]// Proceedings of the Genetic Algorithms in Engineering and Computer Science. New York: Wiley, 1995:127-140.

[16]Grefenstette J J. Optimization of control parameters for genetic algorithm [J]. IEEE Trans Systman and Cybern, 1986,1 6:122-128.

[17] Koza .J. R. Genetic Programming I [M]. MIT Press, Cambridge, MA, 1992.

[18] Koza .J. R. Genetic Programming II [M]. MIT Press, Cambridge, MA, 1994.

[19]边增远.基于级坐标方向寻优的进化策略及应用[D].广东工业大学工学硕士学位论文. 2006.

[20]王战权.进化算法的研究及其应用[D].东北大学博士学位论文. 1999.

[21]Thomas Back, D.B.Fogel, Zbigniew Michalewicz. Evolutionary Computation 1: Basic Algorithms and Operators [J]. Bristol and Philadelphia: Institute of Physics Publishing, 2000.

[22]I.Rechenberg, Cybernetic Solution Path of an Experimental Problem [M].Royal Aircraft Establishment Library Translation,1965.

[23]I.Rechenberg. Evolutionsstrategie: Optimierung texhnischer Systeme nacb Prinzipien der biologischen Evolution [M]. Holzboog, 1973.

[24]T.Back，Evolutionary Algorithms in Theory and Practice [M].New York:Oxford University Press, 1996.

[25]Schwefel H P. Numerical Optimization of Compute Models [M]. Chichester: Wiley, 1981.

[26]S.M.Yang, D.G.Shao, Y.J.Luo. A novel evolution strategy for multiobjecitve optimization problem [J]. Applied mathematics and computation. 2005.

[27]C.Ebenau, J.Rottschafer, G.Thierauf. An advanced evolutionary strategy with an adapitve penalty funciton for mixed ~discrete structural optimizaiton [J]. Advances

in engineering software. 2005, 36(1):29-38.

[28]夏慧明. 进化策略在数值计算中的一些应用研究[D]. 中国优秀硕士学位论文全文数据库, 2008 .

[29]E.M.Monies and C.A.C.Coello. A simple multi-membered evolution strategy to solve constrained optimizaiton problems [J].IEEE Trans.On Evolutionary computation. 2005, 9(1):1-17.

[30]D.V.Arnold and H.G.Beyer.Investigation of the (μ,λ)-ES in the presence of noise [C]. Evolutionary Computation, 2001.Proceedings of the 2001 Congress, Volume:1, 2001: 332-339.

[31]曹邦武，姜长生，文戎.基于进化策略的模糊控制系统优化设计方法[J].数据采集与处理. 2003.

[32]R.Berlich, M.Kunze.Parallel evolutionary algorithms [J]. Nuclear instruments & methods in physics research.502 (2003):467-470.

[33]王哲,黄海东, 余英林.进化策略在图像恢复中的应用[J].通信学报.1998.

[34]柯晶, 姜静, 李歧强.改进进化策略及其在神经网络训练中的应用.计算机工程与应用[J].2006.

[35]Kursawe, F. Towards self adapting evolution strategies[C].Evolutionary Computation, IEEE Intenraitonal Conference,Volume:1 ,29 Nov.-1 Dec. 1995 :283 288.

[36] L.Gruenz, H.G Beyer. Some observations on the interaction of recombination and self adaptation in evoluiton strategies[C]. Evolutionary Computation, 1999.CEC 99. Proceedings of the 1999 Congress, Volume: 1: 6 9.

[37]Y.Matsumura，K.Ohkura，K.Ueda. Advantages of global discrete ercombination in (μ / μ , λ)-evolutions trategies[C].Evolutionary Computation,2002.CEC'02. Proceedings of the 2002 Congress, Volume:2, 2002:1848-1853.

[38]C.Kappler. A evolutionary algorithms imporved by large mutation[C].Voigt H M, Ebeling w, Rochenberg I eds.Proceedings of the Parallel Problem Solving from Nature IV.Lecture Notes in Computer Science 1141.Berlin, Germany:Springer -Verlag, 1996,346-355.

[39]YAO XIN，LIU YONG. Fast evolution strategies [A]. Control and Cybernetics [C]. Berlin: Spring-Verlag，1997: 196-206.

[40]林丹，李敏强，寇纪淞．进化规划和进化策略中突变算子的若干研究[J]．天津大学学报（自然科学版），2000, 33(5): 627~630.

[41]Gunter R. Local convergence rates of simple evolutionary algorithms with Cauchy mutations[C]. IEEE Transactions on Evolutionary Computation,1997,4(1):249-258.

[42]王云诚，唐焕文.单峰函数最优化问题的一个快速收敛进化策略[J]，小型微型计算机系统.2002, 23 (11):1390~1392.

[43]刘若辰，杜海峰，焦李成.基于柯西变异的免疫单克隆策略[J]，西安电子科技大学学报，2004, 31 (4):551~556.

[44]Change Ming, Kazuhiro O, Kanji U.Some experimental observations of (μ,λ) evolution strategies [J]. Proceedings of the 2001 Congress on Evolutionary Computation.Seoul: Coex Center, 2001, 1:663-670.

