学校代码：10491　　研究生学号：220070154

中国地质大学
博士学位论文

差分演化算法的改进及其在聚类分析中的应用研究

博 士 生：龚文引

学科专业：地学信息工程

指导教师：蔡之华　　教授

Charles Ling 教授

本文研究得到了国家 863 计划(2009AA12Z117)、高等学校博士学科点专项科研基金课题(博导类)(20090145110007) 、国家建设高水平大学公派出国留学计划、中国地质大学(武汉)优秀博士论文创新基金的资助。

二〇一〇年五月

A Dissertation Submitted to China University

of Geosciences for the Doctor Degree of Engineering

Differential Evolution Algorithm and Its Application in Clustering Analysis

Ph.D. Candidate： Wenyin Gong

Major： Geosciences Information Engineering

Supervisor： Prof. Zhihua Cai

Prof. Charles Ling

China University of Geosciences

Wuhan 430074 P. R. China

中国地质大学（武汉）研究生学位论文原创性声明

本人郑重声明：本人所呈交的博士学位论文《差分演化算法的改进及其在聚类分析中的应用研究》，是本人在导师的指导下，在中国地质大学（武汉）攻读博士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果，对论文的完成提供过帮助的有关人员已在文中说明并致以谢意。

本人所呈交的博士学位论文没有违反学术道德和学术规范，没有侵权行为，并愿意承担由此而产生的法律责任和法律后果。

学位论文作者（签字）：　　日期：　　年　　月　　日

作者简介

龚文引，男，土家族，1979 年 10 月生，湖南永顺人，分别于 2004 年 7 月和 2007 年 7 月在中国地质大学(武汉)获得工学学士和硕士学位。2007 年 9 月至今在中国地质大学(武汉)计算机学院攻读博士学位，师从导师蔡之华教授。2008 年 10 月初到 2009 年 12 月底，在国家留学基金委的资助下，公派到加拿大西安大略大学计算机系进行联合培养博士研究，师从导师 Charles Ling 教授。主要研究方向是演化计算和演化数据挖掘。博士期间完成了相关课程学习，修满了要求的学分。目前，已在国内外期刊和国际会议上发表论文 20 多篇，其中 SCI-E 检索 3 篇、EI 光盘版检索 9 篇、ISTP 检索 10 篇。

博士期间参加的项目主要有：

1. 国家民用航天项目(涉密);
2. 国家 863 计划(2009AA12Z117): 基于演化数据挖掘的铀矿床高光谱遥感信息自动提取技术研究;
3. 高等学校博士学科点专项科研基金课题(博导类)(20090145110007).

博士期间所获奖励和资助：

1. 2009.01 – 2010.07 中国地质大学(武汉)优秀博士论文创新基金;
2. 2008.10 – 2009.12 国家建设高水平大学公派出国留学计划;
3. 2009.06 GECCO-09 学生旅行奖学金;
4. 2008.12 湖北省优秀硕士毕业论文奖.

博士期间发表的主要学术论文（“△”表示相应的论文已公开发表）：

期刊论文:

1. **W. Gong**, and Z. Cai, “An improved multiobjective differential evolution based on Pareto-adaptive ε-dominance and orthogonal design,” *European Journal of Operational Research*. Elsevier Press. Oct. 2009, 198(2): 576 - 601. (SCI-E & EI 检索) △
2. **龚文引**, 蔡之华, “基于 ε 占优的正交多目标差分演化算法研究,” 计算机研究与发展. 2009, 46(4): 655 - 666. (EI 检索) △
3. **W. Gong**, Z. Cai, and L. Zhu, “An efficient multiobjective differential evolution algorithm for engineering design,” *Structural and Multidisciplinary Optimization*. Springer-Verlag. Apr. 2009, 38(2): 137 - 157. (SCI-E & EI 检索) △
4. **W. Gong**, Z. Cai, and L. Jiang, “Enhancing the performance of differential evolution using orthogonal design method,” *Applied Mathematics and Computation*. Elsevier Press. Dec. 2008, 206(1): 56 - 69. (SCI-E & EI 检索) △
5. Z. Cai, **W. Gong**, C.X. Ling, and H. Zhang, “A clustering-based differential evolution

for global optimization," *Applied Soft Computing*. Elsevier Press. (SCI-E & EI 刊源, 已录用)

6. **W. Gong**, Z. Cai, and C.X. Ling, "DE/BBO: A hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*. Springer-Verlag. (SCI-E & EI 刊源, 已录用)
7. **W. Gong**, Z. Cai, C.X. Ling, and H. Li, "A real-coded biogeography-based optimization with mutation," *Applied Mathematics and Computation*. Elsevier Press. (SCI-E & EI 刊源, 已录用)
8. 蔡之华, **龚文引**, C.X. Ling, "基于进化规划的新型生物地理学优化算法研究," 系统工程理论与实践. (EI 刊源, 已录用)

会议论文:

9. **W. Gong**, A. Fialho, Z. Cai, "Adaptive strategy selection in differential evolution," In *Proc. of Genetic and Evolutionary Computation Conference (GECCO 2010)*. 2010.7. Portland, USA. (EI & ISTP 刊源, 已录用)
10. **W. Gong**, Z. Cai, C.X. Ling, and J. Du, "Hybrid differential evolution based on fuzzy c-means clustering," In *Proc. of Genetic and Evolutionary Computation Conference (GECCO 2009)*. 2009.7: 523 - 530. Montreal, Canada. (EI & ISTP 检索) △
11. **W. Gong**, and Z. Cai, "A multiobjective differential evolution algorithm for constrained optimization," In *Proc. of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, IEEE press. 2008.6: 181 - 188. Hong Kong, China. (EI & ISTP 检索) △
12. **W. Gong**, Z. Cai, C.X. Ling, and B. Huang, "A point symmetry-based automatic clustering approach using differential evolution," In *Proc. of the 4th International Symposium on Intelligence Computation and Applications (ISICA 2009)*, LNCS 5821, Springer-Verlag. 2009.10: 151 - 162. Huangshi, China. (ISTP 检索) △
13. **W. Gong**, Z. Cai, and Y. Huang, "A simple and fast differential evolution for unconstrained global optimization," In *Proc. of the 2nd International Symposium on Intelligence Computation and Applications (ISICA 2007)*. 2007.9: 163 - 167. Wuhan, China. (ISTP 检索) △
14. X. Liu, Z. Cai, and **W. Gong**, "An improved gene expression programming for fuzzy classification," In *Proc. of the 3rd International Symposium on Intelligence Computation and Applications (ISICA 2008)*, LNCS 5370, Springer-Verlag. 2008.12: 520 - 529. Wuhan, China. (EI & ISTP 检索) △

博士期间所参加的学术活动:

1. Oral presentation: CEC-2008;
2. Oral presentation: GECCO-2009;
3. Reviewer: IEEE Transactions on Evolutionary Computation;
4. Reviewer: Information Sciences;
5. Reviewer: European Journal of Operational Research

6. Reviewer: Advances in Engineering Software;
7. Reviewer: Applied Mathematics and Computation;
8. Reviewer: ISICA-09;
9. Reviewer: CCC-10.

差分演化算法的改进及其在聚类分析中的应用研究

博士生：龚文引　　导师：蔡之华 教授, Charles Ling 教授

摘　要

在工程设计、运筹学、生物医学、数据挖掘等领域经常遇到优化问题。如何设计有效的算法求解优化问题，一直是研究的热点。近年来，利用演化算法求解优化问题得到越来越多的关注。由于演化算法具有自适应、自组织、隐并行性等特点，以及对所求解问题不需要连续、可导、单峰等要求，因此利用演化算法求解复杂的优化问题是可行的，且具有普适性。

差分演化算法是演化算法的一种，它是一种简单、高效、快速的基于群体的全局优化算法，具有结构简单、容易实现、鲁棒性强等特点。尽管该算法已经在很多领域取得了成功的应用，但也存在一些不足之处，主要有：1) 缺乏局部搜索能力，使算法在演化后期收敛速度变慢，从而不能满足算法在较少适应值函数评价次数下快速收敛到问题最优解的要求; 2) 传统差分演化算法对缩放因子 F 和杂交概率 CR 设置是很敏感的，对不同的问题需要选择不同的 F 和 CR 值; 3) 在差分演化算法中，虽然提出了多种变异策略，但是，不同策略具有不同的性能，适合求解不同的问题，因此如何根据不同问题选择最优的变异策略具有一定的困难。

聚类分析是数据挖掘中一种重要的数据分析方法，它是一种无指导分类算法。它把无标记的数据集通过一定的相似性评价标准分成很多组(簇)，即相似性高的个体被分在一组，而不相似的个体则分在不同组。大多数聚类分析算法都可以转化成一个多峰优化问题。从优化的观点来看，聚类问题是一个 NP-难问题。由于聚类问题的复杂性，因此，大多数传统的聚类分析算法都是局部最优算法。为了弥补传统聚类算法的不足，把演化算法与传统聚类算法相结合，形成的演化聚类分析算法能增强原有聚类算法的有效性。但是，目前已有的演化聚类算法存在以下两个缺点：1) 由于演化算法中存在很多控制参数，而大部分参数对不同数据集是敏感的，因此影响了算法的通用性；2) 演化聚类算法的运行时间远大于传统聚类算法。

本文在综述了当前差分演化算法的改进以及应用的基础上，针对其存在的不足进行了改进研究，并把差分演化算法应用于多目标优化问题和聚类问题的求解中。本文主要工作概括如下:

(1) 针对差分演化算法存在利用能力不足的缺点，将其与生物地理学优化(BBO)算法相结合，提出了混合的差分演化算法, DE/BBO 算法。算法中设计了一种通用的混合移动算子，该算子把 BBO 算法的移动算子与差分变异算子相混合，既能利用差分变异算子对搜索空间进行开采(exploration)，对新空间进行搜索，增加算法找出全局最优解的可能性，从而

增强了算法的鲁棒性；又能结合 BBO 算法移动算子利用(exploitation)能力强的特点，对当前群体的信息进行有效利用，进而加快算法的收敛速度。基于该算子的 DE/BBO 算法能有效平衡算法的开采能力和利用能力。通过标准测试函数对算法进行测试，并与其他改进的差分演化算法进行比较，验证了 DE/BBO 算法的优越性。此外，还对群体大小、问题维数、不同变异策略和自适应参数控制对 DE/BBO 和 DE 算法的影响进行了研究，实验结果进一步表明了 DE/BBO 算法在解的求解精度和收敛速度上优于传统 DE 算法。

(2) 针对差分演化算法对不同问题进行策略选择困难的缺点，提出了一种简单、通用的个体策略产生方法，并在此基础之上设计了三种自适应多策略选择技术，从而实现算法对策略库中策略的自适应选取，增强算法的性能。把所提出的自适应多策略选择技术与 JADE 算法相结合，应用于函数优化问题的求解中，取得了很好的效果。同时，对算法的复杂度和简单性进行了分析。通过实验验证了 SaJADE 算法的优越性能，表明 SaJADE 算法对策略选取具有自适应的特征，且该算法增强了 JADE 算法的性能。

(3) 为了能使差分演化算法有效求解多目标优化问题，提出了一种基于 ε 占优的正交多目标快速差分演化算法，ε-ODEMO 算法。算法采用正交实验设计方法来产生初始群体，从而使群体中的个体能在决策变量空间上均匀分布，增加找出较好个体的可能性，而这些好的个体可以在演化过程中得到利用，并加快算法的收敛。利用 Archive 群体来保存演化过程中找出的非劣解，使得算法始终可以保存精英解。为了对 Archive 群体进行有效更新，保证非劣解集的收敛性和分布的多样性，采用基于 ε 占优技术的群体更新方法。提出了一种混合选择策略，对差分变异算子中的基向量采用混合选择机制选取。该混合选择策略在演化初期强调算法对搜索空间的开采，寻找新的搜索空间，而在演化中后期则强调对 Archive 群体中个体信息的利用，加速算法的收敛。通过多个 2 目标和 3 目标标准测试函数对算法的有效性进行了验证；同时，对混合选择策略的控制参数和杂交概率进行了初步实验分析。

(4) 在 ε-ODEMO 算法的基础上，提出了一种适合求解约束多目标工程优化问题的 *pa* ε-ODEMO 算法。该算法在保留了 ε-ODEMO 算法优点的基础上，还具有以下特征: 1) 采用自适应 ε 占优技术代替 ε-ODEMO 算法中原有的 ε 占优技术，从而弥补了传统 ε 占优技术容易丢失一些重要的非劣解的缺点；2) 采用基于约束 Pareto 占优的约束函数处理技术，增强算法处理带约束的多目标问题的能力。*pa* ε-ODEMO 算法在约束多目标标准测试函数和工程优化问题中进行测试，并通过与 NSGA-II 算法进行比较，验证了改进算法处理带约束多目标问题的有效性。

(5) 针对当前演化聚类算法的不足，把简单、高效的差分演化算法用于聚类分析中，提出差分演化聚类分析算法, DEPS-C 算法。该算法采用基于点对称距离作为相似性准则来对数据点进行分配，同时考虑了对称的封闭性，以增强对原有点对称距离标准的鲁棒性。采用基于Kd树的最邻近点搜索方法来寻找对称点，从而减少了算法的复杂度；并设计了适合聚类分析的演化群体中个体的表示方法，并对演化中错误个体进行修正。通过标准测试数据集和 UCI 数据集对算法性能进行了测试。此外，对差分演化算法杂交概率 *CR* 的不同取值对算法性能的影响进行了研究，实验结果表明: 对于具有对称性的数据集，DEPS-C 算法能对其进行正确的聚类划分；且该算法控制参数少，对杂交概率 *CR* 的不同取值不敏感，从而增加了该算法的实用性和通用性。

(6) 尽管 DEPS-C 算法具有很好的性能，但该算法在对给定数据集进行聚类之前需要

用户给定簇的准确个数，这在很多实际问题中是很难的。为此，在 DEPS-C 算法基础上提出了自动差分演化聚类算法(ACDEPS 算法)，使算法能通过演化自动确定簇的个数，并得到相应最优的聚类划分。在 ACDEPS 算法中，采用了有效的个体表示以满足差分演化算法进行自动聚类的需求，同时提出了改进的基于点对称距离的聚类有效性验证索引(CVI) *Sym*'-index，利用对个体所包含簇个数进行动态惩罚来避免算法在演化初期找出过多的簇数目，从而影响算法的收敛。通过实验验证了: 1) 采用 *Sym*'-index 的 ACDEPS 算法对于具有对称性特征的数据集能自动确定簇的正确个数，且能找到其对应的最优聚类划分；2) 改进的基于点对称的 CVI 标准 *Sym*'-index 在 ACDEPS 算法中能增强原始 *Sym*-index 的性能; 3) 基于差分演化算法的聚类算法性能优于基于遗传算法的聚类算法。

综上所述，本文在分析了基本差分演化算法和已有改进差分演化算法的基础上，指出了差分演化算法的不足；同时，对当前演化聚类算法的关键技术进行了综述，分析了当前演化聚类算法的缺点。在此基础之上，重点研究了基于生物地理学优化算法的混合差分演化算法、自适应多策略差分演化算法、基于 ε 占优技术的正交多目标快速差分演化算法、改进的正交多目标差分演化算法在工程优化中的应用、给定 K 值的差分演化聚类算法和自动差分演化聚类分析算法等。在研究中通过实验的方法证明了各算法的有效性。

关键词：差分演化算法；全局优化；聚类分析；自适应；多目标优化

DIFFERENTIAL EVOLUTION ALGORITHM AND ITS APPLICAION IN CLUSTERING ANALYSIS

Ph.D. Candidate: Wenyin Gong Supervisor: Prof. Zhihua Cai, Prof. Charles Ling

ABSTRACT

Global optimization is a common problem in almost every field, such as engineer design, operational research, bio-medical science, data mining, and so on. How to design an efficient algorithm to solve this problem is essential in science and real applications. Recently, using Evolutionary Algorithms (EAs) to solve the global optimization problem is more popular. Since EAs are self-adaptive, self-organized, parallel, using EAs for global optimization is reasonable and available.

Differential evolution (DE), which is an evolutionary algorithm, is a simple, fast, and efficient population-based direct search algorithm for global optimization. The DE algorithm has been widely used in many fields. Among its advantages are its simple structure, ease of use, robustness, and fast convergence. However, the original DE algorithm has some pitfalls: 1) DE is good at exploring the search space and locating the region of global minimum, but it is slow at the exploitation of the solution; 2) The parameters of DE are problem dependent and the choice of them is often critical for the performance of DE; 3) There are many strategies in the DE literature, however, choosing the best among different mutation strategies available for DE is also not easy for a specific problem.

Clustering analysis is an important technique for the data analysis in data mining and machine learning. Clustering is the unsupervised classification of objects (patterns) into different groups, or more precisely, the partitioning of a dataset into subsets (clusters), so that the data in the same clusters is similar and the data in different clusters is dissimilar according to some defined distance measure. Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis, and bioinformatics. Most of the clustering algorithm can be converted into a global optimization problem. From the optimization point of view, clustering is an NP-hard problem. Due to the complexity of clustering, most of the traditional clustering algorithm is a local optimal algorithm. In order to solve the clustering problem more effectively, one possible way is the hybridization

of EAs with the traditional clustering algorithm, so that the evolutionary clustering algorithm is able to enhance the robustness and efficiency of the traditional clustering algorithm. However, there are two main drawbacks in the evolutionary clustering algorithm: 1) In EAs, there are many control parameters, such as crossover probability and mutation probability. These parameters are problem-dependent and sensitive. Hence, this drawback may degrade the commonality of the evolutionary clustering algorithm. 2) The execution time of EAs is significantly higher than that of other traditional clustering algorithms (e.g. K-means and FCM), especially when applied to large datasets.

In this dissertation, firstly, I described the advances of the DE algorithm. Then, to remedy some of DE's pitfalls, I proposed two variants of DE. Thirdly, the ε-domination based orthogonal DE algorithm was proposed for the multi-objective optimization problems and the engineering design problems. Finally, the DE algorithm based on the point symmetry metric was proposed for the clustering problem. The main contributions of this dissertation are as follows.

(1) Biogeography-Based Optimization (BBO) is a new biogeography inspired algorithm. It mainly uses the biogeography-based migration operator to share the information among solutions. To enhance the exploitation ability of DE, a hybrid DE with BBO, namely DE/BBO, is proposed for the global numerical optimization problem. DE/BBO combines the exploration of DE with the exploitation of BBO effectively, and hence it can generate the promising candidate solutions. To verify the performance of the proposed DE/BBO approach, 23 benchmark functions with a wide range of dimensions and diverse complexities are employed. Experimental results indicate that the proposed approach is effective and efficient. Compared with other state-of-the-art DE approaches, DE/BBO performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate. In addition, the influence of the population size, dimensionality, different mutation schemes, and the self-adaptive control parameters of DE are also studied.

(2) The fusion of multi-methods is an efficient technique to enhance the performance of the single algorithm. As the above-mentioned, the choice of the best mutation strategy is difficult for a specific problem. To alleviate this drawback and enhance the performance of DE, a family of improved DE that attempts to adaptively choose the suitable strategies for different problems is presented. In addition, in the proposed strategy adaptation mechanism (SaM) different parameter adaptation methods of DE can be used for different strategies. In order to test the efficiency of our approach, the proposed SaM is combined with JADE, which is a recently proposed DE variant, for numerical optimization. Twenty widely used scalable benchmark problems are chosen from the literature as the test suit. Experimental results verify the expectation that SaM is able to adaptively choose the suitable strategy for a specific problem without any *prior* knowledge. Compared with other state-of-the-art DE variants, the proposed approach performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate.

(3) Evolutionary multi-objective optimization (EMO) has become a very popular topic in the last few years. However, to design an efficient and effective EMO algorithm to find the

near-optimal and near-complete Pareto front is a challenging task. A novel differential evolution algorithm, ε-ODEMO, is proposed to solve multi-objective optimization problems (MOPs) efficiently. The proposed approach uses an Archive population to retain the obtained non-dominated solutions; also it adopts the orthogonal design method with quantization technique to generate an initial population of points that are scattered uniformly over the feasible solution space, so that the algorithm can evenly scan the feasible solution space once to locate good points for further exploration in subsequent iterations. Moreover, it is based on the ε-dominance concept to obtain a good distribution of Pareto-optimal solutions and gets them in a small computational time. To make the algorithm converge faster, the new approach employs a hybrid selection mechanism in which a random selection and an elitist selection are interleaved. Experiments on eight benchmark problems of diverse complexities show that the proposed approach is able to obtain a good distribution in all cases. Compared with several other state-of-the-art evolutionary algorithms, it achieves not only comparable results in terms of convergence and diversity metrics, but also a considerable reduction of the computational effort. Furthermore, the discussion of the influence of different CR value and the parameter value of hybrid selection mechanism to the performance of the algorithm is conducted experimentally.

(4) Solving engineering design and resources optimization via multi-objective evolutionary algorithms (MOEAs) has attracted much attention in the last few years. An efficient multi-objective differential evolution algorithm is presented for engineering design. The approach is the extension of ε-ODEMO; and it has some modifications: 1) An archive (or secondary population) is employed to keep the non-dominated solutions found and it is updated by a new relaxed form of Pareto dominance, called Pareto-adaptive ε-dominance (*pa*ε-dominance), at each generation. 2) To handle the constraints, a new constraint-handling method is employed, which does not need any parameters to be tuned for constraint handling. The proposed approach is tested on four benchmark-constrained problems to illustrate the capabilities of the algorithm in handling mathematically complex problems. Furthermore, four well-studied engineering design optimization problems are solved to illustrate the efficiency and applicability of the algorithm for multi-objective design optimization. Compared with NSGA-II, one of the best MOEAs available at present, the results demonstrate that my approach is found to be statistically competitive. Moreover, the proposed approach is very efficient and is capable of yielding a wide spread of solutions with good coverage and convergence to true Pareto-optimal fronts.

(5) To remedy some of the drawbacks of the evolutionary clustering algorithm, a DE-based clustering approach (DEPS-C), which combines several features of previous clustering techniques in a unique manner, is presented. It is characterized by: (a) employing a recently proposed point symmetry-based distance measure as the similarity measure, (b) incorporating the closure property when assigning a data point to a cluster, and (c) using the Kd-tree based nearest neighbor search to reduce the complexity of finding the closest symmetric point. Experiments have been conducted on 10 artificial data sets and 8 real-life data sets of diverse complexities. The results indicate that DEPS-C is suitable for both the symmetrical intra-clusters and the

symmetrical inter-clusters. Compared with GAPS, a recently proposed genetic algorithm with point symmetry distance based clustering algorithm, the proposed approach performs better, or at least comparably, in terms of the quality and stability of the final solutions. Moreover, DEPS-C is faster than GAPS with respect to the execution time.

(6) Based on the DEPS-C algorithm, an automatic clustering DE technique (ACDEPS) for the clustering problem is proposed. This approach can be characterized by: (i) proposing a modified point symmetry-based cluster validity index (CVI) as a measure of the validity of the corresponding partitioning, (ii) using the Kd-tree based nearest neighbor search to reduce the complexity of finding the closest symmetric point, and (iii) employing a new representation to represent an individual. Experiments have been conducted on 6 artificial data sets of diverse complexities. The results demonstrate that ACDEPS is suitable for both the symmetrical intra-clusters and the symmetrical inter-clusters. In addition, ACDEPS is able to find the optimal number of clusters of the dataset. Furthermore, based on the comparison with the original point symmetry-based CVI, the modified point symmetry-based CVI shows better performance in terms of the F-measure and the number of clusters found.

In summery, based on the analysis of the original DE algorithm and its variants, this dissertation pointed out the drawbacks of DE. Meanwhile, I described the key techniques of the evolutionary clustering algorithms and analyzed their pitfalls. Based on the above-mentioned analysis, this dissertation is mainly studied: the BBO-based hybrid DE algorithm, DE with the adaptive multi-strategies selection, ε-domination based orthogonal DE for MOPs, *pa*ε-domination based constrained ε-ODEMO for engineering design, DE-based clustering algorithm with predefined K, and automatic DE-based clustering algorithm. In addition, for each improved algorithms, the experimental study was conducted to validate its performance.

Key Words: Differential evolution; global optimization; clustering analysis; adaptation; multi-objective optimization.

目　录

第一章 绪论

本章首先阐述了本文的选题背景及研究意义，然后分别对演化算法、差分演化算法及聚类分析等进行简要的描述，最后重点介绍了本文的主要工作及本文其他章节的安排。

§1.1 引言

20 世纪 80 年代中期以来，演化计算(Evolutionary Computation)已成为计算机领域的一大研究热点[1,2]。近 30 年来,研究学者们提出了多种算法，主要包括遗传算法(Genetic Algorithms)、演化策略(Evolution Strategies)、进化规划(Evolutionary Programming)、遗传规划(Genetic Programming)等[3]。

作为一种优化算法，演化算法对所优化的问题具有广泛的适用性，而且不像经典优化算法那样要求所优化问题的可导、连续、单峰等特点。近年来，演化算法已广泛应用于工程优化设计、运筹学、图像处理、地学工程、经济学、电力负荷分配等领域[1]。

差分演化算法(Differential Evolution)[4,5]是演化算法的一种，是 1995 年由 Price 和 Storn 首次提出[6]。该算法是一种简单、高效和快速的演化算法，具有结构简单，易实现，快速收敛等优点。但是，传统差分演化算法存在以下几方面的不足: 1) 缺乏局部搜索能力[7]，使算法在演化后期收敛速度变慢，从而不能满足算法在较少适应值函数评价次数下快速收敛到问题最优解的要求; 2) 传统差分演化算法对缩放因子 F 和杂交概率 CR 设置是很敏感的，对不同的问题需要选择不同的 F 和 CR 值[8,9]; 3) 在差分演化算法中，提出了多种变异策略[5,10]，但是，不同策略具有不同的性能，适合求解不同的问题，因此，如何根据不同问题选择最优的变异策略是很困难的[11]。为此，本文研究在传统差分演化算法上加入局部搜索算子，以增强算法的局部搜索能力，加快算法的收敛; 同时，设计自适应差分演化算法，使得 F, CR 和变异策略的设置与选取具有自适应性，从而避免人为设置参数的不合理性以及增强算法对不同求解问题的自适应性。此外，在差分演化算法上加入约束函数处理技术和多目标处理技术，以增强算法求解约束函数和多目标函数的能力。

聚类分析[12]在数据分析和模式识别中具有非常重要的作用。传统的聚类分析方法(如 K-均值、K-中心、基于密度的聚类、分层聚类等)具有一定的不足之处，如 K-均值方法对初始聚类中心的选取十分敏感，层次聚类算法的伸缩性不强等，而且传统聚类分析算法均为单目标聚类分析方法。为了克服传统聚类分析算法的不足，本文把改进的差分演化算法应用到聚类分析中，设计出能高效求解聚类问题的差分演化算法，同时，设计自动聚类差分演化算法使其能自适应选取 K 值等。

本文的研究在分析传统差分演化算法不足的基础上，对其进行改进，提出的混合算法和

自适应算法不仅增强了算法的有效性和高效性，而且使算法的自适应性得到了增强，从而有利于算法得到更广泛的应用。此外，把传统演化算法应用于多目标优化领域和约束优化领域中，扩展了算法的求解能力。把差分演化算法与传统聚类分析算法相结合所提出的差分演化聚类算法一方面提高了传统聚类算法的性能，另一方面为算法今后在图像分析处理等领域的应用提供了参考。

§1.2 演化算法与差分演化算法

1.2.1 演化算法

在人工智能(Artificial Intelligence)领域中，演化算法(Evolutionary Algorithms)[12]是演化计算的一个分支。它是一种基于群体的元启发式优化算法，具有自适应、自搜索、自组织和隐并行性等特点。近年来，很多学者将演化算法应用到优化领域中，取得了很大的成功，并已引起了人们的广泛关注。越来越多的研究者加入到演化优化的研究之中，并对演化算法作了许多改进，使其更适合各种优化问题。目前，演化算法已广泛应用于求解无约束函数优化、约束函数优化、组合优化、多目标优化等多种优化问题中。

演化算法求解问题的基本思想受到生物演化的启发，主要包括复制(reproduction)、变异(mutation)、重组(recombination)和选择(selection)等步骤。其基本思路是：由问题的候选解组成一个群体，然后通过随机变异，重组和选择等算子对群体进行演化。其中随机变异和重组算子提供了发现新解的机制，选择算子则确定保持哪些解作为下一步搜索的基础。演化算法具有以下几个优点：1) 以优化变量的遗传编码为运算、搜索对象，不仅可以用于优化数值优化问题，还可用于优化非数值优化问题；2) 只利用“适应值”信息，而不需利用目标函数的具体值及其他辅助信息，如连续、可导、平滑、无噪音等，从而使得演化算法可以应用于多种优化问题中；3) 非单点操作，使用群体搜索策略。这使得算法可以进行并行搜索，并且可以一次优化得到多个优化结果，特别适合于进行多目标优化决策；4) 使用随机搜索机制：随机搜索机制的一个优点是相应算法的健壮性(robustness) 得到增强。总的来说，演化算法具有通用、并行、稳健、简单和全局优化能力强等突出优点。

1.2.2 差分演化算法

差分演化算法是演化算法的一种，是一类基于群体的随机优化算法，主要采用实数编码，用于求解实数优化问题。近年来，该算法也用于求解离散优化问题。差分演化算法于 1995 年由 Price 和 Storn 首次提出，并在 1996 首届 IEEE 演化计算竞赛中得到三等奖[13]。差分演化算法的主要算子为差分变异(differential mutation)算子，该算子利用不同个体间的差分信息，在搜索方向和搜索步长上具有自适应性[5]。与其他演化算法相比，该算法具有结构简单、容易使用、收敛快和鲁棒性强等优点。这些优点使算法得到了广泛的应用，如数据挖掘[14,15]、模式识别、数字滤波器设计、人工神经网络等[5,16,17]。目前，该算法也被用于求解全局互斥组合优化问题(global permutation-based combinational optimization)中[18]。

§1.3 聚类分析

聚类(Clustering)分析是一种无指导分类算法，它将无标记的数据集(模式)根据一定的相似性评价标准分成很多组(簇)。在分组过程中，相似的个体被分在一组，而不相似的个体则要分在不同组。这里，相似性评价标准可以采用很多不同标准，如欧拉距离、曼哈顿距离等.文献[11]和文献[19]对聚类分析进行较好的综述。

聚类分析在数据分析、模式识别和机器学习中具有非常重要的作用。传统的聚类分析方法主要有 K-均值、K-中心、基于密度的聚类、分层聚类、模糊聚类、EM 算法等。其中，K 均值聚类算法是一种的经典聚类分析算法，在 2006 年香港召开的 ICDM'06[20]中被评为至今 10 大数据挖掘算法之一[21]。K 均值聚类算法在很多领域得到了广泛的应用，包括数据挖掘、机器学习、模式识别等领域。

K 均值算法具有如下优点:

1) 算法简单，容易实现;
2) 算法复杂度低，适合求解大规模数据集问题。

但是，传统 K 均值算法存在以下几点不足之处:

1) 算法是非数据集独立的(data-dependent);
2) 算法采用贪心搜索算法，受初始条件的影响，容易收敛于问题的局部最优解;
3) 算法需要用户事先给定 K 值，这在很多实际运用中有一定的难度。

由于传统 K 均值算法存在以上不足，一些学者将演化算法与 K 均值算法相结合，提出了演化 K 均值聚类算法，如基于遗传算法的 K 均值聚类算法[22-28]、基于进化规划的 K 均值聚类算法[29]、基于粒子群优化算法的 K 均值聚类算法[30]、基于差分演化算法的 K 均值聚类算法[31,32]等。尽管这些算法在聚类的实现细节上存在一定的差距，但总体框架基本类似。此外，由于聚类问题的特殊性，在设计演化 K 均值聚类算法的时候需根据问题的特殊性采用新的个体编码方法，适应值计算等技术。

§1.4 本文工作及章节安排

本文研究得到了国家 863 计划(2009AA12Z117)、高等学校博士学科点专项科研基金课题(博导类) (20090145110007) 、国家建设高水平大学公派出国留学计划、中国地质大学(武汉)优秀博士论文创新基金的资助。

1.4.1 本文主要工作

本文的研究工作主要包含两大部分: 1) 基本差分演化算法的改进，以及差分演化算法在多目标优化和约束优化问题中的应用; 2) 基于差分演化算法的 K 均值聚类算法研究，包括给

定 K 值的差分演化聚类算法和自动(未给定 K 值)的差分演化聚类算法研究。主要工作包括以下几个方面:

(1) 对基本差分演化算法进行分析，并对当前差分演化算法的研究进行综述;

(2) 针对传统差分演化算法缺乏利用能力的缺点，提出基于生物地理学优化算法的混合差分演化算法，并通过将该算法应用于函数优化问题的求解中验证其性能;

(3) 在差分演化算法中提出了多种变异策略，不同策略具有不同的特点，因此适合求解不同的问题，但是，针对不同问题选择最优变异策略是很困难的。为了克服此缺点，提出一种多策略自适应选择差分演化算法。新算法包含多个变异策略，通过一定的机制使算法能够根据不同优化问题自适应选择最合适的变异策略，从而增强了算法的性能和普适性;

(4) 为了进一步扩展差分演化算法的求解能力，将 ε 占优和正交实验设计与差分演化算法相结合，提出一种求解多目标问题的高效快速的差分演化算法，并通过实验验证算法的性能。同时，在此算法的基础上，把自适应 ε 占优与上述算法相结合，并提出一种新的约束函数处理技术，从而增强了算法处理约束函数优化问题的能力;

(5) 在对演化聚类分析算法进行综述的基础上，提出了基于点对称标准的差分演化 K 均值聚类算法，包括给定 K 值的算法和自动聚类算法，并通过实验验证了两类聚类算法的有效性。

1.4.2 论文章节安排

本文主要章节安排如下:

(1) 第一章为引言部分，对本文研究工作的背景和研究意义进行了简述，同时，简要介绍了本文研究所涉及的一些基本内容，如演化算法、差分演化算法、演化聚类算法等。

(2) 第二章对差分演化算法进行了综述，包括对基本差分演化算法的介绍，搜索原理的分析，当前对基本差分演化算法的改进，差分演化算法在复杂优化环境(如多目标优化、约束优化等)下的应用，离散差分演化算法的研究，以及差分演化算法的一些主要应用等。

(3) 第三章提出一种基于生物地理学优化(BBO)算法的混合差分演化算法以增强算法的利用能力。该算法将 BBO 算法与基本差分演化算法相结合，提出了一种通用的混合移动算子，该算子一方面结合了差分变异算子开采能力强的优点，另一方面利用了 BBO 算法移动算子的利用能力强的特点，从而能有效平衡算法的开采(exploration)能力和利用(exploitation)能力。通过标准测试函数为算法进行了测试。本章的实验部分还包括就算法参数(如: 群体大小、问题维数、不同差分变异策略、自适应控制参数等)的不同设置对算法性能影响的实验分析。同时，与其他混合差分演化算法进行了对比研究。

(4) 第四章针对传统差分演化算法对变异策略选择的困难，提出一种新的多策略自适应差分演化算法。算法提出一种新的多策略选择机制，在此基础上，提出了三种选择方法，在对三种选择方法进行实验分析后，重点对其中一种方法进行了研究，包

括算法的鲁棒性、收敛速度、成功率、自适应性、简单性等。

(5) 第五章提出一种基于 ε 占优和正交实验设计的快速多目标差分演化算法。算法将 ε 占优技术和正交实验设计方法应用到差分演化算法中，利用正交实验设计产生初始群体，从而可以均匀搜索整个决策空间，加速算法收敛；采用 ε 占优技术保存非劣解，可以使所保存的非劣解集均匀分布。同时，算法提出一种混合选择机制，以保证算法的开采能力和收敛速度。算法通过多个高维标准测试函数验证了其高效性。

(6) 第六章把 Pareto 自适应 ε 占优应用到上一章所提出的算法中，并采用一种新的约束函数处理技术，以实现对约束条件的处理。最后，将改进算法应用于约束多目标函数的优化中，通过标准测试函数和实际工程优化实例验证了算法的性能。

(7) 通过前几章对差分演化算法的改进及其对算法性能的验证，表明差分演化算法的有效性和高效性。为此，第七章把差分演化算法与传统的 K 均值算法相结合，提出了差分演化聚类算法。本章首先对目前已有的演化 K 均值聚类算法中所涉及的一些关键技术进行了综述，然后，在此基础上，针对当前演化聚类算法存的一些不足，提出了基于差分演化算法的聚类分析算法。算法采用改进的基于点对称距离的评价标准来进行点的分配，且利用适合聚类分析的演化群体个体表示方法。通过 10 个人工数据集和 8 个 UCI 数据集对改进算法进行了有效性验证。

(8) 第八章针对差分演化聚类分析算法需要人工给定各数据集簇的准确数目的不足，在第七章工作的基础上，提出了自动差分演化聚类算法。算法采用的有效地适合自动聚类的个体表示方法，提出了改进的基于点对称距离的聚类有效性索引。通过 6 个人工数据集并与 GUCK 算法进行对比，验证了算法的可行性和有效性。

(9) 第九章对全文工作进行总结，包括对本文研究工作创新点的说明以及对今后工作的展望。

第二章 差分演化算法概述

§2.1 基本差分演化算法

2.1.1 算法简介

差分演化算法是一类基于群体的自适应全局优化算法，该算法属于演化算法的一种，具有结构简单(算法只有三个控制参数)、容易实现(其核心代码用 C 语言实现仅有 30 余行)、收敛快速、鲁棒性好等优点。算法于 1995 年由 Price 和 Storn 首次提出[6]，主要用于求解实数优化问题。差分演化算法中最重要的算子为差分变异(differential mutation)算子，该算子将同一群体中两个个体向量进行差分和缩放，并与该群体中第三个个体向量相加等到一个变异个体向量(mutant)；然后变异个体向量与父个体向量进行杂交(crossover)形成尝试个体向量(trial vector)；最后，尝试个体向量与父个体向量进行适应值比较，较优者保存在下一代群体中。这样，差分演化算法利用差分变异，杂交和选择等算子对群体不断进行演化，直到达到终止条件退出。由于差分演化算法采用的是一对一选择算子(one-to-one selection)，因此，该算法属于稳态(steady state)演化算法。目前，差分演化算法已经被广泛应用到数据挖掘[14,15]、模式识别、数字滤波器设计、人工神经网络等[5,16,17]等领域，也被用于求解全局互斥组合优化问题(global permutation-based combinational optimization)中[18]。

2.1.2 算法基本流程

差分演化算法作为演化算法的一种，具有与其他演化算法相似的流程，如群体初始化、个体适应值评价、通过遗传算子对群体进行演化等。图 2-1 给出了基于 DE/rand/1/bin 策略的基本差分演化算法流程图，其中从第 5 行到第 13 行是 DE/rand/1/bin 策略；第 9 行是 DE/rand/1 变异算子；第 17 行到第 19 行是选择算子。在图 2-1 中，涉及到基本差分演化算法的三个控制参数，分别为群体大小 NP，缩放因子 F 和杂交概率 CR。rndint(1, D)表示在[1, D]区间随机均匀产生的整数，D 为所求解问题的自变量维数；rndreal[0, 1]表示在[0, 1]之间随机均匀产生的实数。算法的终止条件可以是最大演化代数、最大适应值评价次数等，一般需要用户事先给定。从图 2-1 可以看出，基本差分演化算法的结构简单，容易实现。

图 2-1 给出了差分演化算法的基本流程，算法中包含个体的编码与初始化、差分变异算子、杂交算子和选择算子等，下面将对其进行简述，以便对算法有更详细的了解。

2.1.2.1 编码及初始化

经典差分演化算法采用实数编码，这使得算法更适合于求解实数优化问题。假设所求解问题自变量有 D 维，则群体中第 i 个个体 x_i 表示如下:

1: 产生初始群体P
2: 评价初始群体P中所有个体适应值
3: **while** 如果终止条件没有满足 **do**
4: **for** $i = 1$ to NP **do**
5: 随机均匀选择$r_1 \neq r_2 \neq r_3 \neq i$
6: $j_{rand} = \text{rndint}(1, D)$
7: **for** $j = 1$ to D **do**
8: **if** $\text{rndreal}_j[0,1) < CR$ or $j == j_{rand}$ **then**
9: $U_i(j) = X_{r_1}(j) + F \times \left(X_{r_2}(j) - X_{r_3}(j)\right)$
10: **else**
11: $U_i(j) = X_i(j)$
12: **end if**
13: **end for**
14: **end for**
15: **for** $i = 1$ to NP **do**
16: 评价新个体U_i
17: **if** U_i优于X_i **then**
18: $X_i = U_i$
19: **end if**
20: **end for**
21: **end while**

图 2-1 基于 DE/rand/1/bin 的差分演化算法流程

$$X_i=\{x_i(j),x_i(j),\mathbf{L},x_i(j)\} \tag{2.1}$$

其中，$x_i(j)\in[l(j),u(j)]$，$i=1,\mathbf{L},NP$，$j=1,\mathbf{L},D$．$x_i(j)$ 为一实数在其自变量范围 $[l(j),u(j)]$ 内随机均匀初始化，即 $x_i(j)=\text{rndreal}[l(j),u(j)]$．

2.1.2.2 差分变异算子

差分演化算法中最重要的算子是差分变异算子，该算法也正因此算子而得名。如上所述，在图 2-1 中，第 9 行是 DE/rand/1 变异算子。除了该算子外，Price 等还提出了 9 个变异算子[5]，为了区别这些算子，我们采用“DE/a/b”来表示，其中“DE”表示差分演化算法；“a”代表基向量的选择方式，一般有 rand 和 best 两种；“b”表示算子中差分向量的个数。在多个变异算子中比较常用的有:

DE/best/1: $$V_i=X_{best}+F(X_{r2}-X_{r3}) \tag{2.2}$$

DE/rand/2: $$V_i=X_{r1}+F(X_{r2}-X_{r3})+F(X_{r4}-X_{r5}) \tag{2.3}$$

DE/current-to-best/1: $$V_i=X_i+F(X_{best}-X_i)+F(X_{r2}-X_{r3}) \tag{2.4}$$

DE/rand-to/best/1: $$V_i=X_{r1}+F(X_{best}-X_{r1})+F(X_{r2}-X_{r3}) \tag{2.5}$$

其中，X_{best} 为当前群体的最优个体；X_i 为父个体；$r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$ 为群体中随机选择的 5 个个体；V_i 是变异向量(mutant)；$X_{r2} - X_{r3}$ 为差分向量；$F \in [0,1+)$ 为缩放因子，用于对差分向量进行缩放，从而可以控制搜索步长。

2.1.2.3 杂交算子

差分演化算法采用离散杂交算子，其中包括二项式杂交(binomial crossover)和指数杂交(exponential crossover)，图 2-1 中第 6 行到第 13 行采用的是二项式杂交。杂交算子把通过变异算子产生变异向量V_i与父个体向量 X_i 进行离散杂交得到尝试向量U_i。二项式杂交算子也可表示为:

$$U_i(j) = \begin{cases} V_i(j) & \text{if } (\text{rndreal}_j[0,1] < CR \text{ or } j = j_{rand}) \\ X_i(j) & \text{otherwise} \end{cases} \tag{2.6}$$

其中，$j = 1, \mathbf{L}, D$，j_{rand} 是$[1, D]$之间的一个随机整数，保证尝试向量U_i中至少有一维是来自于变异向量V_i，从而避免与父个体向量 X_i 相同。在差分演化算法中各个变异算子均可以与指数杂交算子相结合，图 2-2 为基于 DE/rand/1/exp 策略的基本差分演化算法流程图。

```
1: 产生初始群体P
2: 评价初始群体P中所有个体适应值
3: while 如果终止条件没有满足 do
4:   for i = 1 to NP do
5:     随机均匀选择r1 ≠ r2 ≠ r3 ≠ i
6:     L = 0
7:     U_i = X_i
8:     repeat
9:       U_i(j) = X_r1(j) + F × (X_r2(j) − X_r3(j))
10:      j_rand = (j_rand + 1) mod D
11:      L = L + 1
12:    until rndreal_j[0, 1) > CR or L > D
13:  end for
14:  for i = 1 to NP do
15:    评价新个体U_i
16:    if U_i优于X_i then
17:      X_i = U_i
18:    end if
19:  end for
20: end while
```

图 2-2 基于 DE/rand/1/exp 的差分演化算法流程

2.1.2.4 选择算子

差分演化算法通过变异算子和杂交算子产生孩子群体之后，采用一对一竞标赛选择算子将子个体与相应的父个体进行比较，较优者保存到下一代群体中。对于最小化优化问题其选择算子可以描述为:

$$X_i = \begin{cases} U_i & \text{if } (f(U_i) \le f(X_i)) \\ X_i & \text{otherwise} \end{cases} \tag{2.7}$$

其中 $f(X_i)$ 为个体 X_i 的适应值。由于差分演化算法采用的是一对一竞标赛选择，因此该算法是一种保存精英个体的稳态演化算法。

一旦新的群体形成之后，差分演化算法继续通过变异、杂交和选择算子对群体不断进行演化，直到达到终止条件退出程序。

2.1.3 控制参数对算法性能的影响

从 2.1.2 节对差分演化算法介绍中可以看出，该算法有 3 个控制参数，即群体大小 ***NP***、缩放因子 ***F*** 和杂交概率 ***CR***。一般而言，这些控制参数会影响算法搜索最优解和收敛速度[9]，针对不同的问题需要进行不同的设置。各个参数对算法性能的影响可以简要归纳如下:

(1) **群体大小 *NP* 的影响**: 较大群体会增加群体个体的多样性，加大对搜索空间的搜索，增加搜索到最优解的可能性，但同时会减低收敛速度；而较小群体则会加快算法收敛，但是，容易导致算法局部收敛或停滞演化。

(2) **缩放因子 *F* 的影响**: 较小 ***F*** 值会加速算法收敛，但是，容易使算法局部收敛。因此，为了避免早熟，应避免设置太小的 ***F*** 值。较大 ***F*** 值会增加算法跳出局部最优解的可能性，但是，*F*>1 会减低收敛速度。

(3) **杂交概率 *CR* 的影响**: ***CR*** 值的设置主要取决于所求解的问题。一般而言，对于自变量之间相互独立的问题，***CR*** 可设置较小的值；而对于旋转问题(自变量相互依赖)，***CR*** 应设置较大的值。

§2.2 差分演化算法的改进

尽管差分演化算法在很多领域取得了成功的应用，但是，该算法也存在一些不足，主要有: 1) 缺乏局部搜索能力[7]，使算法在演化后期收敛速度变慢，从而不能满足算法在较少适应值函数评价次数下快速收敛到问题最优解的要求；2) 传统差分演化算法对缩放因子 ***F*** 和杂交概率 ***CR*** 设置是很敏感的，对不同的问题需要选择不同的 ***F*** 和 ***CR*** 值[8]；3) 在差分演化算法中，提出了多种变异策略[5,9]，但是，不同策略具有不同的性能，适合求解不同的问题，因此，如何根据不同问题选择最优的变异策略是很困难的[11]。针对以上不足，许多学者对差分演化算法进行了改进，主要体现在以下几个方面。

2.2.1 混合差分演化算法研究

混合算法由于其在求解实际问题中所取得的成功得到了越来越多的关注[33]。把差分演化算法与其他算法相结合，形成混合差分演化算法是差分演化算法的一个研究方向。Fan 和

Lampinen 提出了一种三角变异算子来取代差分演化算法中传统的变异算子[34]。通过一些标准测试函数和实际优化问题，他们验证了改进算法优于传统的差分演化算法。Sun 等[35]把分布评估算法(Estimation of Distribution Algorithm, EDA)与差分演化算法相结合，利用 EDA 算法的概率模型确定有效搜索区域，并与差分演化算法的变异算子相结合对搜索空间进行搜索，从而可以提高搜索的有效性。通过标准测试函数验证了改进算法的性能。Gong 等[36, 37]把正交实验设计与差分演化算法相结合，设计了基于二水平正交设计的杂交算子，增强了对搜索空间的开采和利用能力，从而加速了算法的收敛速度。通过标准测试函数对算法进行策略，显示了改进算法在求解精度、收敛速度和稳定性上优于传统差分演化算法。Noman 和 Iba[38]提出基于杂交算子的局部搜索算子来对群体中最优个体进行利用，从而以求解高维函数优化问题。在此工作基础上，他们把基于爬山算法的自适应局部搜索算子与差分演化算法相结合以加速算法的搜索速度[7]. 通过实验，他们验证了算法的有效性. Wang 等[39]提出了一种基于动态聚类的混合差分演化算法. 算法中基于分层技术的动态聚类算法不断对群体进行划分，使差分演化算法能更有效利用群体中相似个体的信息，从而增强了算法的性能. Rahnamayan 等[40]提出了一种基于逆向学习(opposition-based learning)技术的混合差分演化算法，算法中逆向学习技术不仅用于初始群体的产生，同时，也被用于在演化中对群体进行逆向搜索，从而增强了对群体搜索空间的利用能力。作者通过大量的测试函数和一些评价标准验证了改进算法的有效性。Caponio 等提出了一种基于局部搜索算法的 Memetic 差分演化算法 SFMDE 算法[41]。在 SFMDE 算法中，粒子群算法(Particle Swarm Optimization, PSO)用于在演化初期对群体进行演化以便产生较好的子群体。在演化中后期，通过差分演化算法和两个局部搜索算法自适应地对群体进行演化，从而加速了算法的收敛速度。Gong 等把模糊 C 均值聚类算法应用与差分演化算法中，增强了算法的利用能力。Gong 等[43]把差分演化算法与生物地理学优化(Biogeography-Based Optimization, BBO)算法[44]相结合提出了一种新的混合杂交算子，该算子可以使新算法在开采性(exploration)和利用性(exploitation)上得到平衡: 即一方面利用差分演化算法的变异算子对搜索空间进行开采，另一方面利用 BBO 的算子对群体信息进行有效利用。作者通过大量实验验证了混合算法的有效性。

2.2.2 参数自适应算法研究

针对差分演化算法对其控制参数(*CR*, *F* 和 *NP*)的设置很敏感，且随不同问题需要进行不同的设置的不足，许多学者提出了自适应参数控制的差分演化算法。Liu 和 Lampinen 提出了一种基于模糊控制的自适应差分演化算法[45]，算法利用模糊逻辑控制器对缩放因子 *F* 和杂交概率 *CR* 进行自适应控制。实验表明该模糊自适应差分演化算法优于传统差分演化算法。Brest 等[8]提出了一种自适应差分演化算法，算法把控制参数 *F* 和 *CR* 作为个体的一部分编码到个体中，并通过自适应控制机制对它们进行控制，这样 *F* 和 *CR* 就能随着演化自适应改变。Salman 等提出了一种自适应差分演化算法(SDE)用于消除用户对参数的设置[46]。在 SDE 中，缩放因子 *F* 通过类似差分演化算法的变异算子自适应控制。Nobakhti 和 Wang 提出一种随机化的自适应差分演化算法(RADE)[47]。在 RADE 中，一种简单的随机自适应机制对缩放因子 *F* 进行控制。Das 等[48]提出了两种改进的差分演化算法 DERSF 和 DETVSF 用于对缩放因子 *F* 进行动态控制。实验表明了两种改进算法优于传统差分演化算法。Teo 提出了一种动态自

适应群体大小的差分演化算法[49]，算法中群体大小是自适应变化的。通过 5 个 De Jong 测试函数对算法进行验证，表明改进算法是有效的。Brest 和 Maucec 提出了一种改进的差分演化算法[50]，算法中群体大小是逐渐减小的。作者通过实验验证了改进算法在有效性和鲁棒性上优于传统差分演化算法。Teng 等[51]提出了一种自适应群体差分演化算法，这里两个(绝对编码和相对编码)不同的机制来对群体大小进行自适应控制。实验结果表明，采用相对编码机制的算法取得了较好的性能[51]。Qin 等[52]提出了一种自适应策略的差分演化算法，算法中缩放因子 F 采用随机正态分布产生，杂交概率 CR 则通过演化过程中的反馈信息进行自适应调整。Zhang 和 Sanderson[53,54]提出了一种新的自适应差分演化算法(JADE)。在 JADE 中，F 和 CR 均利用演化过程中的反馈信息，并通过不同的自适应机制进行调整。实验结果表明了新算法具有很好的性能，在解的求解精度、收敛速度和鲁棒性上均优于现有的一些改进的差分演化算法。

2.2.3 多策略自适应算法研究

差分演化算法中提出了多种变异策略，不同策略具有不同的性能，适合于求解不同的问题。Feoktistov 和 Janaqi[10]提出了一种一般化的变异策略框架，以方便用户选择合适的变异算子和设计新的变异算子。Kaelo 和 Ali[55]针对传统差分演化算法变异算子存在利用能力不足的缺点提出了一种改进的变异算子，即通过竞标赛策略来选择基向量，从而可以增强算法对基向量附近区域的利用能力。受 PSO 算法的启发，Das 等[56]提出了一种基于邻域搜索的 DE/target-to-best/1 算子，该算子与传统的 DE/target-to-best/1 相结合对群体进行演化，从而可以有效平衡算法的开采能力和利用能力。Zhang 和 Sanderson[53,54]在 JADE 中提出了一种改进的 DE/current-to-best/1 算子，即 DE/current-to-$p$best/1 算子。DE/current-to-$p$best/1 的不同之处在于算法每次随机从 100p%个最优个体中选择一个作为 $p$best 个体，从而可以避免算法的局部收敛。针对各个变异算子具有不同性能的特点，Qin 和 Suganthan[57]提出了一种自适应策略的差分演化算法，算法中有两个变异策略 DE/rand/1/bin 和 DE/best/2/bin 利用反馈信息自适应的交替使用。在此基础上，Qin 等[52]采用 4 个变异策略，同时对不同策略采用不同的 CR 值，且 CR 值通过一个自适应机制进行调整。通过实验对比，作者验证了该算法优于传统差分演化算法和一些改进的自适应差分演化算法[52]。

§2.3 复杂环境下差分演化算法研究

差分演化算法除了应用于连续单目标无约束函数的优化问题的求解之外，还被广泛应用于一些复杂环境下优化问题的求解中，如多目标函数优化、约束函数优化等。

2.3.1 多目标函数优化

演化多目标算法一直是演化算法的一个研究热点[58]。由于差分演化算法在单目标优化问题中所取得的成功，一些学者将其应用到多目标优化问题的求解中。Madavan 提出一种基于 Pareto 占优的多目标差分演化算法[59]，算法把 Deb 等[60,61]提出的非劣解排序和排序选择机制

应用到差分演化算法中。作者通过 10 个无约束多目标测试函数验证了算法的有效性。Xue 等提出了 MODE 算法[62]。MODE 算法采用 $(m+l)$ 选择, Pareto 排序和拥挤距离等机制来保存非劣解集。利用 5 个高维多目标测试函数对算法进行测试, 结果辨明 MODE 算法优于 SPEA 算法[63]。Babu 和 Jehan 采用加权的方法把二目标优化问题转化成单目标优化问题, 并通过改变权值得到不同的解[64]。受 VEGA 算法[66]的启发, Parsopoulos 等提出了 VEDE 算法[65], 该算法是一种并行的多群体差分演化算法。通过 4 个二目标测试函数对 VEDE 算法进行测试并与 VEGA 算法进行比较, 结果表明 VEDE 优于 VEGA 算法。Iorio 和 Li[67]提出了 NSDE 算法, 并把它应用到旋转 MOPs 的求解中,该算法与 NSGA-II[61]的唯一区别在于采用 DE 算子代替 NSGA-II 里面的交叉和变异算子。Kukkonen 和 Lampinen 提出了 GDE 算法[68], 与传统差分演化算法相比较其最主要区别在于选择算子的不同, 使之更适合于求解多目标优化问题。Robič 和 Filipič 提出了 DEMO 算法[69], 该算法采用一个子群体来保存当子个体和父个体非劣时的个体, 并且在子群体非空时把子群体与演化群体合并, 采用非劣解排序和拥挤距离对合并的群体进行裁剪, 然后, 裁剪后的群体作为下一代的演化群体。DEMO 结合 DE 算法的优点, 把 NSGA-II 中的非劣解排序和拥挤距离引入到算法中, 使算法能较快收敛于真正 Pareto 前沿。Kukkonen 和 Lampinen 提出的 GDE3 算法[70], 该算法是对 GDE[68]算法的改进, 增强了算法对约束函数处理的能力。张利彪等[71]提出了基于极大极小距离密度的多目标差分进化算法, 利用极大极小距离密度来保存非劣解集。Abbass 和 Sarker 提出了 PDE 算法[72], 算法利用正态随机数产生初始群体, 并采用自适应交叉变异算子对对群体进行演化。Santana-Quintero 和 Coello 提出了 ε-MyDE 算法[73], 算法把 Deb 等[74]提出的 ε-占优技术和差分演化算法相结合以保存非劣解集。龚文引等提出了 ε-ODEMO 算法[75]。在 ε-ODEMO 算法中, 正交实验设计[76]被用于产生初始群体, 使得初始群体可以均匀地分布于决策空间, 同时, Deb 等提出的 ε-占优技术被用于保存非劣解集, 提出了一种基于随机选择和精英选择的混合选择机制以加快算法的收敛。通过多个二目标和三目标高维测试函数, 并与其他算法相比较验证了 ε-ODEMO 算法的有效性。在 ε-ODEMO 算法基础上, Gong 和 Cai 把 Pareto 自适应 ε-占优技术[77]应用到 ε-ODEMO 算法中提出了 *pa*ε-ODEMO 算法[78], 在 *pa*ε-ODEMO 算法中同时还提出了边界点保存策略, 以保存因 ε-占优技术而丢失的非劣解边界点。此外, Gong 等提出了新的约束函数处理技术, 并将此技术与 *pa*ε-ODEMO 算法相结合用于求解约束多目标优化问题(包括工程优化问题)[79]。

2.3.2 约束函数优化

利用演化算法求解约束函数优化问题主要有以下几类技术[80]: 1) 罚函数法; 2) 特殊表示和算子; 3) 修复算法; 4) 把约束函数和目标函数分开处理; 5) 混合算法等。近年来一些学者把差分演化算法用于求解约束函数优化问题[81]。差分演化算法的创始人之一 Storn 提出了一种约束自适应技术[82]用于差分演化算法中以求解约束优化问题。算法把所有约束条件都放松使得所有个体在初始群体中都是可行的, 然后, 通过一定的机制不断缩减约束条件直到达到最初的约束为止。作者指出此种方法不适合于求解带等式约束的优化问题, 但适合于求解约束满足问题[82]。Lampinen 提出了一种约束差分演化算法[83]。为了处理约束函数, Lampinen 提出了三个约束处理规则: 1) 如果两个个体均是可行个体, 则具有较好目标函数适应值的个

体赢; 2) 如果一个个体是可行个体, 而另一个个体是不可行个体, 则可行个体赢; 3) 如果两个个体均为不可行个体, 则子个体只有在具有较低或者相等的总约束函数值时才能取代其相应的父个体。此类约束处理技术的优点在于不需要额外的参数。Becerra 和 Coello 提出了一种基于文化算法的混合约束差分演化算法[84], 算法中文化算法利用不同知识信息影响差分演化算法的算子。Takahama 和 Sakai 提出了 εDE 算法[85], 算法采用 ε 约束处理技术来处理约束函数, 同时对不可行个体采用基于梯度的变异算子进行变异直到成为可行个体. 为了验证 εDE 的有效性, 作者采用 CEC2006 约束函数测试数据集[86]对算法进行测试, 实验结果表明该算法不仅能快速找到可行解, 而且在求解精度和成功率上均取得了很好的结果。Huang 等提出了一种协演化差分演化算法[87], 算法采用一种特殊的惩罚函数来处理约束条件, 并通过一些协演化机制来对解个体和惩罚函数因子进行协同演化, 从而使得解个体和惩罚函数因子可以相互和自适应的演化。Mezura-Montes 等提出了一种多后代差分演化算法[88], 算法对每个父个体产生多个子个体, 并从中选择最优的一个, 从而增加了父个体产生好的子个体的可能性; 同时, 采用一种基于可行性规则和多样性规则的混合约束函数处理技术来处理约束条件。作者通过一些标准约束函数优化问题和工程优化问题验证了改进算法的有效性。受随机排序选择约束处理技术[89]的启发, Zhang 等提出了一种动态随机约束处理技术[90], 并将其应用于差分演化算法中. 实验表明该算法在处理约束函数优化问题时是有效的[90]. Gong 和 Cai 提出了一种基于多目标思想的约束函数处理技术[91], 算法中目标函数和各个约束函数分别作为一个函数, 然后采用 ε 占优技术来保存非劣解集, 最后从中选择可行个体中选择最好个体作为最后可行解。Fan 等[92]把差分演化算法与随机排序选择约束处理技术[89]相结合用于对微电机系统进行优化。Liu 等提出了一种混合的 PSO-DE 算法[93]。在 PSO-DE 算法中, Deb 提出的约束函数处理技术[94]被用于处理约束函数, 同时, 算法对群体中一半个体采用 PSO 进行演化, 而另一半个体采用差分演化算子进行演化从而可以避免 PSO 算法过早停滞演化。

§2.4 离散差分演化算法

差分演化算法提出之处主要采用实数编码, 因此主要用于求解实数优化问题。但是, 离散优化问题在日常生活中也经常遇到, 如 TSP 问题、图论问题、车间调度问题等。一般来说, 离散优化可分为二进制优化和整型优化等。近年来, 一些学者对传统差分演化算法进行了变形以求解离散优化问题。

2.4.1 二进制差分演化算法

Pamparά等提出了一种二进制差分演化算法(Angle Modulated DE, AMDE)[95]。MADE 仍然采用实数编码, 但通过一个三角法函数将实数个体转换成二进制串用于求解二进制优化问题。作者通过标准测试函数验证了 AMDE 的可行性。Gong 和 Tuson 提出了一种基于正则分析的二进制差分演化算法[96], 算法直接采用二进制编码, 并通过正则分析对传统差分演化算法的变异和杂交算子进行了改动。文章通过一些二进制优化问题(如 MAX-ONE 问题, Order-3 欺骗问题等)验证了算法的可行性。He 和 Han 提出了一种新的二进制差分演化算法[97],

算法直接采用二进制编码，传统差分演化算法中的算数运算被逻辑运算代替以适用于二进制编码中。作者通过 5 个标准测试函数对算法进行测试，并与二进制 PSO 算法进行比较，结果表明新算法搜索能力和收敛速度上优于二进制 PSO 算法。Engelbrecht 和 Pamparά提出了一种基于函数转换的二进制差分演化算法[98]。算法采用实数编码，对个体中的每一维实数，通过一个转换函数将其转换成二进制数。Moraglio 和 Togelius 在分析了差分演化算法变异算子几何特性的基础上，提出了几何差分演化算法[99]，并通过海明距离提出了二进制差分演化算法。作者通过 Spears-DeJong 函数(1)验证了算法的可行性。Greenwood 提出了一种简单的转换机制把传统差分演化算转换成二进制差分演化算法并用于求解图论问题[100]。

表 2-1 差分演化算法的主要应用

工程设计	数字滤波器设计[104]，机械工程设计[79,88,105,106]，空气动力学设计[107-109]，无线电频率集成电路设计[110]，造纸工业中缺陷检测滤波器设计[111]，Kalman 滤波器的设计[112]等.
化学工程与生物系统	发酵过程控制[113]，生物过程系统优化控制[114]，绝热苯乙烯反应堆优化[115]，炼钢中热传递模型优化[116]，非线性化学过程优化[117]等.
控制工程	最优控制和最优时间点问题[118,119]，模糊控制器设计[120]，模糊控制器调节[121,122]，多峰优化控制问题[123]，参数识别[124]，路径规划优化[125]，非线性系统控制[126]等.
电力系统	经济负荷分配问题[127-130]，最优电力潮流问题[131,132]，电力系统规划问题[133]，大规模多总线被动谐滤波器优化[134]，发电扩张规划问题[135]等.
神经网络设计和模糊系统设计	前馈网络训练[136]，前馈网络的训练与编码[137]，在线训练[138]，学习神经元网络来诊断乳腺癌[139]，调节模糊隶属度函数[140]，训练神经网络来预测天气[141]，训练网络来进行非线型系统识别[142]等.
数据挖掘	聚类分析[15,31,32,143-147]，文本聚类[148]，数值关联规则挖掘[14]等.
调度问题	植物调度与规划[149]，并行机器车间调度[150]，企业规划[151]，异步多处理器调度[152]，流水线车间调度[101-103,153]，终端分配问题[154]等.
生物信息学	柔性配体对接问题[155]，蛋白质能量最小化问题[156,157]，非线性生物动力系统的结构识别和参数估计问题[158]等.
图像处理	图像匹配[159,160]，图像非指导分类[161,162,163]等.
决策制定	基于混合整型差分演化算法的化工厂模糊决策制定[164]，Choquet 积分识别问题[165]等.

(1) Spears-DeJong函数可以在线下载: http://www.cs.uwyo.edu/~wspears/functs/dejong.c

2.4.2 整型差分演化算法

Nearchou 和 Omirou 提出了一种子范围编码(sub-range encoding)方法使差分演化算法可以处理整数优化问题[101]。作者将该算法应用于序列和时间表问题中，取得了较好的效果。Onwubolu 和 Davendram 提出了两种转换技术把差分演化算法应用于流水式车间调度问题的求解中[102]。这两种转换技术分别是前向转化技术(从整数变量到实数变量的转换)和后向转换技术(从实数变量到整数变量的转换)。Pan 等[103]在差分演化算法中提出一种分解和构造过程来产生变异个体，从而是差分演化算法可以用于整数问题的求解中。通过把算法应用与互斥流水式车间调度问题验证了算法的有效性。

§2.5 差分演化算法的应用

目前，差分演化算法已经被广泛应用到各个领域中，如工程设计、化学工程、控制工程、电力系统、神经网络设计、数据挖掘、决策制定等。表 2-1 列出了近年来差分演化算法的一些主要应用。

§2.6 本章小结

本章对差分演化算法进行概述，简述了基本差分演化算法流程及其各遗传算子(差分变异算子、杂交算子和选择算子)。随后，指出了基本差分演化算法所存在的一些不足之处，如控制参数设置的敏感性，演化后期收敛较慢，变异策略的问题依赖性等。在此基础上，对当前一些改进的差分演化算法进行了综述。此外，简述了差分演化算法对复杂环境(多目标优化和约束函数优化等)优化问题的应用以及当前离散差分演化算法的研究。最后，本章对当前差分演化算法在各个领域的应用进行了简述。通过本章对差分演化算法概述，对该算法有了直观的了解，这对以后各章节具有铺垫作用。

第三章 基于 BBO 的混合差分演化算法

§3.1 引言

优化问题在日常生活中经常遇到，不失一般性，一个最小化优化问题 $f(X)$ 可以描述如下:

$$\text{Minimize: } f(X)$$

$$\text{Subject to: } \begin{cases} g_i(X) \le 0, & i = 1, \mathbf{L}, l \\ h_j(X) = 0, & j = l+1, \mathbf{L}, n \\ l_k \le X(k) \le u_k, & k = 1, \mathbf{L}, D \end{cases} \tag{3.1}$$

其中，$X = [X(1), X(2), \mathbf{L}, X(D)]^T$ 是一个 D 维向量，$f(X)$ 是目标函数，$g_i(X)$ 是第 i 个不等式约束函数，$h_j(X)$ 是第 $j-l$ 个等式约束函数，$X(k)$ 为第 k 维自变量其范围是 $[l_k, u_k]$。当 $n = 0$ 时，则优化问题是无约束优化问题。本章主要是关注于无约束优化问题的求解。

优化问题随着问题的不同具有不同的特性。传统数值优化算法由于对求解问题的特殊要求(如连续、可微、非凸、单峰等)往往不能有效求解复杂的优化问题，因此，许多学者采用演化算法来求解优化问题。近年来，混合优化算法由于其在实际优化问题中所取得的成功已经受到越来越多的关注。混合优化算法能结合各个算法的优点，对单一算法的缺点进行弥补，从而增强算法的有效性和高效性。由于生物地理学优化算法(Biography-based optimization, BBO)能有效弥补传统差分演化算法利用能力差的缺点，因此，本章将 BBO 算法与差分演化算相结合提出了基于 BBO 算法的混合差分演化算法，即 DE/BBO 算法。

§3.2 生物地理学优化算法

生物地理学是一种研究生物物种地理分布的学科。受生物地理学理论的启发，Simon 提出了一种新的全局优化算法——生物地理学优化(BBO)算法[44]。该算法采用整数编码，根据生物物种在地理上分布的特点，设计一种基于概率的个体移动(habitat migration)算子，使得多个个体间可以相互共享信息，每个个体具有各自的移进概率(immigration rate，l)和移出概率(emigration rate，m)，以控制个体信息的移动概率；同时，为了增强群体的多样性，算法中增加了变异算子，每个个体具有各自的变异概率，变异概率由生物地理学理论计算得到[44]。通过把 BBO 算法应用到 14 个标准测试函数中，并与 7 种演化算法相比较验证了算法的有效性。由于个体移动算子的有效设计，BBO 算法具有较强的利用能力。

BBO 算法是一种全局优化算法[44]，它模拟生物物种在地理分布上的特征，采用基于概率的个体移动算子使个体之间共享信息，这类似生物物种在各岛屿之间相互移动。每个个体具有各自的移进概率 l 和移出概率 m，根据生物地理学理论，单个岛屿的物种移动(移进/移出)概率可以用图 3-1 表示。

图 3-1 中 I 表示最大移进概率，E 表示最大移出概率，$S_{\max}$ 为该岛屿中所能容纳物种的最大数目。从图 1 可以看出，当岛屿中物种数目为 0 时，该岛屿具有最大的移进概率 $l=I$，此时，移出概率 $m=0$，随着物种数目增加 l 逐渐减小，而 m 则不断增大，直到岛屿中物种数目达到最大值 $S_{\max}$，此时该岛屿处于饱和状态，因此 $l=0, m=E$。设 $S_{\max}=n$，第 k 个岛屿所包含的物种个数为 k，则图 1 所表示模型的移进概率 l_k 和移出概率 m_k 计算如下[44]:

$$\begin{cases} l_k = I \times \left(1 - \dfrac{k}{n}\right) \\ m_k = E \times \dfrac{k}{n} \end{cases} \tag{3.2}$$

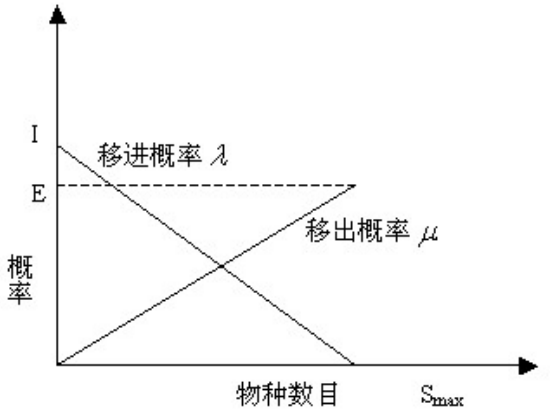

图 3-1 单个岛屿物种移动概率

与遗传算法类似，BBO 算法也是一种基于群体的算法。在基本 BBO 算法中个体采用整数编码，个体表示生物地理学中的岛屿，每个个体具有一个适应值(habitat suitability index, HSI)用于对个体进行评价，个体中的变量为适应性索引变量(suitability index variables, SIVs)。较高 HSI 的个体具有向较低 HSI 个体共享信息的趋势，因此，较差个体可以更多获得较好个体的信息从而得到改进。

§3.3 DE/BBO 算法

3.3.1 研究动机

如第 2.2 节所述，差分演化算法具有很好的开采能力，能对决策变量空间进行有效搜索，从而可以避免算法局部收敛；但是，传统差分演化算法缺乏局部搜索和对群体信息的有效利

用能力，因此，算法的收敛速度有待进一步改进。由于 BBO 算法的个体移动算法能有效利用当前群体中个体的有效信息，加速算法的收敛速度，因此，本章把 BBO 算法的个体移动算子与差分演化算法的变异算子相结合，提出一种新的混合杂交变异算子，该算子能有效平衡算法的开采能力和平衡能力，从而提高差分演化算法的性能。

3.3.2 混合杂交算子

DE/BBO 算法的主要遗传算子是混合杂交算子(或混合移动算子)，该算子是差分变异算子和 BBO 算法的移动算子的混合。图 3-2 描述了该算子的基本流程，其中 l_k 和 m_k 分别第 k 个体的移进概率和移出概率，按照公式(3.2)进行计算。从图 3-2 可以看出，通过该混合算子所产生的子个体 U_i 可能有三部分组成: 1) 差分变异向量，2) 通过移动算子所得到的其他个体成分，和 3) 该子个体相应的父个体 X_i 。

1: **for** $i = 1$ to NP **do**
2: 随机均匀选择$r_1 \neq r_2 \neq r_3 \neq i$
3: $j_{rand} = \text{rndint}(1, D)$
4: **for** $j = 1$ to D **do**
5: **if** $\text{rndreal}(0,1) < \lambda_i$ **then**
6: **if** $\text{rndreal}_j[0,1) < CR$ or $j == j_{rand}$ **then**
7: $U_i(j) = X_{r_1}(j) + F \times \left(X_{r_2}(j) - X_{r_3}(j)\right)$ {原始差分变异算子}
8: **else**
9: 按概率$\propto \mu_k$选择X_k
10: $U_i(j) = X_k(j)$
11: **end if**
12: **else**
13: $U_i(j) = X_i(j)$
14: **end if**
15: **end for**
16: **end for**

图 3-2 DE/BBO 算法混合杂交算子

设计该混合杂交算子基于两点考虑: 1) 群体中好的个体应尽可能不被破坏，而差的个体则要尽可能多的接受较好个体的信息，这样，当前群体的信息就能得到充分的利用。2) 原始差分变异算子可以对搜索空间进行开采，增强算法的鲁棒性。根据文献[166]对混合启发式算法的增强性(intensification)和多样性(diversification)的综述，可以看出混合杂交算子的移动算子强调算法的增强性，而差分变异算子在强调算法的多样性。

从上面的分析可以看出，DE/BBO 算法的混合杂交算子可以使算法的开采能力和利用能力得到有效的平衡。值得指出的是，图 3-2 的第 7 行是以差分演化算法的“DE/rand/1”变异

算子作为示例，其他差分变异算子同样也可以应用到该混合杂交算子中。本章将在后面的实验部分就不同差分变异算子对 DE/BBO 算法性能的影响进行讨论。

```
产生初始群体P
评价初始群体P中所有个体适应值
while 如果终止条件没有满足 do
    对群体按照适应值从好到差进行排序
    对每个个体根绝适应值计算其物种个数
    计算每个个体X_i的移进概率λ_i和移出概率μ_i
    利用图3-2所示的混合杂交算子对群体进行重组
    for i = 1 to NP do
        评价新个体U_i
        if U_i优于X_i then
            X_i = U_i
        end if
    end for
end while
```

图 3-3 DE/BBO 算法基本流程

3.3.3 自变量越界处理

利用图 3-2 所示的混合杂交算子产生的子个体U_i可能会导致部分自变量超过其相应的边界约束，即$U(i)\notin[l_i,u_i],\ i=1,\mathbf{L},D$，因此需要采用一定的修复方法对其进行修复。本章采用如下规则进行修复:

$$U(i)=\begin{cases} l_i+\text{rndreal}_i[0,1]\times(u_i-l_i), & \text{if } U(i)<l_i \\ u_i-\text{rndreal}_i[0,1]\times(u_i-l_i), & \text{if } U(i)>u_i \end{cases} \tag{3.3}$$

其中，$\text{rndreal}_i[0,1]$表示对第 i 自变量在[0,1]之间产生均匀随机数。

3.3.4 DE/BBO 算法流程

把图3-2所示的混合杂交算子嵌入到传统差分演化算法中，则形成了基于BBO算法的混合差分演化算法，即 DE/BBO 算法。该算法的基本流程如图 3-3 所示。与传统差分演化算法相比，DE/BBO 算法只在对群体进行排序和计算个体的移进概率和移出概率上增加了额外的计算代价，但是，这些代价与整个算法的计算代价相比较是可以忽略不计的。DE/BBO 算法的结构同样是很简单的，且容易实现。由于采用了混合杂交算子，DE/BBO 算法可以一方面利用差分变异算子开采搜索空间，另一方面还可以通过 BBO 算法的移动算子对群体信息进行有效利用，这样可以克服传统差分演化算法缺乏利用能力的缺点。值得指出的是 DE/BBO 算

法和差分演化算法的唯一区别在于混合杂交算子的使用。DE/BBO 算法保持了差分演化算法的一对一选择算子，因此，该算法仍然是一种保持精英策略的算法。

§3.4 实验结果及分析

3.4.1 测试函数集

为了验证 DE/BBO 算法的性能，本章从文献[167]中选取了 23 个测试函数，它们的简要介绍见表 3-1，其详细介绍见附录 1 或文献[167]。这 23 个函数可以分为 3 大类: 1) 单峰函数; 2) 多峰多极值函数; 3) 多峰少极值函数。选择这 23 个函数的原因在于: 这些函数具有很好的定义和多种函数特征，且它们被广泛选用[8,37,52,53,54]。

表 3-1 实验标准参试函数。详细请参考附录 1 或文献[167].

函数	函数名称	维数 D	自变量空间 S	已知最优解
f01	Sphere Model	30	$[-100,100]^D$	0
f02	Schwefel's Problem 2.22	30	$[-10,10]^D$	0
f03	Schwefel's Problem 1.2	30	$[-100,100]^D$	0
f04	Schwefel's Problems 2.21	30	$[-100,100]^D$	0
f05	Generalized Rosenbrock's Function	30	$[-30,30]^D$	0
f06	Step Function	30	$[-100,100]^D$	0
f07	Quartic Function	30	$[-1.28,1.28]^D$	0
f08	Generalized Schwefel's Problem 2.26	30	$[-500,500]^D$	$-418.98289\times D$
f09	Generalized Rastrigin's Function	30	$[-5.12,5.12]^D$	0
f10	Ackley's Function	30	$[-32,32]^D$	0
f11	Generalized Griewand Function	30	$[-600,600]^D$	0
f12	Generalized Penalized Function 1	39	$[-50,50]^D$	0
f13	Generalized Penalized Function 2	39	$[-50,50]^D$	0
f14	Shekel's Foxholes Function	2	$[-65.536,65.536]^D$	1
f15	Kowalik's Function	4	$[-5,5]^D$	0.003705
f16	Six-Hump Camel-Back Function	2	$[-5,5]^D$	-1.0316285
f17	Branin Funrion	2	$[-5,10]\times[0,15]$	0.398
f18	Glodstein-Price Function	2	$[0,1]^D$	3
f19	Hartman's Function 1	3	$[0,1]^D$	-3.86
f20	Hartman's Function 2	6	$[0,1]^D$	-3.32
f21	Shekel's Function 1	4	$[0,10]^D$	-10.1532
f22	Shekel's Function 2	4	$[0,10]^D$	-10.4029
f23	Shekel's Function 3	4	$[0,10]^D$	-10.5364

函数 f01 – f13 是高维的维数可变的函数。函数 f01 – f05 是单峰函数; f06 是步长函数，该函数有一个最小值且是不连续函数; f07 是一个噪声函数。函数 f08 – f13 是多峰多极值函数，这些函数的局部极值个数为函数维数的增加呈指数级增长。这类函数对大多数算法都是很难求解的。函数 f14 – f23 是多峰少极值的低维函数。

3.4.2 实验参数设置

对于 DE/BBO 算法，本章所采用的参数设置主要参考已有文献的一些设置，而没有去寻找最优参数设置，因为这已经超出本章的研究范围。对于所有测试函数如果没有特殊说明其参数设置如表 3-2 所示。每个函数的最大适应值评价次数(Max_NFFEs)见表 3-3 的第二列。

表 3-2 DE/BBO 算法参数设置

参数名称	参数值
群体大小: *NP*	100[7, 8, 46, 167]
最大移进概率: *I*	1.0[44]
最大移出概率: *E*	1.0[44]
缩放因子: *F*	rndreal(0,1) [5, 17]
杂交概率: *CR*	0.9[6, 40, 45, 49]
差分变异策略:	DE/rand/1[6, 45, 49]
求解精度: *VTR*	10^{-8} [168]，除了函数 f07，$VTR = 10^{-2}$

此外，为了能对算法结果进行有效统计，每个函数每次独立优化 50 次，统计其平均最优值(Mean)和标准方差(Std Dev)。在对不同算法进行比较时，每次独立运行各算法采用相同的初始群体，这样使得各算法之间的比较是公平的。所有算法采用标准 C++语言实现。

3.4.3 评价标准

为了对算法性能进行评估，本章采用了文献[40]和[168]中提出的 5 中评价标准，它们是:

1) **误差值(Error)**[168]: 解向量 X 的误差值为 $f(X)-f(X^*)$，其中 X^* 表示优化问题已经最优解向量。程序对每次运行达到 Max_NFFEs 是统计其最小的误差值，然后对 50 次运行结果统计其均值和方差。
2) **适应值评价次数(NFFEs)**[168]: 当算法在每次运行时，在当前适应值评价次数没有达到 Max_NFFEs 且最优解误差值达到或小于 VTR 时记录其 NFFEs 值，然后对 50 次运行结果统计其均值和方差。
3) **成功运行次数(SR)**[168]: 如果当前适应值评价次数没有达到 Max_NFFEs 且最优解误差值达到或小于 VTR，则表示此次运行是成功的。
4) **收敛曲线图**[168]: 收敛曲线图表示算法在所有 50 次运行中最优解的平均误差值曲线图。
5) **加速率(AR)**[40]: 加速率用于比较DE/BBO算法与其他算法之间的收敛速度，其计算如下:

$$AR = \frac{\mathrm{NFFEs}_{\mathrm{other}}}{\mathrm{NFFEs}_{\mathrm{DE/BBO}}} \tag{3.4}$$

如果 $AR > 1$ 则表示 DE/BBO 算法收敛速度快于所比较的算法。

3.4.4 实验结果

3.4.4.1 DE/BBO 算法一般性能

为了展示 DE/BBO 算法的优越性，本小节把该算法与传统差分演化(DE)算法和 BBO 算法进行比较。DE/BBO 和 DE 算法的参数如表 3-2 所示，BBO 算法的参数设置与文献[44]一样。每个测试函数独立运行 50 次。表 3-3 列出了 DE/BBO、DE 和 BBO 算法的最优误差值的统计结果；表 3-4 则列出了三个算法的 NFFEs 的平均值和方差。此外，图 3-4 展示了一些典型的收敛曲线图。

表 3-3 DE/BBO、DE 和 BBO 算法的最优误差值的统计结果，其中“1vs2”表示“DE/BBO vs DE”，“1vs3”表示“DE/BBO vs BBO”。表中较好结果用**黑体**表示，下同。

F	NFFEs	DE/BBO			DE			BBO			1 vs 2	1 vs 3
		Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	*t*-test	*t*-test
f01	150000	8.66E-28	5.21E-28	50	1.10E-19	1.34E-19	50	8.86E-01	3.26E-01	0	-5.81†	-19.21†
f02	200000	0.00E+00	0.00E+00	50	1.66E-15	8.87E-16	50	2.42E-01	4.58E-02	0	-13.24†	-37.36†
f03	500000	2.26E-03	1.58E-03	0	8.19E-12	1.65E-11	50	4.16E+02	2.02E+02	0	10.10‡	-14.58†
f04	500000	1.89E-15	8.85E-16	50	7.83E+00	3.78E+00	0	7.76E-01	1.72E-01	0	-14.65†	-31.96†
f05	500000	1.90E+01	7.52E+00	0	8.41E-01	1.53E+00	6	9.14E+01	3.78E+01	0	16.73‡	-13.28†
f06	150000	0.00E+00	0.00E+00	50	0.00E+00	0.00E+00	50	2.80E-01	5.36E-01	38	0	-3.69†
f07	300000	3.44E-03	8.27E-04	50	3.49E-03	9.60E-04	50	1.90E-02	7.29E-03	4	-0.29	-14.96†
f08	300000	0.00E+00	0.00E+00	50	4.28E+02	4.69E+02	1	5.09E-01	1.65E-01	0	-6.45†	-21.78†
f09	300000	0.00E+00	0.00E+00	50	1.14E+01	7.57E+00	0	8.50E-02	3.42E-02	0	-10.61†	-17.61†
f10	150000	1.07E-14	1.90E-15	50	6.73E-11	2.86E-11	50	3.48E-01	7.06E-02	0	-16.66†	-34.81†
f11	200000	0.00E+00	0.00E+00	50	1.23E-03	3.16E-03	43	4.82E-01	1.27E-01	0	-2.76†	-26.93†
f12	150000	7.16E-29	6.30E-29	50	2.07E-03	1.47E-02	49	5.29E-03	5.21E-03	0	-1	-7.18†
f13	150000	9.81E-27	7.10E-27	50	7.19E-02	5.09E-01	49	1.42E-01	5.14E-02	0	-1	-19.50†
f14	10000	0.00E+00	0.00E+00	50	2.75E-13	1.55E-12	50	8.85E-06	2.74E-05	14	-1.26	-2.28†
f15	40000	3.84E-12	2.70E-11	50	4.94E-19	5.20E-19	50	5.92E-04	2.68E-04	0	1.01	-15.65†
f16	10000	1.15E-12	6.39E-12	50	1.98E-13	4.12E-13	50	6.75E-04	1.09E-03	0	1.05	-4.37†
f17	10000	2.92E-10	1.38E-09	50	7.32E-10	2.21E-09	49	4.39E-04	4.26E-04	0	-1.19	-7.30†
f18	10000	9.15E-13	6.51E-15	50	9.14E-13	5.01E-15	50	7.86E-03	9.57E-03	0	0.7	-5.81†
f19	10000	0.00E+00	0.00E+00	50	1.36E-14	6.40E-15	50	2.51E-04	2.62E-04	0	-15.05†	-6.76†
f20	20000	0.00E+00	0.00E+00	50	4.76E-03	2.35E-02	47	1.46E-02	3.90E-02	0	-1.43	-2.64†
f21	10000	3.59E-03	1.44E-02	15	6.83E-06	1.26E-05	0	5.18E+00	3.34E+00	0	1.76	-10.95†
f22	10000	3.14E-07	1.36E-06	29	7.26E-06	4.53E-05	1	3.67E+00	3.40E+00	0	-1.08	-7.63†
f23	10000	2.50E-08	5.37E-08	27	3.70E-06	1.75E-05	0	2.73E+00	3.29E+00	0	-1.49	-5.87†

†‡表示在自由度为 $49, a = 0.05$ 时 t 测试结果差异明显；

‡表示相应算法优于 DE/BBO 算法，下同。

表 3-4 算法成功收敛时的 NFFEs 值统计，其中，“NA”表示算法在 Max_NFFEs 内不能成功收敛。

F	DE/BBO			DE			BBO			1 vs 2	1 vs 3
	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	*AR*	*AR*
f01	**59926**	745.5	50	79688	1858.8	50	NA	NA	0	1.33	NA
f02	**82004**	983.9	50	119764	1871.2	50	NA	NA	0	1.46	NA
f03	NA	NA	0	**385990**	17193.4	50	NA	NA	0	NA	NA
f04	**296572**	4969.9	50	NA	NA	0	NA	NA	0	NA	NA
f05	NA	NA	0	**448583**	60428.8	6	NA	NA	0	NA	NA
f06	**21590**	573.3	50	28874	2014.5	50	119042	16882.8	38	1.34	5.51
f07	109574	21005.8	50	**103136**	29677.7	50	205000	23896.2	4	0.94	1.87
f08	**95952**	3126.7	50	251700	0	1	NA	NA	0	2.62	NA
f09	**170226**	8379.0	50	NA	NA	0	NA	NA	0	NA	NA
f10	**91308**	922.7	50	122340	2179.6	50	NA	NA	0	1.34	NA
f11	**62042**	1219.6	50	81986	1795.8	43	NA	NA	0	1.32	NA
f12	**54482**	873.3	50	71183	4700.9	49	NA	NA	0	1.31	NA
f13	**64772**	1133.4	50	93298	16916.8	49	NA	NA	0	1.44	NA
f14	**4532**	719.5	50	6768	766.0	50	5757	2351.3	14	1.49	1.27
f15	24028	3279.3	50	**12590**	977.8	50	NA	NA	0	0.52	NA
f16	**5676**	1012.7	50	5760	632.5	50	NA	NA	0	1.01	NA
f17	**7138**	1404.9	50	7271	1098.7	49	NA	NA	0	1.02	NA
f18	5050	374.3	50	**4610**	292.9	50	NA	NA	0	0.91	NA
f19	**4808**	352.2	50	5434	346.8	50	NA	NA	0	1.13	NA
f20	**9614**	705.1	50	14161	1553.3	47	NA	NA	0	1.47	NA
f21	**9560**	397.9	15	NA	NA	0	NA	NA	0	NA	NA
f22	**9527**	349.4	29	10000	0	1	NA	NA	0	1.05	NA
f23	**9533**	362.7	27	NA	NA	0	NA	NA	0	NA	NA

与 DE 算法比较：从表 3-3 可以看出 DE/BBO 算法在 8 个函数上明显优于 DE 算法，但是对两个函数(f03 和 f05), DE 算法明显优于 DE/BBO 算法。对剩下 13 个函数，*t* 测试结果显示两个算法之间无明显差异。对于多峰多极值函数(f08 – f13), DE/BBO 算法在所有 50 次运行中均能成功收敛：即在各函数 Max_NFFEs 内，最优误差值均小于或等于 $VTR = 10^{-8}$ 。但是，DE 算法对 5 个函数会有时陷入局部最优解。这表明 DE/BBO 算法能有效跳出差的局部最优解，而收敛到全局最优解附近。明显地，从表 3-4 可以看出在 18 个函数上 DE/BBO 算法需要较少的 NFFEs 值来达到 VTR 值。此外，从图 3-4 可以看出对大部分测试函数, DE/BBO 算法的收敛速度快于 DE 算法。

与 BBO 算法比较：从表 3-3、3-4 和图 3-4 可以看出，对于所有测试函数和所有 5 个评价标准, DE/BBO 算法明显优于 BBO 算法。仔细观察图 3-4 可以看出, BBO 算法在演化初期收敛速度快于 DE/BBO 算法，但是，在中后期，DE/BBO 算法优于 BBO 算法。其原因可能是

BBO 算法对当前群体具有较好的利用能力，但是其缺乏开采能力，因此不能有效开拓新的搜索空间。而 DE/BBO 算法则通过其混合杂交算子有效平衡了开采能力和利用能力，从而在各方面性能上优于 BBO 算法。

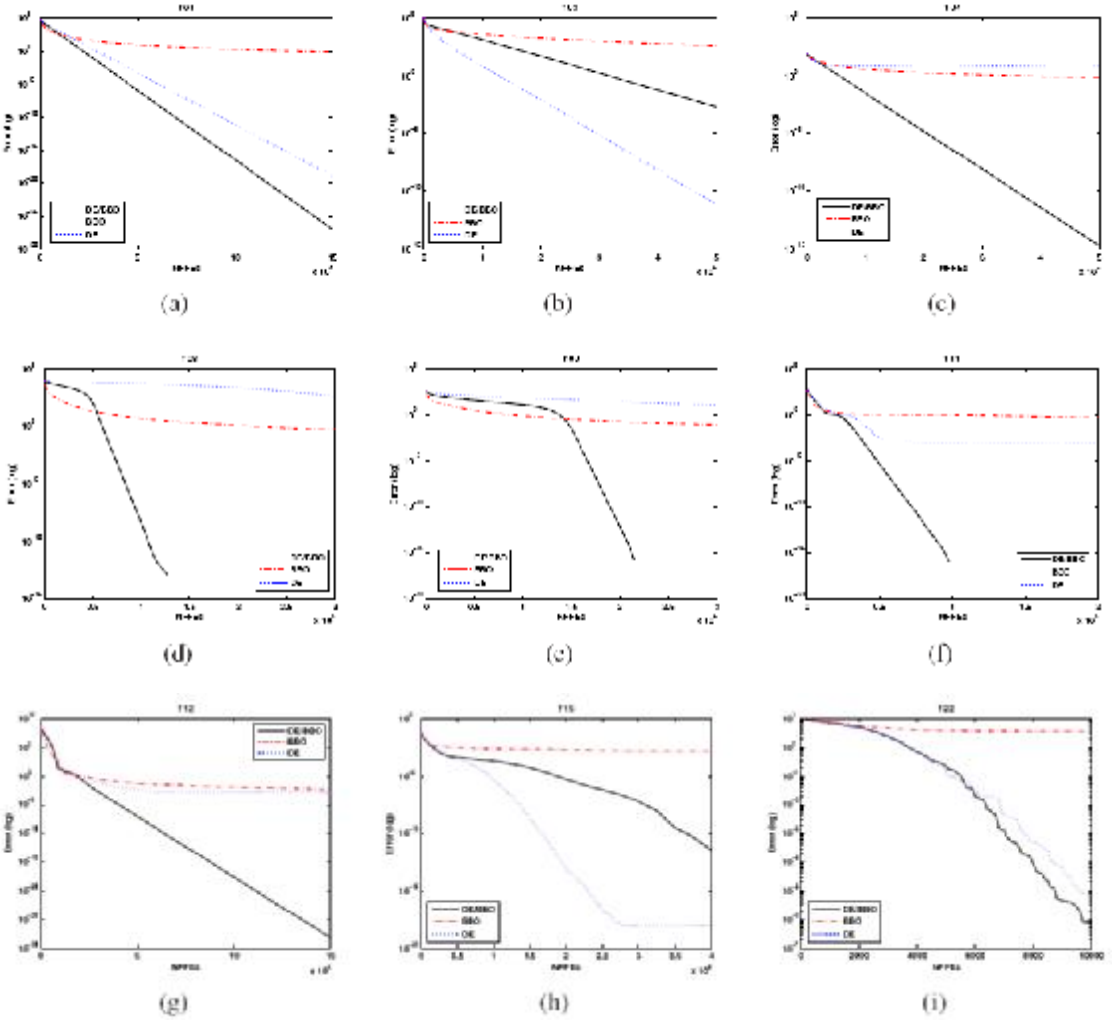

图 3-4 对选择的函数 DE/BBO, DE 和 BBO 算法的收敛曲线图。(a) f01. (b) f03. (c) f04. (d) f08. (e) f09. (f) f11. (g) f12. (h) f15. (i) f22. (彩色图请见相应的电子文件。下同。)

总之, DE/BBO 算法与 DE 算法具有很强的可比性，特别是对于高维优化问题。在所有测试函数中，DE/BBO 算法明显优于 BBO 算法。因为对于大部分低维测试函数(f14 – f23), DE/BBO 算法和 DE 算法之间没有明显差距，在下面的实验中将不再列数这些函数的结果。此外，由于 BBO 算法的性能很差，在后面的试验中将不再与其他算法进行比较。

3.4.4.2 群体大小的影响

对于不同的问题选择最优差分演化算法的群体大小是很困难的[9,49,50]。增加群体大小将增加群体的多样性，增强对搜索空间的开采能力；但是，同时也会减低收敛速度和减少找出最优搜索方向的可能性[11]。本小节就群体大小对算法性能的影响进行实验，除了 $NP = 50,150,200$ 外，其他参数设置与表 3-2 相同。

表 3-5 对函数 f01- f13 ($D=30$)不同群体大小对算法性能的影响。(#)表示成功运行次数，[$a \pm b$] 表示算法成功收敛时 NFFEs 值的均值和方差，下同。

F	NP=50		NP=100	
	DE/BBO	DE	DE/BBO	DE
f01	**4.93E-34±2.44E-33(**50)	6.73E-33±9.09E-33(50)†	**8.66E-28±5.21E-28(**50)	1.10E-19±1.34E-19(50)†
f02	**0.00E+00±0.00E+00(**50)	1.55E-17±4.49E-17(50)†	**0.00E+00±0.00E+00(**50)	1.66E-15±8.87E-16(50)†
f03	7.69E-13±8.68E-13(50)	**3.03E-17±2.10E-16(**50)‡	2.26E-03±1.58E-03(0)	**8.19E-12±1.65E-11(**50)‡
f04	**1.37E-06±6.00E-06(**40)	1.72E+01±5.64E+00(0)†	**1.89E-15±8.85E-16(**50)	7.83E+00±3.78E+00(0)†
f05	**1.10E+01±5.63E+00(**0)	1.27E+01±7.26E+00(0)	1.90E+01±7.52E+00(0)	**8.41E-01±1.53E+00(**6)‡
f06	**[9.39E+03±3.09E+02](**50)	[1.95E+04±6.20E+03](50)†	**[2.16E+04±5.73E+02](**50)	[2.89E+04±2.01E+03](50)†
f07	**1.64E-03±3.67E-04(**50)	4.19E-03±2.05E-03(50)†	**3.44E-03±8.27E-04(**50)	3.49E-03±9.60E-04(50)
f08	**2.37E+01±5.35E+01(**41)	5.57E+02±3.38E+02(0)†	**0.00E+00±0.00E+00(**50)	4.28E+02±4.69E+02(1)†
f09	**5.37E-01±7.84E-01(**31)	1.23E+01±4.17E+00(0)†	**0.00E+00±0.00E+00(**50)	1.14E+01±7.57E+00(0)†
f10	**4.14E-15±0.00E+00(**50)	7.45E-02±2.55E-01(46)†	**1.07E-14±1.90E-15(**50)	6.73E-11±2.86E-11(50)†
f11	**3.45E-04±1.73E-03(**48)	4.83E-03±7.90E-03(32)†	**0.00E+00±0.00E+00(**50)	1.23E-03±3.16E-03(43)†
f12	**1.57E-32±0.00E+00(**50)	4.98E-02±1.26E-01(41)†	**7.16E-29±6.30E-29(**50)	2.07E-03±1.47E-02(49)
f13	**1.36E-32±7.15E-34(**50)	3.61E+00±1.80E+01(36)	**9.81E-27±7.10E-27(**50)	7.19E-02±5.09E-01(49)
F	NP=150		NP=200	
	DE/BBO	DE	DE/BBO	DE
f01	**7.60E-23±3.75E-23(**50)	6.36E-12±4.24E-12(50)†	**1.91E-16±7.51E-17(**50)	7.24E-08±2.98E-08(0)†
f02	**0.00E+00±0.00E+00(**50)	1.12E-09±3.67E-10(50)†	**1.88E-14±4.27E-15(**50)	8.87E-07±2.37E-07(0)†
f03	8.10E-01±4.63E-01(0)	**1.16E-07±2.24E-07(**2)‡	1.98E+01±9.76E+00(0)	**3.27E-05±3.17E-05(**0)
f04	**2.33E-13±1.02E-13(**50)	4.74E+00±3.21E+00(0)†	**6.53E-10±2.04E-10(**50)	2.61E+00±1.95E+00(0)†
f05	1.99E+01±5.26E-01(0)	**1.84E+00±1.58E+00(**0)‡	2.11E+01±4.06E-01(0)	**4.85E+00±1.63E+00(**0)‡
f06	**[2.56E+04±6.27E+02](**50)	[4.27E+04±1.53E+03](50)†	**[3.36E+04±6.97E+02](**50)	[5.78E+04±1.74E+03](50)†
f07	**3.93E-03±7.69E-04(**50)	4.39E-03±1.10E-03(50)†	**5.03E-03±1.09E-03(**50)	5.55E-03±1.75E-03(50)‡
f08	**0.00E+00±0.00E+00(**50)	2.35E+03±1.13E+03(0)†	**0.00E+00±0.00E+00(**50)	3.10E+03±1.04E+03(0)†
f09	**0.00E+00±0.00E+00(**50)	3.27E+01±1.43E+01(0)†	**0.00E+00±0.00E+00(**50)	4.86E+01±1.21E+01(0)†
f10	**1.91E-12±5.21E-13(**50)	6.40E-07±1.98E-07(0)†	**3.11E-09±5.76E-10(**50)	7.04E-05±1.68E-05(0)†
f11	**0.00E+00±0.00E+00(**50)	2.22E-18±1.57E-17(50)	**0.00E+00±0.00E+00(**50)	4.93E-04±1.99E-03(47)
f12	**5.13E-24±3.15E-24(**50)	2.07E-03±1.47E-02(49)	**1.17E-17±5.01E-18(**50)	1.58E-09±1.01E-09(50)†
f13	**6.84E-22±5.10E-22(**50)	2.51E-03±1.78E-02(49)	**1.75E-15±8.37E-16(**50)	9.58E-08±7.78E-08(0)†

对于不同的群体大小, DE/BBO 和 DE 算法的实验结果如表 3-5 所示。从表中可以看出: 1) 对于 $NP=50$，在 10 个函数上 DE/BBO 算法明显优于 DE 算法; 对函数 f03, DE 算法明显优于 DE/BBO 算法; 对函数 f05, DE/BBO 算法稍优于 DE 算法。2) 当群体大小增加到 $NP=100$，与 $NP=50$ 相比 DE 算法可以获得更高的成功运行次数。在 8 个函数上 DE/BBO 算法明显优于 DE 算法; 对函数 f03 和 f05, DE/BBO 算法明显差于 DE 算法; 对剩下 3 个函数, DE/BBO 算

法稍优于 DE 算法。3) 对于 $NP=150$ 和 $NP=200$，DE/BBO 算法分别在 8 个和 9 个函数上明显优于 DE 算法；类似地，对函数 f03 和 f05，DE/BBO 算法明显差于 DE 算法。

总的来说，对于不同的群体大小 DE/BBO 算法的总性能优于 DE 算法。DE/BBO 算法可以获得更高的成功运行次数，较快的收敛速度和更强的鲁棒性。

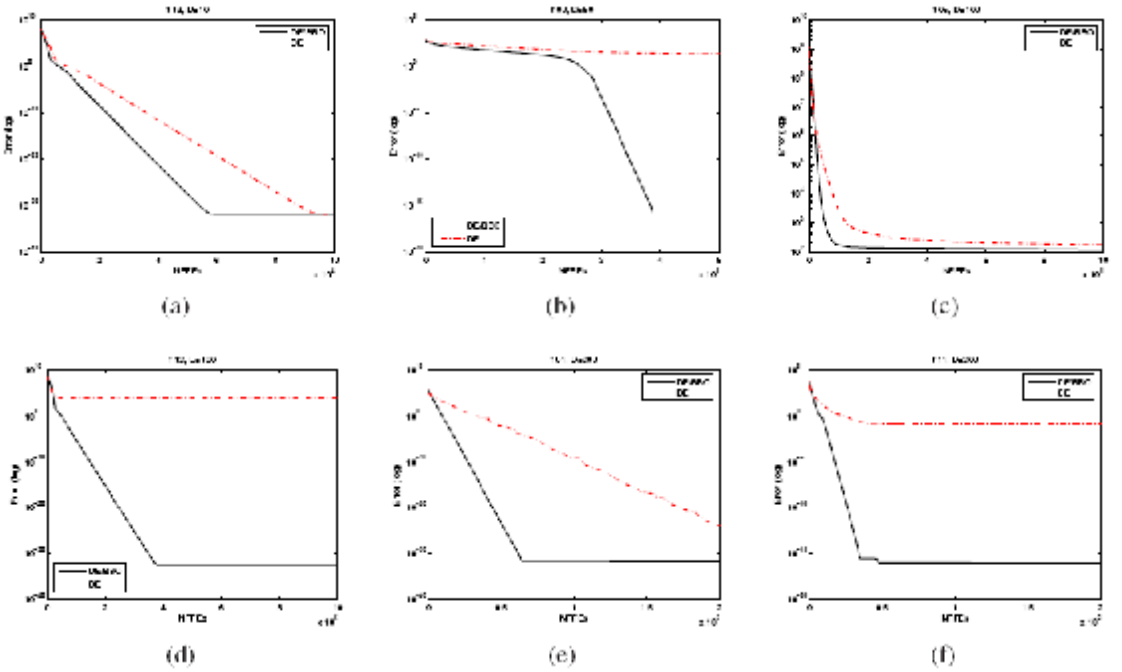

图 3-5 DE/BBO 和 DE 算法在不同维数时的收敛曲线图。(a) f13 (D=10). (b) f09 (D=50). (c) f05 (D=100). (d) f12 (D=100). (e) f01 (D=200). (f) f11 (D=200).

3.4.4.3 问题维数的影响

为了验证问题维数对 DE/BBO 算法性能的影响，本小节对函数 f01 – f13 进行了伸缩性实验($D=10,50,100,200$)，并把实验结果与 DE 算法进行比较。此处，除了最大适应值评价次数 $Max\_NFFEs=D\times10000$ 外，其他参数保持如表 3-2 所示不变。实验结果如表 3-6 所示，一些代表性的收敛曲线图如图 3-5 所示。

从表 3-6 可以看出，随着问题维数的增加 DE/BBO 和 DE 算法总的成功次数(总 SR 值)在不断减少，这是因为由于问题维数的增加导致算法有时不能在 Max_NFFEs 内成功求解该问题。但是，与 $D=30$ 相似，对于不同的维数 DE/BBO 算法在大部分函数上优于 DE 算法。仔细观察实验结果可以看出，对于函数 f05 在 $D=30$ 时，DE 算法优于 DE/BBO 算法。但是，当问题维数增加之后(如 $D=50,100,200$)，DE/BBO 算法要明显优于 DE 算法。此外，从图 3-5 可以看出对于大部分函数在不同维数时，DE/BBO 算法的收敛速度快于 DE 算法。因此，从此处的实验结果可以得出结论：DE/BBO 算法的混合杂交算子能够加快 DE 算法的收敛，尤其是对于高维函数其效果更明显。

3.4.4.4 不同差分变异策略的影响

本节致力于在不同差分变异策略下 DE/BBO 算法与 DE 算法的性能比较。四种不同的策略，即 DE/best/1，DE/rand/2，DE/rand-to-best/1 和 DE/best/2 被用来代替图 3-2 中的 DE/rand/1

变异策略。对于 DE 算法各策略均采用二项式杂交。所有其他参数如表 3-2 所示。表 3-7 给出了 DE/BBO 和 DE 算法在不同变异策略时的结果。基于 t 测试，把实验结果采用“$w/t/l$”来表示，其中 w 表示 DE/BBO 算法赢的函数个数，t 表示 DE/BBO 和 DE 算法相持平的函数个数，l 表示 DE/BBO 算法输的函数个数。从表 3-7 可以看出，对 DE/best/1，DE/rand/2，DE/rand-to-best/1 和 DE/best/2 变异策略，它们分别是 12/1/0，9/2/2，8/3/2，以及 10/2/1。该实验结果表明对于不同的变异策略 DE/BBO 算法与 DE 算法相比较具有更好的鲁棒性。

表 3-6 对函数 f01 – f13 的可伸缩性研究($Max\_NFFEs = D \times 10000$).