[45]王战权，赵朝义，夏云庆进化策略中基于柯西分布的变异算子改进探讨[J].系统工程，1999, 17(4):49~54.

[46]Sang Hwan Lee, Hyo Byung Jun, Kwee Bo Sim.Performance improvement of evolution strategies using reinforcement learning[C].Fuzzy Systems Conference Proceedings,1999.IEEE'99.1991 EEEE Intenrational,Volume:2, 22-25.

[47]Yong Liang, Kwong Sak Leung.Two-way mutation evolutions trategies [C].CEC'02 ,Proceedings of the 2002 Congress on Evolutionary Computation, Volume:1，2002:789-794.

[48]王湘中，喻寿益．适用于高维优化问题的改进进化策略[J]．控制理论与应用，2006, 23(1): 147~151.

[49]夏慧明，梁华，周永权．用双种群进化策略算法求解复函数方程的根[J]．计算机工程与应用，2008，44(7)：78-81.

[50]L.Schonemann. The impact of population sizes and diversity on the adaptability of evolution strategies in dynamic environments[C].evolutionary Computation, CEC2004 Congress，Volume:2, 2004:1270~1277.

[51]Beyer, Hans-Georg. The Theory of Evolution Strategies[C]. Berlin Heidelberg: Springer-Verlag, 2001.

[52]G.Rudolph. Convergence properties of evolutionary algorithms [J]. Kovac, Hamburh, 1997.

[53]G.Yin, G.Rodulph, H .P.Schwefel. Analyzing evolution (1+λ) strategy via stochastic apporximation methods [J]. Evolutionary computation. 1996, 3(4):473~489.

[54]A.E.Eiben, R.Hinterding, Z.Michalewicz.Parameter control in Evolutionary Algorithms [J].IEEE Trans. On Evolutionary computation.1999, 2(3):124~141.

[55]N.G. Kim, J.M.Won, J.S.Lee, etc. Local convergence rate of evolutionary algorithm with combined mutation operator.Evoluitonary Computation [J], 2002. CEC'02. Proceedings of the 2002 Congress on Volume 1, 2002:261-266.

[56]T.Aemeyer, W.Ebeling. Unified description of evolutionary strategies over continuous parameter space [J]. BioSystems, 1997, 41(3):167-178.

[57]周永权，张明，赵斌．基于进化策略方法求任意函数的数值积分[J]．计算机学报，2008，31 (2)：467- 496.

[58]YAO XIN，GUANGMING LIN，LIU YONG．An Analysis of Evolutionary Algorithms Based on Neighborhood and Step Size [A]. Lecture Notes in Computer Science[C]．London： Spring-Verlag，1997：297-307.

[59]王向军，向东，蒋涛，等．一种双种群进化规划算法[J]．计算机学报，2006，29(5)：835-840.

[60]王战权，赵朝义，云庆夏，等．进化策略中变异算子的改进研究[J]．汁算机仿真，1999，16(3)：8-11.

[61]Ehrlich.P.R, Raven.P.H. Butterflies and plants: a study in coevolution evolution[J]. 1965, 18:586-608.

[62]张新征．结合竞争与合作的新型协同进化算法[D]．中国科学技术大学硕士学位论文．2005.

[63]慕彩红，焦李成，刘逸．M-精英协同进化数值优化算法[J]．软件学报，2009，20(11).

[64]Krishna K，Murty M N．Genetic K-Means algorithm [J]．IEEE Trans SystMan Cybern：Part B，1999，29(3)：433-439.

攻读硕士学位期间发表的论文

1. 吴伟民,陈宝财,陈丹,王振华,苏庆. 基于双种群的进化策略算法，计算机应用,2009 年第 5 期. 在学位论文中的相关章节：2.2、2.3.1、2.4

2. AnyviewC: A Visual Practice Platform for Data Structures Course. Weimin Wu, Yongfeng Cao, Baocai chen, Qing Su ， Kailun Li .2009 World Congress on Computer Science and Information Engineering (CSIE 2009).

致 谢

本论文是在我的导师吴伟民教授的悉心指导下完成的。在三年的硕士研究生学习过程中，他在生活上、学习上给予我极大的关怀和帮助。在指导我们的研究方向的时候，他始终细心地参与讨论和研究，给我们提出宝贵的意见。他那渊博的学识、精湛的专业知识、求实的科研精神、严谨的治学态度、诚信的为人风尚、踏实的工作作风，无论是治学还是为人，都是我终生学习的榜样。在此，我要向他表达我崇高的敬意和真挚的感谢。

感谢计算机学院的领导和所有教过我的老师们。感谢他们在学习和生活给我的帮助，他们的教导使我受益匪浅。

感谢林志毅老师和苏庆老师在课题的研究和论文的撰写过程中给予的意见和建议，感谢曹勇锋、陈丹、王振华、黄杰晟、卢琦，以及所有的同学们在学习和生活上给予的帮助，有了他们，我才能顺利完成论文的设计，有了他们，三年的研究生生活才是那么的充实和精彩，再一次感谢他们。

最后，感谢家人对我的支持和鼓励。