F	D=10		D=50	
	DE/BBO	DE	DE/BBO	DE
f01	**[1.89E+04±3.63E+02](**50)	[3.08E+04±8.41E+02](50)†	**0.00E+00±0.00E+00(**50)	1.87E-32±1.64E-32(50)†
f02	**[2.63E+04±4.19E+02](**50)	[4.71E+04±7.61E+02](50)†	**0.00E+00±0.00E+00(**50)	1.44E-17±3.97E-17(50)†
f03	6.07E-14±9.60E-14(50)	**1.35E-21±2.49E-21(**50)‡	5.20E+02±2.48E+02(0)	**1.12E-01±8.06E-02(**0)‡
f04	**1.58E-16±9.88E-17(**50)	1.39E-13±1.22E-13(50)†	**3.42E-03±2.28E-02(**3)	1.95E+01±4.14E+00(0)†
f05	3.40E+00±8.03E-01(0)	**3.53E-11±1.24E-10(**50)‡	**4.43E+01±1.39E+01(**0)	5.33E+01±2.89E+01(0)
f06	**[6.62E+03±3.28E+02](**50)	[1.06E+04±5.50E+02](50)†	**[2.74E+04±6.22E+02](**50)	[5.57E+04±1.64E+04](50)†
f07	**1.11E-03±3.96E-04(**50)	1.52E-03±5.97E-04(50)†	**4.05E-03±9.20E-04(**50)	9.49E-03±3.07E-03(34)†
f08	**0.00E+00±0.00E+00(**50)	1.51E-11±1.05E-10(50)	**2.37E+00±1.67E+01(**49)	1.51E+03±9.97E+02(0)†
f09	**0.00E+00±0.00E+00(**50)	3.75E+00±2.72E+00(5)†	**0.00E+00±0.00E+00(**50)	2.33E+01±8.27E+00(0)†
f10	**5.89E-16±0.00E+00(**50)	8.02E-16±8.52E-16(50)	**5.92E-15±1.79E-15(**50)	3.52E-02±1.74E-01(48)
f11	**0.00E+00±0.00E+00(**50)	8.54E-03±1.56E-02(31)†	**0.00E+00±0.00E+00(**50)	5.08E-03±1.62E-02(9)†
f12	4.71E-32±0.00E+00(50)	4.71E-32±0.00E+00(50)	**9.42E-33±0.00E+00(**50)	4.51E-02±1.40E-01(41)†
f13	1.35E-32±0.00E+00(50)	1.35E-32±0.00E+00(50)	**1.35E-32±0.00E+00(**50)	1.34E-01±5.74E-01(40)
F	D=100		D=200	
	DE/BBO	DE	DE/BBO	DE
f01	**6.16E-34±1.97E-33(**50)	2.30E-31±1.37E-31(50)†	**3.07E-32±2.48E-32(**50)	1.41E-24±3.14E-24(50)†
f02	**0.00E+00±0.00E+00(**50)	7.95E-16±1.05E-15(50)†	**5.83E-17±8.11E-17(**50)	7.38E-09±2.33E-08(46)†
f03	3.10E+04±1.12E+04(0)	**1.31E+02±5.76E+01(**0)‡	2.22E+05±5.90E+04(0)	**3.64E+03±7.60E+02(**0)‡
f04	**2.71E+00±2.58E+00(**0)	3.05E+01±4.00E+00(0)†	**1.59E+01±3.43E+00(**0)	4.27E+01±4.31E+00(0)†
f05	**1.19E+02±3.38E+01(**0)	1.76E+02±4.22E+01(0)†	**2.95E+02±4.48E+01(**0)	4.39E+02±1.11E+02(0)†
f06	**[4.93E+04±1.11E+03](**50)	[3.30E+05±1.34E+05](50)†	**0.00E+00±0.00E+00(**50)	3.00E+00±7.75E+00(20)
f07	**6.87E-03±1.15E-03(**49)	4.84E-02±2.19E-02(0)†	**1.56E-02±2.82E-03(**0)	2.19E-01±7.38E-02(0)†
f08	**7.11E+00±2.84E+01(**47)	6.79E+03±1.07E+03(0)†	**2.01E+02±1.68E+02(**23)	2.47E+04±2.44E+03(0)†
f09	**7.36E-01±8.48E-01(**23)	7.28E+01±1.07E+01(0)†	**1.76E+01±2.89E+00(**0)	2.12E+02±2.12E+01(0)†
f10	**7.84E-15±7.03E-16(**50)	1.87E+00±5.72E-01(1)†	**1.09E-14±1.12E-15(**50)	5.30E+00±8.47E-01(0)†
f11	**0.00E+00±0.00E+00(**50)	8.43E-03±1.71E-02(36)†	**1.11E-16±0.00E+00(**50)	1.33E-01±2.50E-01(0)
f12	**4.71E-33±0.00E+00(**50)	8.07E+03±2.82E+04(26)†	**2.36E-33±0.00E+00(**50)	5.71E+04±7.31E+04(8)†
f13	**1.76E-32±2.68E-32(**50)	1.85E+03±7.38E+03(12)	**8.36E-32±1.44E-31(**50)	1.65E+05±2.91E+05(0)

3.4.4.5 自适应参数控制的影响

如第二章中所述，差分演化算法对于控制参数 *CR* 和 *F* 的选取是很敏感的[8,9]。为此一些学者提出了不同的自适应差分演化算法以消除对参数 *CR* 和 *F* 的控制，如[8]、[45]、[53]等。为了显示 DE/BBO 算法同样可以改进自适应差分演化算法的性能，本节把文献[8]中提出的自适应参数控制技术加入到 DE/BBO 算法中实现对参数 *CR* 和 *F* 的自适应控制。表 3-8 列出了自适应差分演化算法(SADE)和自适应 DE/BBO 算法(SADE/BBO)的实验结果。

表 3-7 不同差分变异策略对 DE/BBO 和 DE 算法的影响

F	DE/best/1		DE/rand/2	
	DE/BBO	DE	DE/BBO	DE
f01	**3.71E-32±3.61E-32(**50)	1.25E+02±1.55E+02(0)†	**1.27E-17±6.15E-18(**50)	1.53E-10±8.92E-11(50)†
f02	**5.46E-16±1.81E-15(**50)	4.16E+00±3.02E+00(0)†	**4.70E-15±1.54E-15(**50)	2.02E-08±9.61E-09(2)†
f03	**8.15E-30±2.66E-29(**50)	2.62E+02±3.04E+02(0)†	1.31E+01±7.15E+00(0)	**7.28E-07±8.90E-07(**0)‡
f04	**2.36E+01±5.72E+00(**0)	2.97E+01±6.06E+00(0)†	**2.04E-09±7.53E-10(**50)	5.18E+00±3.41E+00(0)†
f05	**8.72E+00±1.43E+01(**8)	1.14E+04±1.43E+04(0)†	9.25E+00±1.30E+00(0)	**9.60E-02±5.63E-01(**0)‡
f06	**1.56E+01±2.98E+01(**0)	1.17E+03±5.42E+02(0)†	**[3.16E+04±8.95E+02](**50)	[4.74E+04±1.88E+03](50)†
f07	**4.07E-03±1.39E-03(**50)	9.53E-03±5.73E-03(43)†	**5.21E-03±1.26E-03(**50)	5.41E-03±1.46E-03(49)
f08	**6.58E+02±2.94E+02(**0)	4.84E+03±6.80E+02(0)†	**0.00E+00±0.00E+00(**50)	6.71E+03±2.95E+02(0)†
f09	**1.85E+01±7.55E+00(**0)	7.26E+01±1.51E+01(0)†	**7.60E-05±2.98E-04(**12)	1.16E+02±2.28E+01(0)†
f10	**2.48E+00±1.16E+00(**2)	9.73E+00±1.66E+00(0)†	**8.65E-10±2.00E-10(**50)	3.11E-06±1.04E-06(0)†
f11	**3.22E-02±4.06E-02(**12)	2.13E+00±1.33E+00(0)†	**0.00E+00±0.00E+00(**50)	6.41E-04±2.22E-03(46)†
f12	**6.70E-01±1.15E+00(**20)	2.77E+01±2.24E+01(0)†	**1.31E-18±1.09E-18(**50)	8.03E-12±1.04E-11(50)†
f13	**7.14E-01±1.24E+00(**1)	2.67E+04±1.28E+05(0)	**3.28E-16±3.02E-16(**50)	1.29E-04±9.08E-04(44)
F	DE/rand-to-best/1		DE/best/2	
	DE/BBO	DE	DE/BBO	DE
f01	**0.00E+00±0.00E+00(**50)	1.23E-34±6.10E-34(50)	**1.65E-32±1.37E-32(**50)	1.65E-31±1.93E-31(50)†
f02	**[3.42E+04±3.97E+02](**50)	[4.14E+04±3.72E+03](50)†	**3.11E-17±6.65E-17(**50)	7.88E-15±3.26E-14(50)
f03	3.71E-25±8.69E-25(50)	**5.63E-32±2.72E-32(**50)‡	3.28E-30±5.47E-30(50)	**1.07E-30±8.69E-31(**50)‡
f04	**2.63E+00±1.35E+00(**0)	2.95E+00±1.31E+00(0)	**2.31E-06±6.77E-06(**3)	3.64E-02±4.42E-02(0)†
f05	1.28E+01±3.24E+00(0)	**1.04E+00±1.77E+00(**37)‡	**1.10E+01±5.63E+00(**0)	1.27E+01±7.26E+00(0)
f06	**2.00E-02±1.41E-01(**49)	3.42E+00±4.13E+00(5)†	**2.20E-01±4.65E-01(**40)	1.34E+01±1.95E+01(2)†
f07	**8.43E-04±2.62E-04(**50)	2.17E-03±7.98E-04(50)†	**2.20E-03±7.68E-04(**50)	5.33E-03±2.81E-03(46†
f08	**4.97E+01±7.60E+01(**33)	3.15E+03±5.79E+02(0)†	**2.61E+01±5.50E+01(**40)	4.60E+03±5.21E+02(0)†
f09	**3.98E-02±1.97E-01(**48)	2.37E+01±7.13E+00(0)†	**1.11E+00±1.25E+00(**19)	5.71E+01±1.48E+01(0)†
f10	**6.13E-15±1.78E-15(**50)	1.69E+00±7.01E-01(0)†	**1.77E-01±4.19E-01(**42)	3.99E+00±1.14E+00(0)†
f11	**1.23E-03±3.24E-03(**43)	1.35E-02±1.52E-02(12)†	**6.74E-03±8.82E-03(**23)	2.60E-02±4.33E-02(16)†
f12	**2.07E-03±1.47E-02(**49)	7.06E-02±1.71E-01(34)†	**6.43E-02±1.63E-01(**40)	1.54E+00±2.35E+00(8)†
f13	**2.20E-04±1.55E-03(**49)	2.71E+00±1.05E+01(16)	**5.47E-03±1.93E-02(**41)	1.99E+01±2.41E+01(0)†

根据表 3-8 的结果可以看出: 1) 对于 Error 值, 在所有 50 次独立运行中 SADE/BBO 和 SADE 均能得到 5 个函数(f02, f06, f08, f09 和 f11)的全局最优解; SADE/BBO 算法在 5 个函数上明显优于 SADE 算法; 对两个函数(f03 和 f05), SADE 算法明显优于 SADE/BBO 算法。2) 对于 NFFEs 值, SADE/BBO 在 11 个函数上明显快于 DE 算法。对函数 f03 和 f05, SADE/BBO 不能求解这两个函数。总的来说, 集成所提出的混合杂交算子能够改进 SADE 算法的性能。

表 3-8 自适应参数控制对 DE/BBO 和 DE 算法的影响

F	Error		NFFEs		
	SADE/BBO	SADE	SADE/BBO	SADE	*AR*
f01	**0.00E+00±0.00E+00(**50)	1.75E-27±1.57E-27(50)†	**3.91E+04±8.15E+02**	6.11E+04±1.12E+03†	1.56
f02	0.00E+00±0.00E+00(50)	0.00E+00±0.00E+00(50)	**5.12E+04±8.17E+02**	8.45E+04±1.40E+03†	1.65
f03	2.10E-01±2.85E-01(0)	**4.03E-13±6.20E-13(**50)‡	NA	**3.57E+05±1.84E+04**	NA
f04	**4.11E-16±1.10E-15(**50)	3.44E-14±1.83E-13(50)	**2.25E+05±3.57E+04**	3.09E+05±4.54E+03†	1.38
f05	4.05E+01±2.30E+01(0)	**9.99E-02±1.23E-01(**1)‡	NA	**4.81E+05±0.00E+00**	NA
f06	0.00E+00±0.00E+00(50)	0.00E+00±0.00E+00(50)	**1.46E+04±4.91E+02**	2.30E+04±7.05E+02†	1.57
f07	**1.98E-03±4.35E-04(**50)	3.46E-03±9.00E-04(50)†	**6.33E+04±1.32E+04**	1.11E+05±2.23E+04†	1.75
f08	0.00E+00±0.00E+00(50)	0.00E+00±0.00E+00(50)	**4.87E+04±1.26E+03**	9.58E+04±2.27E+03†	1.97
f09	0.00E+00±0.00E+00(50)	0.00E+00±0.00E+00(50)	**6.45E+04±3.23E+03**	1.19E+05±4.08E+03†	1.85
f10	**4.14E-15±0.00E+00(**50)	1.54E-14±5.48E-15(50)†	**5.92E+04±8.21E+02**	9.31E+04±1.63E+03†	1.57
f11	0.00E+00±0.00E+00(50)	0.00E+00±0.00E+00(50)	**4.04E+04±7.94E+02**	6.47E+04±3.14E+03†	1.60
f12	**1.57E-32±0.00E+00(**50)	1.15E-28±1.15E-28(50)†	**3.56E+04±8.09E+02**	5.52E+04±1.28E+03†	1.55
f13	**1.35E-32±0.00E+00(**50)	3.92E-26±5.22E-26(50)†	**4.22E+04±8.62E+02**	6.71E+04±1.45E+03†	1.59

3.4.4.6 与其他混合差分演化算法比较

本章提出了一种基于 BBO 算法的混合差分演化算法(DE/BBO), 为了进一步研究算法的优越性, 本小节把 DE/BBO 算法与其他混合差分演化算法进行对比。由于目前存在很多不同的混合差分演化算法, 此处选取 DEahcSPX 算法[7]和 ODE 算法[40]与 DE/BBO 算法进行比较。DEahcSPX 利用一种基于杂交算子的自适应局部搜索算子来加快差分演化算法的收敛。通过实验表明 DEahcSXP 在收敛速度上优化传统差分演化算法。ODE 算法则采用基于逆向学习理论来对差分演化算法进行改进。本节把传统差分演化算法、DEahcSPX 和 ODE 算法与 DE/BBO 算法进行比较, 所有算法的参数设置如表 3-2 所示。对 DEahcSPX 参数 $n_p = 3$[7], 对 ODE 参数 $J_r = 0.3$[40]。实验结果如表 3-9 所示, 一些典型函数的收敛曲线如图 3-6 所示。

从表 3-9 可以得知, DE/BBO 在 8 个函数上明显优于 DEahcSPX; 但是, 对于函数 f03 和 f05, DEahcSPX 优于 DE/BBO。对剩下的 3 个函数, 两个算法之间没有明显的差异。然而, 对函数 f12 和 f13, DE/BBO 在所有 50 次运行中均能收敛于全局最优解附近, 而 DEahcSPX 则会陷入其局部最优解。此外, 图 3-6 表明对于大多数函数 DE/BBO 算法的收敛速度快于 DEahcSPX 算法。

与 ODE 算法相比较, t 测试的对比结果是 9/2/2, 这表明 DE/BBO 在 9 个函数上明显优于

ODE 算法。对两个函数(f03 和 f07), ODE 算法优于 DE/BBO。对函数 f01 和 f10, 两个算法之间没有明显区别, 但是, DE/BBO 稍微优于 ODE 算法。此外, 对于大部分函数 DE/BBO 呈现出较快的收敛速度。

表 3-9 DE, DEahcSPX, ODE 和 DE/BBO 之间实验结果比较

F	DE/BBO	DE	DEahcSPX	ODE
f01	**8.66E-28±5.21E-28(**50)	1.10E-19±1.34E-19(50)†	2.90E-20±2.28E-20(50)†	4.33E-25±1.86E-24(50)
f02	**0.00E+00±0.00E+00(**50)	1.66E-15±8.87E-16(50)†	4.47E-16±3.66E-16(50)†	2.81E-13±1.74E-13(50)†
f03	2.26E-03±1.58E-03(0)	8.19E-12±1.65E-11(50)‡	**5.11E-12±9.27E-12(**50)‡	2.50E-11±3.91E-11(50)‡
f04	**1.89E-15±8.85E-16(**50)	7.83E+00±3.78E+00(0)†	7.79E+00±3.18E+00(0)†	9.44E-02±2.33E-01(14)†
f05	1.90E+01±7.52E+00(0)	**8.41E-01±1.53E+00(**6)‡	1.24E+00±1.67E+00(5)‡	2.80E+01±9.24E+00(0)†
f06	**[2.16E+04±5.73E+02](**50)	[2.89E+04±2.01E+03](50)†	[2.81E+04±1.50E+03](50)†	[2.29E+04±1.81E+03](50)†
f07	3.44E-03±8.27E-04(50)	3.49E-03±9.60E-04(50)	3.52E-03±1.20E-03(50)	**1.03E-03±3.38E-04(**50)‡
f08	**0.00E+00±0.00E+00(**50)	4.28E+02±4.69E+02(1)†	4.98E+02±8.42E+02(5)†	1.63E+03±1.27E+03(1)†
f09	**0.00E+00±0.00E+00(**50)	1.14E+01±7.57E+00(0)†	1.30E+01±8.11E+00(0)†	1.65E+01±1.17E+01(0)†
f10	**1.07E-14±0.00E+00(**50)	6.73E-11±2.86E-11(50)†	3.89E-11±1.97E-11(50)†	5.34E-07±3.77E-06(49)
f11	**0.00E+00±0.00E+00(**50)	1.23E-03±3.16E-03(43)†	1.82E-03±5.09E-03(42)†	2.12E-03±4.66E-03(39)†
f12	**7.16E-29±6.30E-29(**50)	2.07E-03±1.47E-02(49)	6.22E-03±2.49E-02(47)	3.44E-18±1.95E-17(50)†
f13	**9.81E-27±7.10E-27(**50)	7.19E-02±5.09E-01(49)	3.22E-02±2.26E-01(46)	2.05E-22±1.44E-21(50)†

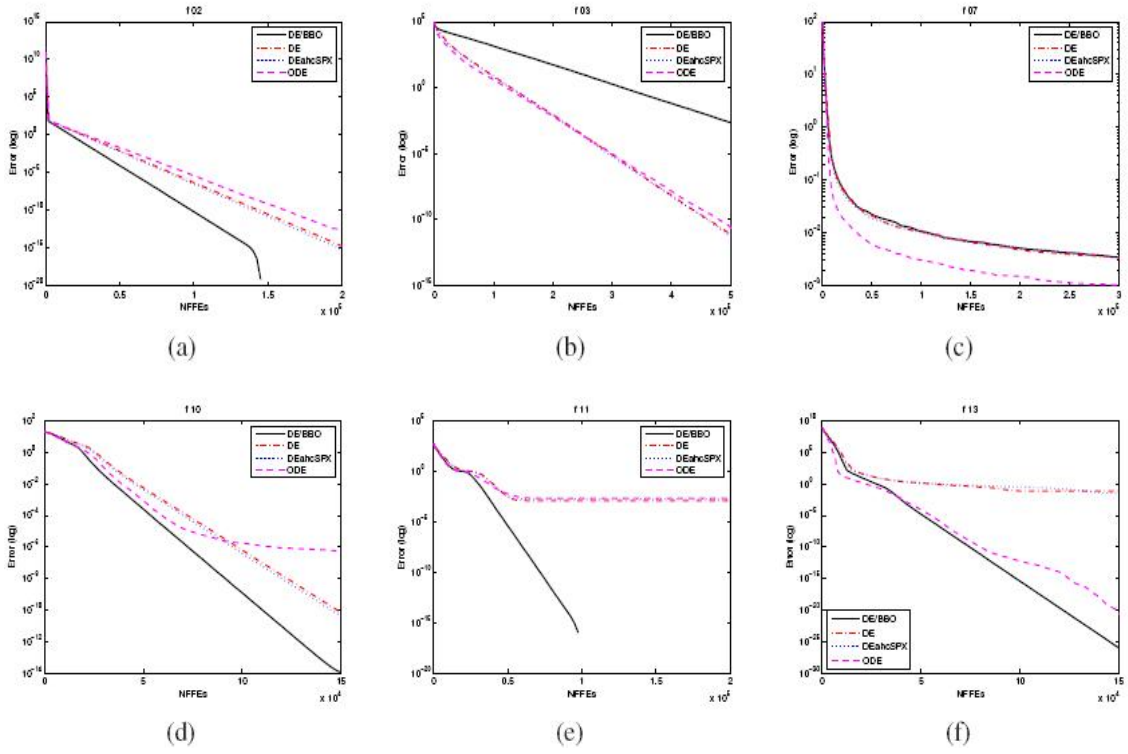

图 3-6 DE, DEahcSPX, ODE 和 DE/BBO 收敛曲线图。(a) f02. (b) f03. (c) f07. (d) f10. (e) f11. (f) f13.

§3.5 实验结论

差分演化算法是一种快速、简单、高效的全局优化算法，但是，该算法存在利用能力差的不足。BBO 算法是一种新颖的优化算法。BBO 算法由于其移动算子的有效设计具有对当前群体较好的利用能力。因此，本章把差分演化算法与 BBO 算法相结合，提出了一种混合杂交算子用于产生好的子个体。从实验结果可以得出如下结论:

1) DE/BBO 算法是一种有效和高效的优化算法，它能得到本章中所测试问题的全局最优解或接近全局最优解。
2) 与 BBO 算法和一些 DE 算法相比较, DE/BBO 算法总的性能优于或具有很高的可比性。
3) 通过把 DE/BBO 和 DE 算法在不同群体大小下进行实验，其结果表明对于大部分函数 DE/BBO 优于 DE 算法。
4) 可伸缩性实验表明 DE/BBO 能够在不同维数下加快 DE 算法的收敛速度，尤其是对于高维函数。
5) 对于不同的差分变异策略, DE/BBO 仍然可以得到比 DE 算法好的结果，表明了所提出混合杂交算子的普适性和可推广性。
6) 自适应参数控制技术可以增强 DE/BBO 和 DE 算法的性能。文章所提出的混合杂交算子显示了其对其他自适应差分演化算法性能增强的潜力。
7) 由于混合杂交算子的有效设计，DE/BBO 算法能平衡算法的开采能力和利用能力。此外，该混合杂交算子能使群体中好的个体分享更多信息给坏的个体，同时，可以避免好的个体在演化中被破坏。
8) 对于函数 f03，在 DE/rand/1 变异策略下 DE/BBO 比 DE 算法要差，但是，对 DE/best/1 策略，DE/BBO 优于 DE 算法。因此，采用如文献[52]中提出的策略自适应极值可能会进一步增强 DE/BBO 算法的性能。我们将在今后的研究中对此进行验证。

§3.6 本章小结

为了平衡差分演化算法的开采能力和利用能力，本章提出了一种混合差分演化算法，即 DE/BBO 算法。该算法基于一种新提出的混合杂交算子，算子把 BBO 算法的移动算子与差分变异算子相结合，一方面利用移动算子对群体信息进行有效利用，另一方面差分变异算子对搜索空间进行开采，探测新的区域，增强了算法的鲁棒性和寻找全局最优解的可能性。

为了验证算法的有效性，通过 23 个标准测试函数对 DE/BBO 进行测试，并与 BBO, DE, DEahcSPX 和 ODE 等算法进行了比较，实验结果验证了 DE/BBO 算法的优越性。此外，本章还对群体大小、问题维数、不同变异策略和自适应参数控制对 DE/BBO 和 DE 算法的影响进行了研究，实验结果进一步表明了 DE/BBO 算法在解的求解精度和收敛速度上优于传统 DE 算法。

第四章 多策略自适应差分演化算法

第三章中就差分演化算法存在利用能力差的缺点，提出了基于 BBO 算法的混合差分演化算法(DE/BBO)，算法中提出一种具有普适性的混合杂交算子。该混合算子把 BBO 算法的移动算子和差分变异算子相结合，从而有效平衡了算法的开采能力和利用能力。如第二章中所述，差分演化算法中存在多个差分演化策略(不同差分遗传算子与杂交算子的结合)，不同策略具有不同的性能，适合于求解不同的问题。但是，要根据所求问题选择最优的策略是很困难的。因此，为了进一步增强差分演化算法的性能，本章提出一种多策略自适应差分演化算法(SaJADE 算法)，把多个策略同时集成到算法中，通过设计有效的多策略自适应机制，使算法能自适应地根据所求问题选择最合适的策略，从而避免了对不同策略选择的不便，在增强算法性能的同时也便于工程使用。

本章第一节简述了与本章算法相关的几种改进的差分演化算法；第二节重点介绍本章多提出的 3 中多策略自适应机制；第三节则结合第二节中所提出的多策略自适应机制描述了 SaJADE 算法流程，并分析了该算法与已有算法的不同之处和算法的复杂度；第四节致力于通过实验对 SaJADE 算法进行测试及其对实验结果的分析；在分析了实验结果的基础上，第五节对实验结论进行阐述；最后一节，第六节小结了本章的工作。

§4.1 相关工作

在差分演化算法的研究文献中，目前已经提出了多种改进的算法，本节简述了其中 3 个算法：即 jDE 算法[8]，SaDE 算法[52]和 JADE 算法[53]。选择这 3 个算法的原因在于它们均具有较好的性能，且所提出的 SaJADE 算法将会与它们进行比较。

4.1.1 jDE 算法简介

jDE 算法[8]是一种参数自适应差分演化算法。该算法把控制参数 CR 和 F 直接编码到个体中，如群体中第 i 个个体 X_i 可以表示为：$X_i = \langle \mathbf{x}_i, CR_i, F_i \rangle$ [1]，其中 $\mathbf{x}_i = [x_{i,1}, \mathbf{L}, x_{i,j,\mathbf{L}}, x_{i,D}]^T$ 为第 i 个个体 X_i 的 D 维自变量向量，$i = 1, \mathbf{L}, NP$，$j = 1, \mathbf{L}, D$，NP 为群体大小。对于每个个体的控制参数，jDE 算法采用如下规则进行更新：

[1] 注：本章所求解的问题是第三章中公式(3.1)所涉及的无约束优化问题，但是，在符号描述上与第三章有所不同，本章将会一一说明.

$$CR_i = \begin{cases} \text{rndreal}_i[0,1], & \text{rndreal}[0,1] < t_1 \\ CR_i, & \text{otherwise} \end{cases} \tag{4.1}$$

$$F_i = \begin{cases} \text{rndreal}_i[0.1,1], & \text{rndreal}[0,1] < t_2 \\ F_i, & \text{otherwise} \end{cases} \tag{4.2}$$

其中，$\text{rndreal}[a,b]$ 是在 $[a,b]$ 区间里产生的均匀随机数；$t_1 = 0.1$ 和 $t_2 = 0.1$ 是调节参数 CR_i 和 F_i 的概率。每个个体的控制参数会在变异和杂交操作之间进行更新，所以，它们会影响随后的重组(变异与杂交)和选择操作。通过一些标准函数进行测试并与其他算法进行比较，作者验证了该方法的有效性[8]。

4.1.2 SaDE 算法简介

Qin 等[52]提出了 SaDE 算法，该算法是一种多策略自适应差分演化算法。算法中四个策略(DE/rand/1/bin, DE/rand-to-best/2/bin, DE/rand/2/bin 和 DE/current-to-rand/1)根据它们所产生有效子个体的经验进行自适应选择。此外，各策略具有不同的杂交概率，且也根据先前经验值自适应更新。

设 p_k 为第 $k = 1, \mathbf{L}, K$ 个策略的概率，K 为策略库中策略的个数。p_k 初始化为 $1/K$，根据 p_k 采用随机广义选择方法对每一个目标向量选择策略。经过 LP 代演化之后，p_k 按如下规则进行更新:

$$p_k = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \tag{4.3}$$

其中:

$$S_{k,G} = \frac{\sum_{g=G-LP}^{G-1} ns_{k,g}}{\sum_{g=G-LP}^{G-1} ns_{k,g} + \sum_{g=G-LP}^{G-1} nf_{k,g}} + e \tag{4.4}$$

其中，$G(G > LP)$ 是当前演化代数；$ns_{k,g}$ 和 $nf_{k,g}$ 分别是在过去 LP 代中第 k 个策略所产生子个体成功或失败进入下一代群体的数目；$S_{k,G}$ 第 k 个策略所产生子个体成功进入下一代群体的成功率；e 是一个很小的常数，用于避免 0 值的出现。

在 SaDE 中，每个个体的缩放因子 F_i 在每一代采用如下方法产生:

$$F_i = \text{rndn}_i(0.5, 0.3) \tag{4.5}$$

其中 $\text{rndn}_i(0.5,0.3)$ 是均值为 0.5，方差为 0.3 的正态分布随机数，对每个个体 i 独立产生.

对于每个个体的第 k 个策略其杂交概率 $CR_{i,k}$ 同样采用随机方法独立产生:

$$CR_{i,k} = \text{rndn}_i(\text{CRm}_k, 0.1) \tag{4.6}$$

当 $CR_{i,k}$ 的值不在[0,1]范围之内时，则采用截断的方法使其在[0,1]之间。CRm_k 初始化为 0.5。

设 CRMemory_k 为前 LP 代中第 k 个策略所产生子个体成功进入下一代时所保存的 CR 值，经过 LP 代演化之后，以后每一代 CRm_k 用 CRMemory_k 中所保存 CR 值的中间值代替。

4.1.3 JADE 算法简介

最近, Zhang 和 Sanderson 提出了一种新的自适应差分演化算法, JADE 算法[53,54]。在 JADE 算法中, 刚开始实现了两种新的变异策略[53], 即不采用归档的 DE/current-to-*p*best 和采用归档的 DE/current-to-*p*best。这两种策略可以描述为:

1) 无归档的 DE/current-to-*p*best:

$$\mathbf{v}_i = \mathbf{x}_i + F_i \times (\mathbf{x}_{best}^p - \mathbf{x}_i) + F_i \times (\mathbf{x}_{r2} - \mathbf{x}_{r3}) \tag{4.7}$$

2) 归档的 DE/current-to-*p*best:

$$\mathbf{v}_i = \mathbf{x}_i + F_i \times (\mathbf{x}_{best}^p - \mathbf{x}_i) + F_i \times (\mathbf{x}_{r2} - \tilde{\mathbf{x}}_{r3}) \tag{4.8}$$

在归档的 DE/current-to-*p*best 策略中, 采用一个数组 $\mathbf{A}$ 用来保存先前被好的子个体替换的父个体。其中, $\mathbf{x}_{best}^p$ 为当前群体 $100p\%$ 个最好解中的任意一个, $p \in (0,1]$。$\mathbf{x}_i, \mathbf{x}_{r2}$ 和 $\mathbf{x}_{best}^p$ 均为当前群体 $\mathbf{P}$ 中个体的自变量向量; $\tilde{\mathbf{x}}_{r3}$ 则为 $\mathbf{P} \cup \mathbf{A}$ 中随机选择的一个个体的自变量向量。随后, 为了求解更高维数的优化问题和进一步增加群体的多样性, 他们提出了两个新的变异策略[54], 即不采用归档的 DE/rand-to-*p*best 和采用归档的 DE/rand-to-*p*best。这两种策略可以描述为:

3) 无归档的 DE/rand-to-*p*bes:

$$\mathbf{v}_i = \mathbf{x}_{r1} + F_i \times (\mathbf{x}_{best}^p - \mathbf{x}_{r1}) + F_i \times (\mathbf{x}_{r2} - \mathbf{x}_{r3}) \tag{4.9}$$

4) 归档的 DE/rand-to-*p*bes:

$$\mathbf{v}_i = \mathbf{x}_{r1} + F_i \times (\mathbf{x}_{best}^p - \mathbf{x}_{r1}) + F_i \times (\mathbf{x}_{r2} - \tilde{\mathbf{x}}_{r3}) \tag{4.10}$$

其中 $\mathbf{x}_{r1}$ 和 $\mathbf{x}_{r2}$ 均为当前群体 $\mathbf{P}$ 中随机选取的两个个体的自变量向量.

在每一代, 对于每一个目标个体 X_i 其杂交概率按如下规则独立产生:

$$\begin{cases} CR_i = \mathrm{rndn}_i(m_{CR}, 0.1) \\ m_{CR} = (1-c) \times m_{CR} + c \times \mathrm{mean}_A(S_{CR}) \end{cases} \tag{4.11}$$

当 CR_i 的值不在[0,1]范围之内时, 则采用截断的方法使其在[0,1]之间。$\mathrm{mean}_A(\bullet)$ 是常用的求算数平均值运算; S_{CR} 是第 g 代所有成功产生子个体[2]的 CR_i 值的集合; $c \in [0,1]$ 是一个常数。

为了增前群体的多样性, 在 JADE 算法中, 对于每一个目标个体 X_i 其缩放因子按如下规则独立产生:

$$\begin{cases} F_i = \mathrm{rndc}_i(m_F, 0.1) \\ m_F = (1-c) \times m_F + c \times \mathrm{mean}_L(S_F) \end{cases} \tag{4.12}$$

当 $F_i > 1.0$ 时, 设置 $F_i = 1.0$; 当 $F_i < 0$ 是则重新产生。$\mathrm{rndc}_i(m_F, 0.1)$ 表示按照定位参数 m_F 和缩放参数 0.1 产生的 Cauchy 的随机数; S_F 是第 g 代所有成功产生子个体的 F_i 值的集合; $\mathrm{mean}_L(\bullet)$ 是求 Lehmer 平均值运算:

$$\mathrm{mean}_L(S_F) = \frac{\sum_{i=1}^{|S_F|} F_i^2}{\sum_{i=1}^{|S_F|} F_i} \tag{4.13}$$

[2] 注: 所谓成功产生子个体即通过差分演化算法的策略所产生的子个体优于其对应的父个体.

§4.2 多策略自适应机制

4.2.1 研究动机

如上所述，差分演化算法中存在多个策略，但是，针对所求解的问题从中选择最优的策略是很困难的[52,57]。到目前为止，对于所有问题没有任何一个单一的策略被证明是最优的，这也符合优化领域中的 No Free Lunch 理论[169]。

适应(adaptation)或者自适应(self-adaptation)是一种很有效的方法用于对控制参数和算子进行调节，特别是在不需要人工交互的情况下[170]。对于演化算法中所采用的参数和算子的调节技术的两篇好的综述文献可参考[171]和[172]。在演化算法中，除了对参数和算子进行自适应调节外，多方法的自适应控制也是很有效的。近来，多方法自适应技术也越来越受到关注。Ong 和 Keane 提出一种自适应 memetic 算法[173]，算法中一个局部搜索算法能够自适应地从局部搜索算法库中选取，并对个体进行改进。SaDE 算法[52,57]实现了差分演化算法的多策略自适应选取。Vrugt 等[174]提出了一种多方法自适应算法用于全局优化。算法中，多个搜索算法同时运行，并通过一种自适应学习机制自动控制不同算法在每一代中所产生子个体的数目。

在众多改进的差分演化算法中，采用多策略方法的改进算法仍然是很少的。文献[175]采用神经网络的方法来更新不同策略之间的权值。Zamuda 等[176]采用预先给定的概率参数来对各策略进行选择。SaDE 算法[52,57]利用先前各策略产生子个体的成功率来自适应选取各策略。基于以上考虑，本章提出新的简单的多策略自适应选择机制，并在此机制下实现了三种不同的方法。我们的方法不同于上面所描述的策略选择方法。

4.2.2 基本思想

在我们的方法中，用一个策略参数 $h \in [0,1)$ 来控制策略的选择。h 被直接编码到个体中，以实现其自适应控制。群体中第 i 个个体 X_i 描述如下:

$$X_i = \langle \mathbf{x}_i, h_i \rangle = \langle x_{i,1}, \mathbf{L}, x_{i,j}, \mathbf{L}, x_{i,D}, h_i \rangle \tag{4.14}$$

设策略库中共有 K 个不同的策略，对于第 i 个目标个体其策略($S_i = \{1,2,\mathbf{L},K\}$)采用以下方式计算:

$$S_i = \lfloor h_i \times K \rfloor + 1 \tag{4.15}$$

例如，如果 $K = 4$ 且 $h_i \in [0,0.25)$，则 $S_i = 1$。这表明如果 X_i 是当前目标个体，则策略库中的第一个策略将会被选取作为该个体产生子个体的策略。

此处描述了对于目标个体运用其策略参数从策略库中选择策略的基本思想，然而，为了实现多策略的自适应选择，必须要处理以下两个问题: 1) 如何从差分演化算法多个策略中选择合适的策略作为策略库？ 2) 如何设计有效的技术来更新各目标个体的策略参数 h_i？这两个问题将在下面章节中进行阐述。

4.2.3 策略库的选取

如上所述，在差分演化算法中存在多个不同的策略，不同策略具有不同的特性，适合求解不同的问题。然而，到目前为止还没有理论研究来证明选择哪些策略和如何设置策略库的大小来达到最优的优化性能[52]。本章中，为了选择不同的策略形成策略库，我们选择文献[53]和[54]中所提出的四个策略作为策略，即(1) 无归档的 DE/current-to-*p*best 策略; (2) 无归档的 DE/rand-to-*p*best 策略; (3) 归档的 DE/current-to-*p*best 策略; 以及(4) 归档的 DE/rand-to-*p*best 策略。这四个策略在 4.1.3 节中有介绍。选择这四个策略主要基于以下两点考虑: a) 它们已经被证明每个独立的策略均有较好的性能[53,54]; b) 一方面两个无归档的策略收敛较快，适合求解低维函数优化问题; 另一方面两个归档的策略可以提供更高的群体多样性，更适合求解高维函数优化问题。值得指出的是此处所选择的这 4 个策略只能看作是一种示例，其他策略同样也可以被选入到策略库中。

4.2.4 参数自适应机制

对于第二个问题，由于策略参数 h_i 是一个实数，因此很多技术有可以用来对其进行更新。本章实现了如下 3 中方法。

4.2.4.1 方法 1

受 JADE 算法中对控制参数 CR 和 F 的自适应技术的启发，在每一代中，对每个个体的策略参数 h_i 采用如下规则产生:

$$h_i = \begin{cases} \mathrm{rndn}_i(m_s, 1/6), & \text{if } g = 1 \\ \mathrm{rndn}_i(m_s, 0.1), & \text{otherwise} \end{cases} \tag{4.16}$$

其中，g 是当前演化代数。如果 $h_i \notin [0,1)$，则把它截断到 $[0,1)$ 之间。在第一代中($g = 1$)，标准方差设置为 1/6，这样可以保证初始时 h_i 能覆盖整个 $[0,1)$ 区间。当 $g > 1$ 后，标准方差设置为 0.1，这与文献[52]、[53]和[54]中的方法相似。

设 H_s 为第 g 代所有成功子个体的策略参数 h_i 的集合。均值 m_s 初始化为 0.5，在每一代结束后按如下方法更新:

$$m_s = (1-c) \times m_s + c \times \mathrm{mean}_A(H_s) \tag{4.17}$$

其中，$c \in [0,1]$ 是一个常数; $\mathrm{mean}_A(\bullet)$ 是常用的求算数平均值运算。

4.2.4.2 方法 2

方法 2 受文献[8]中 jDE 算法中所采用的参数自适应技术启发，对每个个体的策略参数 h_i 采用如下规则更新:

$$h_i = \begin{cases} \mathrm{rndreal}_i[0,1), & \text{if } \mathrm{rndreal}[0,1] < d \\ h_i, & \text{otherwise} \end{cases} \tag{4.18}$$

其中 $d \in [0,1]$ 是策略参数 h_i 的更新概率，本章实验中 $d = 0.1$。

4.2.4.3 方法 3

由于策略参数 h_i 是一个实数，因此，传统的差分变异算子也可以用来对其进行更新，此处，采用如下方法进行更新:

$$h_i = h_i + K_i \times (h_{best} - h_i) + K_i \times (h_{r1} - h_{r2}) \tag{4.19}$$

其中，h_{best} 为当前群体中最好个体的策略参数；$r1, r2 \in [1, NP]$ 且 $r1 \neq r2 \neq i$；$K_i = \text{rndreal}_i[0,1]$.如果 $h_i \notin [0,1)$，则把它截断到 $[0,1)$ 之间。

§4.3 SaJADE 算法

4.3.1 算法流程

为了使描述更清楚，此处把第一种策略自适应方法与差分演化算法相结合在图 4-1 中描述了 SaJADE 算法的流程图。图 4-1 中采用了 JADE 算法[53]中所提出的参数自适应方法，因此，从这点上看，SaJADE 算法是 JADE 算法的一个改进。图 4-1 用“<==”指明 SaJADE 算法与 JADE 算法的主要改进之处。需要指出的是文章中所提出的三种策略自适应方法均可应用到基本差分演化算法或其他改进的差分演化算法中。此外，在所提出的策略自适应方法中，差分演化算法中已有的不同参数自适应技术可以很容易应用到每个单一的策略中。如在图 4-1 中对于 DE/rand-to-*p*best 策略，其缩放因子 $F_i = \text{rndn}_i(m_F, 0.1)$；而对 DE/current-to-*p*best 策略，其缩放因子 $F_i = \text{rndc}_i(m_F, 0.1)$。其原因在于 DE/rand-to-*p*best 策略可以提供较高的群体多样性，因此，对于低维或中等维数的优化问题，采用较小的缩放因子可以加速算法的收敛速度。

4.3.2 与相近工作的区别

如上所述，文献[175]，[176]和[52]也在差分演化算法中采用了多策略对群体进行演化，本章所提出的方法与这些方法相比较主要区别在于:

1) 本章多提出的策略自适应方法通过一个策略参数 h 进行控制，该参数是一个实数，因此，很多演化算法中已有的参数调节技术可以用于更新此参数。
2) 与 SWAF 算法[175]相比，SWAF 算法采用神经网络的方法来更新各策略的权值，这种方法相对比较复杂，且在神经网络中本身存在许多参数需要微调。
3) 在文献[176]中，Zamuda 等对不同的策略预先给定固定的选择概率(selection probability)，因此，该方法不属于自适应方法，且作者没有给出如何对各个策略设置不同的选择概率.
4) 在 SaDE 算法[52]中，每个策略的选择概率通过公式(4.3)和(4.4)进行更新，该方法在实现上相对比较复杂。
5) 总的来说，本章所提出的三种方法不同于上述策略选择方法，且这些方法都很简单，其容易实现。

4.3.3 算法复杂度分析

相对于 JADE 算法而言，图 4-1 所示的 SaJADE 算法在总的算法复杂度上没有增加。SaJADE 算法所增加的额外复杂度是对策略参数 h 的自适应控制，该复杂度是 $O(NP)$。由于 JADE 算法总的复杂度是 $O(G \cdot NP \cdot (D + \log(NP)))$ [54]，其中 G 是最大演化代数，所以，SaJADE 算法总的复杂度也是 $O(G \cdot NP \cdot (D + \log(NP)))$。一般来说，群体大小 NP 的设置与问题维数 D 相关，因此，SaJADE 的总复杂度是 $O(G \cdot D^2)$，这与传统差分演化算法，JADE 算法等相同。

随机初始化群体$\mathbf{P}$, 并计算群体中每个个体适应值
设置$\mu_{CR}=0.5;\mu_F=0.5;\mu_s=0.5;\mathbf{A}=\phi;g=1;K=4$
while 如果终止条件没有满足 **do**
  $S_{CR}=\phi;S_F=\phi;H_s=\phi$ ⇐
  for $i=1$ to NP **do**
    根据公式(4.16)更新策略参数η_i ⇐
    计算策略索引$S_i=\lfloor \eta_i \times K \rfloor+1$ ⇐
    随机选择$\mathbf{x}_{best}^p$个体
    根据公式(4.11)产生CR_i
    if $S_i==2$ **or** $S_i==4$ **then**
      $F_i=\mathrm{rndn}_i(\mu_F,0.1)$ ⇐
    else
      $F_i=\mathrm{rndc}_i(\mu_F,0.1)$
    end if
    if $S_i==1$ **or** $S_i==2$ **then**
      从群体$\mathbf{P}$中随机选取r_1,r_2,r_3, 满足$r_1\neq r_2\neq r_3\neq i$
    else if $S_i==3$ **or** $S_i==4$ **then**
      从群体$\mathbf{P}$中随机选取$r_1,r_2,r_1\neq r_2\neq i$; 从群体$\mathbf{P}\cup\mathbf{A}$中随机选取$r_3$
    end if
    $j_{rand}=\mathrm{rndint}(1,D)$
    for $j=1$ to D **do**
      if $\mathrm{rndreal}_j[0,1)<CR$ **or** $j==j_{rand}$ **then**
        if $S_i==1$ **then**
          $u_{i,j}=x_{i,j}+F_i(x_{best,j}^p-x_{i,j})+F_i(x_{r_2,j}-x_{r_3,j})$
        else if $S_i==2$ **then**
          $u_{i,j}=x_{r_1,j}+F_i(x_{best,j}^p-x_{r_1,j})+F_i(x_{r_2,j}-x_{r_3,j})$
        else if $S_i==3$ **then**
          $u_{i,j}=x_{i,j}+F_i(x_{best,j}^p-x_{i,j})+F_i(x_{r_2,j}-\tilde{x}_{r_3,j})$
        else if $S_i==4$ **then**
          $u_{i,j}=x_{r_1,j}+F_i(x_{best,j}^p-x_{r_1,j})+F_i(x_{r_2,j}-\tilde{x}_{r_3,j})$
        end if
      else
        $u_{i,j}=x_{i,j}$
      end if
    end for
  end for
  for $i=1$ to NP **do**
    评价子个体$\mathbf{u}_i$
    if $f(\mathbf{u}_i)$优于$f(\mathbf{x}_i)$ **then**
      利用父个体$\mathbf{x}_i$更新数组$\mathbf{A}$
      $CR_i\rightarrow S_{CR};F_i\rightarrow S_F;\eta_i\rightarrow H_s$ ⇐
      子个体$\mathbf{u}_i$替换差的父个体$\mathbf{x}_i$
    end if
  end for
  更新μ_{CR},μ_F,和μ_s ⇐
  $g=g+1$
end while

图 4-1 SaJADE 算法基本流程

§4.4 实验结果及分析

4.4.1 测试函数集

为了验证所提出多策略自适应方法的性能，本节选取了 20 个可伸缩的函数作为测试函数集，其中函数 f01 – f13 选择文献[167]，函数 f14 – f16 选择文献[177]，余下四个函数(F06, F07, F09 和 F10)选择文献[168]。函数 f01 – f13 分别为附录一中的函数 f01 – f13；函数 f14, f15 和 f16 分别是附录一中的函数 f24, f25 和 f26(即 Neumaie 3, Solomon 和 Alpine)。对余下几个函数为 CEC2005 标准测试函数集，本论文中不再详细描述，感兴趣的读者请参考文献[168]。函数 f01 – f13 的基本特征在第三章中已叙述，余下 7 个测试函数均为多峰多极值函数。

4.4.2 实验参数设置

实验先对 4.2.4 节中所提出的三种策略自适应方法进行对比，因此，这里有 3 种基于 JADE 的算法: 1) 基于第一种方法的 SaJADE1; 2) 基于第二种方法的 SaJADE2; 和 3) 基于第三种方法的 SaJADE3。同时，我们把文献[52]中提出的策略自适应策略($LP = 50$)嵌入到 JADE 算法中，把此算法成为 SaJADE4。随后，把 SaJADE 算法(基于第一种策略自适应算法)与 jDE[8]，SaDE[52]，JADE-wo[53]和 JADE-w[53]进行直接实验比较。这里 JADE-wo 表示基于无归档策略的 JADE 算法，JADE-w 表示基于归档策略的 JADE 算法，两个 JADE 算法均采用 DE/current-to-*p*best 策略。对于所有实验，在无特殊说明下均采用如下参数设置[3]:

- 问题维数: $D = 30$ 和 $D = 100$；
- 群体大小: 当 $D = 30$ 时，$NP = 100$；当 $D = 100$ 时，$NP = 400$ [53,54];
- $m_{CR} = 0.5,\ m_F = 0.5,\ m_s = 0.5$ [53,54];
- $c = 0.1,\ p = 0.05$ [53,54];
- 求解精度(Value-to-reach, VTR): 对函数 f01 – f06, f08 – f16，$\mathrm{VTR} = 10^{-8}$；对函数 f07, F06, F07, F09 和 F10，$\mathrm{VTR} = 10^{-2}$ [168,53];
- 最大适应值评价次数(Max_NFFEs): 当 $D = 30$ 时，对函数 f01, f06, f10, f12 和 f13, Max_NFFEs = 150,000; 对函数 f03 – f05, Max_NFFEs = 500,000; 对函数 f02 和 f11, Max_NFFEs = 200,000; 对函数 f07 – f09, f14 – f16 和 F06 – F10, Max_NFFEs = 300,000。当 $D = 100$ 时，对于所有函数 Max_NFFEs =1,000,000 (即 $D \times 10,000$)[168]。

此外，为了对各算法性能进行对比，采用第三章中 3.4.3 节中所描述的评价标准。但是，本章中把算法成功运行次数(SR)改为算法运行成功率(S_r): 即成功运行次数除以总运行次数.

所有函数独立优化 50 次，且对于不同算法之间进行比较时，每次独立运行各算法采用相同的初始群体，以保证比较的公平性。

---

[3] 注: 对 jDE 和 SaDE 算法，一些特殊的参数(如 jDE 算法的 t_1 和 t_2，SaDE 算法的 LP)分别按照文献[8]和[52]进行设置.

4.4.3 不同多策略自适应方法比较

首先对不同多策略自适应方法进行比较，即 SaJADE1, SaJADE2, SaJADE3 和 SaJADE4 之间的性能比较。所有测试函数在 $D=30$ 时的实验结果如表 4-1 所示。最好解和第二好解分别用**黑体**和*斜体*进行强调。由于对大多数函数所有 4 个算法能成功求解，因此，表 4-1 中对其 NFFEs 值进行比较。除了函数 f15 和 F10，因为没有算法能成功求解这两个函数，表 4-1 中列出的是其 Error 值。

表 4-1 不同多策略自适应方法的实验对比

F	SaJADE1			SaJADE2			SaJADE3			SaJADE4		
	Mean	Std	*Sr*	Mean	Std	*Sr*	Mean	Std	*Sr*	Mean	Std	*Sr*
f01	**2.40E+04**	**5.55E+02**	1.00	2.88E+04	8.83E+02	1.00	*2.72E+04*	*1.46E+03*	1.00	2.82E+04	7.21E+02	1.00
f02	**3.90E+04**	**1.44E+03**	1.00	5.01E+04	2.46E+03	1.00	*4.66E+04*	*3.33E+03*	1.00	4.91E+04	2.07E+03	1.00
f03	**7.88E+04**	**3.63E+03**	1.00	8.79E+04	4.27E+03	1.00	*8.22E+04*	*8.21E+03*	1.00	8.97E+04	4.59E+03	1.00
f04	**2.09E+05**	**8.25E+03**	1.00	2.88E+05	6.38E+03	1.00	*2.85E+05*	*6.96E+03*	1.00	2.86E+05	6.95E+03	1.00
f05	**1.18E+05**	**3.61E+03**	1.00	1.26E+05	5.08E+03	0.98	*1.25E+05*	*7.94E+03*	0.96	1.26E+05	3.70E+03	0.98
f06	**9.20E+03**	**2.25E+02**	1.00	1.07E+04	3.94E+02	1.00	*1.03E+04*	*7.99E+02*	1.00	1.05E+04	3.26E+02	1.00
f07	**2.26E+04**	**4.59E+03**	1.00	2.61E+04	5.55E+03	1.00	*2.51E+04*	*6.63E+03*	1.00	2.71E+04	6.07E+03	1.00
f08	**1.01E+05**	**3.98E+03**	1.00	1.06E+05	2.11E+03	1.00	1.05E+05	2.50E+03	1.00	*1.04E+05*	*2.54E+03*	1.00
f09	**1.27E+05**	**4.16E+03**	1.00	1.32E+05	2.29E+03	1.00	*1.30E+05*	*2.19E+03*	1.00	1.31E+05	2.40E+03	1.00
f10	**3.61E+04**	**8.47E+02**	1.00	4.45E+04	1.21E+03	1.00	*4.29E+04*	*2.68E+03*	1.00	4.41E+04	1.51E+03	1.00
f11	**2.51E+04**	**7.64E+02**	1.00	3.07E+04	1.40E+03	1.00	*2.95E+04*	*1.16E+03*	1.00	3.03E+04	3.10E+03	1.00
f12	**2.17E+04**	**7.32E+02**	1.00	2.63E+04	1.24E+03	1.00	*2.51E+04*	*2.01E+03*	1.00	2.60E+04	1.18E+03	1.00
f13	**2.55E+04**	**1.07E+03**	1.00	3.32E+04	2.04E+03	1.00	*3.10E+04*	*3.32E+03*	1.00	3.19E+04	1.80E+03	1.00
f14	2.18E+05	2.05E+04	1.00	*2.15E+05*	*2.50E+04*	1.00	**2.06E+05**	**2.59E+04**	0.84	2.23E+05	2.74E+04	1.00
f15★	1.76E-01	4.28E-02	0.00	**1.60E-01**	**4.95E-02**	0.00	*1.74E-01*	*4.43E-02*	0.00	*1.74E-01*	*4.43E-02*	0.00
f16	**1.51E+05**	**7.01E+04**	0.72	2.89E+05	0.00E+00	0.02	*2.17E+05*	*8.46E+04*	0.14	2.34E+05	2.82E+04	0.06
F06	**1.04E+05**	**7.97E+03**	0.96	1.13E+05	8.37E+03	0.86	*1.07E+05*	*6.00E+03*	0.88	1.12E+05	8.56E+03	0.92
F07	**3.29E+04**	**4.02E+03**	0.80	3.57E+04	4.74E+03	0.74	*3.50E+04*	*6.88E+03*	0.70	3.57E+04	5.29E+03	0.80
F09	*1.04E+05*	*2.76E+03*	1.00	1.06E+05	2.42E+03	1.00	**1.04E+05**	**2.33E+03**	1.00	1.05E+05	2.29E+03	1.00
F10★	*2.66E+01*	*4.44E+00*	0.00	2.68E+01	5.20E+00	0.00	**2.43E+01**	**5.07E+00**	0.00	3.11E+01	5.60E+00	0.00
$\sum S_r$	17.48			16.60			16.52			16.76		

★表示所示结果为 Error 统计值，因为没有算法能成功求解该函数。

从表 4-1 可以看出，基于第一种策略自适应方法的 SaJADE1 算法总性能最好。对于 NFFEs 值，它在 16 个函数上排名第一，此外，SaJADE1 算法能提供最大的总成功率 ($\sum S_r = 17.48$)。SaJADE3 能在 NFFEs 值上取得第二好的结果，但是，该算法容易陷入局部最优解，其总成功率最小。SaJADE2 和 SaJADE3 两算法的结果相当。

表 4-2 所有测试函数在 $D=30$ 时最优解误差值统计结果

F	Max_ NFFEs	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f01	150,000	1.46E-28±1.78E-28†	*9.93E-62±5.34E-61*†	2.69E-56±1.41E-55†	3.42E-37±3.63E-37†	**1.10E-79±7.52E-79**
f02	200,000	9.02E-24±6.01E-24†	*5.53E-28±3.16E-27*†	3.18E-25±2.05E-24†	3.51E-25±2.74E-25†	**1.35E-47±7.53E-47**
f03	500,000	1.16E-13±1.73E-13†	1.93E-56±7.02E-56†	**6.11E-81±1.62E-80**‡	1.54E-14±4.56E-14†	*1.17E-77±3.39E-77*
f04	500,000	2.44E-14±1.65E-13†	1.34E-09±5.71E-10†	5.29E-14±2.05E-14†	**6.39E-27±8.27E-27**‡	*1.26E-19±1.35E-19*
f05	500,000	*1.04E-03±1.37E-03*†	4.78E-01±1.31E+00†	1.59E-01±7.89E-01	7.98E-02±5.64E-01†	**1.60E-30±6.32E-30**
f06	10,000	6.13E+02±1.72E+02†	*3.12E+00±1.54E+00*†	5.62E+00±1.87E+00†	5.07E+01±1.34E+01†	**0.00E+00±0.00E+00**
	150,000	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
f07	300,000	3.35E-03±8.68E-04†	6.59E-04±2.43E-04†	*6.14E-04±2.55E-04*†	2.06E-03±5.21E-04†	**4.10E-04±1.48E-04**
f08	100,000	**1.70E-10±1.71E-10**‡	4.14E-05±2.37E-05†	2.62E-04±3.59E-04†	*1.13E-08±1.08E-08*‡	6.83E-07±2.70E-06
	300,000	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
f09	100,000	**3.32E-04±6.39E-04**‡	*2.68E-03±1.90E-03*‡	1.33E-01±9.74E-02	2.43E+00±1.60E+00†	1.54E-01±2.25E-01
	300,000	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
f10	50,000	2.37E-04±7.10E-05†	*1.10E-09±7.45E-10*†	3.35E-09±2.84E-09†	3.81E-06±8.26E-07†	**1.12E-12±1.07E-12**
	150,000	8.26E-15±1.32E-15	4.14E-15±0.00E+00	4.14E-15±0.00E+00	4.14E-15±0.00E+00	4.14E-15±0.00E+00
f11	50,000	7.29E-06±1.05E-05†	*1.44E-14±7.05E-14*†	1.57E-08±1.09E-07†	2.52E-09±1.24E-08†	**0.00E+00±0.00E+00**
	200,000	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
f12	50,000	7.03E-08±5.74E-08†	*1.84E-17±4.52E-17*†	1.67E-15±1.02E-14†	8.25E-12±5.12E-12†	**2.10E-23±6.89E-23**
	150,000	5.99E-30±5.87E-30	1.57E-32±0.00E+00	1.57E-32±0.00E+00	1.57E-32±0.00E+00	1.57E-32±0.00E+00
f13	50,000	1.80E-05±1.42E-05†	*3.16E-13±9.79E-13*†	1.87E-10±1.09E-09†	1.93E-09±1.53E-09†	**3.83E-21±1.56E-20**
	150,000	1.80E-27±2.62E-27	1.35E-32±0.00E+00	1.35E-32±0.00E+00	1.35E-32±0.00E+00	1.35E-32±0.00E+00
f14	300,000	7.31E-01±1.19E+00†	1.72E-03±3.05E-03†	**1.68E-09±1.97E-09**‡	1.25E+02±2.68E+02†	*2.88E-09±2.43E-09*
f15	300,000	1.98E-01±1.41E-02†	2.02E-01±1.41E-02†	2.00E-01±1.63E-12†	**1.56E-01±5.01E-02**	*1.76E-01±4.28E-02*
f16	300,000	**6.08E-10±8.36E-10**	2.61E-06±1.21E-06†	2.78E-05±8.43E-06†	2.94E-06±3.47E-06†	*1.44E-07±4.92E-07*
F06	300,000	2.93E+01±2.79E+01†	7.00E+00±1.87E+01†	*2.56E+00±6.22E+00*	1.68E+01±2.60E+01†	**1.59E-01±7.89E-01**
F07	300,000	1.17E-02±9.90E-03	1.57E-02±1.13E-02†	**5.96E-03±7.39E-03**‡	1.54E-02±9.60E-03†	*1.04E-02±8.48E-03*
F09	100,000	**1.30E-05±3.17E-05**‡	*2.87E-03±1.82E-03*‡	1.35E+00±6.08E-01†	1.46E+00±1.02E+00†	1.13E-01±1.60E-01
	300,000	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
F10	300,000	5.54E+01±9.44E+00†	2.88E+01±5.33E+00†	*2.82E+01±5.32E+00*	7.57E+01±1.02E+01†	**2.66E+01±4.44E+00**
w/t/l		15/2/3	18/0/2	13/4/3	17/1/2	–

†表示在 $a=0.05$ 时 Wilcoxon 测试结果表明 SaJADE 明显优于所比较的算法。

‡表示相应算法明显优于 SaJADE 算法，下同。

由于 SaJADE1 算法优于基于其他三个策略自适应的 JADE 算法，因此，在后面的实验中将只对 SaJADE1 算法进行讨论，由于不会出现混淆，后面实验中 SaJADE1 算法简称为 SaJADE 算法。

表 4-3 所有测试函数在 $D=100$ 时最优解误差值统计结果

F	Max_ NFFEs	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f01	1,000,000	2.09E-20±9.27E-21†	5.13E-62±4.82E-62†	*1.21E-85±2.27E-85*†	1.09E-27±6.65E-28†	**1.62E-92±2.92E-92**
f02	1,000,000	1.82E-12±4.30E-13†	5.19E-36±7.12E-36†	*9.20E-42±2.96E-41*†	1.09E-15±2.10E-16†	**4.03E-51±1.41E-50**
f03	1,000,000	7.47E+03±7.43E+03†	6.85E-03±5.87E-03†	**4.79E-05±4.63E-05**	4.96E+00±1.61E+00†	*5.06E-05±5.74E-05*
f04	1,000,000	1.60E+00±1.34E-01†	1.62E-01±3.05E-02†	*3.09E-03±3.90E-03*	1.90E-01±1.80E-01†	**2.42E-03±3.73E-03**
f05	1,000,000	9.20E+01±1.41E+01†	4.96E+01±1.15E+01†	*2.77E+01±6.81E+00*†	8.49E+01±1.04E+01†	**2.45E+01±1.43E+00**
f06	40,000	3.18E+04±2.52E+03†	*1.14E+02±1.42E+01*†	1.25E+02±1.40E+01†	1.59E+03±1.70E+02†	**4.19E+01±5.27E+00**
	1,000,000	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00	0.00E+00±0.00E+00
f07	1,000,000	2.08E-02±2.88E-03†	2.04E-03±4.28E-04†	*1.60E-03±3.33E-04*†	6.85E-03±1.34E-03†	**8.59E-04±1.39E-04**
f08	1,000,000	**2.81E-08±2.30E-08**‡	3.94E+03±2.75E+02‡	9.11E+03±4.18E+02	*1.89E+01±3.51E+01*‡	9.04E+03±3.88E+02
f09	1,000,000	**6.01E+00±2.36E+00**‡	*1.03E+02±3.95E+00*‡	1.82E+02±8.44E+00†	1.05E+02±4.84E+00‡	1.67E+02±7.59E+00
f10	200,000	4.20E-01±5.34E-02†	7.79E-06±1.89E-06†	*4.05E-07±1.06E-07*†	5.98E-03±6.95E-04†	**7.28E-09±3.49E-09**
	1,000,000	1.73E-11±3.15E-12	7.69E-15±0.00E+00	7.84E-15±7.03E-16	1.05E-14±1.76E-15	7.69E-15±0.00E+00
f11	200,000	9.25E-01±5.88E-02†	8.87E-04±3.22E-03†	*3.54E-10±2.34E-09*†	3.84E-03±8.15E-03†	**2.39E-15±2.80E-15**
	1,000,000	0.00E+00±0.00E+00	8.87E-04±3.25E-03	0.00E+00±0.00E+00	2.96E-04±1.46E-03	0.00E+00±0.00E+00
f12	200,000	1.44E+00±3.11E-01†	2.38E-11±1.20E-11†	*4.62E-14±3.70E-13*†	8.96E-06±2.10E-06†	**1.66E-17±1.48E-17**
	1,000,000	4.47E-21±1.90E-21	4.71E-33±0.00E+00	4.71E-33±0.00E+00	6.75E-30±4.63E-30	4.71E-33±0.00E+00
f13	200,000	6.04E+01±1.10E+01†	2.83E-08±3.48E-08†	*1.13E-10±1.58E-10*†	7.81E-03±3.27E-03†	**7.24E-15±1.01E-14**
	1,000,000	1.91E-17±1.08E-17	1.35E-32±0.00E+00	1.35E-32±0.00E+00	5.56E-27±5.23E-27	1.35E-32±0.00E+00
f14	1,000,000	2.07E+05±4.46E+04†	1.51E+05±1.98E+04†	**7.90E+04±1.43E+04**	1.69E+05±1.39E+04†	*8.20E+04±1.47E+04*
f15	1,000,000	3.80E-01±3.93E-02†	3.28E-01±4.54E-02†	*2.98E-01±1.41E-02*†	3.60E-01±4.93E-02†	**2.66E-01±4.79E-02**
f16	1,000,000	4.78E-03±4.63E-04†	*1.12E-11±5.08E-11*†	9.55E-05±3.98E-04†	5.78E-03±1.71E-03†	**1.84E-23±1.30E-22**
F06	1,000,000	8.90E+01±4.16E-01†	1.26E+02±2.92E+01†	*3.60E+01±2.78E+01*	1.85E+02±4.26E+01†	**3.14E+01±2.83E+01**
F07	1,000,000	6.68E-01±1.33E-01†	1.90E-01±2.83E-01	*8.50E-02±4.50E-01*	2.81E-01±3.50E-01†	**7.29E-02±2.04E-01**
F09	1,000,000	**1.37E-01±1.26E-01**‡	1.13E+02±5.72E+00‡	2.00E+02±6.26E+00†	*1.07E+02±5.24E+00*‡	1.77E+02±7.07E+00
F10	1,000,000	4.76E+02±2.71E+01†	**4.22E+02±1.52E+01**‡	4.67E+02±2.93E+01†	5.31E+02±2.31E+01†	*4.51E+02±1.73E+01*
w/t/l		17/0/3	15/1/4	14/6/0	17/0/3	–

4.4.4 与其他差分演化算法比较

本节把 SaJADE 与 jDE, SaDE, JADE-wo 和 JADE-w 的性能进行实验比较，主要在以下三个方面进行对比: 1) 最终解的质量; 2) 算法收敛速度; 3) 成功率。对所有比较算法其参数设置如 4.4.2 节所示。所有测试函数在 $D=30$ 和 $D=100$ 时进行测试。此外，对 JADE-wo 和 JADE-w 算法使用 DE/rand-to-*p*best 策略也进行测试，这两个算法称为 rJADE-wo 和 rJADE-w。但是，由于纸张空间限制，我们没有列出实验结果，只在图 4-2 中绘出了它们在部分函数上的收敛曲线图。在 $a=0.05$ 时 Wilcoxon 测试被用来比较不同算法之间差异的明显性。Wilcoxon

测试是一种无参数统计测试。当数据不能满足正态分布时，Wilcoxon 测试可以视为 *t* 测试一种替换测试[178]。本章采用 Wilcoxon 测试基于以下两点考虑: 1) 尽管 *t* 测试在演化计算中经常被采用[167,56]，但是，最近的研究表明这种参数统计分析是不合适的，尤其对于处理多问题结果[179,180]。2) Wilcoxon 测试已被包含在很多著名的商业软件(如 SPSS, SAR, OriginPro, Matlab 等)中，很容易使用。

表 4-4 对函数在 $D=30$ 时, NFFEs 值的统计结果

F	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f01	5.89E+04±1.15E+03(1.00)	*2.90E+04±8.72E+02*(1.00)	3.03E+04±8.54E+02(1.00)	4.35E+04±6.06E+02(1.00)	**2.40E+04±5.55E+02**(1.00)
f02	8.12E+04±1.27E+03(1.00)	*5.08E+04±2.49E+03*(1.00)	5.48E+04±2.89E+03(1.00)	7.19E+04±9.36E+02(1.00)	**3.90E+04±1.44E+03**(1.00)
f03	3.56E+05±1.58E+04(1.00)	1.02E+05±4.60E+03(1.00)	**7.78E+04±3.88E+03**(1.00)	3.11E+05±2.09E+04(1.00)	*7.88E+04±3.63E+03*(1.00)
f04	2.93E+05±1.39E+04(1.00)	4.50E+05±1.06E+04(1.00)	3.08E+05±5.18E+03(1.00)	**1.68E+05±4.82E+03**(1.00)	*2.09E+05±8.25E+03*(1.00)
f05	4.66E+05±0.00E+00(0.02)	1.53E+05±5.50E+03(0.88)	*1.22E+05±5.43E+03*(0.96)	2.81E+05±1.10E+04(0.88)	**1.18E+05±3.61E+03**(1.00)
f06	2.22E+04±8.48E+02(1.00)	*1.09E+04±4.14E+02*(1.00)	1.15E+04±3.73E+02(1.00)	1.58E+04±4.66E+02(1.00)	**9.20E+03±2.25E+02**(1.00)
f07	1.06E+05±2.59E+04(1.00)	*2.79E+04±5.86E+03*(1.00)	2.99E+04±7.48E+03(1.00)	5.40E+04±1.15E+04(1.00)	**2.26E+04±4.59E+03**(1.00)
f08	**9.09E+04±2.05E+03**(1.00)	1.21E+05±1.91E+03(1.00)	1.17E+05±2.21E+03(1.00)	*9.94E+04±1.97E+03*(1.00)	1.01E+05±3.98E+03(1.00)
f09	**1.17E+05±3.84E+03**(1.00)	1.32E+05±2.06E+03(1.00)	1.43E+05±1.93E+03(1.00)	1.35E+05±3.22E+03(1.00)	*1.27E+05±4.16E+03*(1.00)
f10	8.95E+04±1.50E+03(1.00)	*4.54E+04±1.17E+03*(1.00)	4.72E+04±1.58E+03(1.00)	6.85E+04±8.06E+02(1.00)	**3.61E+04±8.47E+02**(1.00)
f11	6.20E+04±2.01E+03(1.00)	*3.20E+04±1.96E+03*(1.00)	3.44E+04±5.12E+03(1.00)	4.58E+04±1.47E+03(1.00)	**2.51E+04±7.64E+02**(1.00)
f12	5.34E+04±1.30E+03(1.00)	*2.74E+04±1.11E+03*(1.00)	2.91E+04±1.39E+03(1.00)	3.89E+04±7.64E+02(1.00)	**2.17E+04±7.32E+02**(1.00)
f13	6.43E+04±1.59E+03(1.00)	*3.51E+04±1.99E+03*(1.00)	3.76E+04±3.26E+03(1.00)	4.71E+04±1.10E+03(1.00)	**2.55E+04±1.07E+03**(1.00)
f14	NA±NA(0.00)	NA±NA(0.00)	**2.10E+05±2.41E+04**(1.00)	NA±NA(0.00)	*2.18E+05±2.05E+04*(1.00)
f15	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)
f16	*2.55E+05±1.79E+04*(1.00)	NA±NA(0.00)	NA±NA(0.00)	2.84E+05±0.00E+00(0.02)	**1.51E+05±7.01E+04**(0.72)
F06	2.53E+05±0.00E+00(0.02)	1.33E+05±1.11E+04(0.74)	*1.09E+05±6.15E+03*(0.84)	2.46E+05±1.49E+04(0.28)	**1.04E+05±7.97E+03**(0.96)
F07	1.01E+05±1.66E+04(0.66)	4.52E+04±5.76E+03(0.56)	*3.49E+04±2.14E+04*(0.80)	1.02E+05±1.44E+04(0.62)	**3.29E+04±4.02E+03**(0.80)
F09	**8.34E+04±3.69E+03**(1.00)	*9.60E+04±1.85E+03*(1.00)	1.15E+05±2.09E+03(1.00)	1.10E+05±2.76E+03(1.00)	1.04E+05±2.76E+03(1.00)
F10	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)
Sr	14.70	15.18	16.60	14.80	17.48

4.4.4.1 最终解质量分析

对于所有测试函数在 $D=30$ 和 $D=100$ 时不同算法在 50 次运行中最优解的统计值分别如表 4-2 和表 4-3 所示。与文献[53]中所采用方法类似，当多个算法对某个函数都能得到其全局最有解时表中还列出了其中间结果值。因此，对于这些函数，Wilcoxon 测试只针对中间结果值进行统计。在各表格的最后一行中 *w*/*t*/*l* 表示 SaJADE 算法与其他算法相比赢了 *w* 个函数，平 *t* 个函数，输了 *l* 个函数。

从表 4-2 可以看出, SaJADE 算法在解的最终质量上优于其他几个算法。SaJADE 在 15, 18, 13 和 17 个函数上分别明显优于 jDE, JADE-wo, JADE-w 和 SaDE 算法。对函数 f08, f09 和 F09, jDE 明显优于 SaJADE 算法。SaJADE 分别在 2, 3 和 2 个函数上劣于 JADE-wo, JADE-w 和

SaDE 算法。

表 4-6 对函数在 $D=100$ 时, NFFEs 值的统计结果

F	jDE	JADE-wo	JADE-w	SaDE	SaJADE
f01	5.38E+05±5.12E+03(1.00)	1.96E+05±3.46E+03(1.00)	*1.64E+05±2.61E+03(*1.00)	3.67E+05±4.84E+03(1.00)	**1.34E+05±2.82E+03(**1.00)
f02	7.45E+05±5.74E+03(1.00)	2.85E+05±3.80E+03(1.00)	*2.61E+05±3.97E+03(*1.00)	5.80E+05±5.09E+03(1.00)	**2.03E+05±2.21E+03(**1.00)
f06	2.04E+05±3.30E+03(1.00)	1.03E+05±8.58E+04(1.00)	*6.23E+04±4.68E+03(*1.00)	1.14E+05±3.14E+03(1.00)	**5.05E+04±8.58E+02(**1.00)
f07	NA±NA(0.00)	2.09E+05±3.09E+04(1.00)	*1.94E+05±2.66E+04(*1.00)	6.27E+05±1.03E+05(0.98)	**1.45E+05±2.04E+04(**1.00)
f10	7.82E+05±5.45E+03(1.00)	2.88E+05±4.04E+03(1.00)	*2.37E+05±2.81E+03(*1.00)	5.69E+05±6.21E+03(1.00)	**1.97E+05±3.77E+03(**1.00)
f11	5.30E+05±6.46E+03(1.00)	1.95E+05±6.33E+03(0.92)	*1.66E+05±6.39E+03(*1.00)	3.65E+05±1.60E+04(0.96)	**1.35E+05±2.55E+03(**1.00)
f12	5.11E+05±6.17E+03(1.00)	1.58E+05±3.65E+03(1.00)	*1.38E+05±2.48E+03(*1.00)	2.91E+05±3.64E+03(1.00)	**1.12E+05±1.94E+03(**1.00)
f13	6.53E+05±1.01E+04(1.00)	2.05E+05±1.56E+04(1.00)	*1.71E+05±4.11E+03(*1.00)	3.82E+05±6.64E+03(1.00)	**1.37E+05±3.77E+03(**1.00)
f16	NA±NA(0.00)	3.92E+05±1.55E+05(1.00)	**3.34E+05±4.89E+04(**0.86)	NA±NA(0.00)	*3.47E+05±1.03E+05(*1.00)
F07	NA±NA(0.00)	2.90E+05±1.69E+04(0.82)	**1.75E+05±4.40E+03(**0.80)	5.47E+05±2.43E+04(0.78)	*1.80E+05±4.08E+03(*0.88)
F08	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	9.50E+05±0.00E+00(0.02)
F09	9.98E+05±0.00E+00(0.02)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)	NA±NA(0.00)
Sr	7.02	9.74	9.66	8.72	9.90

表 4-5 对函数在 $D=30$ 时, *AR* 值(SaJADE 与其他算法相比)

F	jDE	JADE-wo	JADE-w	SaDE		F	jDE	JADE-wo	JADE-w	SaDE
f01	2.46	1.21	1.26	1.82		f10	2.48	1.26	1.31	1.90
f02	2.08	1.30	1.41	1.84		f11	2.47	1.28	1.37	1.83
f03	4.52	1.29	0.99	3.95		f12	2.46	1.27	1.34	1.79
f04	1.40	2.16	1.48	0.81		f13	2.52	1.38	1.48	1.85
f05	3.95	1.30	1.04	2.38		f14	NA	NA	0.96	NA
f06	2.42	1.18	1.25	1.72		f16	1.69	NA	NA	1.88
f07	4.66	1.23	1.32	2.38		F06	2.44	1.29	1.05	2.37
f08	0.90	1.20	1.17	0.99		F07	3.08	1.37	1.06	3.08
f09	0.92	1.04	1.13	1.07		F09	0.80	0.92	1.11	1.05
						Avg	2.1671	1.1544	1.0988	1.7247

表 4-7 对函数在 $D=100$ 时, *AR* 值(SaJADE 与其他算法相比)

F	jDE	JADE-wo	JADE-w	SaDE		F	jDE	JADE-wo	JADE-w	SaDE
f01	4.01	1.46	1.22	2.74		f12	4.57	1.41	1.23	2.60
f02	3.66	1.40	1.28	2.85		f13	4.76	1.49	1.25	2.78
f06	4.04	2.03	1.23	2.26		f16	NA	1.13	0.96	NA
f07	NA	1.44	1.34	4.32		F07	NA	1.61	0.97	3.04
f10	3.98	1.47	1.21	2.90		Avg	3.5714	1.3320	1.0610	2.6411
f11	3.93	1.44	1.23	2.71						

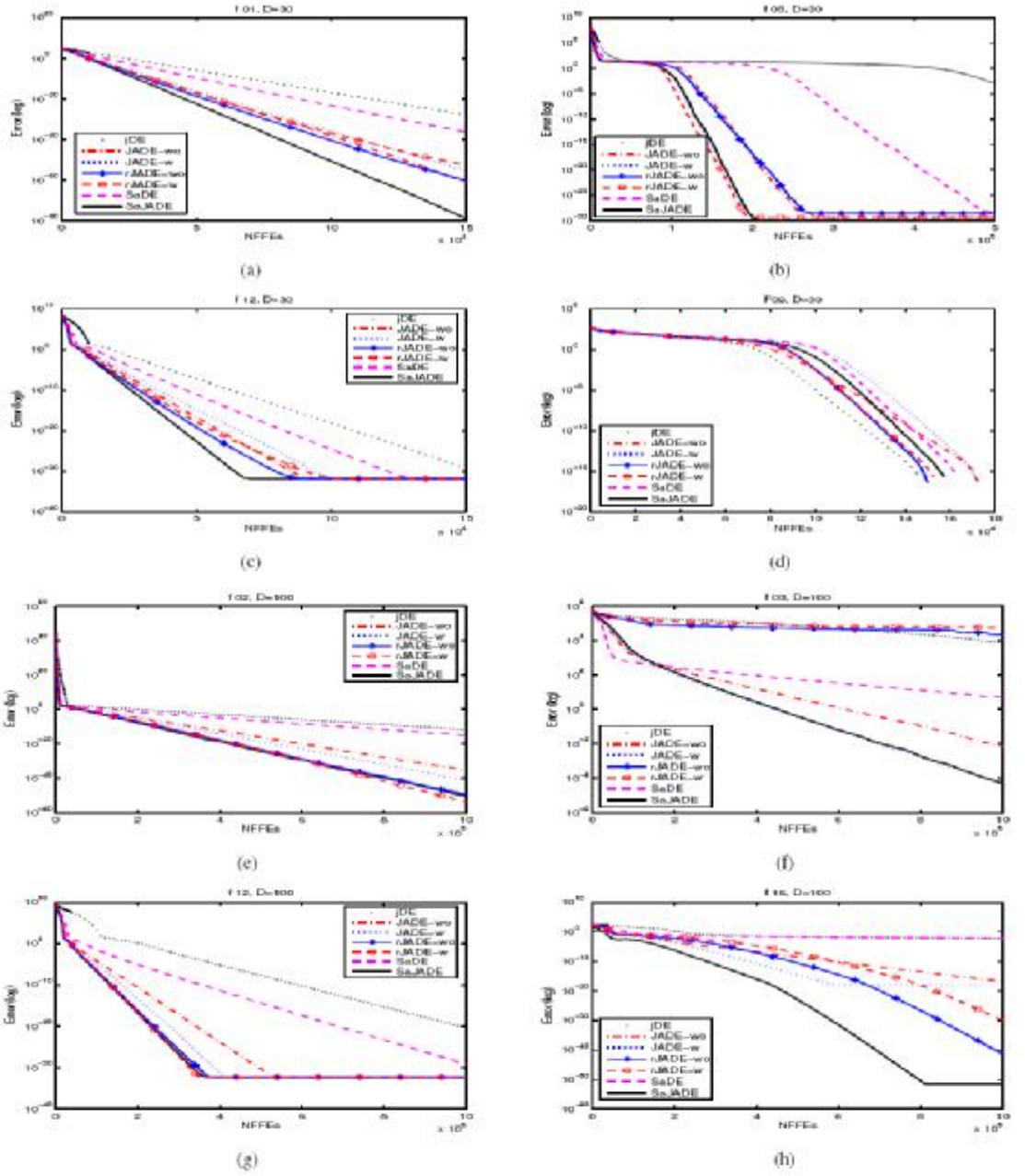

图 4-2 jDE, JADE-wo, JADE-w, rJADE-wo, rJADE-w, SaDE 和 SaJADE 算法的收敛曲线图。(a) f01 (*D*=30)。(b) f05 (*D*=30). (c) f12 (*D*=30). (d) F09 (*D*=30). (e) f02 (*D*=100). (f) f03 (*D*=100). (g) f12 (*D*=100). (h) f16 (*D*=100).

对于函数维数 $D=100$ 时，从表 4-3 中可以得出类似于表 4-2 的结论，即在绝大部分函数上 SaJADE 算法所得到的最终解质量明显优于其他算法。

总之，在 $D=30$ 和 $D=100$ 时, SaJADE 在所测试的函数上能够获得最优的最终解质量。所提出的策略自适应选择方法能在最终解的质量上明显增强 JADE 算法的性能。

4.4.4.2 收敛速度和成功率分析

除了解的最终质量外，算法的收敛速度和成功率同样是十分重要的评价标准。表 4-4 和 4-6 分别列出了算法在成功运行时在 $D=30$ 和 $D=100$ 下的 NFFEs 值的统计结果。此外，在表 4-5 和 4-7 中列出了多个算法能成功求解的函数的 AR 值。对算法 jDE，JADE-wo，JADE-w，rJADE-wo，rJADE-w，SaDE 和 SaJADE 的一些典型的收敛曲线如图 4-2 所示。

表 4-8 SaJADE 算法与其他算法已发表结果比较。对每个函数，第一行为平均最优解，第二行为标准差(在括号中)。*a* 表示相应结果被转化成误差值。

F	Max_NFFEs	SaJADE	JADE-wo	JADE-w	jDE	Adaptive LEP	Best Levy
f01	150,000	**1.10E-79**	*1.8E-60*	1.3E-54	1.1E-28	6.32E-04	6.59E-04
		(7.52E-79)	*(8.4E-60)*	(9.2E-54)	(1.0E-28)	(7.6E-05)	(6.4E-05)
f03	150,000	*2.95E-20*	2.8E-15	**8.5E-22**	0.090075	0.041850	30.628906
		(6.99E-20)	(8.2E-15)	**(3.6E-21)**	(0.080178)	(0.059696)	(22.113122)
f05	150,000	**2.48E-15**	3.2E-01	5.6E-01	*3.1E-15*	43.40	57.75
		(1.70E-14)	(1.1E+00)	(1.4E+00)	*(8.3E-15)*	(31.52)	(41.60)
f08	150,000	**0.00E+00**	4.7E+00	2.4E+00	**0.0E+00**[a]	1100.3[a]	670.6[a]
		(0.00E+00)	(2.3E+01)	(1.7E+01)	**(7.3E-12)**	(58.2)	(52.2)
f09	150,000	*8.87E-14*	1.4E-11	3.8E-11	**1.5E-15**	5.85	12.50
		(2.51E-13)	(1.0E-11)	(2.0-11)	**(4.8E-15)**	(2.07)	2.29
f10	150,000	**4.14E-15**	**4.4E-15**	**4.4E-15**	7.7E-15	1.9E-02	3.1E-02
		(0.00E+00)	**(0.0E+00)**	**(0.0E+00)**	(1.4E-15)	(1.0E-03)	(2.0E-03)
f11	150,000	**0.00E+00**	**0.0E+00**	2.0E-04	**0.0E+00**	2.4E-02	1.8E-02
		(0.00E+00)	**(0.0E+00)**	(1.4E-03)	**(0.0E+00)**	(2.8E-02)	(1.7E-02)
f12	150,000	**1.57E-32**	**1.6E-32**	**1.6E-32**	6.6E-30	6.0E-06	3.0E-05
		(0.00E+00)	**(5.5E-48)**	**(5.5E-48)**	(7.9E-30)	(1.0E-06)	(4.0E-06)
f13	150,000	**1.35E-32**	**1.3E-32**	**1.3E-32**	5.0E-29	9.8E-05	2.6E-04
		(0.00E+00)	**(1.1E-47)**	**(1.1E-47)**	(1.4E-15)	(1.2E-05)	(3.0E-05)

从表 4-4 和 4-6 可以看出，与其他差分演化算法相比，SaJADE 在大部分函数上只要较低的 NFFEs 值来达到各问题的求解精度(VTR)。SaJADE 也可以得到最高的总成功率，当 D=30 时 $\sum S_r = 17.48$；当 D=100 时，$\sum S_r = 9.90$。并且，从表 4-5 和 4-7 可以看出，SaJADE 算法与其他算法相比较收敛更快，特别是对于 D=100 时。例如：与 JADE-wo 算法相比，平均 AR 值是 1.3320，这表示 SaJADE 比 JADE-wo 算法在所成功求解的函数上要快 33.30%。在 D=100 时，与 JADE-w 相比，尽管平均 AR 值是 1.0610，但是，对于其中 10 个中的 8 个函数其 AR 值均大于 1.20，这表明 SaJADE 比 JADE-w 算法在这些函数上要快 20%以上。此外，从图 4-2 也可以看出，在大部分函数上 SaJADE 算法能提供更快的收敛速度。

4.4.5 与已发表结果比较

本节把 SaJADE 算法与一些演化算法文献中报道的结果进行简介比较。表 4-8 列出了各

算法的实验结果。该表中所比较算法的结果有：从文献[181]表 III 中自适应 LEP 和最优 Levy 算法的结果，文献[54]表 4.10 中 JADE-wo 和 JADE-w 算法的结果，文献[8]表 III 中 jDE 算法的结果。表 4-8 中体现了 SaJADE 算法在最终解质量上的优越性。SaJADE 在 9 个中的 7 个函数上取得了最好的结果。在剩余两个函数上 SaJADE 算法的结果排在第二位。

4.4.6 多策略自适应分析

在 SaJADE 算法中，多策略自适应方法被嵌入到 JADE 算法中以实现对不同求解问题来自适应地选择最好的策略对群体进行演化。为了对 SaJADE 算法自适应特征进行研究，本节把参数 m_s 的变化趋势在图 4-3 中展示出，图中的误差棒(Error Bars)是 50 次独立运行中参数 m_s 的均值和方差。它们可以清晰地显示出参数 m_s 的变化趋势。此外，差分演化算法的控制参数 m_{CR} 和 m_F 的变化趋势也在图中显示出来。

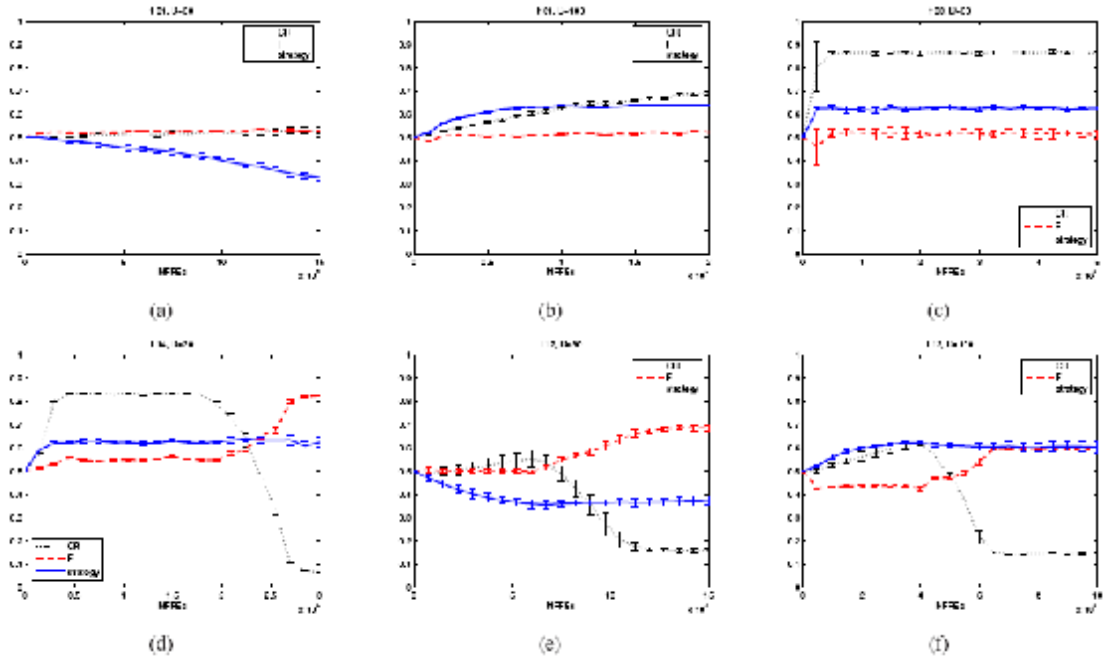

图 4-3 参数 m_s, m_{CR}, m_F 的自适应特征图示。(a) f01 (D=30). (b) f01 (D=100). (c) f03 (D=30). (d) f14 (D=30). (e) f12 (D=30). (f) f12 (D=100).

从表 4-2 和 4-3 以及图 4-2 中可以看出，参数 m_s 能随不同问题自适应地进行调节，这表明所提出的多策略自适应方法能根据不同问题自适应地选择最优的策略对问题进行优化。比如，对函数 f01 在 D=30 时，JADE-wo 优于 JADE-w，所以，参数 m_s 趋向于较小的值(如图 4-3(a)所示)；而当 D=100 时，JADE-w 优于 JADE-wo，因此，参数 m_s 倾向于较大的值(如图 4-3(b)所示)。对于其他函数也可以得出相似的结论。此外，图 4-3 中还表明了 SaJADE 算法保持了 JADE 算法对控制参数 m_{CR} 和 m_F 的自适应特征[53,54]。

4.4.7 算法简单性分析

Ong 和 Keane 指出算法的简单性也是一个很重要的指标[173]，因为复杂的算法对于实际

应用是很困难的。文献[173]中对算法简单性的定义是容易实现和尽量少的控制参数。本章 4.2.4 节所实现的三个多策略自适应方法都是很简单且容易实现的。它们没有增加基本差分演化算法的复杂性。与 JADE 算法[53,54]相比，第一种方法引入了参数 m_s，这个参数对不同问题是不敏感的 [4]。与文献[8]类似，在第二种方法中参数 d 是很容易按照 t_1 和 t_2 进行设置的。而在第三种方法中，则没有引入新的参数。

§4.5 实验结论

差分演化算法是一种有效、通用的基于群体的全局优化算法，该算法已经广泛应用到各个领域。在传统差分演化算法和一些改进算法中提出了多种策略，不同策略具有不同的特性。但是，对于不同问题选择最优的策略是很困难的。目前，只要较少的文献研究了多策略的差分演化算法。基于这些考虑，本章先提出了一种通过的策略产生方法，然后，基于此方法，实现了三种多策略自适应选择机制，并把它们嵌入到 JADE 算法中，通过 20 个标准测试函数对算法进行了实验验证。从实验结果可以得出如下结论:

1) 在 4 中不同的策略自适应选择方法中，与 JADE 算法结合的第一种方法具有最好的性能，因此，在本章其他实验中重点对其进行讨论。
2) 根据 4.4.6 节的实验分析可以得知，所提出的第一种策略自适应选择方法能够根据不同的问题自适应从策略库中选择最优的策略对其进行优化。此外，所提出的 3 种方法都是很简单其容易实现的。
3) 与其他算法相比，SaJADE 算法(JADE 算法与第一种方法的结合)优于其他算法。且 SaJADE 算法能在最终解的质量和收敛速度上增强 JADE 算法的性能。
4) 尽管 SaJADE 算法具有很好的性能，但是对一些函数，该算法会有时陷入其局部最优解。如对函数 f16 和 F06。其原因可能是群体多样性的丢失。可能的改进方法有群体重新启动技术[182,183]。但是，这已经超出本章的研究范围，我们将在今后的工作中对其进行研究。

§4.6 本章小结

本章针对传统差分演化算法中存在策略选择困难的不足，提出一种通用的策略产生方法，该方法通过一个策略参数 h 来控制目标个体策略的选择。$h \in [0,1)$ 是一个实数，因此，目前演化算法中多种参数自适应技术可以用于对该参数进行更新，从而实现多策略自适应选择。在此方法基础上实现了 3 种多策略自适应选择方法，这 3 种方法都很简单和容易实现。通过把这些方法与 JADE 算法相结合，利用 20 个标准测试函数对算法进行测试，并与其他算法进行比较，验证了 SaJADE 算法的优越性能。此外，实验结果进一步表明了 SaJADE 算法

[4] 注: 本章对不同 m_s 的初始值进行了实验，但是，由于空间限制，不再给出相关结果.

对策略选取具有自适应的特征，且该算法增强了 JADE 算法的性能。

对 SaJADE 算法可能的改进方向有采用群体重新启动技术来避免算法的局部收敛。另外，由于所提出的策略参数 h 是一个实数，其他参数自适应技术也可以对其进行更新，因此，寻求更有效的参数自适应技术以进一步提高 SaJADE 算法的性能也是一个可行的研究。

第五章 基于 ε 占优的快速正交多目标差分演化算法

第三章和第四章主要针对传统差分演化算法所存在的一些不足进行改进，并把改进算法应用到无约束函数的优化中。在现实生活中存在多种复杂的优化问题，多目标优化问题(Multi-objective optimization problems, MOPs)就是其中一类。近年来，采用演化算法求解 MOPs 已经成为一个重要的研究方向[58]。差分演化算法是演化算法的一种，如前几章所述，该算法在很多领域取得了广泛的应用。把差分演化算法应用到多目标优化问题中，也是该算法的一个研究热点。

本章把 ε 占优技术和正交实验设计与差分演化算法相结合，提出一种快速的多目标差分演化算法。算法利用正交实验设计产生初始群体，从而可以使初始个体均匀分布在决策变量空间，有利于使好的个体在演化中得到利用。利用 Archive 群体保存非劣解集，并使用 ε 占优技术对该群体进行更新，使得非劣解能均匀分布。此外，为了加速算法的收敛，提出一种混合选择机制来对差分变异策略中的基向量进行选择。通过标准多目标测试函数对算法进行测试，验证了新算法的有效性。本章 5.1 节简述一些相关工作 5.2 节重点详细介绍了新算法(ε -ODEMO 算法)的一些改进技术。实验部分安排在 5.3 节中，并对实验结果进行了分析讨论。最后一节, 5.4 节小结了本章的工作。

§5.1 相关工作

5.1.1 问题描述

不失一般性，一个 MOP 问题包含 n 个自变量(决策变量)，k 个目标函数和 m 个约束函数。该问题可以描述为:

$$\begin{aligned} &\text{minimize:} && \mathbf{y}=\mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),\mathbf{L},f_k(\mathbf{x})) \\ &\text{subject to:} && \mathbf{e}(\mathbf{x})=(e_1(\mathbf{x}),\mathbf{L},e_m(\mathbf{x}))\ge 0 \\ &\text{where:} && \mathbf{x}=[x_1,\mathbf{L},x_n]^T\in\mathbf{X} \\ & && \mathbf{y}=[y_1,\mathbf{L},y_n]^T\in\mathbf{Y} \\ & && l_i\le x_i\le u_i \end{aligned} \tag{5.1}$$

其中，$\mathbf{x}$ 是决策向量，$\mathbf{y}$ 是目标函数向量，$\mathbf{X}$ 表示决策空间，$\mathbf{Y}$ 表示目标空间。约束函数 $\mathbf{e}(\mathbf{x})\ge 0$ 定义了可行解空间，当约束函数个数 $m=0$ 时，(5.1)所描述的问题是无约束多目标优化问题。

定义 5.1 Pareto 占优(Pareto Dominance): 一个向量 $\mathbf{y}=[y_1,\mathbf{L},y_k]^T$ 被另一个向量 $\mathbf{x}=[x_1,\mathbf{L},x_k]^T$ Pareto 占优, 即 $\mathbf{x}\ \mathbf{p}\ \mathbf{y}$, 当仅当满足下列条件:

$$\forall i=1,\mathbf{L},k:\ x_i\le y_i \quad \text{且} \quad \exists j=1,\mathbf{L},k:\ x_i<y_i \tag{5.2}$$

定义 5.2 Pareto 最优(Pareto Optimality): 一个解 $\mathbf{x}\in\mathbf{X}$ 是 Pareto 最优当且仅当满足下列条件:

$$\neg\exists\mathbf{y}\in\mathbf{X}:\ \mathbf{v}\ \mathbf{p}\ \mathbf{u} \quad \text{其中:}\ \ \mathbf{u}=\mathbf{f}(\mathbf{x}),\quad \mathbf{v}=\mathbf{f}(\mathbf{y}) \tag{5.3}$$

定义 5.3 ε 占优(ε -dominance): 对于两个目标向量 $\mathbf{f},\mathbf{g}\in\ ¡^k$, 设 $e>0$, 如果 $\mathbf{f}$ ε 占优 $\mathbf{g}$, 即 $\mathbf{f}\ \mathbf{p}_e\ \mathbf{g}$,当仅当满足下列条件:

$$\forall i=1,\mathbf{L},k:\ \ (1-e)\cdot f_i\le g_i \tag{5.4}$$

定义 5.4 Pareto 最优解集(Pareto-optimal Set): Pareto 最优解集(*POS*)定义为所有 Pareto 最优决策向量集合, 即 $POS=\{\mathbf{x}\in\mathbf{X}\mid\neg\exists\mathbf{y}\in\mathbf{X},\ \mathbf{f}(\mathbf{y})\ \mathbf{p}\ \mathbf{f}(\mathbf{x})\}$。

定义 5.5 Pareto 最优前沿(Pareto-optimal Front): Pareto 最优前沿(*POF*)定义为所有 *POS* 中决策向量所对应的目标函数值, 即 $POF=\{\mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),\mathbf{L},f_k(\mathbf{x}))\mid\mathbf{x}\in POS\}$。

5.1.2 多目标演化算法

在求解 MOP 问题时, 与传统的多目标算法不同, 多目标演化算法(Multi-objective Evolutionary Algorithms, MOEAs)能够通过一次运行得到一组 Pareto 解集[61](或称为非劣解集), 这就有利于决策者根据不同问题的需要从多个解中选择最合适的一个解作为最终解。对于 MOEA 算法, 为了使决策者能很容易从所得到的 Pareto 解集中选择合适的解, 所得到的 Pareto 解集需要满足下面两个特点: 1) 由 Pareto 解集所得到的 Pareto 前沿应尽可能逼近真的 Pareto 最优前沿; 2) 所得到的非劣解集应尽可能均匀分布。

在近三十年里, 很多学者提出了多种 MOEA 算法用于求解 MOP 问题, 如国外一些学者提出的 NSGA-II[61], SPEA[63], PAES[184], DEMO[69]等。在国内, 很多学者在 MOEA 算法上也有大量的研究, 如曾三友等[185]等提出了基于正交设计的小生境 MOEA 算法。崔逊学等[186]提出了利用偏好信息选择个体的多目标遗传算法。石川等[187]提出基于 Pareto 占优树来分配适应值的 MOEA 算法。张利彪等[71]提出了一种多目标差分演化算法, 算法采用极大极小距离密度来保证 Pareto 解集的分布。龚文引等[75]提出了基于 ε 占优技术和正交实验设计的多目标差分演化算法。

差分演化算法也被用于求解 MOP 问题, 第二章 2.3.1 节对当前多目标差分演化算法进行了简述, 此处就不再赘述。

5.1.3 正交设计在 EA 中的应用

正交实验设计[76]是一种在多因素多水平实验中通过较少实验次数来找出最优或者较好实验条件的方法。该算法通过设计合适的正交表, 并按照正交表安排实验, 最后, 通过分析实验结果来找出较好的实验方案。通过正交实验设计能得到在不同因素和不同水平所包含的所有实验方案中较少的均匀的实验方案, 从而可以有效节约实验开支。

近年来, 一些学者把正交设计应用到演化算法中, 通过正交设计来初始化群体或设计遗传算子, 以增强演化算法的性能。Leung 和 Wang 提出了一种基于正交设计的遗传算法[188], 该

算法先通过量化方法把连续自变量空间离散化，然后，利用正交实验设计的方法来产生初始群体，使初始群体中的个体均匀分布在决策空间。此外，在算法中还通过正交实验设计提出了正交杂交算子，以增强算法的有效性。Gong 等[36]提出了基于正交实验设计的二水平正交杂交算子，并利用统计信息选择最优子个体。该算子与差分演化算法相结合，增强了基本差分演化算法的性能。曾三友等[185]等利用正交实验设计来划分小生境，算法用于求解多目标优化问题。

5.1.4 基于 ε 占优方法的 MOEAs

为了使算法在多目标优化中快速收敛，一种松弛的 Pareto 占优技术：ε 占优技术(定义 5.3)被 Laumanns 等提出[189]。ε 占优技术采用一种归档策略(archiving strategy)来保证演化多目标优化所要达到的两个标准：即收敛于真正的 Pareto 前沿和非劣解集均匀分布。ε 占优技术通过 Archive 来保存算法所得到的 Pareto 解，并采用 ε 占优技术对 Archive 不断更新。该技术在最近几年得到了很多关注。Deb 等[190,74]把 ε 占优技术应用于一种稳态演化算法中，提出了 ε-MOEA；同时，算法中提出了有效的父个体和 Archive 更新策略。实验结果表明 ε-MOEA 在逼近 Pareto 最优前沿，解的多样性和计算时间上均有较好的性能。同时，作者指出了 ε 占优技术的两个优点：1) 它可以减少 Pareto 解集中解的个数；2) 它可以保证解分布的多样性。Santanan-Quintero 和 Coello[73]以及 Cai 等[191]把 ε 占优技术与差分演化算法相结合，提出了多目标差分演化算法。

§5.2 ε-ODEMO 算法

为了扩展差分演化算法的功能，增强多目标差分演化算法的优化性能，提出一种基于 ε 占优技术和正交实验设计的快速差分演化算法：ε-ODEMO 算法，其研究动机主要基于以下几点考虑：

1) 目前，所存在的大部分 MOEA 算法都采用的是随机群体初始化，这样很难是初始群体中的个体均匀分布在决策空间中，也很难产生好的初始个体，从而降低了算法对初始群体的利用能力；
2) 文献[185]中采用正交实验设计来产生小生境群体，但是，这种方法增加了算法的复杂度，很难有效应用于大规模优化问题中；
3) 绝大部分多目标差分演化算法都只采用传统的 Pareto 占优技术来更新群体，而没有采用 ε 占优技术；
4) 文献[23]中采用了 ε 占优技术和 Archive 群体来保存非劣解集，但是，由于所有子个体的产生都利用 Archive 群体中的个体，从而会导致因为过多利用 Archive 群体的个体信息而误导演化，特别是当 Archive 群体中包含很多无效个体时。

本章中所提出的 ε-ODEMO 算法是对我们以前工作[191]的一个扩展，但是，与文献[191]中算法相比较，本章中的算法设计了更有效的混合选择机制来产生子个体，同时，进行了扩展实验，验证了杂交概率 CR 和选择控制参数 l 对算法性能的影响。

5.2.1 正交初始群体

传统的演化算法一般都采用的是随机群体初始化，这样很难是初始群体中的个体均匀分布在决策空间中，也很难产生好的初始个体，从而降低了算法对初始群体的利用能力。采用正交实验设计方法产生初始群体，可以使初始群体中的个体均匀地分布于整个决策空间，而且可以产生好的个体，这些个体可以增强算法在随后演化过程中的利用能力，加速算法的收敛[188]。为了增强算法的有效性和加快算法收敛速度，采用正交实验设计的方法产生初始 Archive 群体和演化群体。

5.2.1.1 产生正交数组

在产生正交初始群体之前，算法先根据不同的问题产生不同的正交矩阵(Orthogonal Array，产生正交矩阵的详细算法请参考文献[188])，在本章的研究中用 $L_R(Q^C)$表示具有 Q 个不同水平的正交矩阵，这里 Q 为奇数，$R=Q^J$ 表示正交矩阵的行数，正交指数 J 是一个正整数，满足以下条件:

$$C=\frac{(Q^J-1)}{(Q-1)} \tag{5.5}$$

C 表示正交矩阵的列数，为了使所产生的正交矩阵能满足正交初始化群体的需要，C 必须大于或等于优化问题自变量个数。因此，要设计一个合适的正交矩阵则需满足(一般而言，为了减少产生初始群体所需的评价次数，取指数 J=2, Q 取满足下面条件的最小值即可):

$$R=Q^J \geq NP \qquad \text{s.t.: } C \geq n \tag{5.6}$$

其中，n 表示所要优化问题的自变量个数，NP 表示演化群体的大小。

5.2.1.2 搜索空间量化

为了能把正交实验设计的方法应用到连续决策变量空间中，需要对连续变量空间进行量化(quantization)。设所优化问题某一自变量 x_k，$k=1,\mathbf{L},n$ 的范围是 $[l_k,u_k]$，将其量化为 Q 个水平 $a_1,a_2,\mathbf{L},a_Q$，其中 Q 为奇数，a_i 由如下公式计算:

$$a_i=l_k+(i-1)\frac{u_k-l_k}{Q-1},\quad i=1,\mathbf{L},Q \tag{5.7}$$

这样 $[l_k,u_k]$ 连续空间就被量化成 $Q-1$ 个相等的部分。

5.2.1.3 初始群体的产生

根据所优化的问题按公式(5.6)产生了合适的正交矩阵和对搜索空间量化之后，则可根据正交矩阵产生正交群体 OP(具体方法请参考文献[188])，由公式(5.6)可知，此时 OP 的个体数目大于演化群体的种群数 NP，为此，在本章研究中，先通过传统 Pareto 占优的方法从 OP 中选择非劣个体，形成初始 Archive 群体，设此时 Archive 群体大小为 ar_size。产生完初始 Archive 群体后，再产生初始演化群体。如果 $ar\_size \geq NP$，则从 Archive 群体中随机选择 NP 个个体作为初始演化群体；否则，除把 Archive 群体中的所有个体加入到初始演化群体外，剩下 $NP-ar\_size$ 个个体从正交群体中随机选取。与文献[74]的主要区别是: 1) 将正交群体初始化应用于多目标演化算法中，根据 Pareto 占优技术从 OP 中产生初始 Archive 群体; 2) 由初始 Archive 群体和 OP 共同产生初始演化群体。值得指出的是对于指定 Q 和 J，算法每次产生的

正交矩阵是固定的，因此可以采用离线方式产生正交矩阵，从而避免了文献[185]采用正交实验设计产生小生境而增加算法的复杂度。

5.2.2 混合选择机制

在基本差分演化算法中，采用如下差分变异策略对产生变异向量:

$$V_i = X_{r1} + F \times (X_{r2} - X_{r3}) \tag{5.8}$$

其中，X_{r1} 称为基向量(base vector)，$r1 \neq r2 \neq r3 \neq i$ 是从群体中随机选择的三个父个体的索引。为了加快算法收敛速度和利用 Archive 群体中的个体指导演化，在 ε-ODEMO 中采用基于随机选择和精英选择的混合选择机制:

$$\text{selection} = \begin{cases} \text{random} & eval < (l \times Max\_eval) \\ \text{elitist} & \text{otherwise} \end{cases} \tag{5.9}$$

其中，*eval* 是当前适应值评价次数，*Max_eval* 是最大评价次数，$l \in [0.1, 1.0]$ 为选择机制控制参数。在随机选择中，三个父个体从演化群体中随机选取，而在精英选择中，基向量 $\boldsymbol{X}_{r1}$ 从 Archive 群体中随机选取，其他两个个体则从演化群体中随机选取。采用随机选择和精英选择的原因是: 在演化初期，Archive 群体中的个体大部分不是最终的非劣解，对演化的指导作用小; 随着演化的进行，Archive 群体中的好个体数目增加，利用其中的个体作为基向量，可以加快算法收敛到全局最优解的速度。与文献[73]中方法不同之处是该方法中精英选择只把 Archive 中个体作为基向量; 而[73]中所有父个体均从 Archive 中选取，当 Archive 中包含很多无效个体时有可能会误导演化; 且由于所有父个体均从 Archive 中选取使演化群体失去意义。

5.2.3 非劣解存储

为了能够在较短的时间内获得均匀分布的非劣解集，本章采用文献[74]中提出的 ε 占优技术来更新 Archive 群体，存储非劣解。每个解在 Archive 群体中分配一个识别数组 (identification array)$\boldsymbol{B}=(B_1, B_2, \ldots, B_k)$, k 为目标函数个数:

$$B_j(f) = \left\lfloor (f_j - f_j^{\min}) / e_j \right\rfloor \quad \text{for minimizing } f_j \tag{5.10}$$

其中，f_j^{min} 是第 j 个目标函数的可能最小值; ε_j 是第 j 个目标函数容差。利用识别数组 $\boldsymbol{B}$ 把整个目标空间划分成许多网格，每个解根据相应的识别数组判断优劣，按 ε 占优技术存储在网格中，并规定每个网格只能存放一个解。Archive 群体的更新过程如图 5-1(a)中算法 1 所示。

5.2.4 ε-ODEMO 算法流程

ε-ODEMO 算法的流程大致与文献[69]的 DEMO 算法流程相似，其区别是 ε-ODEMO 采用正交实验方法产生初始群体，利用 Archive 群体保存非劣解并通过 ε 占优的方法更新 Archive 群体; 当子群体非空时，把子群体与演化群体合并，对合并群体的裁剪只使用非劣解排序的方法，而用随机选取的方法代替拥挤距离选取。这是因为，ε-ODEMO 算法所得到的非劣解集通过 Archive 群体保存，而 Archive 群体的多样性通过 ε 占优方法保持，因此，演化群体的多样性就可以不考虑。此外，ε-ODEMO 算法采用了混合选择机制，利用 Archive 群体中

Algorithm 1: Updating of Archive with ε-dominance

```
If the offspring C ε-dominates some solutions in Archive then
    Delete all solutions ε-dominated by C in Archive
    Insert C into Archive
Else if C is ε-dominated by any solution in Archive then
    Reject C
Else // C is ε-nondominated with respect to Archive solutions
    If C shares the same grid with an Archive solution D then
        If C usual-dominates D or C is closer to the grid than D then
            Delete D from Archive and accept C
        Else
            Reject C
        End if
    Else // C does not share the same grid with any Archive solution
        Insert C into the Archive
    End if
End if
```

(a)

Algorithm 2: Procedure of the proposed ε-ODEMO

```
Generate a proper OA and generate the orthogonal population OP
Create the initial Archive E(0) with ε-nondominated solutions from OP
Create the initial orthogonal evolutionary population P(0) from E(0) and OP
while eval < Max_eval do
    for i=1 to N
        if eval <λ×Max_eval then
            Random selection
            Produce the new solution with DE/rand/1/bin strategy
        else
            Elitist selection
            Produce the new solution with DE/rand/1/bin strategy
        end if
        if the new solution dominates the target solution P(j)[i] then
            Replace the target solution with the new solution
        else if the new solution is non-dominated by the target solution P(j)[i]
then
            Add the new solution in the child population CP
            child_size++
        else
            Discard the new solution
        end if
        Update the Archive E(j) with ε-dominance concept
    end for
    if the child population is not empty then
        Combine CP and P(j)
        Prune the mixed population using non-dominated ranking method only
        Get the next evolutionary population P(j+1)
    end if
end while
```

(b)

图 5-1 算法 1 和 2 的流程图。(a) ε 占优更新 Archive 群体; (b) ε-ODEMO 算法流程图

个体的信息指导演化，加快了算法的收敛。其算法流程如图 5-1(b)中算法 2 所示。其中 *eval* 表

示函数评价次数，随目标函数被评价而增加；*Max_eval* 是最大函数评价次数，当 $eval \geq Max\_eval$ 时循环终止；*P(j)*[*i*]表示第 *j* 代群体 *P(j)*中的第 *i* 个个体；DE/rand/1/bin 策略如第二章中图 2-1 所示。

§5.3 算法测试及结果分析

5.3.1 测试函数

为了验证改进算法的性能，选择 8 个无约束多目标函数作为标准测试函数集(表 5-1 简述了这些函数的基本特性，详细信息见附录 2)，其中 6 个 ZDT 函数只有 2 个目标函数；前 5 个函数(ZDT1-4，ZDT6)来自文献[63]，这 5 个函数是变量无关的函数。第 6 个函数 ZDT1.1[192]是 ZDT1 函数的变形，它是一个变量相关的函数，只有当所有变量与第一个变量的值相等时，才能收敛到真正 Pareto 前沿。DTLZ1 和 DTLZ7 选自文献[193]，它们具有 3 个目标函数，在本章的研究中 DTLZ1 的自变量个数为 12 个(文献[193]中为 7 个自变量)，这增加了此函数的求解难度。

表 5-1 实验测试函数集

函数名称	*n*	自变量范围	性能
ZDT1	30	$[0, 1]^n$	高维凸函数
ZDT2	30	$[0, 1]^n$	高维非凸函数
ZDT3	30	$[0, 1]^n$	高维不连续凸函数
ZDT4	10	$x_1 \in [0,1]$，$x_i \in [-5,5]^{n-1}$	非凸函数，具有多个局部 Pareto 前沿
ZDT6	10	$[0, 1]^n$	非凸，非均匀分布
ZDT1.1	10	$[0, 1]^n$	凸函数，其变量相关
DTLZ1	12	$[0, 1]^n$	3 个目标函数，超平面空间
DTLZ7	22	$[0, 1]^n$	3 个目标函数，不连续

5.3.2 评价标准

为了评价算法的有效性并与其他算法进行比较，选用文献[61]中提出的两个演化多目标算法评价标准。假设 *Q* 为算法所得到的非劣解集，P^*为问题的真正 Pareto 前沿近似集。

1) **收敛性** g。收敛性用来评价 *Q* 与 P^*之间的逼近程度:

$$g = (\sum_{i=1}^{|Q|} d_i)/(|Q|) \tag{5.11}$$

其中，d_i 表示 *Q* 中第 *i* 个个体与 P^*之间在目标空间的最小欧氏距离。g 值越小，表明算法所得非劣解集 *Q* 越接近所求问题的真正 Pareto 前沿，算法收敛性越好。

2) **多样性** Δ。多样性用于评价算法的分布广度和均匀程度;

$$\Delta = (d_f + d_l + \sum_{i=1}^{|Q|-1} |d_i - \bar{d}|)/(d_f + d_l + (|Q|-1)\bar{d}) \tag{5.12}$$

其中，d_i 是非劣最优目标域 Q 中连续两个非劣解向量间的欧氏距离；$\bar{d}$ 是所有 d_i 的均值；d_f 和 d_l 分别表示真正 Pareto 前沿近似集 P^* 中边界点和非劣解集 Q 边界点间的距离。Δ 值越小，表明算法的多样性越好。

表 5-2 各算法收敛性比较(**黑体**部分表示算法所得到的较好解)

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	ZDT1.1	DTLZ1	DTLZ7
NAGS-II[61]	0.000894	0.000824	0.43411	3.22764	7.80680	0.00392	3.75997	0.03241
	0	0	0.000042	7.30763	0.001667	0.001976	1.89255	0.00457
SPEA2[194]	0.023285	0.16762	0.018409	4.9271	0.23255	0.02572	3. 235731	0.03898
	0	0.000815	0	2.703	0.004945	0.00245	1.38236	0.00365
MMDD-DE[71]	0.000968	0.000752	0.001196	0.001017	0.000598	NA	NA	NA
	0	0	0	0.000120	0.000032	NA	NA	NA
MOEO[195]	0.001277	0.001355	0.004385	0.008145	0.000630	NA	NA	NA
	0.000697	0.000897	0.001910	0.004011	0.000033	NA	NA	NA
DEMO	0.00554	0.14345	0.07951	0.00135	0.57403	0.0064	0.06361	0.07058
	0.000692	0.01237	0.00922	0.000710	0.02934	0.00493	0.11229	0.00904
ε-DEMO	0.00476	0.01682	0.0086	0.000938	0.54372	0.00232	0.040365	0.05346
	0.00518	0.00911	0.000807	0.000265	0.61489	0.000353	0.02364	0.00359
ε-ODEMO	0.000187	0.000534	0.000218	0.000187	0.000615	0.001880	0.00438	0.02036
	0.000012	0.000570	0.000011	0.000012	0.000014	0.000438	0.000250	0.00105

表 5-3 各算法多样性比较(**黑体**部分表示算法所得到的较好解)

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	ZDT1.1	DTLZ1	DTLZ7
NAGS-II[61]	0.463293	0.435112	0.575606	0.479475	0.644477	0.65028	0.95023	0.57615
	0.0414622	0.024607	0.005078	0.009841	0.035042	0.06389	0.2654	0.06149
SPEA2[194]	0.154723	0.33945	0.4691	0.8239	1.04422	0.70987	0.8507665	0.37781
	0.0008738	0.001755	0.005265	0.002883	0.158106	0.0826	0.63254	0.05698
MMDD-DE[71]	0.253462	0.302548	0.308754	0.348541	0.421578	NA	NA	NA
	0.098544	0.003215	0.002143	0.002143	0.026842	NA	NA	NA
MOEO[195]	0.327140	0.285062	0.965236	0.275567	0.225468	NA	NA	NA
	0.065343	0.056978	0.046958	0.183704	0.033884	NA	NA	NA
DEMO	0.33648	0.68299	0.70669	0.3252	0.8625	0.68095	0.46383	0.50651
	0.03744	0.04095	0.05365	0.03349	0.04453	0.05608	0.07482	0.07238
ε-DEMO	0.32187	0.36593	0.48575	0.3187	0.51864	0.58715	0.43465	0.50235
	0.02262	0.0347	0.02748	0.01268	0.27761	0.057	0.08316	0.06324
ε-ODEMO	0.31656	0.27578	0.46081	0.31758	0.18415	0.36348	**0.15784**	**0.33393**
	0.01536	0.02396	0.02587	0.01015	0.01512	0.02714	0.01434	0.0503

5.3.3 参数设置

对所有测试问题，演化群体大小 *NP*=100；最大适应值评价次数 *Max_eval*=10,000；缩放因子 *F*=0.5；选择控制参数λ=0.1；正交设计指数 *J*=2。对 ZDT1-4 和 ZDT6, DTLZ1, DTLZ7，杂交概率 CR=0.1，对 ZDT1.1,CR=0.9。对 ZDT1-3，正交设计水平数 Q=29；对 ZDT4，ZDT6 和 ZDT1.1，DTLZ1，DTLZ7，Q=21；为了使 Archive 群体中获得的非劣解个数大致为 100，对 ZDT1-2，ZDT4 和 ZDT1.1，ε=$[0.0075,0.0075]^T$，对 ZDT3，ε=$[0.0025,0.0025]^T$，对 ZDT6,ε=$[0.0060,0.0070]^T$，对 DTLZ1, ε=$[0.02,0.02,0.05]^T$，对 DTLZ7, ε=$[0.05,0.05,0.05]^T$。

算法采用标准C++语言实现，每个问题采用不同的随机种子独立运行30次，统计平均值和方差。测试环境为: CPU: P-IV 2.8G；内存: 512M；操作系统: MS Windows XP。

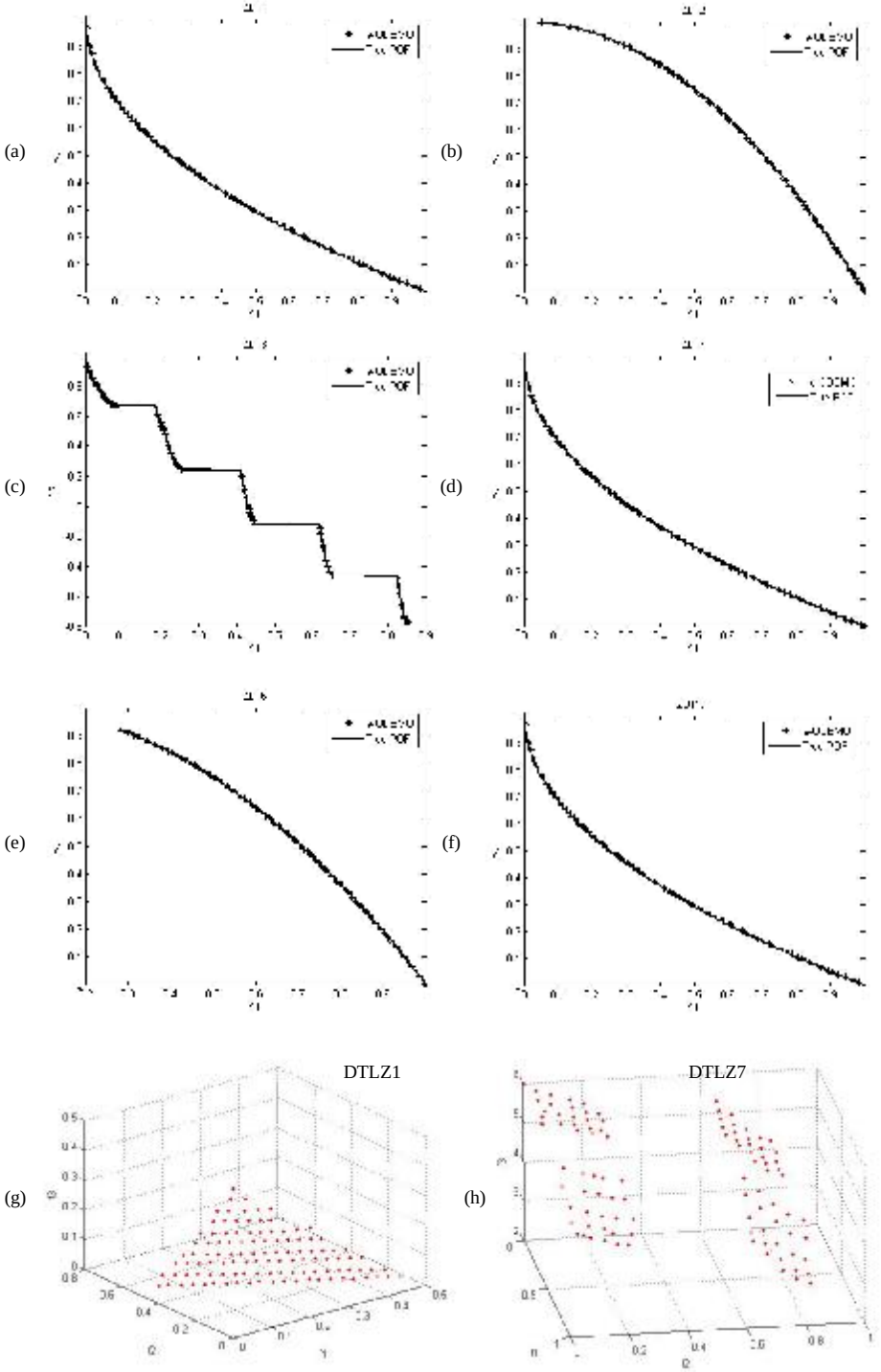

图 5-2 ε-ODEMO 所得非劣解集的 Pareto 前沿。(a) ZDT1; (b) ZDT2; (c) ZDT3; (d) ZDT4; (e) ZDT6; (f) ZDT1.1; (g) DTLZ1; (h) DTLZ7.

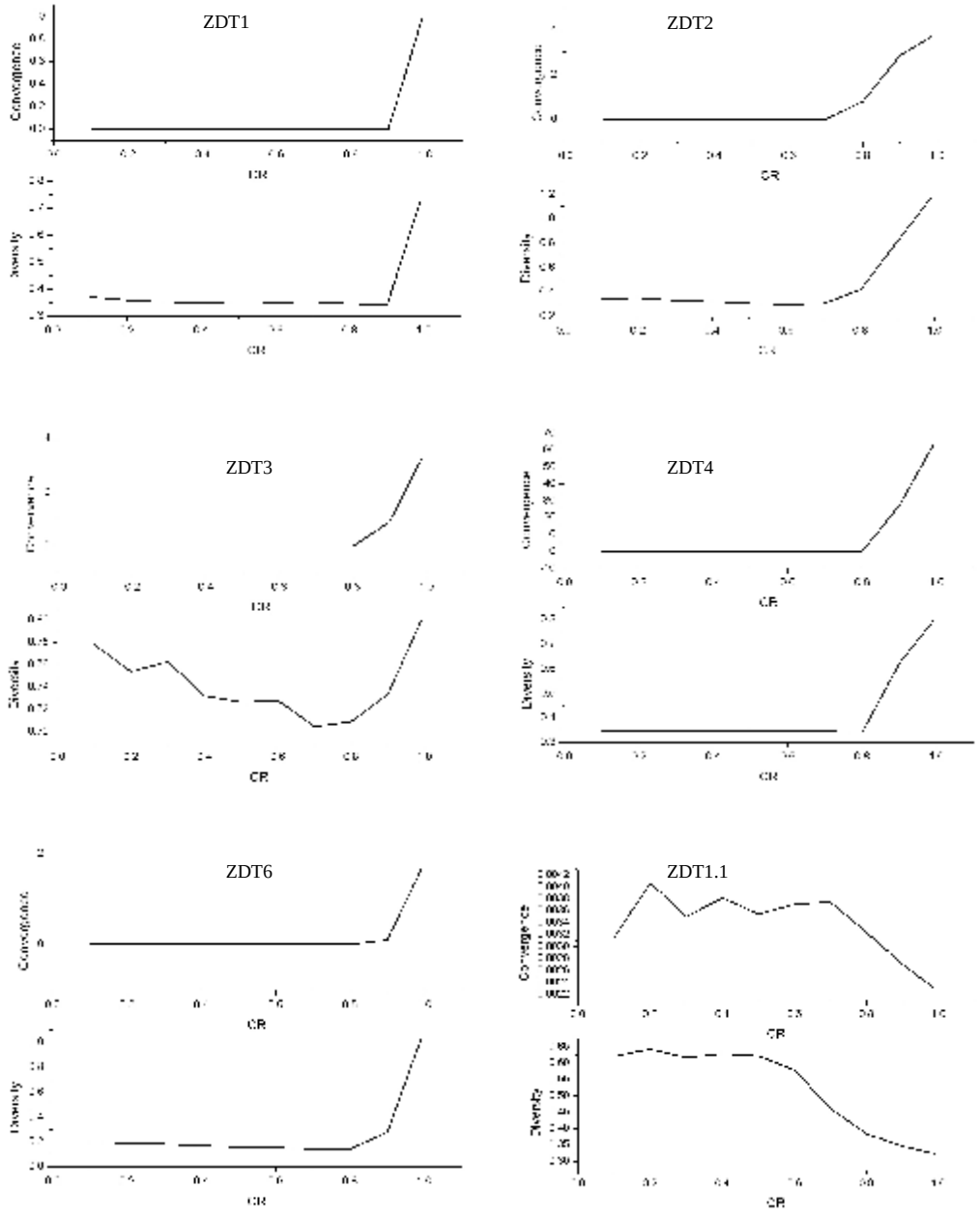

图 5-3 在不同 CR 下各测试函数 Convergence 和 Diversity 评价标准变化曲线图.

5.3.4 实验结果及分析

本节把 ε-ODEMO 的结果与 NSGA-II[61], SPEA2[194], MMDE-DE[71]和 MOEO[195]进行比较。同时, 在其他参数设置与 ε-ODEMO 相同的情况下, 本章对 ε-DEMO(采用 ε 占优方法和随机初始化)和 DEMO(采用随机初始化但不采用 ε 占优方法)进行测试, 以验证正交初始化和 ε 占优方法的性能。值得指出的是 NSGA-II, SPEA2, MMDE-DE 和 MOEO 的最大函数评价次数均为 25,000, 都远大于本章中算法的最大函数评价次数 10,000。表 5-2 为各算法收敛性统

计结果，表 5-3 为多样性统计结果 (各算法第一行表示平均值，第二行表示方差)。

从表 5-2 可以看出，ε-ODEMO 具有很好的收敛性，其收敛性对所有测试函数均优于所比较的四个算法。且收敛性的方差很小，表明 ε-ODEMO 具有很高的稳定性，能在所有 30 次运算中均收敛于所优化问题的真正 Pareto 前沿。NSGA-II 和 SPEA2 的收敛性较差，在 ZDT4 ，ZDT6 和 DTLZ1 上，这两个算法容易陷入问题的局部 Pareto 前沿。DEMO 收敛性较差，特别在 ZDT2 和 ZDT6 上其性能较差，表明该算法在求解非凸 Pareto 前沿问题时性能很差。在引入 ε 占优方法后，ε-DEMO 的收敛性相对于 DEMO，具有一定的改进，但与 ε-ODEMO 相比在所有测试问题上其收敛性都差很多.这表明了 ε 占优方法和正交群体初始化可以有效增强算法的收敛性，指导算法收敛于真正 Pareto 前沿。MMDD-DE 和 MOEO 算法只测试了 ZDTs 问题，其收敛性较好，但比 ε-ODEMO 的性能稍差。

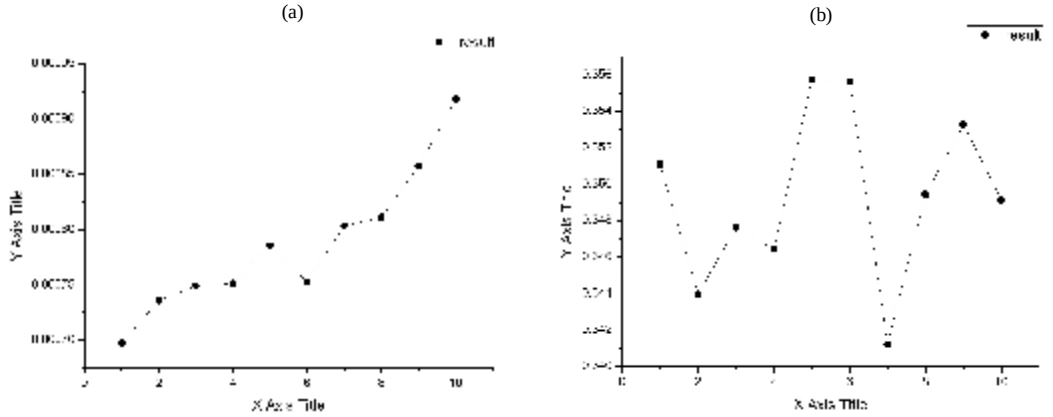

图 5-4 不同 λ 取值下 ZDT1 函数 Convergence 和 Diversity 变化曲线。(a) Convergence; (b) Diversity.

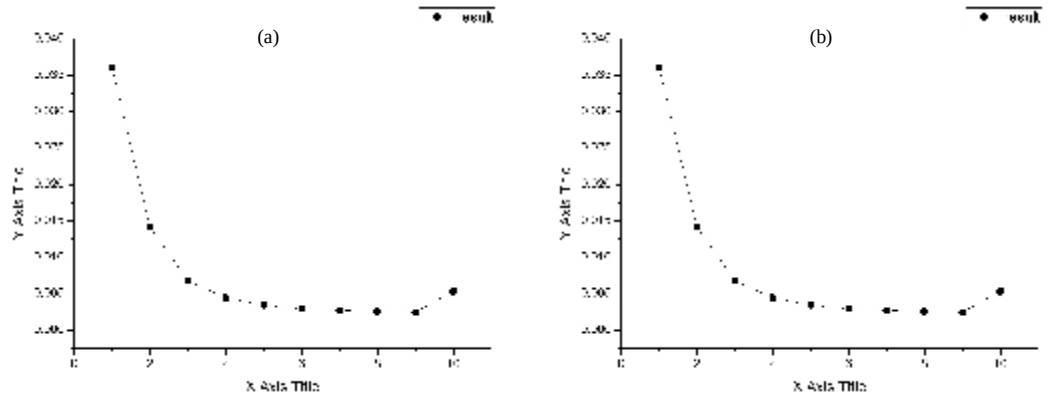

图 5-5 不同 λ 取值下 ZDT1.1 函数 Convergence 和 Diversity 变化曲线。(a) Convergence; (b) Diversity.

表 5-3 表示各算法多样性统计结果。从表中可以看出，ε-ODEMO 在 ZDT2, ZDT3, ZDT4, ZDT6, ZDT1.1, DTLZ1 和 DTLZ7 上优于所比较的算法。在 ZDT1 上 SPEA2 的多样性最好，但从表 5-2 可以看出其收敛性很差。对 DEMO, ε-DEMO 和 ε-ODEMO 这三个算法，从表 5-3 中可以得出与表 5-2 类似的结果，ε-ODEMO 的多样性最好，其次是 ε-DEMO，而 DEMO 的在所

有测试问题上多样性最差。这进一步验证了ε占优方法和正交群体初始化对算法性能的改进。与收敛性相似, MMDD-DE 和 MOEO 的收敛性稍差于 ε-ODEMO。

Zitzler 等[196]指出仅采用数值评价标准不能完全反映算法性能的优劣, 为此, 本章绘制出 ε-ODEMO 所得的非劣解集, 以更直观地了解该算法的性能。图 5-2 为 ε-ODEMO 一次运行所得非劣解集绘制的 Pareto 前沿(其中 True POF 表示所优化问题的真正 Pareto 前沿)。从图 5-2 可以看出, ε-ODEMO 的 Pareto 前沿极好地逼近了所求解问题的真正 Pareto 前沿, 完全覆盖了整个 Pareto 前沿且比较均匀。

5.3.5 CR 取值的影响

在差分演化算法中, 交叉概率 CR 的取值对算法的性能影响较大, 为此, 对 CR 按步长 0.1 取 0.1, 0.2, …, 1.0 的 10 个不同取值进行实验研究, 算法对每个取值独立运行 30 次, 图 5-3 给出了 ε-ODEMO 在不同 CR 值下各测试函数平均收敛性和多样性变化曲线。(注:不同 CR 值对 DTLZ1 和 DTLZ7 的影响与对变量无关的 ZDT 函数影响类似, 此处不再做单独讨论)。

由图 5-3 可以看出, 对于变量无关的问题(ZDT1-4, ZDT6), 当 CR 的值大于 0.8 时, ε-ODEMO 的性能不断减低, 容易陷入局部 Pareto 前沿; 当 CR 的值较小时, 该算法的性能较好。而对于变量相关的问题 ZDT1.1, ε-ODEMO 的性能随 CR 值的增大在提高, 当 CR 值在[0.8,1.0]之间取值时, 算法的性能较好。其原因在于, 对于变量无关的问题, 当 CR 的值比较大时, 差分演化算子容易破坏原有较好的模式, 从而导致算法性能下降。而对变量相关问题, 由于各变量之间存在相关性, 因此, 较大 CR 值可以更多利用差分演化算子的搜索信息, 探索好的模式, 提高算法性能。

5.3.6 混合选择机制性能验证

为了进一步验证混合选择机制对算法性能的改进, 笔者在选择控制参数 l 取 0.1, 0.2, …, 1.0 这 10 个值时对所有测试函数进行测试, 其他参数与 3.3 节中设置一样, 每个函数运行 30 次。值得提出的是当 $l=1.0$ 时, 表示算法只从演化群体中选择个体, 不利用 Archive 中个体的信息。图 5-4 为 ε-ODEMO 在 ZDT1 上收敛性和多样性变化曲线; 图 5-5 为 ε-ODEMO 在 ZDT1-1 上收敛性和多样性变化曲线。其中 X 轴表示 l (1 对应 0.1, 2 对应 0.2, 以此类推); Y 轴表示收敛性或多样性的值; 左图为收敛性变化曲线; 右图为多样性变化曲线。

从图 5-4 可以看出, 虽然 λ 取值对多样性的影响规律不明显, 但是, 当 λ 取值增大时, 收敛性总体上在不断变差。在图 5-5 中, 除了 λ=1.0 外, 其收敛性和多样性都随 λ 增大而得到改进。由此可以得出, 对于变量无关问题, 选择控制参数 λ 应取较小的值, 因为算法能够较快得到较好的解, 并存储在 Archive 中, λ 取较小的值可使算法尽快利用 Archive 中的有效个体指导演化; 而对变量相关问题 λ 应取较大的值, 由于所优化问题变量间相互关联, 在演化初期 Archive 中存储的解不一定是有效解, 对演化的指导作用不大, 如果过多利用 Archive 中的个体信息有可能会使算法陷入局部 Pareto 前沿(如当 λ=0.1 时, 算法的收敛性和多样性都很差)。但是, 当 λ=1.0 时, 无论对变量无关问题还是变量相关问题, 其性能都较差, 这表明合理利用 Archive 中个体的信息可以有效指导演化, 提高算法性能。

§5.4 本章小结

本章提出了一种基于 ε 占优的快速正交多目标差分演化算法 ε-ODEMO。新算法采用正交实验设计和连续空间量化的方法产生初始群体代替传统演化算法中的随机产生方法，增强了算法的利用能力和有效性。ε-ODEMO 采用 Archive 群体来保存得到的非劣解，并利用 ε 占优方法对其进行更新，保证了算法的收敛性和多样性。同时，为了加快算法的收敛，新算法利用基于随机选择和精英选择的混合选择机制来选择差分变异策略中的基向量，一方面精英选择增强了 Archive 中的个体对算法的指导作用，另一方面随机选择增强了算法的开采能力，避免演化早期 Archive 中差的个体误导演化。

通过 8 个标准测试函数对 ε-ODEMO 进行测试，并与 NSGA-II，SPEA2，MMDD-DE，MOEO，DEMO 和 ε-DEMO 算法进行对比，表明 ε-ODEMO 具有很好的收敛性和多样性，且该算法的稳定性很好，在 30 次运算中均能有效收敛于所有测试问题的真正 Pareto 前沿。并且，通过实验对比，验证了正交群体初始化，混合选择机制和 ε 占优方法的有效性。同时，本章通过实验的方法初步讨论了差分演化算法中 CR 取值对算法性能的影响，结论是：对于变量无关的问题，CR 的取值应该较小；而对于变量相关问题，CR 的取值较大。此外，还讨论了选择控制参数 l 取值对算法的影响，结论与 CR 的取值类似：对于变量无关的问题，l 的取值应该较小；而对于变量相关问题，l 的取值较大。今后的工作将致力于在算法中引进约束函数处理机制，以增强算法的约束函数处理能力，以便把算法应用到工程优化中。

第六章 正交多目标差分演化算法在工程优化中的应用

第五章中把 ε 占优技术和正交实验设计与基本差分演化算法相结合，提出了求解多目标优化问题的多目标差分演化算法： ε -ODEMO 算法。该算法通过 8 个无约束多目标标准测试函数进行了验证，实验结果表明了算法的有效性。但是，在实际生活(如工程优化设计等)中经常会遇到各个带有约束条件的多目标优化问题。为了能处理该类问题，需要设计有效的约束函数处理技术，同时，所设计的算法应该涉及尽可能少的控制参数，这样可以方便工程人员的使用。

为此，本章在第五章所提出的 ε -ODEMO 算法的基础上，对其进行了一些改进，主要有: 1) 提出了一种有效的约束函数处理技术来处理约束条件; 2) 采用自适应 ε 占优技术替换 ε -ODEMO 算法中的 ε 占优技术，以减少对不同问题 ε 参数的设置。所提出的改进算法称为 *pa* ε -ODEMO 算法，即 Pareto-adaptive ε -ODEMO 算法。本章余下章节进行如下安排: 第 6.1 节简述了本章的研究背景。在第 6.2 节中，对所改进算法的一些关键技术进行了详细描述。第 6.3 节致力于算法性能的实验验证和结果分析。最后一节是对本章工作的小结。

§6.1 研究背景

第五章中指出 ε 占优技术[189] 可以减少 Archive 群体中解的个数和保证解分布的多样性[74]。但是，针对不同的问题，为了得到合适数目的非劣解需要对 ε 值进行不同的设置。此外，文献[189]中的 ε 占优技术存在一个最大的缺点是: 由于在每个盒子里面只能保存一个解，因此该方法会丢失一些重要的非劣解[197]。为了弥补此不足, Hernández-Díaz 等[197]提出了一种新的 Pareto 自适应 ε 占优技术，即 *pa* ε -dominance。在 *pa* ε -dominance 方法中，根据不同问题 Pareto 前沿的几何特征划分不同的 ε 占优区域，从而可以找出更多的有效非劣解。

利用 MOEA 算法求解工程设计和资源优化等问题已经受到越来越多的关注。一般来说，工程优化问题都涉及很多约束条件，因此，要求解此类问题必须要对约束函数进行处理，即要在算法中加入约束函数处理机制。在演化计算中，不同约束处理机制对算法的性能影响很大。本章为了能有效求解工程优化问题，除了采用上述的 *pa* ε -dominance 方法外，还设计了有效的约束函数处理技术来处理工程优化问题中的约束条件。通过一些标准约束多目标测试函数对算法性能进行可行性验证，在此基础上，把所提出算法应用到 4 个工程优化实际问题中，对问题进行优化，并与其他 MOEA 算法进行了对比研究与分析。本章所涉及的约束多目标优化问题如第五章中公式(5.1)所示。

§6.2 改进的 ε -ODEMO 算法

如上所述，为了能有效处理工程优化问题，在第四章中所提出的 ε -ODEMO 算法的基础上，采用了新的 ε 占优技术(即 pa ε 占优技术[197])来更新 Archive 群体，以保证得到更多有效的非劣解个体；同时，设计有效的约束函数处理技术，以实现对约束条件的处理。本节将对这两个改进进行详述。其他在 ε -ODEMO 算法中已有的技术，如正交群体初始化，混合选择机制等如第四章中所述，此处不再列出。

6.2.1 Archive 群体更新

Zitzler 等[198]指出精英策略可以使 MOEA 算法获得更好的收敛性。在 *pa* ε -ODEMO 算法中，通过一个 Archive 群体来保存演化过程中所得到的非劣解从而实现了算法的精英策略。为了实现较快的收敛速度，算法采用文献[197]中提出的自适应 ε 占优机制，即 *pa* ε -dominance，来更新 Archive 群体。在每一代，新个体在保存到 Archive 群体时需要与当前 Archive 群体中的每一个个体进行 *pa* ε -dominance 比较，满足 *pa* ε -dominance 的个体才能保存在 Archive 群体中，并把被新个体 *pa* ε 占优的个体从 Archive 群体中删除。

```
if 子个体c的B_c ε占优Archive中的任何一个个体a的B_a then
    删除Archive所有被占优的个体
    接受子个体c
else if B_c被Archive中的任何一个个体a的B_a所ε占优 then
    拒绝子个体c
else
    if c和Archive中的个体a共享同一个盒子 then
        if cPareto占优a或者c比a更接近盒子 then
            从Archive中删除a并接受c
        else
            拒绝子个体c
        end if
    else
        接受子个体c
    end if
end if
```

图 6-1 利用 *pa* ε -dominance 更新 Archive 群体

对于 Archive 群体中的每一个个体分配一个识别数组 $\mathbf{B}=(B_1,\mathbf{L},B_k)^T$，其中 k 是目标函数个数:

$$B_i(\mathbf{f}) = \left\lfloor \frac{\log\left(\frac{e_1^i p^{v_i} - (p^{v_i} - 1) f_i}{e_1^i}\right)}{\log\left(\frac{1}{p^{v_i}}\right)} + 1 \right\rfloor \tag{6.1}$$

其中，p 控制曲线(或曲面)的形状，e_1^i 是每一维第一个盒子的大小，v_i 控制曲线(或曲面)改变速度。这些参数满足如下公式:

$$\begin{cases} e_1^i = \dfrac{(p^{v_i} - 1) p^{(T-1)v_i}}{p^{Tv_i - 1}} \\ (1 - 2^{1/p}) p^{Tv_i - 1} + 2^{\frac{1}{p}} p^{\frac{Tv_i}{2}} - 1 = 0 \end{cases} \tag{6.2}$$

其中，T 是决策者所期望获得的非劣解数目。通过识别数组把整个目标空间划分成多个盒子(hyper-box)。对每个子个体 c 和 Archive 中的个体 a 计算其相应的识别数组，子个体 c 采用如图 6-1 所示的过程对 Archive 群体进行更新，其中 $\mathbf{B}_t$ 表示解 t 的识别数组。

采用 *pa* ε -dominance 技术来对 Archive 群体进行更新，可以使算法保证原有 ε 占优技术的优点：即在较短的时间内使所得到的非劣解集有较好的收敛性和解分布的多样性；同时，可以弥补传统 ε 占优技术的缺点：即丢失 Pareto 前沿上一些重要的非劣解。

6.2.2 约束函数处理技术

一般来说，大多数工程优化设计问题具有多个约束条件。因此，在求解此类问题时设计有效的约束处理技术是必要的。文献[80]中对当前在演化算法中所采用的一些主要的约束处理技术进行了综述。但是，在大多数约束处理技术中都涉及一些控制参数，这对于工程应用是不便的。为此，在本节中，采用 Oyama 等[199]提出的一种新的方法，该方法在处理约束函数时不需要任何控制参数。它是一种基于“约束优先，目标次之”的方法[200]，这里首先强调个体的可行性，在个体可行之后在对目标函数进行优化。这是因为在工程设计中，任何不可行个体都可能对设计是致命的。该方法利用约束 Pareto 占优来决定两个解的优劣，描述如下:

定义 6.1 约束 Pareto 占优(Constrained Pareto dominance)：解 $\mathbf{x}_j$ 被解 $\mathbf{x}_i$ 约束占优，记为 $\mathbf{x}_i \ \mathbf{p}_c \ \mathbf{x}_j$，则满足下列任何一个条件即可:

1. 解 $\mathbf{x}_i$ 和解 $\mathbf{x}_j$ 均可行，在目标空间解 $\mathbf{x}_i$ 占优解 $\mathbf{x}_j$；
2. 解 $\mathbf{x}_i$ 可行，而解 $\mathbf{x}_j$ 不可行;
3. 解 $\mathbf{x}_i$ 和解 $\mathbf{x}_j$ 均不可行，在约束空间解 $\mathbf{x}_i$ 占优解 $\mathbf{x}_j$.

其中，在目标空间的 Pareto 占优如定义 5.1 所示，而在约束空间的 Pareto 占优则如下所示:

定义 6.2 约束空间 Pareto 占优(Constraint space Pareto dominance)：解 $\mathbf{x}_j$ 被解 $\mathbf{x}_i$ 在约束空

产生合适的正交矩阵和正交群体OP
利用OP中的非劣解创建初始Archive群体AR_1
从AR_1和OP中创建初始演化群体EP_1
设置$t = 1$, $flag = 0$
while $eval < Max\_eval$ **do**
  $child\_size = 0$
  for $i = 1$ to NP **do**
    if $eval < \lambda \times Max\_eval$ **then**
      随机选择
      利用DE/rand/1/bin产生子个体c
    else
      精英选择
      利用DE/rand/1/bin产生子个体c
    end if
    评价子个体c, $eval$++
    if 子个体cPareto占优EP_t^i **then**
      $EP_t^i = c$
    else if c不被EP_t^iPareto占优 **then**
      把c加入到子群体CP中, $child\_size$++
    else
      丢弃c
    end if
    if $flag == 0$ **then**
      利用Pareto占优更新Archive群体
    else
      $pa\epsilon$-dominance占优更新Archive群体
    end if
  end for
  if $child\_size \neq 0$ **then**
    把CP和EP_t合并
    采用非劣解排序方法对混合群体进行裁剪, 得到下一代群体EP_{t+1}
  end if
  if $ar\_size \geq N\_F$ and $flag == 0$ **then**
    产生$pa\epsilon$-dominance网格
    $flag = 1$
  end if
  t++
end while

图 6-2 *pa* ε -ODEMO 算法基本流程

间占优，记为 $\mathbf{x}_i \ \mathbf{p}_{cs} \ \mathbf{x}_j$，则需同时满足下列两个条件:

1. 在所有约束函数上，解 $\mathbf{x}_i$ 不比解 $\mathbf{x}_j$ 差，即:

$$\forall E_k(\mathbf{x}_i) \geq E_k(\mathbf{x}_j) \tag{6.3}$$

2. 解 $\mathbf{x}_i$ 至少有一个约束函数优于解 $\mathbf{x}_j$，即:

$$\exists E_k(\mathbf{x}_i) > E_k(\mathbf{x}_j) \tag{6.4}$$

其中，$E_k(\mathbf{x}) = \min(0, e_k(\mathbf{x}))$，$k = 1, \mathbf{L}, m$.

6.2.3 *pa* ε -ODEMO 算法流程

为了使算法描述更清晰，图 6-2 给出了 *pa* ε -ODEMO 算法的流程图。算法首先利用正交实验设计的方法产生临时的正交初始群体 *OP* 并评价群体中个体的适应值，随后，利用 *OP* 中的非劣解个体创建初始 Archive 群体 *AR*，再从 *AR* 和 *OP* 中创建初始演化群体 *EP*。在每一代，采用混合选择策略选择差分变异算子中的基向量，然后通过 DE/rand/1/bin 策略产生子个体，如果该子个体优于其相应的父个体，则立即替换该父个体；如果父个体优于子个体，则该子个体被丢弃。否则，如果子个体和父个体不被彼此占优，则把子个体加入到临时的孩子群体 *CP* 中。然后，利用 *pa* ε -dominance 技术对 Archive 群体进行更新。以上步骤重复 *NP* 次，随后把当前演化群体 *EP* 和孩子群体 *CP* 混合，采用非劣解排序的方法从混合群体中选择 *NP* 个个体，作为下一代演化群体。

§6.3 实验结果与分析

为了验证改进算法的性能，本节首先利用 7 个约束多目标标准测试函数对算法进行测试，以验证其求解复杂数学问题的能力。随后，选择 4 个被广泛用于测试 MOEA 算法有效性和应用型的工程设计优化问题来对 *pa* ε -ODEMO 算法进行测试。并且，把 *pa* ε -ODEMO 算法与 NSGA-II[61](该算法被认为是当今最后的 MOEA 算法之一)进行对比。

6.3.1 参数设置

所有比较方法在所有测试函数上都采用不同的随机数种子独立运行 50 次。对不同的问题，其最大适应值评价次数随问题的复杂性而不同，将在后面的实验中对每个函数的 *Max_eval* 进行说明。对不同方法，其参数设置如下:

- 对 NSGA-II[61]: 与文献[61]中基本设置一致，算法采用模拟二进制杂交算法(SBX)和多项式变异算子。杂交概率 $p_c = 0.9$，变异概率 $p_m = 1/n$，其中 n 是自变量个数。杂交和变异算子的分布索引分别是 $h_c = 20$ 和 $h_m = 20$。群体大小 $NP = 100$。
- 对 *pa* ε -ODEMO: 杂交概率 $CR = 0.9$，缩放因子 $F = 0.5$。演化群体大小 $NP = 100$。初始近似 Pareto 前沿解数目 $N\_F = 100$。决策者期望获得的非劣解个数 $T = 100$。选择参数 $l = 0.9$。

- 对于产生正交矩阵 OA，采用参数为 $J=2$，如果 $n<10$，则 $Q=11$；否则 $Q=n-1$。

表 6-1 *pa*ε-ODEMO(*X1*)与 NSGA-II(*X2*)在约束多目标标准测试函数上的性能比较，**黑体**表示算法得到较好的解；各问题第一行为均值，第二行为方差，下同。

Problem	Statistic	*C*		γ		△		Time(s)	
		C(X1,X2)	*C(X2,X1)*	*X1*	*X2*	*X1*	*X2*	*X1*	*X2*
OSY	Mean	**0.26040**	0.16947	**1.31837**	1.56145	0.80109	**0.74815**	**0.64000**	0.72648
	SD	0.11974	0.05659	0.16359	0.22319	0.07555	0.08818	0.02640	0.02360
SRN	Mean	**0.12900**	0.01263	**0.06575**	0.23971	**0.15080**	0.39062	**0.14470**	0.22188
	SD	0.03190	0.00935	0.00882	0.03456	0.01632	0.04516	0.00697	0.01411
TNK	Mean	**0.11880**	0.09949	**0.00146**	0.00228	**0.367440**	0.84805	**0.42648**	0.49932
	SD	0.03243	0.03350	2.28E-4	3.59E-4	0.042210	0.10055	0.02604	0.02156
Kita	Mean	**0.55318**	0.01100	**0.01964**	0.07199	**0.29728**	0.81069	**0.11532**	0.17340
	SD	0.05346	0.01093	0.00145	0.07644	0.03246	0.14839	0.00767	0.01007
Tamaki	Mean	**0.17960**	0.08153	**0.01903**	0.02027	**0.34264**	0.51283	**0.13316**	0.22682
	SD	0.04708	0.02684	0.00075	0.00249	0.01735	0.04984	0.00793	0.03209
DTLZ8	Mean	**0.35160**	0.00373	**0.00982**	0.03549	**0.34844**	0.58427	**2.03898**	2.75628
	SD	0.06052	0.00402	7.86E-4	0.00593	0.02166	0.03546	0.07374	0.18688
DTLZ9	Mean	**0.96980**	0.81995	**0.02018**	0.02530	**0.74592**	0.78633	**2.09844**	2.56438
	SD	0.04851	0.13507	0.00573	0.00356	0.02456	0.03416	0.05457	0.12258

6.3.2 评价标准

本章中除了采用第五章 5.3.2 节所提及的两个评价标准(收敛性 g 和多样性 Δ)外，还采用了文献[63]中所提出的一个二元评价标准(覆盖标准 C)。该标准用来体现一个算法所得到的最终非劣解集占优另一个算法所得的最终非劣解集的程度。覆盖标准 C 的计算公式如下:

$$C(X',X'')=\frac{|a''\in X'';\ \exists a'\in X':\ a'\mathbf{p}=a''|}{|X''|} \tag{6.5}$$

其中 $X',X''\in X$ 是两个目标向量集，$a'\mathbf{p}=a''$ 表示 a' 覆盖(cover) a'' 当且仅当 $a'\mathbf{p}\ a''$ 或 $a'=a''$。函数 C 把排序的配对 (X_i,X_j) 映射到[0,1]区间，其中，X_i 和 X_j 表示算法 i 和算法 j 所得到了最终非劣解集。$C(X_i,X_j)=1$ 表示 X_j 中的所有解被 X_i 中的解覆盖。相反，$C(X_i,X_j)=0$ 表示没有 X_j 中的解被 X_i 中的解覆盖。值得提出的是 $C(X_i,X_j)$ 与 $C(X_j,X_j)$ 是相互独立的，因为它们具有不同的意义。

6.3.3 标准约束多目标函数

首先，我们选择 7 个约束多目标标准测试函数(OSY, SRN, TNK, Kita, Tamaki, DTLZ8 和

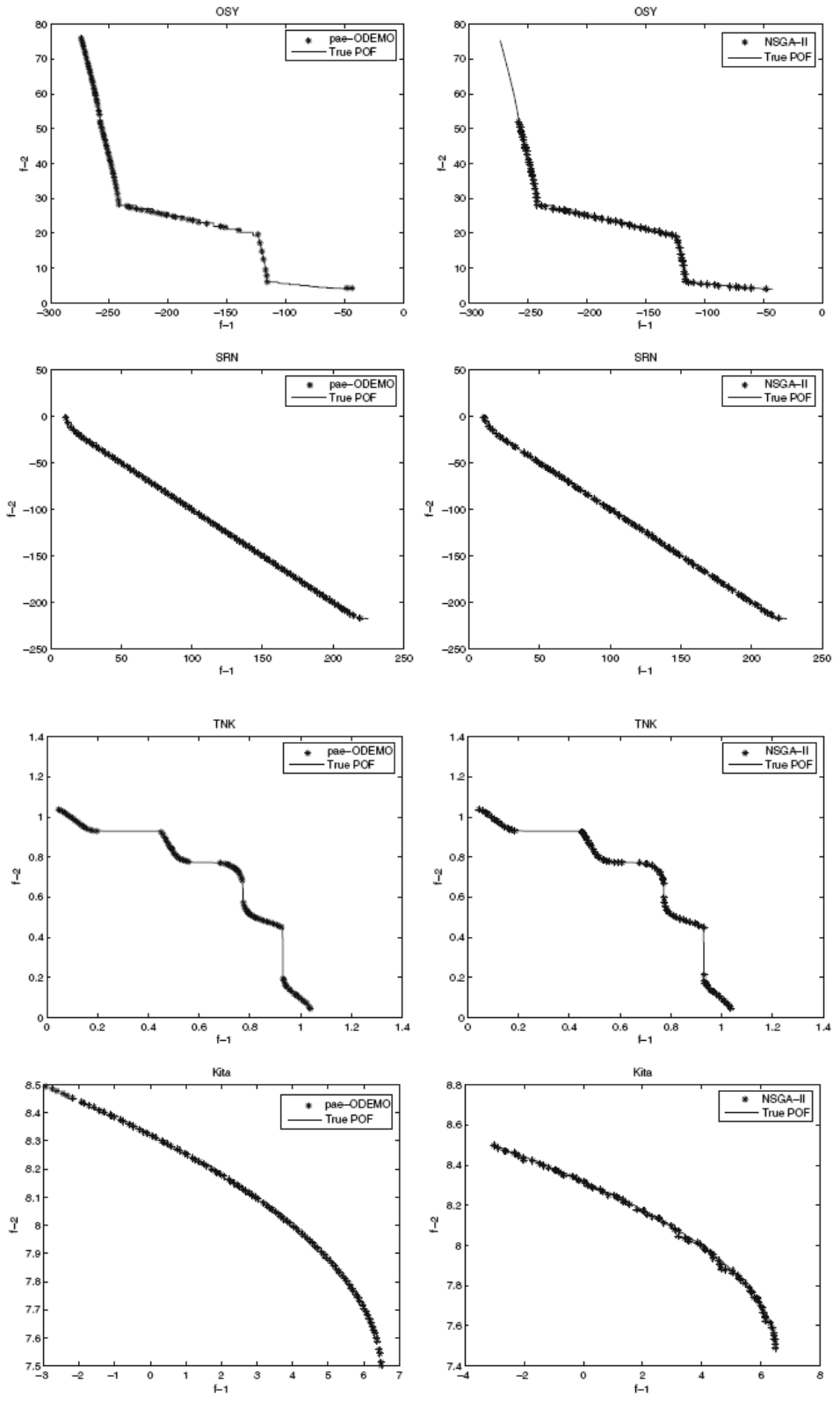

图 6-3 *pa* ε -ODEMO(左)和 NSGA-II(右)在二目标测试函数上的非劣解集分布图

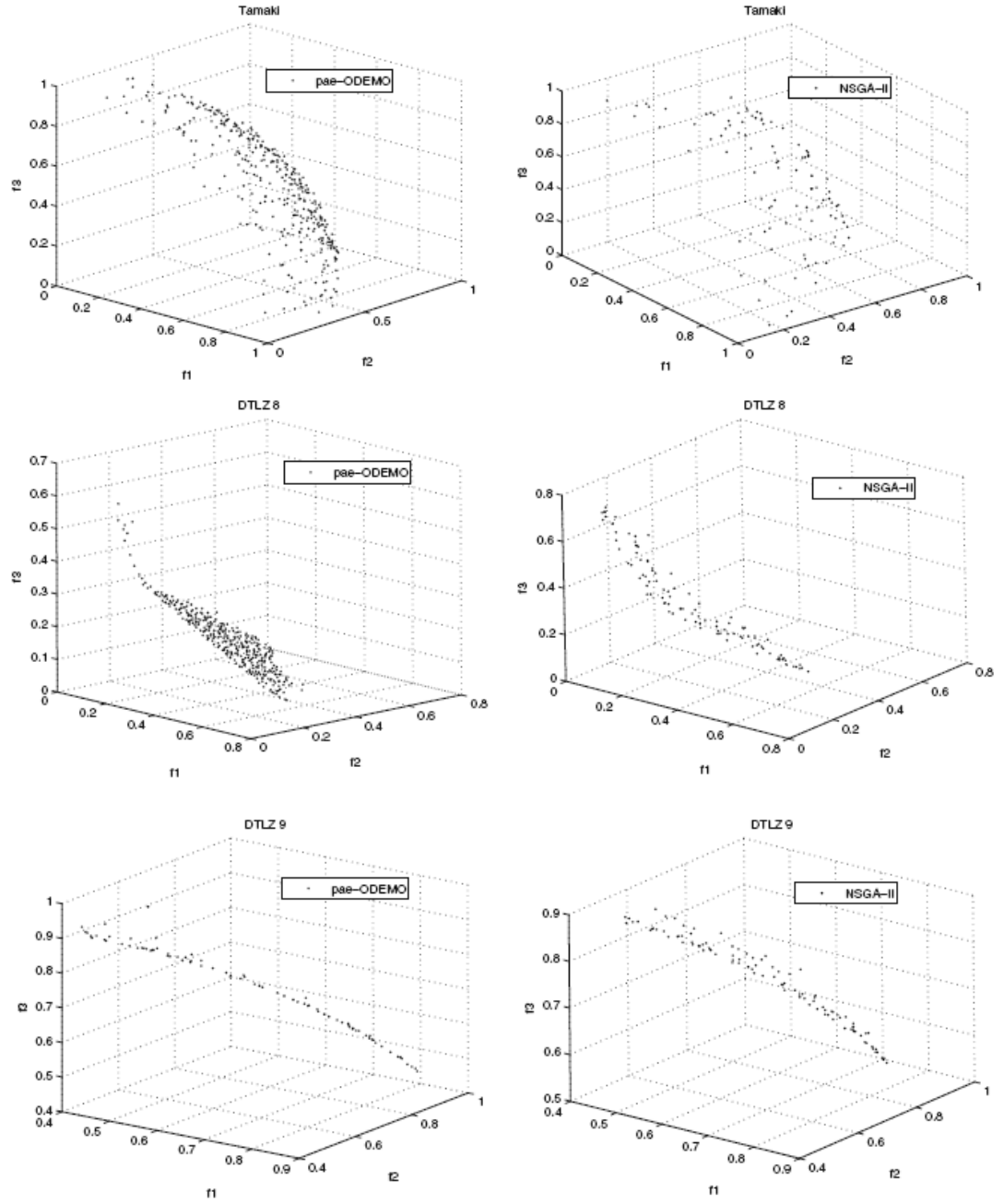

图 6-4 *pa* ε -ODEMO(左)和 NSGA-II(右)在 3 目标测试函数上的非劣解集分布图

DTLZ9)来测试算法求解复杂数学问题的能力，它们的最大适应值评价次数分别为 25000, 5000, 20000, 5000, 5000, 50000 和 50000。这些函数见附录 2，其中 OSY, SRN 和 TNK 来自文献[201], Kita 和 Tamaki 来自文献[73], DTLZ8 和 DTLZ9 来自文献[193]。对于 OSY, SRN, TNK 和 Kita，它们是二目标函数；其他三个函数是 3 目标函数。这些函数的特征描述如下:

- 对 OSY，它有 6 个约束函数。其 Pareto 前沿是由 5 个区域连接而成，每个区域位于一些约束的交界处。由于每一个 Pareto 前沿区域需要 MOEA 算法群体维持一个子群体来保

存不同的非劣解，所以，这个问题对 MOEA 是很难的。

- 对 SRN，它是一个较容易求解的问题。其难点在于由于约束函数的引入而使得原始无约束最优解变得不可行。
- 对 TNK，它是一个不连续的 MOP 问题，由于约束条件使 Pareto 最优解集变得不连续。由于 Pareto 最优解位于非线性约束空间，因此，要找出分布于整个 Paareto 最优区域的解是很困难的。
- 对 Kita，它具有不连续、非凸的 Pareto 最优前沿，且其 Pareto 最优解集也是不连续的。Pareto 最优解集位于可行和不可行区域的边界。
- 对 Tamaki，它是一个 3 目标 MOP 问题，具有连续的 Pareto 最优前沿。
- 对 DTLZ8，它是一个高维的 3 目标 MOP 问题，其 Pareto 最优前沿由一条直线和一个超平面组成。MOEA 算法很难同时找出这两个 Pareto 最优前沿，且很难保证解在超平面上均匀分布。
- 对 DTLZ9，它也是一个高维的 3 目标 MOP 问题。它的 Pareto 最优前沿是 $f_1 = f_2$ 是的一个曲线。Pareto 最优前沿位于两个约束的交界处，这使得该问题很难求解。

表 6-1 给出了 *pa*ε-ODEMO 与 NSGA-II 算法在 7 个约束多目标标准测试函数中的性能比较结果。从表中可以看出，对于所有测试问题，*pa*ε-ODEMO 能够收敛与真正 Pareto 最优前沿附近，且所得非劣解具有较好的分布。从覆盖标准 C 可以看出，对所有 7 个问题，与 NSGA-II 相比，*pa*ε-ODEMO 算法能够提供更高的 C 值，这表明 *pa*ε-ODEMO 所得到的最终非劣解集优于 NSGA-II 所得到的最终非劣解集。对于收敛性标准，在所有测试问题上，*pa*ε-ODEMO 也同样优于 NSGA-II。关于多样性标准，除了 OSY 问题外，在其余 6 个函数上 *pa*ε-ODEMO 优于 NDGA-II 算法。尽管 NSGA-II 在 OSY 问题上对于多样性标准要优于 *pa*ε-ODEMO，但是从图 6-3 中可以看出，NSGA-II 所得非劣解集不能分布于整个 Pareto 最优前沿，而 *pa*ε-ODEMO 则能得到分布于整个 Pareto 前沿的非劣解集。此外，*pa*ε-ODEMO 需要更少的运算时间，其原因在两个方面：1) 对于一个给定的 MOP 问题，在所有 50 次运行中对 *pa*ε-ODEMO 所产生的初始 Archive 群体是一样的，因为所产生的正交矩阵一行，所以，对于 *pa*ε-ODEMO 其初始 Archive 群体只需离线产生一次即可。2) 从图 6-2 可知，*pa*ε-ODEMO 算法没有采用拥挤距离标准，因此，可以节约 NSGA-II 算法中因使用拥挤距离分配和拥挤距离排序所分别带来的 $O(k \cdot 2NP \cdot \log(2NP))$ 和 $O(2NP\log(2NP))$ 的复杂度[61]。

图 6-3 和 6-4 分别给出了 *pa*ε-ODEMO 和 NSGA-II 在二目标函数和 3 目标函数上的非劣解分布图。从图中可以明显看出，*pa*ε-ODEMO 在所有二目标函数上能够得到均匀分布的、完整的、接近最优的 Pareto 前沿。且在所有 3 目标函数上，*pa*ε-ODEMO 能得到比 NSGA-II 更好的非劣解。

从上面的分析可以看出，对于约束多目标标准测试函数，*pa*ε-ODEMO 展示了其求解复杂数学问题的能力，且与 NSGA-II 算法相比，得到了更优的结果。*pa*ε-ODEMO 能得到均匀分布的、完整的、接近最优的 Pareto 前沿，且它能够处理不同特征的约束多目标优化问题，如二目标和 3 目标问题，高维和低维问题，不同类型 Pareto 最优前沿问题(非凸、连续、不连续低密度、非均匀分布)，多约束条件等。

6.3.4 参数敏感性测试

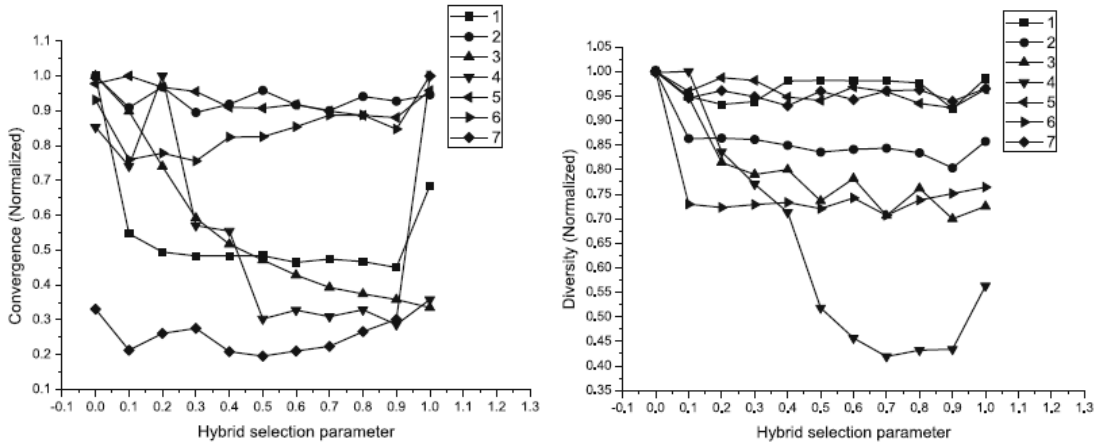

图 6-5 选择控制参数 l 对算法性能的影响，左为收敛性，右为多样性。其中 1-7 分别表示函数 OSY, SRN, TNK, Kita, Tamaki, DTLZ8 和 DTLZ9。图中结果均被归一化，下同。

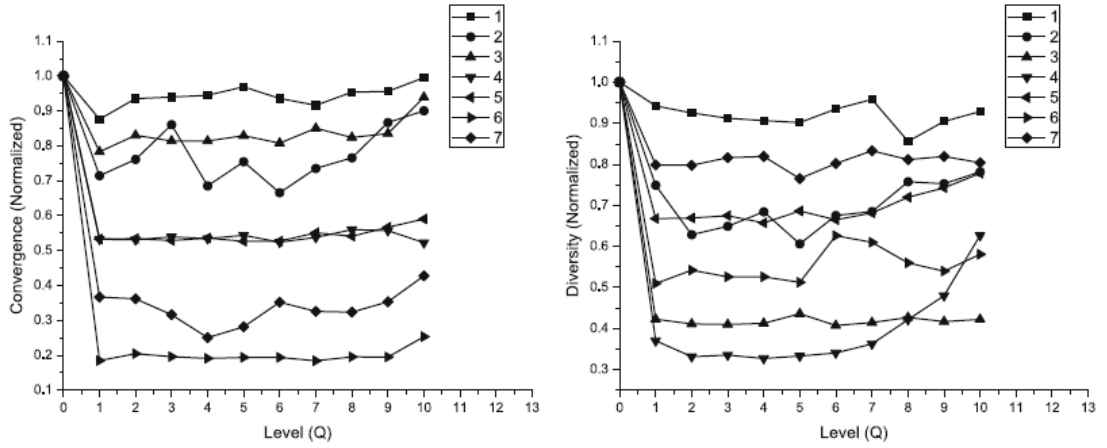

图 6-6 正交实验设计水平数对算法性能的影响，左为收敛性，右为多样性。

在本研究中，对 $pa\varepsilon$-ODEMO 算法我们没有进行严格的尝试来寻求最优的参数设置，因为这已经超出了本研究的范围。但是，本节对不同的参数设置进行了敏感性测试，因为这了解各个参数对算法性能的影响和把算法应用到工程中是有帮助的。

首先，对选择参数 l 在[0,1]范围内按步长 0.1 对其进行实验，以测试该参数对算法性能的影响。其他参数均按照 6.3.1 节所述的设置保持不变。特别地，当 $l=0$ 时，从第五章中公式(5.9)可知，从演化最初DE/rand/1/bin中的基向量就只从Archive群体中选取用于产生子个体。当 $l=1$ 时，表明在整个演化过程中基向量都从演化群体中选取，而 Archive 群体中的个体不参数子个体的生成。图 6-5 给出了在不同选择参数 l 下算法在所有标准测试函数中收敛性和多样性的变化曲线。从图中可以看出，l 的合理取值范围在[0.5,0.9]。此外，图中结果表明在 $l=0$ 时，$pa\varepsilon$-ODEMO 对所有测试函数在收敛性和多样性上表现出最差的性能。这表明过早和过多的采用 Archive 群体中的个体来产生子个体会因为 Archive 中个体差而误导了演化，

这与差分演化算法中的 DE/best/1/bin 策略类似。当 $l=1$时，对于大部分函数，算法的性能都比较差，这表明合适利用 Archive 群体中的个体来产生子个体有利于获得更好的性能。

接下来，我们对正交实验设计中的水平数 Q 选用不同的值来测试该参数对算法性能的影响。除了水平数 Q 外，其他参数均按照 6.3.1 节所示的参数设置。对 OSY, SRN, TNK, Kita 和 Tamaki，Q 设置为 11, 17, 21, 27, 31, 37, 41, 47, 51, 57(图 6-6 中 X 轴的坐标点 1-10)；对 DTLZ8 和 DTLZ9，Q 设置为 29, 35, 39, 45, 49, 55, 59, 65, 69, 75(图 6-6 中 X 轴的坐标点 1-10)。此外，为了验证正交群体初始化的有效性，在 *pa* ε -ODEMO 中采用随机群体初始化代替正交初始化，该值显示在图 6-6 中 X 轴的坐标点 0 上。从图 6-6 可以清楚地看到，对于随机群体初始化，其性能(收敛性和多样性)最差。而且，图中还表明了对于采用不同水平数的正交群体初始化对算法的性能影响很小。

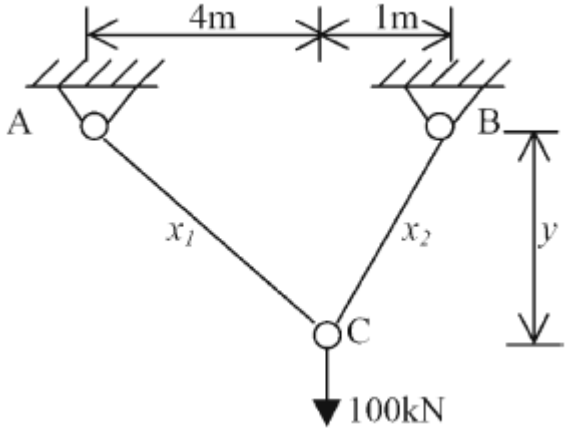

图 6-7 Two-bar truss 设计问题

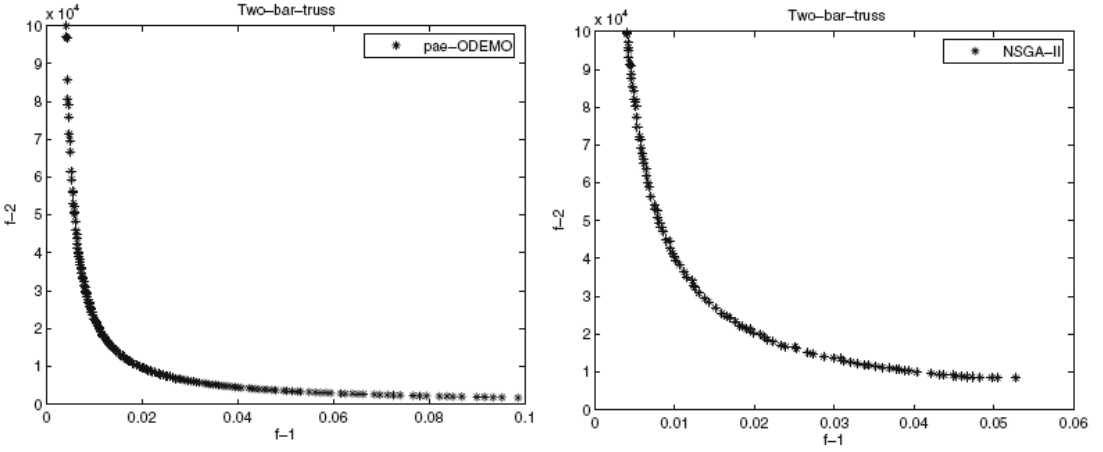

图 6-8 *pa* ε -ODEMO(左)和 NSGA-II(右)在 Two-bar truss 问题上的非劣解集分布图

6.3.5 工程优化实例

从 6.3.4 节中对约束多目标标准测试函数的实验中可以得知 *pa* ε -ODEMO 在处理复杂的数学问题时是很有效的。基于此，本节把该算法用于工程优化设计问题的求解中。4 个工程

优化问题被选取用于验证算法的有效性和应用性，这四个函数的数学描述见附录 2。对于 *pa* ε -ODEMO 和 NSGA-II，各参数的设置如 6.3.1 节所示。算法对各函数分别独立运行 50 次。由于在 6.3.3 节通过标准测试函数已经验证了 *pa* ε -ODEMO 算法的有效性，此处，只给出了 50 次运行中 *pa* ε -ODEMO 和 NSGA-II 所得到的代表性的非劣解集分布图。

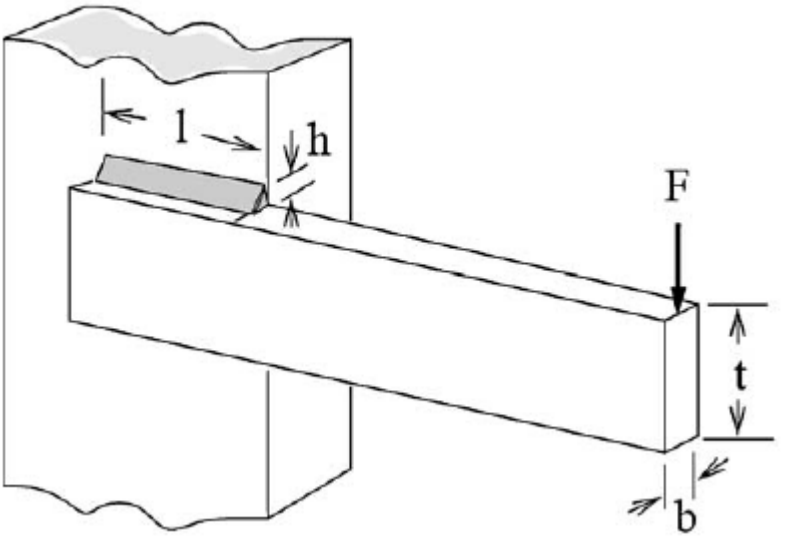

图 6-9 Welded beam 设计问题

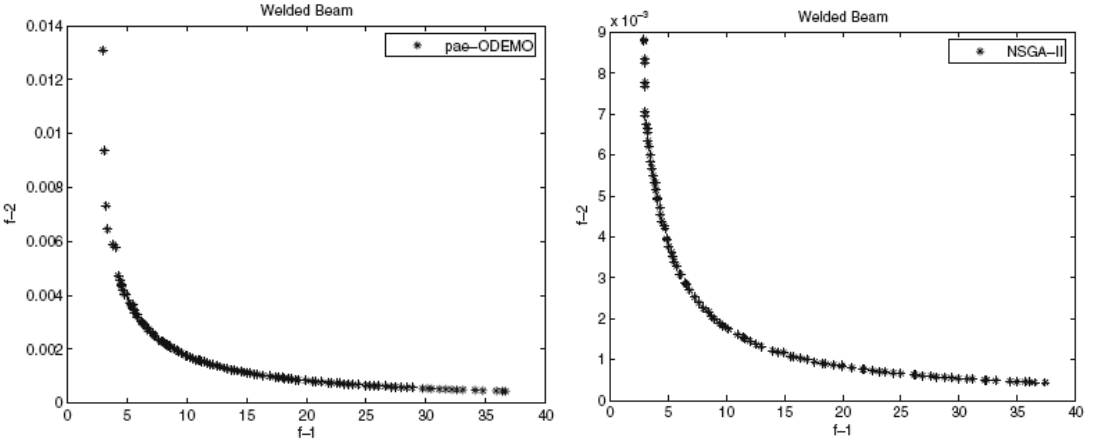

图 6-10 *pa* ε -ODEMO(左)和 NSGA-II(右)在 Welded beam 问题上的非劣解集分布图

6.3.5.1 Two-bar truss 设计问题

第一个工程优化问题是 Tw-bar truss 设计问题，该问题被很多学者选用来验证算法的性能[202-204]。该问题如图 6-7 所示，其数学描述见附录 2。对 *pa* ε -ODEMO 和 NSGA-II，这个问题的最大适应值评价次数为 10000。图 6-7 为 *pa* ε -ODEMO(左)和 NSGA-II(右)所得到的非劣解集分布图。从图 6-8 可以看出，*pa* ε -ODEMO 能得到与 NSGA-II 算法相似的 Pareto 前沿。*pa* ε -ODEMO 所得到的 Pareto 前沿分布在(0.004224 m^3，99987 *kPa*)和(0.098645 m^3，1741.37 *kPa*)；NSGA-II 算法所得到的 Pareto 前沿分布在(0.0043319 m^3，99533.26 *kPa*)和

(0.0.052769 m^3, 8433.61 kPa)。因此, $pa\varepsilon$-ODEMO 能提高分布更广的 Pareto 前沿。此外, 图 6-8 中还表明 $pa\varepsilon$-ODEMO 能保持分布均匀和广阔的 Pareto 最优解。

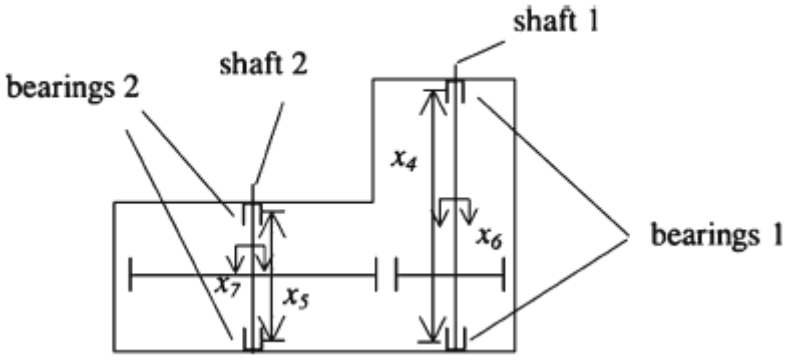

图 6-11 Speed reducer 设计问题

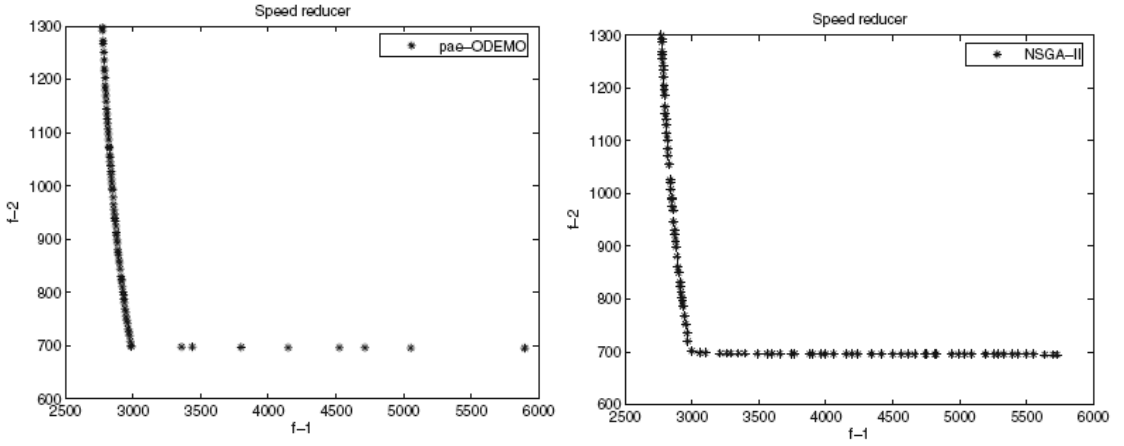

图 6-12 $pa\varepsilon$-ODEMO(左)和 NSGA-II(右)在 Speed reducer 问题上的非劣解集分布图

6.3.5.2 Welded beam 设计问题

Welded beam 设计问题选自文献[205], 它是在满足起重压力, 弯曲压力和抵消电压负荷下获得最小的代价和最小末端倾斜的 Welded beam。其结构图如图 6-9 所示, 数学表示见附录 2。图 6-10 给出了 $pa\varepsilon$-ODEMO(左)和 NSGA-II(右)分别在 15000 次最大适应值评估下所得的非劣解集分布图。图中显示了 $pa\varepsilon$-ODEMO 同样可以获得与 NSGA-II 类似的 Pareto 前沿。$pa\varepsilon$-ODEMO 可以获得具有 0.0131 英寸的最小倾斜下最小代价为 2.8959 和具有 0.00044 英寸的最小倾斜下最小代价为 36.6172 直接的非劣解。对于 NSGA-II, 该算法获得的解在(0.0088, 3.0294)与(0.000439, 37.4018)范围之内。$pa\varepsilon$-ODEMO 算法能很好捕捉到该问题的边界解。通过此问题再次验证了 $pa\varepsilon$-ODEMO 算法的有效性, 它能获得更为广泛的 Pareto 最优解。

6.3.5.3 Speed reducer 设计问题

第三个设计问题是 Speed reducer 设计问题, 该问题如图 6-11 所示。此问题已得到了广泛的研究[73,200,202,203], 其数学描述见附录 2。$pa\varepsilon$-ODEMO(左)和 NSGA-II(右)分别在 15000

次最大适应值评估下所得的非劣解集分布图如图 6-12 所示。*pa* ε -ODEMO 所得到的 Pareto 前沿分布在(2775.02, 1298.01)和(5897.4, 695.348)之间。NSGA-II 所得到的 Pareto 前沿分布在(2772.08, 1299.8)与(5728.87, 696.726)之间。两个算法均能获得广泛的 Pareto 前沿。*pa* ε -ODEMO 与 NSGA-II 在解的收敛性和分布性上有很强的可比性。值得指出的是，在 Pareto 前沿的水平段，*pa* ε -ODEMO 仍然可以得到，这表明了 *pa* ε -ODEMO 算法中所采用的 *pa* ε -dominance 技术的有效性，而传统的 ε 占优技术是很难得到在此问题上水平段的解的。

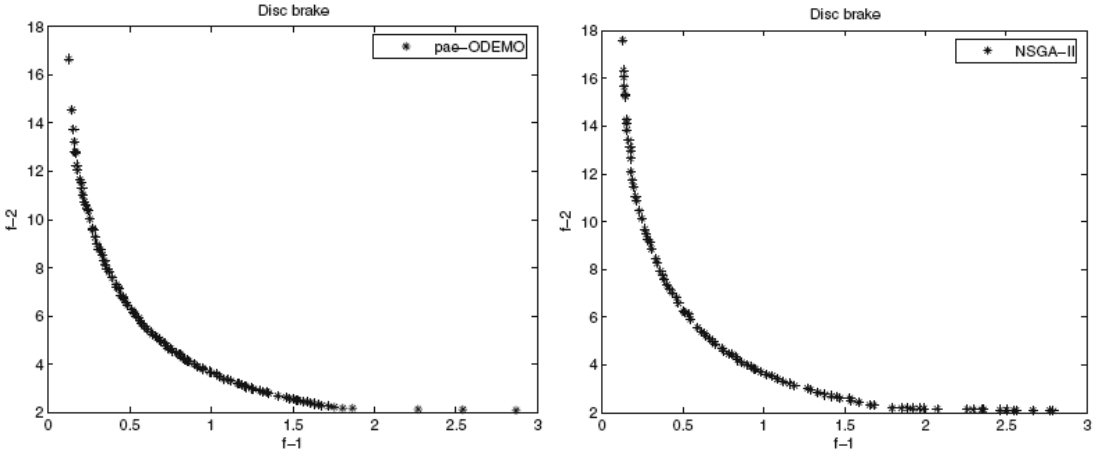

图 6-13 *pa* ε -ODEMO(左)和 NSGA-II(右)在 Disc brake 问题上的非劣解集分布图

表 6-2 *pa* ε -ODEMO(*X1*)与 NSGA-II(*X2*)在工程优化问题上的性能比较

Problem	Statistic	*C*		*γ*		△		Time(s)	
		C(X1,X2)	*C(X2,X1)*	*X1*	*X2*	*X1*	*X2*	*X1*	*X2*
Two-bar truss	Mean	**0.95900**	0.00240	**254.9408**	439.7384	**0.87393**	0.93725	**0.26560**	0.30660
	SD	0.01669	0.00497	48.2813	52.8976	0.05032	0.02425	0.02685	0.00876
Welded beam	Mean	**0.19620**	0.14481	**0.09169**	0.16875	**0.58607**	0.88987	**0.37520**	0.46880
	SD	0.08366	0.04019	0.00733	0.08030	0.04366	0.11976	0.01940	0.02451
Speed reducer	Mean	**0.26524**	0.05000	**2.69281**	3.0771	0.84041	**0.79717**	**0.42500**	0.46260
	SD	0.13005	0.03411	0.24051	0.10782	0.20085	0.06608	0.00671	0.00876
Disc brake	Mean	0.19800	**0.20418**	**0.05175**	0.08203	**0.61511**	0.66542	**0.13780**	0.16880
	SD	0.09348	0.09952	0.00579	0.01957	0.04581	0.01431	0.00716	0.00661

6.3.5.4 Disc brake 设计问题

第四个设计问题用于优化文献[205]中提及的 Disc brake 问题，该问题的数学描述见附录 2。该问题的设计目的是最小化磁盘的刹车片数和最小化停止时间。问题设计四个自变量分别是磁盘的内半径 x_1，外半径 x_2，预定压力 x_3 和摩擦面的数目 x_4。其约束条件包括两个半径间的最小距离、刹车的最大长度、压力、温度和力矩限制等。因此，该问题是一个混合变量的约束多目标优化问题。*pa* ε -ODEMO(左)和 NSGA-II(右)分别在 5000 次最大适应值评估下所得的非劣解集分布图如图 6-13 所示。从图 6-13 可以看出，两个算法均能获得广泛分布的均匀 Pareto 解。*pa* ε -ODEMO 的解分布在(0.1274, 16.6549)与(2.8684, 2.0906)之间; NSGA-II

的解分布在(0.1293, 17.598)和(2.7915, 2.1127)之间。值得指出的是, NSGA-II 得到的解(0.1293, 17.598)被 *pa*ε-ODEMO 得到的解(0.1274, 16.6549)Pareto 占优。

6.3.5.5 数值对比

以上在优化 4 个工程优化问题中 *pa*ε-ODEMO 和 NSGA-II 算法主要是采用算法所得到的 Pareto 前沿进行对比, 此处利用 4 个数值评价标准(收敛性g, 多样性Δ, 覆盖性C和运行时间)对两个算法进行进一步比较。值得指出的是, 由于这 4 个问题是实际工程优化问题, 其真正的 Pareto 前沿目前没有得到工作, 本研究采用穷举和并行处理方法获得它们的近似 Pareto 前沿, 对于某些问题这需要大量的运算时间。

表 6-2 给出了 *pa*ε-ODEMO 和 NSGA-II 在 4 个工程优化问题中的结果比较。算法在每个问题上独立运行 50 次, 统计其均值和方差。从表 6-2 可以看出, NSGA-II 算法与 *pa*ε-ODEMO 相比需要更多的运行时间, 其原因在于 NSGA-II 采用了基于拥挤距离的非劣解排序机制, 从而具有更高的时间复杂度。对于所有 4 个问题, *pa*ε-ODEMO 能获得更好的收敛性。此外, 在 3 个问题中, *pa*ε-ODEMO 能得到更小的多样性值, 这表面在这几个问题上 *pa*ε-ODEMO 在多样性上优于 NSGA-II。对于 Speed reducer 问题上, NSGA-II 的多样性优于 *pa*ε-ODEMO, 其原因在于 *pa*ε-ODEMO 在 Pareto 前沿的水平段获得的解个数少于 NSGA-II 在此段所得到的解的数目。

从上面的分析可以看出, *pa*ε-ODEMO 在非劣解的多样性和收敛性上与 NSGA-II 相比具有很强的可比性。*pa*ε-ODEMO 在求解约束多目标问题时能得到分布广泛且均匀的非劣解集。从对标准测试函数和实际工程优化问题的优化结果可以看出, *pa*ε-ODEMO 算法能得到有效的, 均匀分布的, 接近完整的和接近最优的 Pareto 最优解集。

§6.4 本章小结

为了能有效把第五章中提出的 ε-ODEMO 算法应用于工程优化问题的求解中, 本章在 ε-ODEMO 算法的基础上, 采用 *pa*ε-dominance 技术来代替 ε-ODEMO 中的 ε 占优技术以使算法能得到更多的有效 Pareto 解, 从而更有利于工程人员从所得到的非劣解集中根据偏好和问题需要进行选取。此外, 利用新的基于约束 Pareto 占优来处理工程优化中所遇到的约束条件, 实现对约束函数的有效控制。

本章首先利用 7 个约束多目标标准测试函数对 *pa*ε-ODEMO 算法在求解数学问题上的能力和有效性进行了实验验证, 同时, 把实验结果与 NSGA-II 算法进行比较。实验结果表明了 *pa*ε-ODEMO 算法不仅能有效求解这些函数, 而且与 NSGA-II 算法具有很强的可比性。在此基础上, 把 *pa*ε-ODEMO 算法应用于 4 个工程设计问题的优化中, 实验验证了新算法能取得与 NSGA-II 相似的 Pareto 前沿, 且在多个问题上所得到的结果优于 NSGA-II 算法。*pa*ε-ODEMO 在这 4 个问题上能得到有效的、均匀分布的、接近完整的和接近最优的 Pareto 最优解集。

第七章 基于点对称标准的差分演化聚类算法研究

§7.1 引言

本论文第二章到第六章对基本差分演化算法进行了改进，并把改进算法分别用于无约束单目标函数优化、无约束多目标函数优化以及约束多目标函数优化(包括多目标工程优化设计问题)等问题的求解中，通过实验验证了改进算法的有效性。

聚类分析是数据挖掘中一种重要的数据分析方法，它是一种无指导分类算法。它把无标记的数据集通过一定的相似性评价标准分成很多组(簇)。在每一组中，相似性高的个体被分在一组，而不相似的个体则要分在不同组，这里，相似性评价标准可以采用很多不同标准，如欧拉距离评价标准等。目前，在聚类分析中，已经提出了多种算法，有关聚类分析好的综述文章见文献[12]和[19]。其中，大多数聚类分析算法都可以转化成一个多峰优化问题。从优化的观点来看，聚类问题是一个 NP-难问题[206]。传统的聚类算法在对数据集进行聚类时，容易陷入局部最优解，因此，所得到的数据划分也是局部最优的。为了能尽可能得到较好的划分，把演化算法与传统的聚类分析算法相结合，形成有效的演化聚类分析算法已成为研究聚类算法一大热点[207]。

差分演化算法是一种高效、简单、快速的全局优化算法，它是演化算法的一种。如前所述，该算法已经在很多领域取得了成功的应用。本章结合差分演化算法的优点，把该算法用于聚类分析中，提出基于点对称标准的差分演化聚类分析算法。算法利用改进的点对称规则作为评价标准，同时设计了适合聚类分析的演化群体中个体的表示方法，并对演化中错误个体进行修正，通过标准测试数据集和 UCI[208]数据集对算法性能进行了测试。本章 7.2 节对本研究中的一些相关工作进行了简述，如聚类分析的问题定义，当前演化 K 均值聚类算法的研究，点对称评价标准等。第 7.3 节详细描述了本章中所提出的差分演化聚类分析算法。第 7.4 节致力于算法性能的实验验证，并对差分演化算法杂交概率 CR 的不同取值对算法性能的影响进行了研究。最后一节, 7.5 节是对本章工作的小结。

§7.2 相关工作

本节首先简述了聚类分析的定义，然后对目前已有的演化 K 均值聚类算法进行了简要综述，最后对当前所提出的一些基于点对称距离的评价标准进了描述。

7.2.1 问题定义

聚类问题可以描述为:

给定数据集 $\mathbf{X}=\{\mathbf{x}_1,\mathbf{L},\mathbf{x}_p,\mathbf{L}\,\mathbf{x}_N\}$，其中 $\mathbf{x}_p$ 是 d 维特征空间的一个模式(pattern)，N 是数据集 $\mathbf{X}$ 中模式的个数，则对数据集 $\mathbf{X}$ 的聚类就是把 $\mathbf{X}$ 分成 K 组 $\{C_1,\mathbf{L},C_K\}$，这 K 组子集满足下列条件:

1) 每一个模式必须分配到一个组中，即 $\mathbf{U}_{k=1}^{K}C_k=\mathbf{X}$;

2) 每一个簇中至少包含一个模式，即 $C_k\neq\Phi,k=1,\mathbf{L},K$;

3) 每一个模式只能分配到一个组(对于硬聚类)中，即 $C_i\ \mathbf{I}\ C_j=\Phi,i\neq j$。

目前已有很多聚类分析算法，比如 K 均值聚类算法[209]，EM 算法，分层聚类算法等。其中，K 均值聚类算法 [1] 是一种的经典聚类分析算法，在 2006 年香港召开的 ICDM'06[20]中被评为至今 10 大数据挖掘算法之一[21]。K 均值聚类算法在很多领域得到了广泛的应用，包括数据挖掘、机器学习、模式识别等领域。

K 均值算法具有如下两个有点:

1) 算法简单，容易实现;

2) 算法复杂度低，适合求解大规模数据集问题。

但是，传统 K 均值算法存在以下几点不足之处:

1) 算法是非数据集独立的(data-dependent);

2) 算法采用贪心搜索算法，受初始条件的影响，容易收敛于问题的局部最优解;

3) 算法需要用户事先给定 K 值，这在很多实际运用中是不实际的。

7.2.2 演化聚类算法

由于传统聚类分析算法容易受到初始条件的影响，容易陷入问题的局部最优解，所以很多学者利用演化算法的全局搜索能力把它与聚类分析算法相结合，形成了演化聚类算法。有关演化聚类算法好的综述文章见文献[207]。由于本章主要基于 K 均值聚类算法提出差分演化聚类算法，所以，主要对演化 K 均值聚类算法进行简述。

7.2.2.1 K 均值算法

K 均值聚类算法(K-means clustering algorithm)是数据挖掘中重要的算法之一，已被广泛应用到各种领域。其核心思想是: 算法把 n 个向量 $\mathbf{x}_j(j=1,\mathbf{L},n)$ 划分为 c 个组 $G_i(i=1,\mathbf{L},c)$，并求出每组的聚类中心，使得非相似性标准的评价函数（或目标函数）达到最小。当选择欧几

[1] 注: 本文中将模糊 C 均值算法(Fuzzy C-Means algorithm, FCM)也包含在 K 均值聚类算法中[19].

里德距离为组 j 中向量 $\mathbf{x}_k$ 与相应聚类中心 c_i 间的非相似性标准时，其评价函数可定义为:

$$J=\sum_{i=1}^{c}J_i=\sum_{i=1}^{c}(\sum_{k,\mathbf{x}_k\in G_i}\|\mathbf{x}_k-c_i\|^2) \tag{7.1}$$

其中 $J_i=\sum_{i=1}^{c}(\sum_{k,\mathbf{x}_k\in Gi}\|\mathbf{x}_k-c_i\|^2)$ 是第 i 组内的评价函数。这样 J_i 的值与 G_i 的几何特性和 c_i 的位置有关系。

各组通过划分后一般用一个 $c\times n$ 的二维隶属矩阵 $\mathbf{U}$ 来表示。如果第 j 个数据点 $\mathbf{x}_j$ 属于组 i，则 $\mathbf{U}$ 中的元素 $u_{i,j}=1$，否则，$u_{i,j}=0$。在确定聚类中心 c_i 后，有公式(7.1)得出 $u_{i,j}$ 为:

$$u_{i,j}=\begin{cases}1, & \text{对每个}k\neq i,\text{如果}\|\mathbf{x}_j-c_i\|^2\le\|\mathbf{x}_j-c_k\|^2,\\ 0, & \text{其它}\end{cases} \tag{7.2}$$

如果 c_i 是数据点 $\mathbf{x}_j$ 的最近聚类中心，则 $\mathbf{x}_j$ 属于第 i 组。在 K 均值算法中，因为一个给定数据点只能属于一个组，因此隶属矩阵 $\mathbf{U}$ 具有如下性质:

$$\sum_{i=1}^{c}u_{i,j}=1,\qquad \forall j=1,\ldots,n \tag{7.3}$$

且

$$\sum_{i=1}^{c}\sum_{j=1}^{n}u_{i,j}=n \tag{7.4}$$

此外，如果固定 $u_{i,j}$ 则使式(7.1)达到最小的最佳聚类中心就是第 i 组中所有向量的均值:

$$c_i=\frac{1}{|G_i|}\sum_{k,\mathbf{x}_k\in G_i}\mathbf{x}_k\ , \tag{7.5}$$

这里 $|G_i|$ 是组 G_i 中数据点的个数或 $|G_i|=\sum_{j=1}^{n}u_{i,j}$。

结合以上各公式，K 均值算法描述如下(该算法重复下列步骤，确定聚类中心 c_i 和隶属矩阵 $\mathbf{U}$):

步骤 1: 初始化聚类中心 $c_i, i=1,\mathbf{L},c$;

步骤 2: 利用公式(7.2)计算隶属矩阵 $\mathbf{U}$;

步骤 3: 根据公式(7.1)计算评价函数值。如果它小于某个给定的阀值，或它相对上次评价函数值的改变量小于某个阀值，则终止算法；否则，转入步骤 4。

步骤 4: 利用公式(7.5)修正聚类中心，并返回步骤 2。

由上可知，K 均值算法是一种迭代算法，该算法不能确保收敛于全局最优解。K 均值算法的性能依赖于聚类中心初始位置的选取，因此，为了能取得较好的聚类效果，要么利用专家知识选取较好的初始点中心；要么采用多次运行，每次采用不同初始中心，直到得出较好的聚类结果。

7.2.2.2 演化 K 均值算法

演化 K 均值聚类算法的核心思想是把演化算法与 K 均值聚类算法相结合，根据一定的适应值计算函数，把聚类问题转化为优化问题。一般的，函数优化问题可以描述为:

$$Minimize \quad f(X). \quad X=(x_1,x_2,...,x_d)\in R^d \tag{7.6}$$

其中 $X\in S.\ \ S\subseteq R^d$ 称为搜索空间，$f(X)$ 称为目标函数，通常每个自变量每一维 x_i 满足一定的边界约束条件

$$l_i \le x_i \le u_i, \qquad i=1,2,...,d \tag{7.7}$$

具体应用到聚类分析中，$f(X)$ 表示所要优化的目标适应值函数(如在硬 K 均值聚类算法中可以采用公式(7.1)作为目标适应值函数)，个体 X 代表数据集的一个划分，根据不同的编码方法，个体 X 的表示方法不同，从而会形成不同的搜索空间 S (连续的或者离散的)。

7.2.2.3 个体编码

设计演化 K 均值聚类算法的一个重要步骤是,对所解问题的变量进行编码表示，编码表示方案的选取在很大程度上依赖于问题的性质及遗传算子的设计。通常，在设计演化算法时，只有两个方面与所求问题有关，即问题的编码表示与适应函数的确定。

在多数演化 K 均值聚类算法中主要采用的个体编码方法有三种:

1) **二进制编码(Binary Encoding)**[210]: 二进制编码存在以下三个缺点[213]: (a) 问题冗余; (b) 对传统杂交和变异算法上下文不敏感; (c) 对大数据集由于编码长度过长容易造成编码不稳定。

2) **面向簇的直接编码(Direct Encoding of Object-Cluster Association)**[211]: 这种方法对于 N 个数据模式分配到 K 个簇中，每个个体的长度为 N，个体中每个基因从 $[1,K]$ 中取整数值。但是，这种编码方式存在一个主要缺点: 编码冗余，例如"11322"和"2311"实际上表示同一种划分。

3) **实数编码(Real-value Encoding)**[212]: 每个个体采用一串实数($K\times d$)表示 K 个聚类中心，其中第一 d 个实数表示第一个中心点，第二 d 个实数表示第二个中心点，以此类推。采用实数编码具有以下两个优点: (a) 个体的表示更接近问题的表示(聚类问题一般都转化成实数问题); (b) 可以提供高精度的解。由于实数编码的优越性，目前很多演化 K 均值聚类算法都采用实数编码[213,214,19]。

7.2.2.4 适应值计算

在自然界中,个体的适应值就是其繁殖能力,这将直接关系到其后代的数量。在演化计算中,适应函数是区分群体中个体好坏的标准,是算法演化过程的驱动力,是进行自然选择的唯一依据。改变种群内部结构的操作都通过适应值进行控制。设计演化 K 均值算法的另一个重要步骤就是适应值函数的选取，这其中包括聚类相似性评价方法的选取。

在讨论适应值函数计算之前，首先对在聚类中经常使用的距离公式进行简介，因为两个模式(数据点)之间距离的计算是计算适应值的前提。设在数据集中有 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 两个 d 维数据点，计算这两个点之间的距离可以采用:

1) **欧几里德距离(Eculidean distance)**:

$$d(\mathbf{x}_i,\mathbf{x}_j)=\sqrt{\sum_{p=1}^{d}(\mathbf{x}_{i,p}-\mathbf{x}_{j,p})^2}=\left\|\mathbf{x}_i-\mathbf{x}_j\right\| \tag{7.8}$$

2) **Minowsky 距离(Minowsky distance)**[12]:

$$d^a(\mathbf{x}_i,\mathbf{x}_j)=\left(\sum_{p=1}^{d}(\left|\mathbf{x}_{i,p}-\mathbf{x}_{j,p}\right|)^a\right)=\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^a \tag{7.9}$$

从公式(7.9)可以看出，欧几里德距离实际上是 Minowsky 距离在 $a=2$ 是的特例。

3) 当 $a=1$ 时，则公式(7.9)成为**曼哈顿距离(Manhattan distance)**[215]。

4) **余弦距离(Cosine distance)**：由于 Minowsky 距离对于数据集的维数很高时，其距离不能有效表示远或者近，因此会影响聚类性能[216]。对于这个问题，可以采用如下余弦距离来克服。

$$\left\langle \mathbf{x}_i,\mathbf{x}_j\right\rangle=\frac{\sum_{p=1}^{d}(\mathbf{x}_{i,p}\cdot\mathbf{x}_{j,p})}{\left\|\mathbf{x}_i\right\|\left\|\mathbf{x}_j\right\|} \tag{7.10}$$

5) **Mahalanabis 距离(Mahalanabis distance)**:

$$d_M(\mathbf{x}_i,\mathbf{x}_j)=(\mathbf{x}_i-\mathbf{x}_j)\sum{}^{-1}(\mathbf{x}_i-\mathbf{x}_j) \tag{7.11}$$

其中 $\sum$ 表示数据集的协方差矩阵。

由于在自然界中对称(symmetry)是一个常见的特征，因此一些学者提出了基于点对称的距离 (Point symmetry-based distance)聚类方法。关于点对称距离标准将在 7.2.3 节进行介绍。

在演化 K 均值聚类算法中，最常用的适应值函数是根据簇间误差平方和或者簇间误差平方根值设计，簇间误差平方和由公式(7.1)定义，簇间误差平方根值定义如下:

$$J=\sqrt{\sum_{i=1}^{c}(\sum_{k,x_k\in Gi}\|\mathbf{x}_k-c_i\|^2)} \tag{7.12}$$

文献[25]根据公式(7.1)设计了如下适应值计算函数:

$$fitness=\frac{1}{1+J} \tag{7.13}$$

显然，个体适应值越大表示聚类效果越好，当 $J=0$ 时，适应值取最大值 1。

文献[22]，[212]，[217]和[218]直接采用簇间误差平方和作为适应值计算函数，个体适应值越小，聚类效果越好。

7.2.2.5 遗传算子的设计

遗传算子的设计是演化计算中最富有特色和创造性的部分。但是，在演化 K 均值聚类算法中，由于问题的特殊性，需要重新设计更有效的遗传算子。

针对不同的编码规则和不同的演化算法可以设计出不同遗传算子，本文主要讨论广泛

使用的杂交算子(crossover operator)和变异算子(mutation operator) 。

1) **杂交算子(crossover operator)**

文献[211]采用面向簇的直接编码方法，每个个体采用一个整数串，因此，在文献[211]采用单点杂交算子(single-point crossover operator)[219]。

文献[23]中采用实数编码法方法，利用单点杂交算子产生子个体。

文献[220]采用面向簇的直接编码方法，利用两点杂交算子(two-point crossover operator)产生子个体。

2) **变异算子(mutation operator)**

在演化 K 均值聚类算法中，针对不同的编码规则，在演化算法中常用的变异算子[219]都可以直接使用到演化 K 均值聚类算法中，此处不再赘述。

7.2.2.2.6 控制参数的选取

在设计演化算法时，一方面，需要针对具体问题选择适当的编码方案及相应的遗传算子;另一方面,需要选择算法的控制参数。控制参数的选取和设计在设计演化 K 均值聚类算法更为重要。

在演化 K 均值聚类算法中，对于所设计的演化算法中的参数，如群体大小，变异概率，杂交概率等，一般都采用给定的值，然后在演化过程中固定初始值不变。

文献[214]采用文献[221]中提出的适应杂交概率和变异概率的方法来调节杂交和变异概率，从而可以避免人为设定初始值的不准确性。在演化过程中杂交概率 p_c 和变异概率 p_m 计算如下:

$$p_c = \begin{cases} k_1 \times \dfrac{f_{\max} - f}{f_{\max} - f_{avg}} & \text{if } f > f_{avg} \\ k_3 & \text{otherwise} \end{cases} \tag{7.14}$$

$$p_m = \begin{cases} k_2 \times \dfrac{f_{\max} - f}{f_{\max} - f_{avg}} & \text{if } f > f_{avg} \\ k_4 & \text{otherwise} \end{cases} \tag{7.15}$$

其中，$f_{\max}$ 是当前群体最大适应值，f_{avg} 是当前群体平均适应值，f 为当前个体适应值，$k_1 = k_3 = 1.0$ 是杂交概率控制参数，$k_2 = k_4 = 0.5$ 是变异概率控制参数。

7.2.3 点对称距离标准

如上所述，对称现象在自然界中是经常见到的，如树叶就是对称的。为了能对具有对称属性的数据集进行有效聚类，许多学者提出了基于点对称距离的评价标准，我们将其中典型的几种方法简述如下。

7.2.3.1 Su 和 Chou 的方法

Su 和 Chou 在文献[222]中提出了一种基于点对称距离的评价方法。给定数据点 $\mathbf{x}_j$ 和聚类中心 c，则 $\mathbf{x}_j$ 关于 c 的点对称距离计算如下:

$$d_s(\mathbf{x}_j, c) = \min_{i=1,\mathbf{L},n \text{ and } i\neq j} \frac{\left\|(\mathbf{x}_j - c) + (\mathbf{x}_i - c)\right\|}{\left\|\mathbf{x}_j - c\right\| + \left\|\mathbf{x}_i - c\right\|} \tag{7.16}$$

在公式(7.16)基础上，作者在文献[223]中对其进行了改进，改进方法如下:

$$d_c(\mathbf{x}_j, c) = d_s(\mathbf{x}_j, c) \times d_e(\mathbf{x}_j, c) \tag{7.17}$$

其中，$d_s(\mathbf{x}_j, c)$ 由公式(7.16)计算得到，$d_e(\mathbf{x}_j, c)$ 表示点点 $\mathbf{x}_j$ 和聚类中心 c 的欧几里德距离。

但是，文献[222]和[223]中的方法有可能导致对外部对称的簇造成错误的分类[214,224]，同时，该算法不具有封闭性[224]。

7.2.3.2 Bandyopadhyay 和 Saha 的方法

由于文献[222]和[223]中提出的点对称距离不能检测关于簇间对称的数据集[214,224]，因此, Bandyopadhyay 和 Saha 提出一种新的点对称距离计算方法[214]: 对于数据点 $\mathbf{x}$ 和聚类中心 c，$\mathbf{x}$ 关于 c 的最对称点为 $\mathbf{x}^* = 2c - \mathbf{x}$，设数据集中关于 $\mathbf{x}^*$ 第一临近点和第二临近点的欧几里德距离为 d_1 和 d_2，则 $\mathbf{x}$ 关于 c 的对称距离计算如下:

$$d_{ps}(\mathbf{x}, c) = \frac{(d_1 + d_2)}{2} \times d_e(\mathbf{x}, c) \tag{7.18}$$

7.2.3.3 Chung 和 Lin 的方法

同样，针对文献[222]和[223]中点对称距离的不足，Chung 和 Lin 提出一种新的点对称距离计算方法[224]: 给定数据点 $\mathbf{x}_j$ 和聚类中心 c_k，则 $\mathbf{x}_j$ 关于 c_k 的点对称距离计算如下:

$$SSL(\mathbf{x}_i, c_k) = \max_{X_i \in C_k} \sqrt{\frac{DSL^2(\mathbf{x}_i, c_k, \mathbf{x}_j) + OSL^2(\mathbf{x}_i, c_k, \mathbf{x}_j)}{2}} \tag{7.19}$$

其中 $DSL(\mathbf{x}_i, c_k, \mathbf{x}_j)$ 定义如下:

$$DSL(\mathbf{x}_i, c_k, \mathbf{x}_j) = \begin{cases} 1 - \dfrac{|d_i - d_j|}{d_i} & \text{if } 0 \le \dfrac{d_j}{d_i} \le 2 \\ 0 & \text{otherwise} \end{cases} \tag{7.20}$$

这里 $d_i = d_e(\mathbf{x}_i, c_k)$，$d_j = d_e(\mathbf{x}_j, c_k)$。

$OSL(\mathbf{x}_i, c_k, \mathbf{x}_j)$ 定义为:

$$OSL(\mathbf{x}_i, c_k, \mathbf{x}_j) = \frac{v_i \cdot v_j}{2\|v_i\|\|v_j\|} + 0.5 \tag{7.21}$$

这里 $v_i = \overrightarrow{\mathbf{x}_i c_k} = c_k - \mathbf{x}_i$, $v_j = \overrightarrow{c_k \mathbf{x}_j} = \mathbf{x}_j - c_k$。

```
1: 产生初始群体P
2: 对于每一个个体执行一代K均值算法
3: 评价群体P中每个个体的适应值
4: while 如果终止条件不满足 do
5:   for i = 1 to NP do
6:     随机均匀选择r1 ≠ r2 ≠ r3 ≠ i
7:     利用DE/rand/1/exp策略产生子个体U^i
8:     评价子个体U^i的适应值
9:     if U^i优于P^i then
10:      P^i = U^i
11:    end if
12:  end for
13: end while
```

图 7-1 点对称差分演化聚类(DEPS-C)算法流程图

§7.3 点对称差分演化聚类分析算法

7.2.2 节对当前一些演化 K 均值聚类算法进行了简述, 特别是描述了当前演化 K 均值聚类算法中所采用的一些关键技术, 但是, 这些算法主要存在以下两个不足[12,19]: 1) 大部分演化算法设计很多控制参数, 这些控制参数对不同的数据集是很敏感的; 2) 演化聚类算法的执行时间明显高于传统的聚类算法。为了能弥补当前演化聚类算法的一些不足, 把差分演化算法应用聚类分析中, 并采用点对称距离作为评价标准对数据点进行分配。算法均有以下几个优点: 1) 算法只涉及到两个参数, 即群体大小 NP 和杂交概率 CR。把传统差分演化算法的缩放因子 F 采用随机产生, 且通过对 CR 的敏感性分析得知, CR 对不同问题是不敏感的; 2) 差分演化算法是一种快速的优化算法, 因此, 基于差分演化算法的聚类分析算法能在一定程度上弥补其他演化算法的不足。该算法简称为 DEPS-C(Differential Evolution based on Point Symmetry with Closure for clustering)算法, 算法的流程如图 7-1 所示。其中差分演化算法的

DE/rand/1/exp 策略见第二章中图 2-2。

7.3.1 改进的点对称距离标准

文献[214]中提出的点对称评价标准在基于遗传算法的聚类算法中取得了较好的实验结果[214]，因此，在 DEPS-C 算法中采用类似与[214]中所提出的点对称方法。但是，如文献[224]中所分析，点对称评价标准应具有封闭性[224]，以得到更好的划分。为了弥补此不足，本研究对[214]中的点对称方法进行了改进，即点 $\mathbf{x}_i$ 关于聚类中心 c_t 中的第一和第二对称点($\mathbf{x}_a$ 和 $\mathbf{x}_b$)只能从该聚类中心 c_t 所在的簇 C_t 中寻找，而不是整个数据集。即:

数据集中的每一个点 $\mathbf{x}_i, i=1,\mathbf{L},n$ 分配到簇 C_t 当且仅当 $d_{ps}(\mathbf{x}_i,c_t) \le d_{ps}(\mathbf{x}_i,c_s)$， $s,t=1,\mathbf{L},k$，$s \ne t$， $(d_{ps}(\mathbf{x}_i,c_t)/d_e(\mathbf{x}_i,c_t)) \le q$，且 $\mathbf{x}_a$ 和 $\mathbf{x}_b$ 只能属于簇 C_t。其中 $d_{ps}(\mathbf{x}_i,c_t)$ 按照公式(7.18)进行计算，$d_e(\mathbf{x}_i,c_t)$ 是点 $\mathbf{x}_i$ 到中心点 c_t 的欧氏距离，$q=d_{nn}^{\max}$ 是对称阈值，$d_{nn}^{\max}=\max_{i=1,\mathbf{L},n} d_{nn}(\mathbf{x}_i)$ 是整个数据集中两个相邻点的最大距离[214]。基于该改进的点对称方法的点分配过程如图 7-2 所示。

```
for i = 1 to n do
    for t = 1 to k do
        在簇C_t中找出点x_i关于聚类中心c_t的第一和第二对称点x_a和x_b /* 保证封闭性 */
        计算点对称距离d[t] = d_ps(x_i, c_t)
    end for
    找出t* = Argmin_{t=1,...,k} d[t]
    if (d_ps(x_i, c_t*)/d_e(x_i, c_t*)) ≤ θ then
        分配点x_i到簇C_t*
    else
        按照欧氏距离(即t* = Argmin_{t=1,...,k} d_e(x_i, c_t))分配点x_i到簇C_t*
    end if
end for
```

图 7-2 基于改进的点对称方法的点分配流程

在文献[214]中，作者指出找出关于一个点的对称点的复杂度是 $O(kn^2)$，其中，k 为簇的个数，n 为数据集中数据点的个数。为了减少算法复杂度，我们采用基于 Kd 树(Kd-tree)的最邻近点搜索技术，它可以把复杂度从 $O(kn^2)$ 减少到 $O(kn\log(n))$。DEPS-C 利用文献[225]中开发的基于 C++语言的 ANN 库来寻找最临近点。ANN 库中实现了利用 Kd 树任意高维数据集中寻找最邻近点。

7.3.2 个体表示

在DEPS-C算法中，对于具有 n 个数据点的数据集，每个数据点为 d 维，且用户给定的簇的个数为 k，则 DEPS-C 算法的个体是一个 $d \times k$ 维的实数向量，其中第一个 d 维实数是第一个簇的中心，第二个 d 维实数是第二个簇的中心，以此类推。对于每个个体，群体初始化时其 k 个中心点从数据集中随机选取。

7.3.3 适应值计算

对于每个个体的适应值计算方法与文献[214]中个体适应值的计算方法类似。首先，对于每一个数据点，采用图 7-2 所示的算法分配的一个簇中。然后，每个个体的适应值按照点对称距离进行评价。最后，类似于 K 均值算法对 k 个新簇的新中心点进行计算。个体适应值的计算过程如图 7-3 所示。从图中可以看出，个体的适应值需要被最小化以达到最优解。

1: 按照图7-2所示的算法对每个点进行分配
2: $f = 0$
3: **for** $t = 1$ to k **do**
4: **for** 对每个数据点$x_i, i = 1, \cdots, n$且$x_i \in C_t$ **do**
5: $f = f + d_{ps}(x_i, c_t)$
6: **end for**
7: **end for**
8: 按如下方法计算新的簇中心$c'_1, \cdots, c'_k$: $c'_t = \frac{1}{n_t} \sum_{x_i \in C_t} x_i, t = 1, \cdots, k$ 其中n_t簇C_t中数据点的数目.

图 7-3 个体适应值计算流程

7.3.4 错误个体处理

在 DEPS-C 算法中，图 7-2 中的数据点分配和图 7-3 中的个体适应值计算都需要寻找第一和第二对称点，本研究采用基于 Kd 树的最邻近点搜索技术来找出这两个点，因此，在每个簇中至少应该包含两个数据点。对于一个个体，如果其中一个簇中所包含的数据点少于两个，则对此个体进行重新初始化，即从数据集中随机选择 k 个数据点作为簇的中心。然后，对此个体重新进行数据点的分配。

§7.4 实验结果与分析

7.4.1 参数设置

在DEPS-C算法中，有两个算法控制参数需要实现给定，即群体大小 NP 和杂交概率 CR。

此外，对每个不同问题需要设置一个停机条件，本研究采用最大适应值评价次数 Max_NFFEs 作为停机条件。在基本差分演化算法中的缩放因子 F 我们采用文献[5]和[17]中提出的 dither 技术产生，即对每一个个体 F 在[0.0, 1.0]范围内随机产生。采用 dither 技术的原因在于它能改进差分演化算法的性能；此外，还可以减少用户对该参数的控制。对于所有实现，如果没有特殊说明，采用以下参数设置:

- 群体大小: $NP = 100$；
- 杂交概率: $CR = 0.1$；
- 差分演化算法策略: DE/rand/1/exp;
- 最大适应值评价次数: $\text{Max\_NFFEs} = 3000$。

表 7-1 本研究中数据集描述

数据名	数据描述
dataset-01	选自文献[214, 222]，它由两条带组成，每条带含有 200 个数据点。
dataset-02	选自文献[214,222,226]，它包含 350 个数据点，由环形簇，紧凑簇和线型簇组成。
dataset-03	选自文献[214]，它包含 400 个点，由环形簇，矩形簇和线型簇组成。
dataset-04	选自文献[214,226]，包含 250 个点，有 5 个椭圆形的簇组成，簇之间高度重叠。
dataset-05	选自文献[214]，包含 76 个点，由 3 个簇组成。
dataset-06	选自文献[214,226]，包含 400 个点，在 3 维空间由 4 个不连接的椭球体组成。
dataset-07	选自文献[214,226]，包含 300 个点，由 6 个簇组成。
dataset-08	选自文献[226]，包含 400 个点，由两个相交的椭圆形组成。
dataset-09	选自文献[227]，由包含 9 个簇的 900 个点组成。
dataset-10	选自文献[227]，包含 500 各点，由 10 个簇组成。
iris	$n = 150, d = 4, k = 3$，由 3 类不同的花组成，每类包含 50 个数据点。
breast	$n = 683, d = 9, k = 2$
glass	$n = 214, d = 9, k = 6$
wine	$n = 178, d = 13, k = 3$
vowel	$n = 871, d = 3, k = 6$
liver	$n = 345, d = 6, k = 2$
habermans	$n = 306, d = 3, k = 2$
cmc	$n = 1473, d = 9, k = 3$

在实验中，算法对每个数据集独立运行 50 次。所用算法采用标准 C++语言实现，测试环境是: CPU: P-IV (Core 2) 2.1 GHz; 内存: 1 GB; 操作系统: MS Windows XP。

实验结果与文献[214]的 GAPS 算法结果进行比较。GAPS 算法是一种采用点对称距离标准的基于遗传算法的聚类算法，该算法的性能由于 SBKM 算法[222]和传统 K 均值算法[214]。对于 GAPS 算法，实验中采用与文献[214]相同的参数设置，除了对所有数据集 $\text{Max\_NFFEs} = 3000$ 外。此外，为了验证封闭性对算法性能的影响，另外两个算法也被用来进行对比: 1) GAPS-C，即在 GAPS 算法上采用图 7-2 所示的封闭性点分配方法; 2) DEPS: 即在 DEPS-C 算法上不采用封闭性的点分配方法。

7.4.2 实验数据集

为了验证 DEPS-C 的性能, 选用了 10 个人工数据集和 8 个 UCI 数据集作为测试用例, 其中 dataset-01 到 dataset-10 是人工数据集, 其余 8 个为 UCI 数据集[208]。表 7-1 对这 18 个数据集进行了简要描述。

图 7-2 各算法适应值结果比较。其中**黑体**表示算法得到较好的结果。下同。

Data sets	GAPS		GAPS-C		DEPS		DEPS-C	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
dataset-01	1.24E+02	2.20E+00	1.24E+02	4.13E+00	1.22E+02	6.71E-01	**1.19E+02**	3.92E-01
dataset-02	1.61E+01	7.32E-01	1.67E+01	1.01E+00	1.54E+01	4.83E-01	**1.53E+01**	2.74E-01
dataset-03	1.76E+02	0.00E+00	1.76E+02	0.00E+00	1.76E+02	0.00E+00	1.76E+02	0.00E+00
dataset-04	1.11E+02	7.83E-01	1.10E+02	1.60E+00	1.08E+02	9.17E-01	**1.07E+02**	5.50E-01
dataset-05	1.02E+01	0.00E+00	1.02E+01	0.00E+00	1.02E+01	0.00E+00	1.02E+01	0.00E+00
dataset-06	4.73E+02	4.68E-01	4.73E+02	4.68E-01	**4.72E+02**	2.36E-02	4.73E+02	0.00E+00
dataset-07	1.28E+02	0.00E+00	1.28E+02	0.00E+00	1.28E+02	0.00E+00	1.28E+02	0.00E+00
dataset-08	4.30E+01	1.22E+01	4.21E+01	7.65E+00	3.44E+01	4.53E-01	**3.43E+01**	5.01E-01
dataset-09	8.00E+01	9.82E-02	7.76E+01	1.98E-01	7.77E+01	6.91E-01	**7.64E+01**	3.85E-01
dataset-10	3.59E+02	3.23E-01	3.48E+02	2.77E-02	3.58E+02	1.19E+00	**3.47E+02**	1.03E+00
iris	3.20E+01	7.38E-03	3.20E+01	7.15E-03	3.20E+01	1.37E-01	**3.19E+01**	6.66E-03
breast	1.27E+04	6.24E+00	**1.26E+04**	1.40E+00	1.28E+04	8.52E+00	1.28E+04	8.06E+00
glass	2.23E+02	3.89E+00	2.45E+02	2.69E+00	**2.14E+02**	1.09E+00	2.39E+02	3.85E+00
wine	2.33E+05	2.61E+03	2.41E+05	2.50E+03	**2.31E+05**	2.61E+03	2.36E+05	2.81E+03
vowel	8.89E+06	1.24E+05	1.02E+07	1.45E+05	**8.75E+06**	6.45E+04	9.96E+06	1.57E+05
liver	1.99E+05	4.63E+02	2.00E+05	4.35E+02	1.99E+05	2.94E+02	**1.98E+05**	4.10E+02
habermans	8.84E+03	8.70E+00	9.04E+03	5.98E+01	**8.83E+03**	6.07E+00	8.87E+03	1.89E+01
cmc	9.23E+03	1.12E+01	9.29E+03	8.74E+00	**9.20E+03**	3.31E-01	9.27E+03	1.27E+01

7.4.3 评价标准

为了对 7.4.1 节中所提及的 4 个算法进行比较, 3 个评价标准被选择用来评价它们的性能。这 3 个评价标准描述如下:

1) **适应值(fitness value)**: 每次独立运行的最优个体适应值被记录, 并对 50 次独立运行结果的均值和方差进行统计。显然, 其均值越小, 表明算法的性能越好。

2) **F 评价标准(F-measure)**[228]: F-measure 与信息提取相关, recall 和 precision 评价了聚类算法对一个数据集划分的好坏程度。recall 定义为 $r(i,j)=n_{ij}/n_i$, 其中 n_{ij} 是第 j 个簇中第 i 类数据点的数目, n_i 是第 i 个簇中数据点的数目。precision 定义为

$p(i,j)=n_{ij}/n_j$。因此，对于第 i 类和第 j 个簇, F-measure 定义如下:

$$F(i,j)=\frac{2\cdot p(i,j)\cdot r(i,j)}{p(i,j)+r(i,j)} \tag{7.22}$$

因此，有聚类算法所产生的分类的总的 F-measure 定义为:

$$F=\sum_{i=1}^{k}\left(\frac{n_i}{n}\max_j F(i,j)\right) \tag{7.23}$$

n 是数据集的数据点总数。$F\in[0,1]$，F 值越大，算法性能越好。

3) **算法运行时间(execution time)**: 各算法的运行时间在同一台 PC 上运行得到。统计 50 次独立运行的均值和方差。

表 7-3 各算法 F-measure 结果比较

Data sets	GAPS		GAPS-C		DEPS		DEPS-C	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
dataset-01	0.9497	0.0035	**0.9502**	0.0032	0.9480	0.0051	0.9484	0.0034
dataset-02	0.9279	0.0352	0.9206	0.0381	0.9520	0.0230	**0.9568**	0.0189
dataset-03	1	0	1	0	1	0	1	0
dataset-04	0.9344	0.0109	0.9560	0.0112	0.9276	0.0143	**0.9653**	0.0086
dataset-05	1	0	1	0	1	0	1	0
dataset-06	0.9982	0.0012	**1**	0	0.9975	0.0001	**1**	0
dataset-07	1	0	1	0	1	0	1	0
dataset-08	0.9845	0.0103	0.9856	0.0111	0.9987	0.0034	**0.9989**	0.0027
dataset-09	0.8721	0.0030	0.9186	0.0015	0.8787	0.0128	**0.9250**	0.0072
dataset-10	0.9784	0.0002	**0.9980**	0.0000	0.9877	0.0019	0.9944	0.0045
iris	0.8843	0.0180	0.8816	0.0199	**0.8886**	0.0182	0.8825	0.0180
breast	0.9703	0.0032	0.9705	0.0026	0.9724	0.0029	**0.9727**	0.0033
glass	0.5069	0.1078	0.5317	0.0964	0.6772	0.0130	**0.6776**	0.0169
wine	0.6420	0.0339	0.6438	0.0281	0.6391	0.0380	**0.6721**	0.0212
vowel	0.5378	0.0162	**0.5843**	0.0462	0.5658	0.0198	0.5545	0.0336
liver	0.6204	0.0420	0.6251	0.0417	0.6599	0.0383	**0.6791**	0.0040
habermans	0.5980	0.1024	0.6438	0.1137	0.7443	0.0083	**0.7848**	0.0378
cmc	0.4039	0.0096	0.4083	0.0014	0.4008	0.0084	**0.4096**	0.0093

7.4.4 实验结果

本节把四个算法的性能通过实验进行对比。各算法的参数如 7.4.1 节所示，每个算法在每个数据集上独立运行 50 次。表 7-2, 7-3 和 7-4 分别给出了各个算法在适应值, F-measure 和运行时间上的结果。从表中结果可以看出:

1) 总体上，DEPS-C 和 DEPS 能获得比 GAPS 和 GAPS-C 较小的适应值。与其他 3

个算法相比，在大部分数据集上, DEPS-C 能够提供最高的 F-measure 值。这表明 DEPS-C 能够获得最合适的聚类划分。DEPS 在大多数数据集上的运行时间最少。

2) DEPS 与 GAPS 相比, DEPS 在 13 个数据集上得到较小的适应值。此外, DEPS 在 10 个数据集上得到的 F-measure 值高于 GAPS 的值。在 5 个数据集上, GAPS 的 F-measure 值优于 DEPS。在其他三个数据集上，两个算法得到相同的结果。

表 7-4 各算法运行时间(s)结果比较

Data sets	GAPS		GAPS-C		DEPS		DEPS-C	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
dataset-01	8.47	0.07	9.46	0.04	6.08	0.04	**6.06**	0.03
dataset-02	9.24	0.04	9.99	0.07	**6.04**	0.04	6.09	0.03
dataset-03	13.18	0.06	17.79	0.05	**8.59**	0.04	15.63	0.04
dataset-04	13.65	0.13	16.56	0.07	**7.91**	0.17	12.02	0.16
dataset-05	2.32	0.03	2.78	0.02	**1.58**	0.02	1.75	0.02
dataset-06	29.92	0.27	33.25	0.21	**23.09**	0.16	29.45	0.19
dataset-07	19.46	0.10	22.71	0.08	**12.40**	0.09	19.59	0.11
dataset-08	10.18	0.11	13.48	0.09	**7.04**	0.08	6.33	0.08
dataset-09	87.65	0.65	97.27	0.46	**46.32**	0.38	65.46	0.27
dataset-10	64.90	0.36	75.36	0.39	**42.11**	0.35	50.36	0.25
iris	7.66	0.04	9.46	0.04	**5.59**	0.05	5.89	0.04
breast	65.35	0.38	76.11	0.87	**57.23**	0.19	67.70	0.40
glass	52.27	0.39	67.33	0.34	**34.99**	0.48	42.08	0.20
wine	11.55	0.07	14.98	0.11	**7.20**	0.03	7.44	0.02
vowel	91.43	1.12	101.39	0.77	**58.05**	0.53	68.56	2.29
liver	20.23	0.45	23.61	0.17	**14.43**	0.09	16.24	0.06
habermans	8.04	0.04	9.74	0.04	**5.77**	0.05	6.68	0.04
cmc	208.48	2.07	220.79	1.47	**128.75**	1.46	124.06	0.28

3) DEPS-C 和 GAPS-C 相比，其比较结果与 DEPS 和 GAPS 相似，在所有三个评价标准上, DEPS-C 能够提供更优的或具有很强可以比的结果。

4) GAPS 和 GAPS-C 相比，在适应值的结果上两个算法的结果没有明显差距，但是，对于 F-measure, GAPS-C 在 13 个函数中获得了较好的结果。这表明封闭性提高了 GAPS 算法的聚类划分精度。但是，由于加入了封闭性, GAPS-C 的运行时间略高于 GAPS。

5) DEPS 与 DEPS-C 相比，DEPS-C 在适应值上略优于 DEPS 算法。此外，在 F-measure 评价标准上，DEPS-C 算法在 18 个数据集中的 13 个数据集上优于 DEPS 算法。

为了从直观上看出各算法的聚类结果，此处选择 3 个数据集(dataset-04，dataset-09 和 dataset-10)对各算法进行比较，其结果如图 7-4, 7-5 和 7-6 所示. 从图中可以看出 DEPS 和

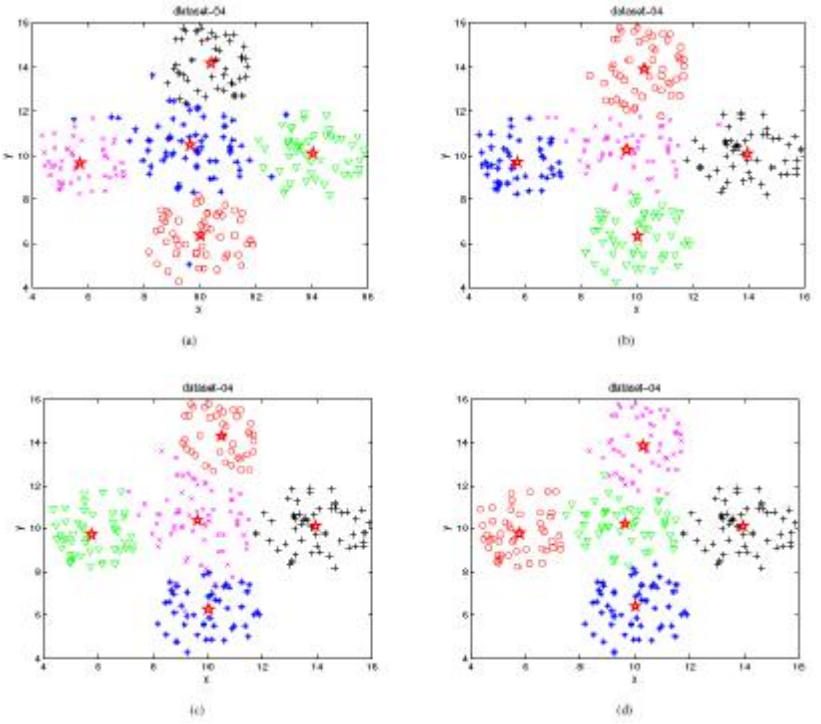

图 7-4 各算法在 dataset-04 上聚类结果. (a) GAPS. (b) GAPS-C. (c) DEPS. (d) DEPS-C. 其中☆表示各簇的聚类中心. 下同. (彩色图形请参考文章的电子文件).

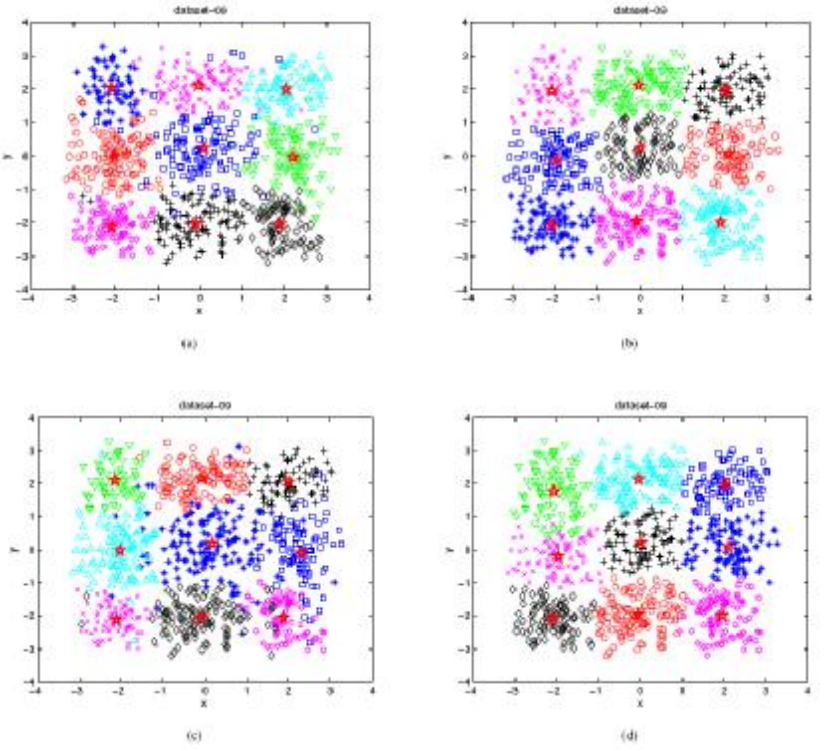

图 7-5 各算法在 dataset-09 上聚类结果.

GAPS 降低了对称的鲁棒性。其原因在于对于大数据集，在公式(7.18)中两个对称点的距离之和趋近于 0，即 $\frac{d_1+d_2}{2}\approx 0$，所以，公式(7.18)所计算的对称距离变得没有意义。然而，在考虑了封闭性之后，这个缺点可以得到弥补，因此, DEPS-C 和 GAPS-C 能够得到更鲁棒的聚类划分结果。这进一步验证了改进的点对称距离分配方法的有效性。

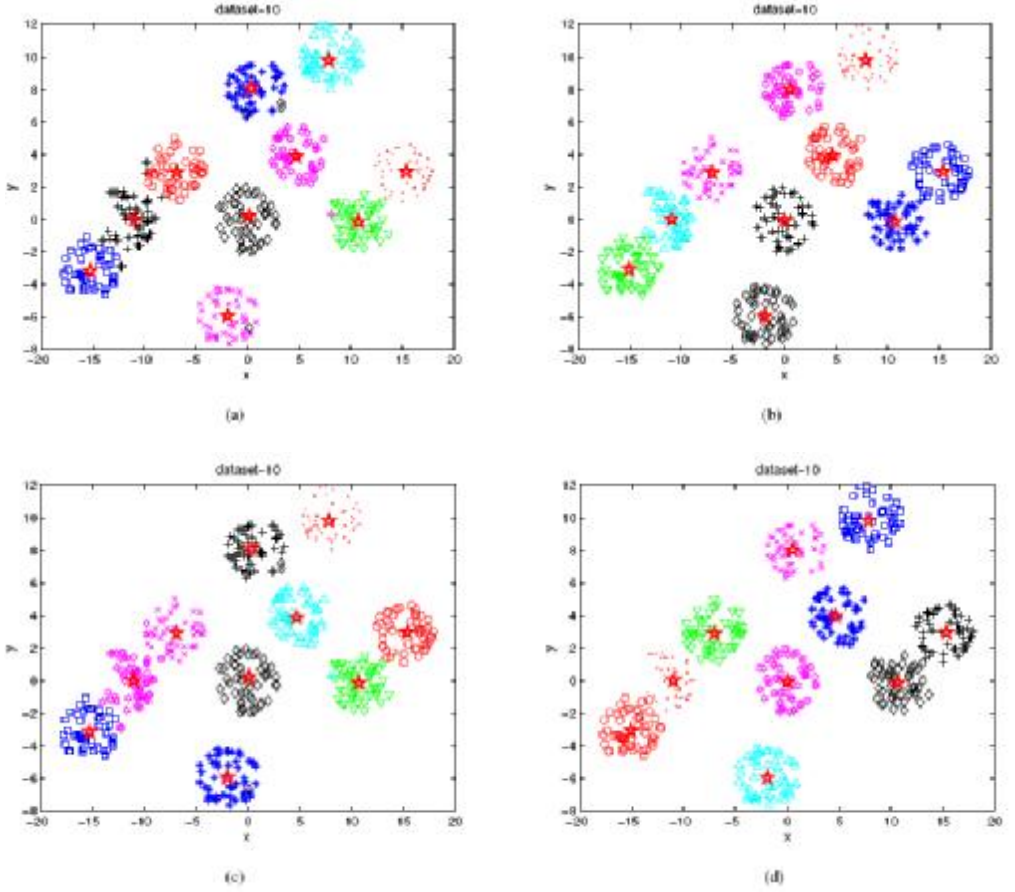

图 7-6 各算法在 dataset-10 上聚类结果.

7.4.5 CR 对算法的影响

文献[12]和[19]中指出当前演化聚类算法所存在一个缺点是: 算法中存在多个控制参数，且参数的设置对不同问题是敏感的。本章所提出的 DEPS-C 算法只有两个控制参数，即群体大小 *NP* 和杂交概率 *CR*。对于群体大小 *NP* 的设置一般随问题的复杂性进行设置。对于杂交概率 *CR*，文献[8,9]中指出，该参数对不同问题的设置是敏感的。为此，本节对杂交概率 *CR* 的不同取值进行实验分析，研究该参数对 DEPS-C 算法的影响。7.4.2 节中的 10 个人工数据集(dataset-01 到 dataset-10)被用来测试，此外, F-measure 被选用作为评价标准。除了 *CR* 取下列不同的值外: $CR=0.05,0.3,0.5,0.7,0.9,0.95$，其他参数按照 7.4.1 节的设置保持不变。图 7-7 和 7-8 显示了在所有人工数据集上 DEPS 和 DEPS-C 算法在不同 *CR* 值设置下的 F-measure 变化曲线。从图中可以明显看出，对于 DEPS 和 DEPS-C 算法，它们的性能对于不同 *CR* 值是不敏

感的。这意味着用户对 *CR* 值的选取是很容易的。

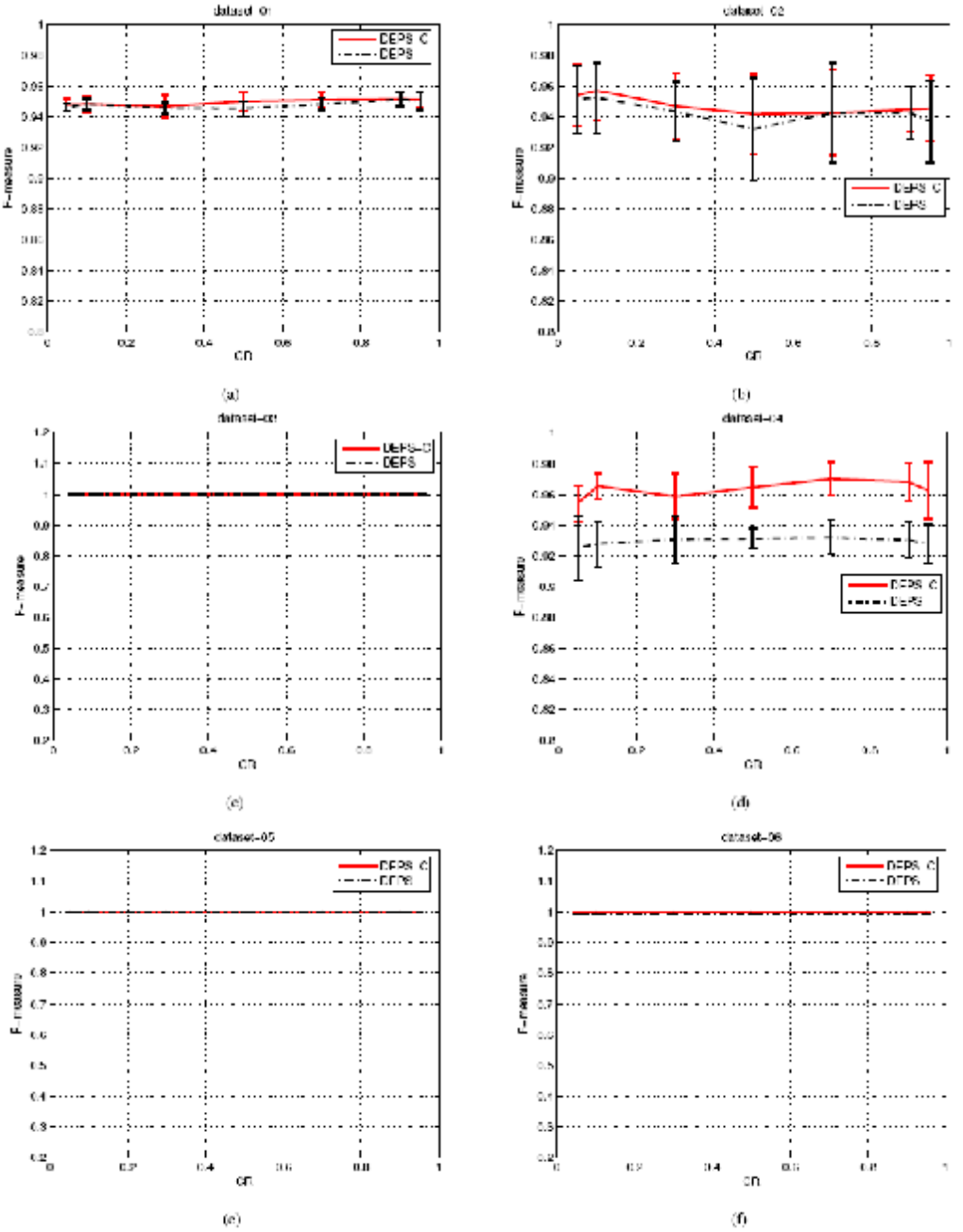

图 7-7 不同 CR 取值对 DEPS 和 DEPS-C 算法性能的影响。(a)到(f)是分别对数据集 dataset-01 到 dataset-06 上 F-measure 的结果。

7.4.6 实验结论

从 7.4.4 和 7.4.5 节的实验结果，我们可以得出如下结论:

1) DEPS-C 算法能够对各种具有对称性的数据集进行有效聚类划分，包括簇内对称和簇外对称的数据集。

2) 与 GAPS 和 GAPS-C 算法相比, DEPS-C 算法在适应值、F-measure 和运行时间上能够获得较好的或具有很强对比性的结果。

3) 封闭性可以增强点对称距离评价标准的鲁棒性。

4) DEPS-C 算法具有很少的控制参数，且杂交概率 *CR* 的不同取值对算法性能的影响很小。因此, DEPS-C 算法是很容易使用的。

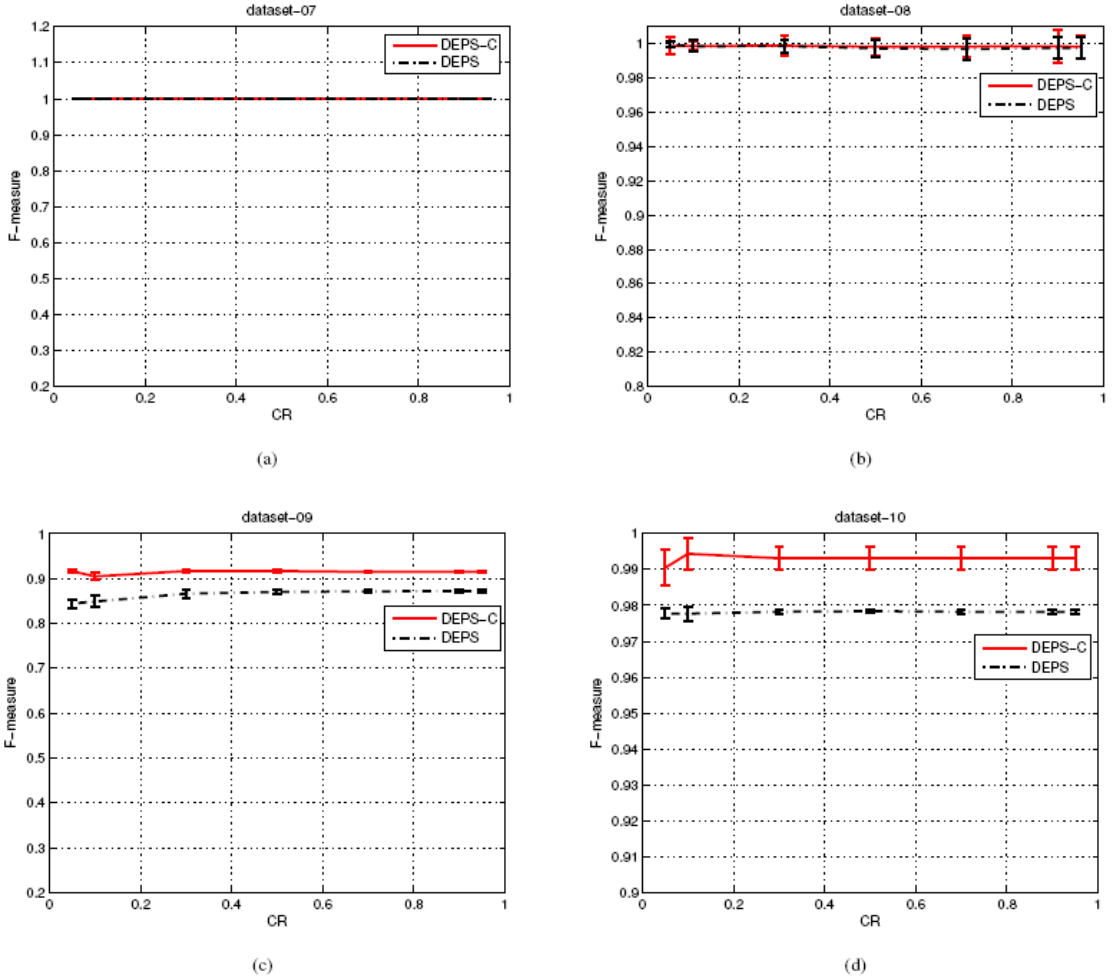

图 7-8 不同 CR 取值对 DEPS 和 DEPS-C 算法性能的影响。(a)到(d)是分别对数据集 dataset-07 到 dataset-10 上 F-measure 的结果。

§7.5 本章小结

差分演化算法是一种快速高效的全局优化算法。本章把差分演化算法与传统的 K 均值聚类算法相结合提出 DEPS-C 算法用于求解聚类分析问题。DEPS-C 算法采用基于点对称距离的标准作为相似性准则，同时，为了增强点对称标准的鲁棒性把封闭性嵌入到该点对称标准中。并且，利用基于 Kd 树的最邻近点搜索方法来降低寻找对称点的复杂度。

为了验证 DEPS-C 算法的有效性，选择了 10 个广泛使用的人工数据集和 8 个 UCI 数据集对算法进行实验分析。此外，DEPS-C 算法还与 GAPS，GAPS-C 和 DEPS 算法进行比较。实验结果表明，DEPS-C 算法在解的质量和稳定性上取得了较好的结果.该算法能够对各种具有对称性的数据集进行有效聚类划分，且算法对参数 CR 的设置是不敏感的。此外，实验结果进一步验证了加入了封闭性的点对称标准能增强原有点对称标准的鲁棒性。

第八章 自动差分演化聚类算法研究

第七章把差分演化算法与传统的 K 均值聚类算法相结合，并采用改进的点对称距离标准作为相似性标准，提出了求解聚类分析问题的 DEPS-C 算法。算法通过人工数据集和 UCI 数据集验证了其有效性。在 DEPS-C 算法中，对各数据集在执行算法之前需要用户给定 K 均值聚类算法中簇的个数。然而，在现实生活中一些数据集预先是很难通过已有知识确定簇的个数的，因此，DEPS-C 算法不能直接应用到此类数据集的聚类中。为此，本章在 DEPS-C 算法的基础上，提出了自动差分演化聚类算法，即 ACDEPS 算法，使算法能够自动确定数据集中簇的个数。为了实现簇个数的自动确定，ACDEPS 算法设计了新的个体表示方法，并提出了改进的基于点对称距离的聚类有效性索(cluster validity index, CVI)引来评价算法所得到划分的有效性。此外，Kd 树同样被用来寻找最邻近对称点以降低算法的复杂度。

本章所研究的聚类问题定义在第七章 7.2.1 节。8.1 节对当前自动聚类算法的一些相关研究进行了简述。8.2 节对本章提出的 ACDEPS 算法的一些关键技术进行详细介绍，包括个体的表示，改进的点对称 CVI 标准和错误个体的修正等。8.3 节通过实验对改进算法的有效性进行验证。最后一节，8.4 节致力于本章工作的小结。

§8.1 相关研究

传统 K 均值算法的一个不足之处在于需要用户事先给定 K 值，这在很多实际应用中是很困难的，用户一般很难给定确定的 K 值。为了克服事先给定 K 值的不足，一些学者利用演化算法的自适应能力设计了演化自动聚类算法[15,29,163,213,226,229-231]，利用演化算法自适应的找到合适的簇的个数和以及相对应合适的划分。对于演化自动聚类算法最为关键的技术在于: 1) 个体编码的设计; 2) 适应值的计算，适应值的计算主要涉及到聚类有效性索引的选取。因此，本节主要讨论这两个关键技术。

8.1.1 个体编码

与 K 值确定是相比，在演化自动聚类中个体的编码要复杂很多，下面主要讨论几种常见的个体编码方法。

1) Bandyopadhyay 和 Maulik 在文献[229]提出一种基于实数和字符#的个体编码方法，其中实数表示簇的中心，而字符#只是用于占据基因位数，不表示簇的中心。在文献[229]中，每个个体具有固定的长度 $K_{\max}\times d$，当簇的个数小于 $K_{\max}$ 时，其他的基因位用字符#代替，但每个个体至少具有 $K_{\min}=2$ 个簇.例如: 设 $K_{\max}=10$，$d=2$，其中一个个体具有 4 个簇，簇的中心点为(0,1), (2,5), (3,3), (6,2)，则这个个体可以表示为##(0,1)#(2,5)(3,3)#(6,2)##。文献[226]

也采用了这种编码方法。

2) 文献[230]提出了一种可变长度个体表示方法，每个个体的长度随所表示簇的给数而改变，个体采用实数编码。如果一个个体 i 具有 M_i 个簇，则这个个体的基因长度为 $M_i \times d$，每个中心视为不可分割的，即在杂交算子中不能从簇的中心处分开。每个个体最大表示 $K_{\max}$ 个簇，最小表示 $K_{\min}=2$ 个簇。此中编码方法同样被文献[213]中采用。

3) Das 等在文献[15,231,232]中提出一种固定长度实数编码方法，每个个体除了表示簇中心的基因外，还有用于控制簇是否激活的基因段。设个体所表示的最大簇个数为 $K_{\max}$，则每个个体的长度为 $K_{\max}+K_{\max}\times d$，其中前 $K_{\max}$ 个基因为[0,1]之间的实数，表示激活基因，后面 $K_{\max}\times d$ 个基因用于表示簇的中心。其个体表示可以用图 1 表示如下:

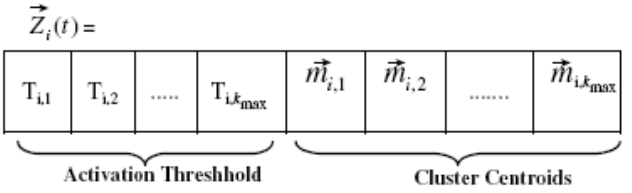

图 8-1 文献[15,231,232]的个体表示方法

在图 8-1 中，当 $T_{i,k}>0.5$ 时，则表示相对应的第 k 个簇是激活的，否则，这个簇在当前群体中不激活，不表示聚类中心。如图 8-2 表示最大为 5 个簇 3 维的个体，其有效簇中心为(4.4,7,6.5),(9.6,5.3,4.2)和(8,4,4)。

0.3	**0.6**	**0.8**	0.1	**0.9**	6.1	3.2	2.1	**6**	**4.4**	**7**	**9.6**	**5.3**	**4.2**	5	8	4.6	**8**	**4**	**4**

图 8-2 文献[15,231,232]的个体示例说明

8.1.2 聚类有效性索引

聚类有效性索引(Clustering validity index, CVI)是对聚类结果进行量化评价的一种方法，其主要有两个目的: 1) 确定数据集中簇的个数; 2) 找出相对应簇个数的最有划分。理想情况下，聚类有效性索引应该具有如下两个特点:

1) **紧凑性(Cohesion)**: 即同一个簇中的模式(数据点)要尽可能相似;

2) **分离性(Separation)**: 不同簇之间应该要很好的区分开。

目前，在聚类的研究中已经提出了很多种聚类有效性索引，其中比较有名的有 Dunn's index (DI)[232], Calinski-Harabasz index (CH)[233], DB index[234], PBM index[235], CS index[236]和 Sym-index[226]等。下面主要介绍 DB index, PBM index, CS index 和 Sym-index。

(1) DB index

DB index[234]是簇内聚类之和(sum of the within-cluster scatter)与簇间距离(between-cluster separation)之间的一个比值。首先定义第 i 个簇的簇内距离为:

$$S_{i,q}=\left[\frac{1}{N_i}\sum_{\mathbf{x}\in C_i}\left\|\mathbf{x}-c_i\right\|_2^q\right]^{1/q} \tag{8.1}$$

再定义第 i 个簇和第 j 个簇之间的距离为:

$$d_{ij,t}=\left\{\sum_{p=1}^{d}|c_{i,p}-c_{j,p}|^{t}\right\}^{1/t}=\left\|c_i-c_j\right\|_t \tag{8.2}$$

其中 c_i 为第 i 个簇的中心，$q,t\geq 1$，q 是整数，q 和 t 独立选取。N_i 是第 i 个簇 C_i 的元素个数。

随后，$R_{i,qt}$ 定义为:

$$R_{i,qt}=\max_{j\in K,j\neq i}\left\{\frac{S_{i,q}+S_{j,q}}{d_{ij,t}}\right\} \tag{8.3}$$

最后，定义 DB index 如下:

$$\mathrm{DB}(K)=\frac{1}{K}\sum_{i=1}^{K}R_{i,qt} \tag{8.4}$$

显然，$\mathrm{DB}(K)$ 值越小表示聚类性能越好。

(2) PBM index

PBM index 是由 Pakhira 等在文献[235]中提出的，其定义为:

$$\mathrm{PBM}(K)=\left(\frac{1}{K}\times\frac{E_1}{E_K}\times D_K\right)^2 \tag{8.5}$$

其中 K 是簇的个数,

$$E_K=\sum_{k=1}^{K}E_k \tag{8.6}$$

$$E_k=\sum_{j=1}^{n}u_{kj}\left\|\mathbf{x}_j-C_k\right\| \tag{8.7}$$

$$D_K=\max_{i,j=1}^{K}\left\|C_i-C_j\right\| \tag{8.8}$$

N 是数据集中点的个数，$U(X)=[u_{kj}]_{K\times N}$ 是划分矩阵。$\mathrm{PBM}(K)$ 的越大表示聚类性能越好。

(3) CS index

最近, Chou 等提出了 CS index[236]用于评价聚类算法的有效性。在计算 CS index 值之前，需要先计算每个簇的中心点:

$$c_i = \frac{1}{N_i}\sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j \tag{8.9}$$

用 $d(\mathbf{x}_i, \mathbf{x}_j)$ 表示两个数据点 $\mathbf{x}_i$ 和 $\mathbf{x}_j$ 之间的距离，则 CS index 定义为:

$$\begin{aligned} \mathrm{CS}(K) &= \frac{\frac{1}{K}\sum_{i=1}^{K}\left[\frac{1}{N_i}\sum_{X_i \in C_i}\max_{X_q \in C_i}\left\{d(\mathbf{x}_i, \mathbf{x}_q)\right\}\right]}{\frac{1}{K}\sum_{i=1}^{K}\left[\min_{j\in K, j\neq i}\left\{d(c_i, c_j)\right\}\right]} \\ &= \frac{\sum_{i=1}^{K}\left[\frac{1}{N_i}\sum_{X_i \in C_i}\max_{X_q \in C_i}\left\{d(\mathbf{x}_i, \mathbf{x}_q)\right\}\right]}{\sum_{i=1}^{K}\left[\min_{j\in K, j\neq i}\left\{d(c_i, c_j)\right\}\right]} \end{aligned} \tag{8.10}$$

以 DB index 类似, CS index 也是簇内聚类之和(sum of the within-cluster scatter)与簇间距离(between-cluster separation)之间的一个比值。CS index 在处理不同密度和不同大小数据时的聚类性能要优于其他聚类有效性索引[236]，但是，此索引的计算复杂度随数据集中点的个数和簇的个数增加很快[15]。

(4) *Sym*-index

Sym-index 是文献[226]中提出的一种基于点对称距离的 CVI。该索引与 PBM index 类似，其定义如下:

$$Sym(K) = \left(\frac{1}{K} \times \frac{1}{x_K} \times D_K\right) \tag{8.11}$$

其中 K 是簇的个数,

$$x_K = \sum_{k=1}^{K} x_k \tag{8.12}$$

$$x_k = \sum_{j=1}^{n_k} d_{ps}^{*}(\mathbf{x}_j^k, c_k) \tag{8.13}$$

且

$$D_K = \max_{i,j=1}^{K} \left\| c_i - c_j \right\| \tag{8.14}$$

其中 K 为簇的个数，D_K 是簇之间的最大欧氏距离，$d_{ps}^{*}(\mathbf{x}_j^k, c_i)$ 按第七章中公式(7.18)在一定约束下所计算得的对称距离。这里，$\mathbf{x}_j$ 关于 c_i 的对称点 $2\times c_i - \mathbf{x}_j$ 的第一和第二邻近点只能在第 i

个簇中寻找。当公式(8.11)达到最大值时，得到数据集的实际簇的个数。有关 *Sym*-index 的详细信息见文献[226]。

§8.2 自动差分演化聚类算法

在前几章中已经说明了差分演化算法是一种有效且通用的全局优化算法，基于差分演化算法的聚类算法能够获得比基于遗传算法的聚类算法更好的性能[31]。在上一章 DEPS-C 算法的基础上，本章提出一种基于差分演化算法的自动聚类算法，即 ACDEPS 算法，其关键改进指出在以下几个小节中进行叙述。

8.2.1 个体表示

为了能使差分演化算法应用与自动聚类中，对个体的表示方法必须进行改进。在 ACDEPS 算法中，采用文献[15]中所提出的个体表示方法。对具有 n 个点的数据集，每个数据点有 d 维，每个个体所能包含的最大簇个数为 $k_{\max}=\sqrt{n}$ [213]，ACDEPS 的个体是一个 $k_{\max}+k_{\max}\times d$ 维的实数向量。群体中第 i 个个体 X_i 表示如下:

$$X_i=(T_{i,1},T_{i,2},\mathbf{L},T_{i,k_{\max}},c_{i,1},c_{i,2},\mathbf{L},c_{i,k_{\max}})^T \tag{8.15}$$

其中前 $k_{\max}$ 个实数 $T_{i,1},\mathbf{L},T_{i,k_{\max}}\in[0,1]$ 用来控制其相应的簇中心是否有效。只有当 $T_{i,j}>0.5$，表示第 i 簇的第 j 个簇中心是有效的。$c_{i,j}$ 是 d 维第 i 簇的第 j 个簇中心向量。

8.2.2 改进的点对称 CVI 标准

从公式(8.11)可以看出，*Sym*-index 由三部分组成，$1/K$，$1/x_K$ 和 D_K。其中第一部分随 K 值的增加而减小。第二部分和第三部分随着 K 值的增加而增加。由于 *Sym*-index 在最大化时获得最优的聚类划分，第二和第三部分均倾向于较多的簇数目，仅有第一部分倾向于较少的簇数目。因此，在演化初期，个体倾向于找出更多的簇。为了使算法能较快得到最优的簇数目，从而得到最后的聚类划分，在 ACDEPS 算法中，我们提出一种改进的基于点对称聚类的 CVI，其中，聚类数目 K 被动态惩罚。该方法描述为:

$$Sym'(K)=\left(\frac{1}{K'}\times\frac{1}{x_K}\times D_K\right) \tag{8.16}$$

其中 $1/x_K$ 和 D_K 分别按公式(8.12)和(8.14)计算。K' 定义为:

$$K'=K^{2^t} \tag{8.17}$$

且

$$t=1.0-a\times\frac{gen}{G_{\max}} \tag{8.18}$$

其中，gen 是当前演化代数，$G_{\max}$ 是最大演化代数。$a\geq 1$ 是常数，它用于对个体中簇的数目 K

进行动态惩罚。$a=1$表示K一直被惩罚，除了演化的最后一代。$a=2$表示当$gen<0.5\times G_{\max}$时K被惩罚，而当$gen\geq 0.5\times G_{\max}$时，对$K$不做惩罚，即个体中所包含的簇的个数就是计算中的簇个数。基于此动态惩罚的 *Sym*'-index，算法能避免在演化初期找出太多的簇个数，从而影响后面的演化过程。

8.2.3 错误个体的修正

与 DEPS-C 算法中对错误个体的修正类似，ACDEPS 算法在计算 *Sym*'-index 是需要寻找第一和第二对称点，本研究采用基于 Kd 树的最邻近点搜索技术来找出这两个点，因此，在每个簇中至少应该包含两个数据点。对于一个个体，如果其中一个簇中所包含的数据点少于两个，则对此个体进行重新初始化，即从数据集中随机选择k个数据点作为簇的中心。然后，对此个体重新进行数据点的分配。

此外，在利用差分演化策略产生新个体之后，如果$T_{i,j}\notin[0,1]$则采用随机的方法是该机在[0,1]范围之内。如果$T_{i,j}>0.5$的个数少于 2 时，则随机选取两个$T_{i,j}$，并把它们在[0.5, 1.0]范围内随机初始化。这样，可以保证每个个体至少包含两个簇。

§8.3 实验结果与分析

为了对算法性能进行评估，ACDEPS 算法对 *Sym*'-index 和 *Sym*-index 进行了测试，因此，这里有两种方法，称为 ACDEPS1 和 ACDEPS2。此外，把 ACDEPS 算法与 GUCK 算法[229]进行比较[1]。为了比较的公平性，在 CUGK 算法中也同时采用了 *Sym*'-index 和 *Sym*-index，算法被称为 GUCK1 和 GUCK2。

8.3.1 参数设置

在 ACDEPS 算法中，有四个参数需要实现给定: 1) 群体大小 *NP*; 2) 杂交概率 *CR*; 3) 最大演化代数$G_{\max}$; 4) 惩罚控制参数a。对于差分演化算法中的缩放因子 F 采用与 DEPS-C 算法类似的方法产生，即采用 dither 技术在[0,1]之间随机产生。在所有实验中，采用如下参数设置。

- 群体大小：$NP=100$;
- 杂交概率：$CR=0.3$;
- 差分演化策略: DE/rand/1/bin;
- 惩罚控制参数：$a=2.0$;
- 最大演化代数：$G_{\max}=30$，即 3000 次适应值评价。

对于 GUCK 算法，其参数设置与文献[229]中的参数设置一致。对每个数据集，算法独立

[1] 由于我们不能得到 VGAPS 算法[226]的代码，本研究中不与该算法进行对比。

运行 10 次。所用算法采用标准 C++语言实现，测试环境是: CPU: P-IV (Core 2) 2.1 GHz; 内存: 1 GB; 操作系统: MS Windows XP。

表 8-1 各算法 F-measure 结果比较。**黑体**表明算法获得较好的解。下同。

Data	ACDEPS1			ACDEPS2			GUCK1			GUCK2		
	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR
data1	**0.982**	0.015	**1.0**	0.084	0.266	0.1	0.492	0.519	0.5	0.000	0.000	0.0
data2	**0.942**	0.021	**1.0**	0.645	0.453	0.7	0.762	0.402	0.8	0.278	0.448	0.3
data3	0.999	0.001	**1.0**	**1.000**	0.000	**1.0**	0.300	0.482	0.3	0.100	0.316	0.1
data4	**1.000**	0.000	**1.0**	0.300	0.483	0.3	**1.000**	0.000	**1.0**	0.899	0.316	0.9
data5	**1.000**	0.000	**1.0**	0.800	0.242	0.0	**1.000**	0.000	**1.0**	0.900	0.316	0.9
data6	0.365	0.472	0.4	**0.699**	0.482	**0.7**	0.197	0.415	0.2	0.690	0.477	0.6

表 8-2 各算法所得簇的个数结果比较。

Data	ACDEPS1			ACDEPS2			GUCK1			GUCK2		
	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR	Mean	Std	SR
data1	**2.000**	0.000	**1.0**	4.200	2.201	0.1	3.100	1.663	0.5	7.200	1.229	0.0
data2	**3.000**	0.000	**1.0**	2.700	0.483	0.7	3.200	0.422	0.8	5.600	1.897	0.3
data3	**3.000**	0.000	**1.0**	**3.000**	0.000	**1.0**	4.400	0.966	0.3	4.800	0.632	0.1
data4	**4.000**	0.000	**1.0**	2.600	0.966	0.3	**4.000**	0.000	**1.0**	4.100	0.316	0.9
data5	**6.000**	0.000	**1.0**	5.800	0.323	0.0	**6.000**	0.000	**1.0**	6.100	0.316	0.9
data6	5.800	0.789	0.4	**5.200**	0.422	**0.7**	7.000	1.333	0.2	4.100	1.449	0.6

8.3.2 实验数据集

为了验证算法的有效性，本研究从已有文献中选取 6 个人工数据集，这些数据集简要描述如下:

- data1: 选自文献[226]，由两个相交的椭圆形组成，每个椭圆形含有 200 个数据点。
- data2: 选自文献[214, 222, 226]，它包含 350 个数据点，由环形簇，紧凑簇和线型簇组成。
- data3: 选自文献[214]，它包含 400 个点和 3 个簇，由环形簇，矩形簇和线型簇组成。
- data4: 选自文献[214, 226]，包含 400 个点，在 3 维空间由 4 个不连接的椭球体组成。
- data5: 选自文献[214,226]，包含 300 个点，由 6 个簇组成。
- data6: 选自文献[226]，包含 850 个点，分布在 5 个簇中。

8.3.3 评价标准

本研究选取 3 个评价标准对本章的 4 个算法进行对比，这 3 个评价标准描述如下:

- **F 评价标准(F-measure)**: 见第七章 7.4.3 节。
- **簇的个数(Number of clusters)**: 算法最优解所包含的簇的数目，统计 10 次独立运行中的均值和方差。

- **成功率(Successful rate, SR)**: 如果算法能够找出数据集中的正确簇的个数，则该算法在此次运行中是成功的。SR 值是成功运行次数除于总的运行次数。

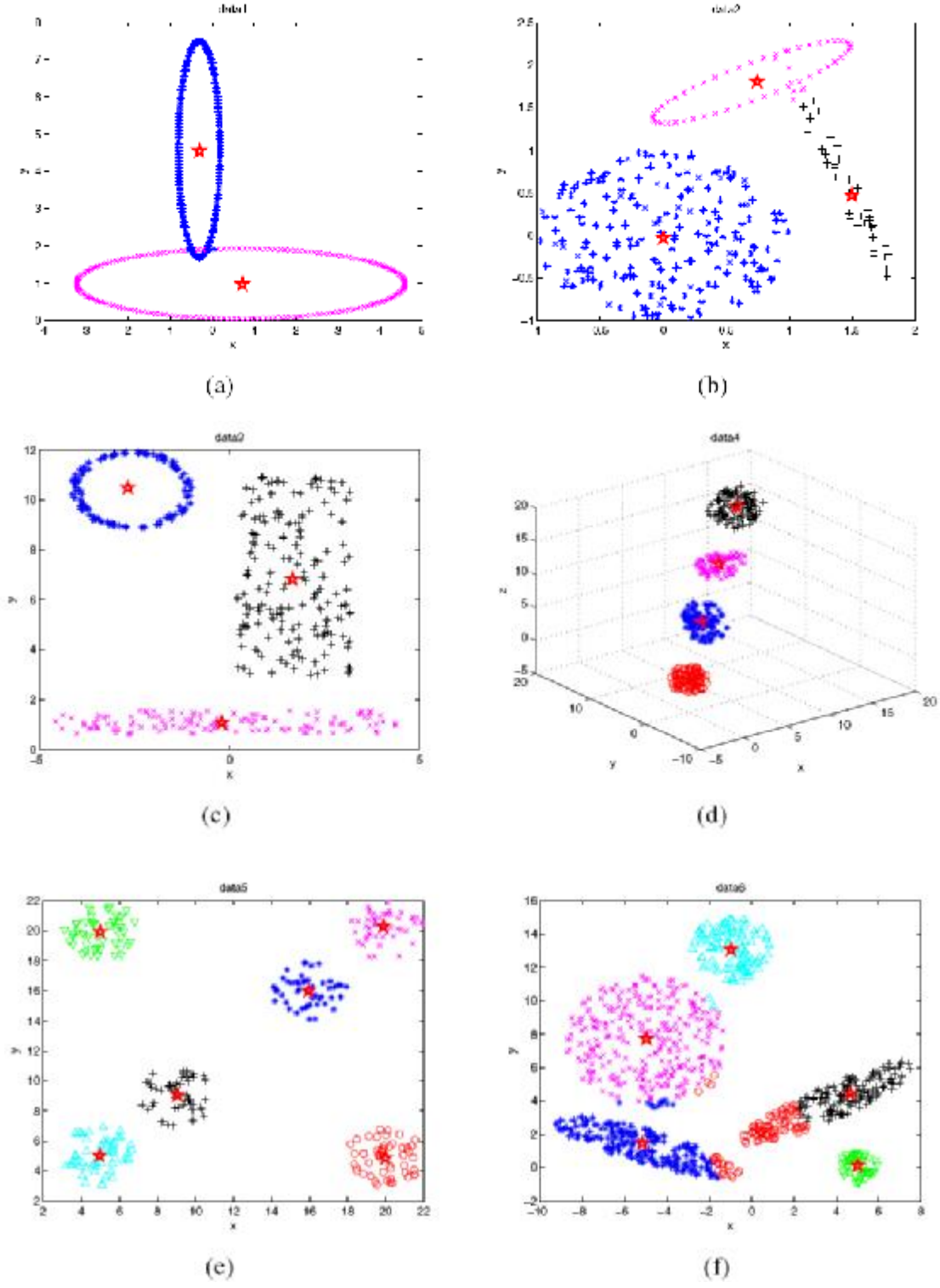

图 8-1 ACDEPS1 算法在所有数据集上的聚类结果。(a)到(f)分别是 data1 到 data6。☆表示各簇的中心。

8.3.4 实验结果

本小节把 4 个算法进行实验对比，各算法的参数如 8.3.1 节所示。算法在每个数据集独立运行 10 次，其实验结果如 表 8-1 和表 8-2 所示。算法 ACDEPS1 的一些典型的聚类结果图如图 8-1 所示。

从表 8-1 可以看出，ACDEPS1 在 5 个测试数据集(data1 – data5)中能每次都找出正确的簇的个数，且在这些数据集上平均 F-measure 值接近 1.0。这表明 ACDEPS1 算法能对这些数据集进行接近最优的划分。ACDEPS1 与 ACDEPS2 相比，可以明显看出 ACDEPS1 在 F-measure 和成功率上均优于 ACDEPS2 算法，这说明了 *Sym*'-index 在 ACDEPS 算法中优于 Sym-index。ACDEPS 与 GUCK 相比(即 ACDEPS1 vs GUCK1, ACDEPS2 vs GUCK2)，实验结果表明在大部分数据集上 ACDEPS 算法的性能优于 GUCK 算法的性能。GUCK 算法只在 data6 数据集上优于 ACDEPS 算法。这说明基于差分演化算法的聚类算法性能优于基于遗传算法的聚类算法，这与文献[31]的结论相似。

从表 8-2 中可以得知，对于 5 个数据集(data1 – data5)，ACDEPS1 算法能够在所有的运行中都找出最优的簇的个数及其最优的划分。但是，对于 data6 数据集，ACDEPS1 找出了 6 个簇，多于原本数据集中的 5 个簇。

此外，表 8-1 和 8-2 中 ACDEPS1 的结果在图 8-1 中也可以明显看出。

总的来说，对于本研究中大部分数据集，ACDEPS1 算法能自动确定数据集的最优簇的个数，并找出其相应最优的聚类划分，只要数据集具有对称性的特征。而且，本章提出的改进的基于点对称的 CVI 标准 *Sym*'-index 在 ACDEPS 算法中能增强原始 *Sym*-index 的性能。

§8.4 本章小结

本章结合差分演化算法的有效性，将其应用到自动聚类分析中，提出基于差分演化算法的自动聚类算法，即 ACDEPS 算法。为了能使算法对给定数据集进行有效地自动聚类，算法设计了有效的个体表示方法，提出了改进的基于点对称距离的聚类有效性索引 *Sym*'-index。*Sym*'-index 能避免算法在演化初期找出过多的簇，从而影响演化朝有利的方向进行。此外，基于 Kd 树的最邻近点搜索算法被用于在计算 *Sym*'-index 时快速寻找最邻近对称点。

为了验证算法的有效性，实验选择了 6 个人工数据集，并把 ACDEPS 算法分别采用 *Sym*'-index 和 *Sym*-index 进行实验对比分析，同时，与文献[229]中提出的 GUCK 算法进行对比。实验结果表明，ACDEPS1(采用 *Sym*'-index 的 ACDEPS 算法)对于具有对称性特征的数据集能自动确定簇的正确个数，且能找到其对应的最优聚类划分。并且，实验结果还验证了改进的基于点对称的 CVI 标准 *Sym*'-index 在 ACDEPS 算法中能增强原始 *Sym*-index 的性能。此外，把 ACDEPS 算法与 GUCK 算法进行对比，得出了与文献[31]类似的结论，即基于差分演化算法的聚类算法性能优于基于遗传算法的聚类算法。

在 *Sym*'-index 中，利用控制参数 a 对个体簇的数据进行动态惩罚，该参数与问题相关，今后的工作将致力于对该参数进行研究。

第九章 全文总结及今后工作

§9.1 全文总结

差分演化算法是一种快速、简单、高效的全局优化算法，该算法已成功应用于多个研究领域。本文首先对差分演化算法进行了分析，并总结出算法存在的不足之处。然后，对当前改进的差分演化算法进行了综述，在此基础之上，重点研究了基于生物地理学优化算法的混合差分演化算法、多策略自适应差分演化算法、基于 ε 占优的正交多目标差分演化算法、多目标差分演化算法在工程优化中的应用、给定 K 值的差分演化聚类算法以及自动差分演化聚类算法等。本文工作的主要创新点如下:

(1) 提出了一种基于生物地理学优化算法(BBO 算法)的混合差分演化算法，DE/BBO 算法。算法中提出了一种混合移动算子，该算子把差分变异算子和 BBO 算法的移动算子有效结合，既保持了差分变异算子开采能力，又把 BBO 算法移动算子能有效利用群体信息优点结合起来。因此，该算子能有效平衡算法的开采(exploration)能力和利用(exploitation)能力。此外，该混合算子是一种通用的算子，它能增强差分演化算法其他变异算子的性能。基于该混合算子，提出了 DE/BBO 算法。通过大量实验分析，验证了改进算法的有效性能。

(2) 针对传统差分演化算法在差分演化策略选择困难的缺点，提出了一种通用、简单的个体策略产生机制，该机制通过一个策略控制参数来选择目标个体的差分演化策略。由于该参数是一个在[0,1]之间的实数，演化算法中很多参数控制方法均能很容易地用于对其进行更新。为此，实现了三种自适应策略选择方法，并把这三种方法与 JADE 算法相结合，将其应用于标准测试函数的优化中。通过实验分析，表明所提出的自适应多策略选择方法的有效性。在不需要任何先验知识情况下，自适应多策略选择方法能自动从策略库中选择求解当前问题最优的策略，从而增强了算法的性能。此外，本文对所提出算法的算法复杂度和简单性进行了简要分析。

(3) 利用差分演化算法求解多目标优化问题是当前的一大研究热点。本文提出了一种基于 ε 占优的正交多目标快速差分演化算法，ε -ODEMO 算法。算法采用正交实验设计方法来产生初始群体，从而使群体中的个体能在决策变量空间上均匀分布，增加找出较好个体的可能性，而这些好的个体可以在演化过程中得到利用，并加快算法的收敛。利用 Archive 群体来保存演化过程中找出的非劣解，使得算法始终可以保存精英解。为了对 Archive 群体进行有效更新，保证非劣解集的收敛性和分布的多样性，采用基于 ε 占优技术的群体更新方法。提出了一种混合选择策略，对差分变异算子中的基向量采用混合选择机制选取。该混合选择策略在演化初期强调算法对搜索空间的开采，寻找新的搜索空间，而在演化中后期则强调对 Archive 群体中个体信息的利用，加速算法的收敛。本文通过多个 2 目标和 3 目标标准测试

函数对算法的有效性进行了验证；同时，对混合选择策略的控制参数和杂交概率进行了初步实验分析。

(4) 在 ε -ODEMO 算法的基础上，针对带有约束条件的多目标工程优化问题，提出了一种适合求解该类问题多目标差分演化算法，*pa* ε -ODEMO 算法。该算法是对 ε -ODEMO 算法的改进，主要改进为：1) 采用自适应 ε 占优技术代替 ε -ODEMO 算法中原有的 ε 占优技术，从而弥补了传统 ε 占优技术容易丢失一些重要的非劣解的缺点；2) 采用基于约束 Pareto 占优的约束函数处理技术，增强算法处理带约束的多目标问题的能力。*pa* ε -ODEMO 算法在约束多目标标准测试函数和工程优化问题中进行测试，并通过与 NSGA-II 算法进行比较，验证了改进算法在处理带约束多目标问题的有效性。

(5) 提出了一种基于点对称距离的差分演化聚类算法——DEPS-C 算法。该算法把差分演化算法与传统 K 均值算法相结合，采用点对称距离作为相似性准则来对个体进行分配，同时，为了增强该对称性规则的鲁棒性，在对原有点对称距离的计算时考虑了封闭性原则。并且，利用基于 Kd 树的最邻近点搜索方法来降低寻找对称点的复杂度。实验结果表明，对于具有对称性的数据集 DEPS-C 算法能对其进行正确的聚类划分；且该算法控制参数少，对杂交概率 *CR* 的不同取值不敏感，这增加了该算法的实用性。

(6) 尽管 DEPS-C 算法具有良好的性能，但是，该算法在对给定数据集进行聚类之前需要用户给定簇的准确个数，这在很多实际问题中是很困难的。为此，在 DEPS-C 算法基础上提出了自动差分演化聚类算法(ACDEPS 算法)，使算法能通过演化自动确定簇的个数，并得到相应最优的聚类划分。在 ACDEPS 算法中，采用了有效的个体表示以满足差分演化算法进行自动聚类的需求；同时，提出了改进的基于点对称距离的聚类有效性验证索引 *Sym*'-index，利用对个体所包含簇个数进行动态惩罚来避免算法在演化初期找出过多的簇数目，从而影响算法的收敛。通过实验验证了：1) 采用 *Sym*'-index 的 ACDEPS 算法对于具有对称性特征的数据集能自动确定簇的正确个数，且能找到其对应的最优聚类划分；2) 改进的基于点对称的 CVI 标准 *Sym*'-index 在 ACDEPS 算法中能增强原始 *Sym*-index 的性能；3) 基于差分演化算法的聚类算法性能优于基于遗传算法的聚类算法。

§9.2 今后工作展望

本文对差分演化算法进行了研究，包括对基本差分演化算法的改进，与多目标处理技术相结合提出适合求解多目标问题的差分演化算法，以及利用差分演化算法进行聚类分析等。并且通过实验的方法对改进算法进行了有效性验证，并提出了自己的见解。但是，由于涉及该算法的研究内容很广泛，还有很多问题值得进一步研究。结合当前工作，今后的研究工作主要在以下几个方面展开:

(1) 目前，对差分演化算法的研究主要是通过实验的方法，对算法的理论研究还有待进一步深入探讨，包括算法的收敛性证明，差分变异算子搜索原理的分析，控制参数 *CR* 和 *F* 对收敛性影响的理论分析等。在今后的工作中将尝试利用随机过程理论对其进行理论研究.

(2) 结合机器学习理论，通过学习的方法来实现差分演化策略的自适应选择。本文第四

章对自适应多策略选择进行了一定的研究，今后的研究工作将在此基础上把机器学习方法(如概率匹配、自适应追逐、Multi-armed bandit 算法等)与差分演化算法相结合，提出基于机器学习的自适应多策略差分演化算法，进一步增强算法的性能。

(3) 把差分演化算法应用到离散优化问题中是将来的研究方向之一。由于许多离散优化问题被证明是 NP 难问题，且当前对离散差分演化算法的研究善处于初步阶段，因此，设计有效的机制使差分演化算法能应用于该类问题的求解中，不仅可以扩展差分演化算法的应用范围，而且可以为求解离散优化问题提供一种可行的选择。

致 谢

首先衷心感谢我的导师蔡之华教授。蔡老师在我六年的研究生学习中，对我的学习、生活、科研等给予了极大的帮助和悉心的指导。蔡老师学识渊博且治学严谨，具有深厚的学术功底，他严谨求实、锐意创新的学术精神使我受益匪浅。蔡老师不仅在学术上有很深的造诣，而且为人正直，宽厚仁和，我从他那里学到的不仅仅是学问，而且还有做人的道理。在我的科研中，蔡老师对我的每一篇论文进行了仔细的修改，提出宝贵的建议。在蔡老师的大力支持和推荐下，我有机会在加拿大西安大略大学进行深造，对我的学习和科研有了很大的帮助。我为能够成为蔡老师的学生而深感荣幸。

感谢我的联合培养导师 Charles Ling 教授。Ling 老师对我在加拿大期间的生活和科研上给予了大力帮助与支持，使我除在科研上取得了成果之外，增长了见识和研究水平。

感谢国家留学基金委的大力资助，使我有机会在加拿大西安大略大学计算机系进行留学访问。通过 15 个月的国外学习和生活，对我在英语水平和科研能力上都有了很大的提高。

感谢中国地质大学(武汉)优秀博士论文创新基金的资助，使我能够顺利地完成我的博士毕业论文。

感谢Y.-S. Ong教授、D. Simon教授、J. Brest教授、P. N. Suganthan教授、A.G. Hernández-Díaz博士、A. Fialho博士、孙怿博士、王勇博士等国内外知名专家学者对我在科研中的帮助。

感谢我的师兄颜雪松博士、蒋良孝博士、杜均博士、李曲博士对我在生活和科研中的帮助。

感谢我的同窗好友蒋思伟、郭嫄嫄、谷琼、刘小波、罗治情、杨鸣、邝达、倪爱玲、李骁、陈瀛等的帮助，在我攻读博士学位期间，我们相互沟通和交流，对我在生活和科研上均有很大的帮助，让我受益匪浅。

感谢计算机学院李晖书记、刘亚东书记和计算机学院其他老师对我的帮助与支持。

感谢我的母校中国地质大学（武汉），在这十年的学习生涯中，在校训“艰苦朴素、求真务实”的感召下，我逐渐成为了一个对社会有用的人。在此期间我学到了很多有用的知识与技能以及懂得了做人的道理。

深深感谢我的父母对我的关爱与支持。感谢我的妻子朱翠云女士。正是他们不懈地支持、鼓励和无怨无悔地付出才使我能够安心顺利地完成学业。

感谢所有关心和支持我的人，对你们表示由衷的祝福！

最后，感谢所有参与论文评审的各位专家，感谢你们在百忙之中对本论文的批评指正和建议，这对本论文的改进有极大的帮助。

参考文献

[1] 潘正君，康立山，陈毓屏．演化计算．北京：清华大学出版社，广西科学技术出版社，2000.

[2] T. Bäck, U. Hammel, and H-P. Schwefel, Evolutionary computation: Comments on the history and current state, IEEE Transactions on Evolutionary Computation, 1997, 1(1): 3-17.

[3] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford Univ. Press, 1996.

[4] R. Storn and K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 1997, 11: 341 – 359.

[5] K. Price, R. Storn, and J. Lampinen, Differential evolution: A practical approach for global optimization, Berline, Springer-Verlag, 2005.

[6] R. Storn and K. Price, Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report: TR-95-012, 1995.

[7] N. Noman and H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Transactions on Evolutionary Computation, 2008, 12(1): 107 – 125.

[8] J. Brest, S. Greiner, B. Bošković, *et al.*, Self-adapting control parameters in differential evolution: A comprehensive study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646 – 657.

[9] R. Gäperle, S.D. Müler, P. Koumoutsakos, A parameter study for differential evolution, WSEAS NNA-FSFS-EC, 2002. [Online]: http://citeseer.ist.psu.edu/526865.html.

[10] V. Feoktistov and S. Janaqi, Generalization of the strategies in differential evolution, in Proceedings of the 18th International Parallel and Distributed Processing Symposium, pp. 165a, Apr. 2004.

[11] E. Mezura-Montes, J. Velázquez-Reyes, and C.A.C. Coello, A comparative study of differential evolution variants for global optimization, In proceedings of Genetic Evolutionary Computation Conference (GECCO-2006), 2006, 485 – 492.

[12] A.K. Jain, M.N. Murty, and P.J. Flynn, Data clustering: A review, ACM Computing Surveys, 1999, 31(3): pp. 264 - 323.

[13] R. Storn and K. Price, Home page of differential evolution, 2008. [Online]. Available: http://www.ICSI.Berkeley.edu/~storn/code.html

[14] B. Alatas, E. Akin, and A. Karci, MODENAR: Multi-objective differential evolution algorithm for mining numeric association rules, Applied Soft Computing, vol. 8, no. 1, pp. 646–656, Jan. 2008.

[15] S. Das, A. Abraham, and A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Transaction on Systems Man and Cybernetics: Part A, vol. 38, no. 1, pp. 218–237, Feb 2008.

[16] V. Feoktistov, Differential Evolution: In Search of Solutions. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[17] U. Chakraborty, Advances in Differential Evolution. Berlin: Springer-Verlag, 2008.

[18] G. C. Onwubolu and D. Davendra, Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization. Berlin: Springer-Verlag, 2009.

[19] M.G.H. Omran, A.P. Engelbrecht, and A. Salman, An Overview of Clustering Methods, Intelligent Data Analysis, 2007, 11, 583 - 605.

[20] IEEE ICDM'06, The 6th IEEE International Conference on Data Mining, Dec. 18 – 22, 2006, Hong Kong.

[21] X. Wu, V. Kumar, J.R. Quinlan, et al, Top 10 Algorithms in Data Mining, Knowledge and Information Systems, 2008, 14(1): 1 - 37.

[22] K. Krishna and M.N. Murty, Genetic K-Means Algorithm, IEEE Transactions on Systems Man and Cybernetics 29(3) (1999) 433 - 439.

[23] S. Bandyopadhyay and U. Maulik, An Evolutionary Technique based on K-means Algorithm for Optimal Clustering in R^N, Information Sciences 146 (2002) 221 - 237.

[24] 李海民, 遗传算法性能及其在聚类分析中应用的研究: [学位论文], 西安电子科技大学博士学位论文, 1999.

[25] 吕　强, 俞金寿. 基于混合遗传算法的K-Means最优聚类算法, 华东理工大学学报(自然科学版), 2005, 31(2): 219 - 222.

[26] 邢宗义, 张　永, 侯远龙, 等, 基于模糊聚类和遗传算法的具备解释性和精确性的模糊分类系统设计, 电子学报, 2006, 34(1): 83 - 88.

[27] 许文杰, 刘希玉. 基于改进免疫遗传算法的聚类分析研究与应用. 计算机科学, 2008, 35(1): 204 - 205.

[28] M. Laszlo and S. Mukherjee, A Genetic Algorithm that Exchanges Neighboring Centers for K-means Clustering, Pattern Recognition Letters, 2007, 28: 2359 - 2366.

[29] M. Sarkar, B. Yegnanarayana, and D. Khemani, A Clustering Algorithm Using an Evolutionary Programming-based Approach, Pattern Recognition Letters, 2997, 18: 975 - 986.

[30] C.Y. Cheo and F. Ye, Particle Swarm Optimization Algorithm and Its Application to Clustering Analysis, Proceedings of the 2004 IEEE International Conference on Networking, 2004, 789 - 794.

[31] S. Paterlini and T. Krink, High Performance Clustering with Differential Evolution, Proceedings of the 2004 IEEE Congress on Evolutionary Computation, 2004, 2004 - 2011.

[32] S. Paterlini and T. Krink, Differential evolution and particle swarm optimisation in partitional

clustering, Computational Statistics & Data Analysis, 2006, 50: 1220 – 1247.

[33] C. Grosan, A. Abraham, and H. Ishibuchi, Hybrid Evolutionary Algorithms. Springer-Verlag, Berlin, Germany, 2009.

[34] H.Y. Fan and J. Lampinen, A trigonometric mutation operation to differential evolution. Journal of Global Optimization, 2003, 27(1): 105–129.

[35] J. Sun, Q. Zhang, and E.P.K. Tsang, DE/EDA: A new evolutionary algorithm for global optimization. Information Sciences, 2005, 169(3-4): 249–262.

[36] W. Gong, Z. Cai, and C.X Ling, ODE: A fast and robust differential evolution based on orthogonal design. In: AI 2006: Advances in Artificial Intelligence, 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006, Proceedings, LNAI 4304, 2006, 709–718.

[37] W. Gong, Z. Cai, and L.X. Jiang, Enhancing the Performance of Differential Evolution Using Orthogonal Design Method, Applied Mathematics and Computation. Elsevier Press. Dec. 2008, 206(1): 56 - 69.

[38] N. Noman and H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization. In: H.G. Beyer, U.M. O'Reilly (eds.) Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, 2005, 967–974.

[39] Y.J. Wang, J.S. Zhang, and G.Y. Zhang, A dynamic clustering based differential evolution algorithm for global optimization. European Journal of Operational Research, 2007, 183(1): 56–73.

[40] S. Rahnamayan, H. Tizhoosh, M. Salama, Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64–79.

[41] A. Caponio, A., F. Neri, and V. Tirronen, Super-fit control adaptation in memetic differential evolution frameworks. Soft Computing, 2009, 13(8-9): 811 - 831.

[42] W. Gong, Z. Cai, C.X. Ling, and J. Du, Hybrid Differential Evolution based on Fuzzy C-means Clustering, proceedings of Genetic and Evolutionary Computation Conference (GECCO 2009). ACM Press. 2009, 523 - 530.

[43] W. Gong, Z. Cai, and C.X. Ling, DE/BBO: A Hybrid Differential Evolution with Biogeography-Based Optimization for Global Numerical Optimization, Soft Computing. 2010, (In Press).

[44] D. Simon, Biogeography-Based Optimization, IEEE Transactions on Evolutionary Computation, 2008, 12(6): 702-713.

[45] J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm. Soft Computing, 2005, 9(6): 448–462.

[46] A. Salman, A.P. Engelbrecht, and M.G.H. Omran, Empirical analysis of self-adaptive differential evolution. European Journal of Operational Research, 2007, 183(2): 785–804.

[47] A. Nobakhti, and H. Wang, A simple self-adaptive differential evolution algorithm with application on the ALSTOM gasifier. Applied Soft Computing, 2008, 8(1): 350–370.

[48] S. Das, A. Konar, A., and U.K. Chakraborty, Two improved differential evolution schemes for faster global search. In: H.G. Beyer, U.M. OReilly (eds.) Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005, 991–998.

[49] J. Teo, Exploring dynamic self-adaptive populations in differential evolution. Soft Computing, 2006, 10(8): 673–686.

[50] J. Brest, and M.S. Maucec, Population size reduction for the differential evolution algorithm. Applied Intelligence, 2008, 29(3): 228 - 247.

[51] N.S. Teng, J. Teo, and M.H.A. Hijazi, Self-adaptive population sizing for a tune-free differential evolution. Soft Computing, 2009, 13(7): 709 - 724.

[52] A. K. Qin, V. L. Huang, and P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398 - 417.

[53] J. Zhang and A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945 - 958.

[54] J. Zhang and A. C. Sanderson, Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization. Berlin: Springer-Verlag, 2009.

[55] P. Kaelo and M.M. Ali, A numerical study of some modified differential evolution algorithms. European J of Operational Research, 2006, 169(3): 1176 - 1184.

[56] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Transactions on Evolutionary Computation, 2009, 13(3): 526 – 553.

[57] A. K. Qin and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in IEEE Congress on Evolutionary Computation (CEC2005). IEEE, 2005, pp. 1785–1791.

[58] C.A.C. Coello, Evolutionary multi-objective optimization: A historical view of the field, IEEE Computational Intelligence Magazine, 2006, 1(1): 28 - 36.

[59] N.K. Madavan, Multiobjective Optimization Using a Pareto Differential Evolution Approach. In Congress on Evolutionary Computation (CEC2002), 2002, 2: 1145 - 1150.

[60] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Gunter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, Proceedings of the Parallel Problem Solving from Nature VI Conference, LNCS 1917, 2000, 849 - 858.

[61] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182 - 197.

[62] F. Xue, A.C. Sanderson, and R.J. Graves, Pareto-based Multi-Objective Differential Evolution.

In Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003), 2003, 2: 862 - 869.

[63] E. Zitzler and L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, 1999, 3(4): 257 - 271.

[64] B.V. Babu and M. Jehan, Differential Evolution for Multi-Objective Optimization. In Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003), 2003, 4: 2696 - 2703.

[65] K.E. Parsopoulos, D.K. Taoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis. Vector Evaluated Differential Evolution for Multiobjective Optimization. In 2004 Congress on Evolutionary Computation (CEC2004), 2004, 1: 204 - 211.

[66] J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, 1985, 93 - 100.

[67] A. Iorio A and X. Li, Solving rotated multi-objective optimization problems using differential evolution. Proc. of Advances in Artificial Intelligence. Berlin: Spering-Verlag, 2004, LNAI 3339, 861–872.

[68] S. Kukkonen and J. Lampinen, An extension of generalized differential evolution for multi-objective optimization with constraints. Proc. of Parallel Problem Solving from Nature. Berlin: Spering-Verlag, 2004, LNCS 3242, 752–761.

[69] T. Robič and B. Filipič, DEMO: Differential evolution for multiobjective optimization. Proc. of EMO'05. Berlin: Spering-Verlag, LNCS 3410, 2005: 520– 533.

[70] S. Kukkonen and J. Lampinen, GDE3: The third evolution step of generalized differential evolution. Proc. of CEC'05. Piscataway, NJ: IEEE, 2005, 1: 443- 450.

[71] 张利彪, 周春光, 马铭, 等. 基于极大极小距离密度的多目标微分进化算法. 计算机研究与发展, 2007, 44 (1) : 177 - 184.

[72] H. Abbass and R. Sarker, The Pareto differential evolution algorithm. International Journal on Artificial Intelligence Tools, 2002, 11(4): 531–552.

[73] L.V. Santana-Quintero and C.A.C. Coello, An algorithm based on differential evolution for multi-objective problems, International Journal of Computational Intelligence Research, 2005, 1(2): 151 - 169.

[74] K. Deb, M. Moha, and S. Mishra, Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. Evolutionary Computation, 2005, 13(4): 501-525.

[75] 龚文引, 蔡之华, 基于ε占优的正交多目标差分演化算法研究, 计算机研究与发展. 2009, 46(4): 655 - 666.

[76] 方开泰, 马长兴. 正交与均匀实验设计. 北京: 科学出版社, 2001.

[77] A.G. Hernandez-Diaz, L.V. Santana-Quintero, C.A.C. Coello, et al., Pareto-adaptive

ε-dominance, Evolutionary Computation, 2007, 15(4): 493–517.

[78] W. Gong and Z. Cai, An Improved Multiobjective Differential Evolution based on Pareto-adaptive ε-dominance and Orthogonal Design, European Journal of Operational Research. Elsevier Press. Oct. 2009, 198(2): 576 - 601.

[79] W. Gong, Z. Cai, and L. Zhu, An Efficient Multiobjective Differential Evolution Algorithm for Engineering Design, Structural and Multidisciplinary Optimization. Springer-Verlag. Apr. 2009, 38(2): 137 - 157.

[80] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering, 2002, 191(11-12): 1245-1287.

[81] 刘波，王凌，金以慧，差分进化算法研究进展，控制与决策, 2007, 22(7): 721 – 729.

[82] R. Storn, System design by constraint adaptation and differential evolution, IEEE Transactions on Evolutionary Computation, 1999, 3(1): 22–34.

[83] J. Lampinen, A constraint handling approach for the differential evolution algorithm, in: Proceedings of the Congress on Evolutionary Computation 2002 (CEC2002), 2002, 2: 1468–1473.

[84] R.L. Becerra and C.A.C. Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering, 2006, 195: 4303 - 4322.

[85] T. Takahama and S. Sakai, Constrained Optimization by the ε Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites, in 2006 IEEE Congress on Evolutionary Computation (CEC'2006), 2006, 308 - 315.

[86] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, et al, Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization, 2006. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan/cec2006/technical report.pdf

[87] F.-Z Huang, L. Wang, and Q. He, An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and Computation, 2007, 186(1): 340-356.

[88] E. Mezura-Montes, C. Coello, J. Velazquez-Reyes, et al, Multiple trial vectors in differential evolution for engineering design. Engineering Optimization, 2007, 39(5): 567-589.

[89] T.P. Runarsson and X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactins on Evolutionary Computation, 2000, 4(3): 284 - 294.

[90] M. Zhang, W. Luo, and X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences, 2008, 178(15): 3043 – 3074.

[91] W. Gong and Z. Cai, A Multiobjective Differential Evolution Algorithm for Constrained Optimization, in 2008 Congress on Evolutionary Computation (CEC'2008), 2008, 181-188

[92] Z. Fan, J. Liu, T. Sorensen, et al, Improved Differential Evolution Based on Stochastic Ranking for Robust Layout Synthesis of MEMS Components, IEEE Transactions on Industrial Electronics, 2009, 56(4): 937-948.

[93] H. Liu, Z. Cai, and Y. Wang, Hybridizing particle swarm optimization with differential

evolution for constrained numerical and engineering optimization, Applied Soft Computing, 2010, 10(2): 629-640.

[94] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, 2000, 186: 311–338.

[95] G. Pampará, A.P. Engelbrecht, and N. Franken, Binary Differential Evolution, 2006 IEEE Congress on Evolutionary Computation (CEC2006), 2006, 1873 - 1879.

[96] T. Gong and A.L. Tuson, Differential Evolution for Binary Encoding, In A. Saad et al. (Eds.): Soft Computing in Industrial Applications, ASC 39, 2007, 251–262.

[97] X. He and L. Han, A novel binary differential evolution algorithm based on artificial immune system, 2007 IEEE Congress on Evolutionary Computation (CEC2007), 2007, 2267 – 2272.

[98] A.P. Engelbrecht and G. Pampará, Binary differential evolution strategies, 2007 IEEE Congress on Evolutionary Computation (CEC2007), 2007, 1942 – 1947.

[99] A. Moraglio and J. Togelius, Geometric Differential Evolution, In proceedings of Genetic Evolutionary Computation Conference (GECCO-2009), 2009, 1705– 1712.

[100] G.W. Greenwood, Using Differential Evolution for a Subclass of Graph Theory Problems, IEEE Transactions on Evolutionary Computation, 2009, 13(5): 1190 - 1192.

[101] A.C. Nearchou and S.L. Omirou, Differential evolution for sequencing and scheduling optimization, Journal of Heuristics, 2006, 12, 395 - 411.

[102] G. Onwubolu and D. Davendram, Scheduling flow shops using differential evolution algorithm. European Journal of Operational Research, 2006, 171(2): 674 - 692.

[103] Q.-K. Pan, M.F. Tasgetiren, and Y.-C. Liang, A Discrete Differential Evolution Algorithm for the Permutation Flowshop Scheduling Problem, In proceedings of Genetic Evolutionary Computation Conference (GECCO-2007), 2007, 126– 133.

[104] R. Storn, Differential evolution design of an IIR-filter with requirements for magnitude and group delay. Technical Report TR-95-026, International Computer Science Institute, Berkeley, CA, June 1995.

[105] J. Lampinen and I. Zelinka, Mechanical engineering design optimization by differential evolution. In David Corne, Marco Dorigo, and Fred Glover, editors, New Ideas in Optimization, pages 127–146. London, McGraw-Hill, 1999.

[106] 宋立明, 李军, 丰镇平, 跨音速透平扭叶片的气动优化设计研究, 西安交通大学学报, 2005, 39(11): 1277 – 1281.

[107] T. Rogalsky, Aerodynamic shape optimization of fan blades. Master's thesis, University of Manitoba. Department of Applied Mathematics, 1998.

[108] T. Rogalsky, S. Kocabiyik, and R.W. Derksen. Differential evolution in aerodynamic optimization. Canadian Aeronautics and Space Journal, 2000, 46(4): 183–190.

[109] T. Rogalsky and R.W. Derksen. Hybridization of differential evolution for aerodynamic design. In Proceedings of the 8th Annual Conference of the Computational Fluid Dynamics Society of Canada, 2000, 729–736.

[110] P. Vancorenland, C. De Ranter, M. Steyaert, et al, Optimal RF design using smart evolutionary algorithms. In Proceedings of 37th Design Automation Conference, Los Angeles, 5–9 June 2000.

[111] V. Tirronen, F. Neri, T. Karkkainen, et al, An Enhanced Memetic Differential Evolution in Filter Design for Defect Detection in Paper Production, Evolutionary Computation, 2008, 16(4): 529–555.

[112] 龚文引, 演化 Kalman 滤波及其应用研究[学位论文]. 中国地质大学(武汉), 2007.5.

[113] J.-P. Chiou and F.-S. Wang. Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. Computers and Chemical Engineering, 1999, 23(9): 1277–1291.

[114] J.-P. Chiou and F.-S. Wang. A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. In Proceedings of IEEE International Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 1998, 627–632.

[115] B.V. Babu, P.G. Chakole and J.H.S. Mubeen, Multiobjective differential evolution (MODE) for optimization of adiabatic styrene reactor, Chemical Engineering Science, 2005, 60: 4822 - 4837.

[116] N. Chakraborti, K. Deb, A. Jha, A genetic algorithm based heat transfer analysis of a bloom re-heating furnace. Steel Research, 2000, 71(10): 396 - 402.

[117] B.V. Babu and R. Angira, Modified differential evolution (MDE) for optimization of non-linear chemical processes, Computers and Chemical Engineering, 2006, 30: 989–1002.

[118] F.-S. Wang and J.-P. Chiou. Differential evolution for dynamic optimization of differential-algebraic systems. In Proceedings of the IEEE International Conference on Evolutionary Computation - ICEC'97, 1997, 531–536.

[119] F.-S. Wang and J.-P. Chiou. Optimal control and optimal time location problems of differential-algebraic systems by differential evolution. Ind. Eng. Chem. Res, 1997, 36: 5348–5357.

[120] F. Cheong and R. Lai. Designing a hierarchical fuzzy logic controller using differential evolution. In Proceedings of IEEE International Fuzzy Systems Conference, FUZZ-IEEE'99, 1999, 1: 277–282.

[121] C.S. Chang, D.Y. Xu, and H.B. Quek. Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. In IEE Proceedings on Electric Power Applications, 1999, 146: 577–583.

[122] C.S. Chang and D.Y. Xu. Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system. In IEE Proceedings on Electric Power Applications, 2000, 147: 206–212.

[123] I.L. Lopez Cruz, L.G. Van Willigenburg, and G. Van Straten. Parameter control strategy in differential evolution algorithm for optimal control. In M.H. Hamza, editor, Proceedings of

the IASTED International Conference Artificial Intelligence and Soft Computing (ASC 2001), 2001, 211–216.

[124] R.K. Ursem and P. Vadstrup. Parameter identification of induction motors using differential evolution. In Proceedings of the 5th Congress on Evolutionary Computation (CEC2003), 2003, 2: 790–796.

[125] H. Santos, J. Mendes, P.B. de M. Oliveira, et al. Path planning optimization using the differential evolution algorithm. In ROBOTICA2003, 2003.

[126] C.-H. Chen, C.-J. Lin, and C.-T. Lin, Nonlinear System Control Using Adaptive Neural Fuzzy Networks Based on a Modified Differential Evolution, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2009, 39(4): 459 - 473.

[127] S.-K. Wang, J.-P. Chiou, and C.-W. Liu, Non-smooth/non-convex economic dispatch by a novel hybrid differential evolution algorithm, IET Gener. Transm. Distrib., 2007, 1(5): 793–803.

[128] D. He, F. Wang, and Z. Mao, A hybrid genetic algorithm approach based on differential evolution for economic dispatch with valve-point effect, Electrical Power and Energy Systems, 2008, (30): 31–38.

[129] N. Noman and H. Iba, Differential evolution for economic load dispatch problems, Electric Power Systems Research, 2008, (78): 1322–1331.

[130] J.-P. Chiou, A variable scaling hybrid differential evolution for solving large-scale power dispatch problems, IET Gener. Transm. Distrib., 2009, 3(2): 154–163.

[131] H.R. Cai, C.Y. Chung, and K. P. Wong, Application of Differential Evolution Algorithm for Transient Stability Constrained Optimal Power Flow, IEEE Transactions on Power Systems, 2008, 23(2): 719 - 728.

[132] M. Varadarajan and K.S. Swarup, Solving multi-objective optimal power flow using differential evolution, IET Gener. Transm. Distrib., 2008, 2(5): 720–730.

[133] G.Y. Yang, Z.Y. Dong, and K.P. Wong, A Modified Differential Evolution Algorithm With Fitness Sharing for Power System Planning, IEEE Transactions on Power Systems, 2008, 23(2): 514 - 522.

[134] Y.P. Chang and C.J. Wu, Optimal multiobjective planning of large-scale passive harmonic filters using hybrid differential evolution method considering parameter and loading uncertainty, IEEE Transactions on Power Delivery, 2005, 20(1): 408 - 416.

[135] S. Kannan, S.M.R. Slochanal, N.P. Padhy, Application and comparison of metaheuristic techniques to generation expansion planning problem. IEEE Transactions on Power Systems, 2005, 20(1): 466 - 475.

[136] J. Ilonen, J.-K. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. Neural Processing Letters, 2003, 17(1): 93–105.

[137] C. Chen, D. Chen, and G. Cao. An improved differential evolution algorithm in training and encoding prior knowledge into feedforward networks with application in chemistry.

Chemometrics and Intelligent laboratory systems, 2002, (64): 27–43.

[138] G.D. Magoulas, V.P. Plagianakos, and M.N. Vrahatis. Hybrid methods using evolutionary algorithms for on-line training. In Proceedings of IJCNN'01, International Joint Conference on Neural Networks, 2001, 3: 2218–2223.

[139] H.A. Abbass, An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine, 2002, 25(3): 265 - 281.

[140] J.G. Marin-Blazquez, Q. Shen, and A. Tuson. Tuning fuzzy membership functions with neighbourhood search techniques: A comparative study. In Proceedings of the 3rd IEEE International Conference on Intelligent Engineering Systems, 1999, 337–342.

[141] H.M. Abdul-Kader, Neural Networks Training Based on Differential Evolution Algorithm Compared with Other Architectures for Weather Forecasting, International Journal of Computer Science and Network Security, 2009, 9(3): 92 - 99.

[142] B. Subudhi and D. Jena, A differential evolution based neural network approach to nonlinear system identification, Applied Soft Computing, 2010, (In press).

[143] D. Zaharie, Density Based Clustering with Crowding Differential Evolution, In Proceedings of the Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05), 2005, 343 - 350.

[144] L. Zhang, M. Ma, X. Liu, et al, Differential Evolution Fuzzy Clustering Algorithm Based on Kernel Methods, In G. Wang et al. (Eds.): Proceedings of the 1st International Conference on Rough Sets and Knowledge Technology (RSKT 2006), LNAI 4062, 2006, 430–435.

[145] W. Gong, Z. Cai, C.X. Ling, et al, A Point Symmetry-based Automatic Clustering Approach Using Differential Evolution, proceedings of the 4th International Symposium on Intelligence Computation and Applications (ISICA 2009), LNCS 5821, Springer-Verlag. 2009, 151 - 162.

[146] S. Das, A. Abraham, and A. Konar, Metaheuristic Clustering, Springer-Verlag, Berlin Heidelberg, 2009.

[147] K. Suresh, D. Kundu, S. Ghosh, et al, Data Clustering Using Multi-objective Differential Evolution Algorithms, Fundamenta Informaticae, 2009, XXI: 1001–1024.

[148] A. Abraham, S. Das, and A. Konar, Document Clustering Using Differential Evolution, 2006 IEEE Congress on Evolutionary Computation (CEC2006), 2006, 1784 - 1791.

[149] Y.-C. Lin, K.-S. Hwang, and F.-S. Wang. Plant scheduling and planning using mixed-integer hybrid differential evolution with multiplier updating. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC2000), 2000, 1: 593 - 600.

[150] M. Ruttgers. Design of a new algorithm for scheduling in parallel machine shops. In Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing, 1997, 3: 2182–2187.

[151] F. Xue, A.C. Sanderson, and R.J. Graves. Multiobjective differential evolution and its application to enterprise planning. In Proceedings of IEEE International Conference on Robotics and Automation, 2003 (ICRA'03), 2003, 3: 3535 - 3541.

[152] K. Rzadca and F. Seredynski. Heterogeneous multiprocessor scheduling with Differential Evolution. In The 2005 IEEE Congress on Evolutionary Computation (CEC2005), 2005, 3: 2840 - 2847.

[153] L. Wang, Q.-K. Pan, P.N. Suganthan, et al, A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems, Computers and Operations Research, 2010, 37(3): 509 - 520 .

[154] E.M. Bernardino, A.M. Bernardino, J.M. Sanchez-Perez, et al, A Hybrid Differential Evolution Algorithm for Solving the Terminal Assignment Problem, In S. Omatu et al. (Eds.): IWANN 2009, Part II, LNCS 5518, 2009, 179–186.

[155] R. Thomsen. Flexible ligand docking using differential evolution. In Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003), 2003, 4: 2354–2361.

[156] M. Ali and A. Torn. Optimization of carbon and silicon cluster by differential evolution. In C.A. Floudas and P. Pardalos, editors, Optimization in Computational Chemistry and Molecular Biology, 2000, 287–300.

[157] N.P. Moloi and M.M. Ali. An iterative global optimization algorithm for potential energy minimization. Computational Optimization and Applications, 2005, 30(2): 119 - 132.

[158] K.Y. Tsai and F.S. Wang, Evolutionary optimization with data collocation for reverse engineering of biological networks. Bioinformatics, 2005, 21(7): 1180 - 1188.

[159] P. Thomas and D. Vernon. Image registration by differential evolution. In Proceedings of the First Irish Machine Vision and Image Processing Conference IMVIP-97, 1997, 221–225.

[160] M. Salomon, G.-R. Perrin, and F. Heitz. Parallelizing differential evolution for 3D medical image registration. Technical report, ICPS, Univ.-Strasbourg, September 2000.

[161] M.G.H. Omran, A.P. Engelbrecht, and A. Salman, Differential Evolution Methods for Unsupervised Image Classification, 2005 IEEE Congress on Evolutionary Computation (CEC2005), 2005, 966 - 973.

[162] M.G.H. Omran and A.P. Engelbrecht, Self-Adaptive Differential Evolution Methods for Unsupervised Image Classification, 2006 IEEE Conference on Cybernetics and Intelligent Systems (CIS2006), 2006, 800 - 805.

[163] S. Das and A. Konar, Automatic image pixel clustering with an improved differential evolution, Applied Soft Computing, 2009, (9): 226 - 236.

[164] H.-J. Huang and F.-S. Wang. Fuzzy decision-making design of chemical plant using mixed-integer hybrid differential evolution. Computers and Chemical Engineering, 2002, 26(12): 1649–1660.

[165] V. Feoktistov and S. Janaqi. Classical identification problem solved by differential evolution: Choquet integral. In R. Matousek and P. Osmera, editors, 10th International Conference on Soft Computing (MENDEL2004), 2004, 62–67.

[166] M. Lozano and C. Garcia-Martinez, Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. Computers

& Operations Research, 2010, 37(3): 481–497.

[167] X. Yao, Y. Liu, and G. Lin, Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82–102.

[168] P.N. Suganthan, N. Hansen, J.J. Liang, et al, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization (2005). Available Online at: http://www.ntu.edu.sg/home/EPNSugan

[169] D.H. Wolpert and W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67 - 82.

[170] J. Brest, B. Boskovic, S. Greiner, et al, Performance comparison of self-adaptive and adaptive differential evolution algorithms, Soft Computing, 2007, 11(7): 617 - 629.

[171] J.E. Smith and T.C. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft Computing, 1997, 1(2): 81 - 87.

[172] A.E. Eiben, R. Hinterding, and Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on Evolutionary Computation, 1999, 3(2): 124 - 141.

[173] Y.-S. Ong and A. J. Keane, Meta-Lamarckian learning in memetic algorithms, IEEE Transactions on Evolutionary Computation, 2004, 8(2): 99 - 110.

[174] J. Vrugt, B. Robinson, and J. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Transactions on Evolutionary Computation, 2009, 13(2): 243 - 259.

[175] X.-F. Xie and W.-J. Zhang, SWAF: Swarm algorithm framework for numerical optimization, in Proc. Genetic Evol. Comput. Conf., GECCO 2004, Part I, ser. Lecture Notes in Computer Science, vol. 3102, 2004, 238–250.

[176] A. Zamuda, J. Brest, B. Boskovic, et al, Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution, in 2008 IEEE World Congr. on Comput. Intell. IEEE Press, 2008, 3719–3726.

[177] M.M. Ali, C. Khompatraporn, and Z.B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, Journal of Global Optimization, 2005, 31(4): 635–672.

[178] C. Shaw, K. Williams, and R. Assassa, Patients' views of a new nurse-led continence service, Journal of Clinical Nursing, 2003. 9(4): 574–584.

[179] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal Of Machine Learning Research, 2006, 7: 1–30.

[180] S. Garcia and F. Herrera, An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons, Journal Of Machine Learning Research, 2008, 9: 2677–2694.

[181] C.-Y. Lee and X. Yao, Evolutionary programming using mutations based on the Levy probability distribution, IEEE Transactions on Evolutionary Computation, 2004, 8(1): 1–13.

[182] A. Auger and N. Hansen, A restart CMA evolution strategy with increasing population size, in

IEEE Congr. on Evol. Comput. (CEC2005), vol. 2. IEEE, 2005, 1769–1776.

[183] F. Peng, K. Tang, G. Chen, and X. Yao, Multi-start JADE with knowledge transfer for numerical optimization, in IEEE Congr. on Evol. Comput. (CEC2009). IEEE, 2009, 1889–1895.

[184] J. Knowlesand D. Corne, Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary Computation, 2000, 8(2): 149 – 172.

[185] 曾三友, 魏巍, 康立山, 等. 基于正交设计的多目标演化算法. 计算机学报, 2005, 28(7): 1153-1162.

[186] 崔逊学, 林闯. 一种基于偏好的多目标调和遗传算法. 软件学报, 2005, 16(5): 761 - 770.

[187] 石川, 李清勇, 史忠植. 一种快速的基于占优树的多目标进化算法. 软件学报, 2007, 18(3): 505 - 516.

[188] Y. Leung and Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Transactions on Evolutionary Computation, 2001, 5(1): 41 – 53.

[189] M. Laumanns, L. Thiele, K. Deb, et al, Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation, 2002, 10: 263–282

[190] K. Deb, M. Mohan, S. Mishra, Towards a quick computation of well-spread Pareto-optimal solutions. In: C.M. Fonseca et al (eds) Proceedings of the second international conference on evolutionary multi-criterion optimization (EMO- 03). Springer, Berlin Heidelberg, 2003, 222–236.

[191] Z. Cai, W. Gong, Q. Huang YQ, A novel differential evolution algorithm based on ε-domination and orthogonal design method for multiobjective optimization. In: S. Obayashi, et al (eds) Proceedings of the fourth international conference on evolutionary multi-criterion optimization (EMO-07). Springer, Berlin Heidelberg, 2007, 286–301.

[192] A. Zhou, Q. Zhang, Y. Jin, et al. A model-based evolutionary algorithm for bi-objective optimization. Proc of CEC'05 (CEC2005). Piscataway, NJ: IEEE, 2005, (3): 2568 - 2575.

[193] K. Deb, L. Thiele, M. Laumanns, et al. Scalable test problems for evolutionary multi-objective optimization. Tech. Rep. 112, Zurich, Switzerland, 2001.

[194] E. Zitzler, M. Laumanns, and L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm . Technical Report 103, Zurich, Switzerland, 2001.

[195] M. Chen and Y. Lu, A novel elitist multiobjective optimization algorithm: Multiobjective extremal optimization, European Journal of Operational Research, 2008, 188(3): 637-651.

[196] E. Zitzler, L. Thiele, M. Laumanns, et al. Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation, 2003, 7(2): 117 – 132.

[197] A.G. Hernández-Díaz, L.V. Santana-Quintero, C.A.C. Coello, et al, Pareto-adaptive ε-dominance, Evolutionary Computation, 2007, 15(4): 493 - 517.

[198] E. Zitzler, K. Deb K, L. Thiele, Comparison of multiobjective evolutionary algorithms:

empirical results. Evolutionary Computation, 2000, 8: 173–195.

[199] A. Oyama, K. Shimoyama, K. Fujii, New constraint-handling method for multi-objective and multi-monstraint evolutionary optimization. Trans Jpn Soc Aeronaut Space Sci, 2007, 50: 56–62

[200] A. Kurpati, S. Azarm, J. Wu, Constraint handling improvements for multiobjective genetic algorithms. Struct Multidisc Optim, 2002, 23: 204–213.

[201] K. Deb, A. Pratap, T. Meyarivan, Constrained test problems for multi-objective evolutionary optimization. In: E. Zitzler, et al (eds) Proceedings of the first international conference on evolutionary multi-criterion optimization (EMO-01). 2001, 284–298.

[202] A. Farhang-Mehr, S. Azarm, Entropy-based multi-objective genetic algorithm for design optimization. Struct Multidisc Optim, 2002, 24: 351–361.

[203] C.A.C. Coello, G. Pulido, Multiobjective structural optimization using a microgenetic algorithm. Struct Multidisc Optim, 2005, 30: 388–403.

[204] M.J. Reddy, D.N. Kumar, An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design. Eng Optim, 2007, 39: 49–68.

[205] T. Ray, K.M. Liew, A swarm metaphor for multiobjective design optimization. Eng Optim, 2002, 34: 141–153.

[206] E. Falkenauer, Genetic Algorithms and Grouping Problems. New York: Wiley, 1998.

[207] E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, et al, A Survey of Evolutionary Algorithms for Clustering, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2009, 39(2): 133 - 155.

[208] D.N.A. Asuncion, UCI Machine Learning Repository. Online available at: http://archive.ics.uci.edu/ml/. 2008.

[209] S.P. Lloyd, Least Squares Quantization in PCM. Unpublished Bell Lab. Tech. Note, portions presented at the Institute of Mathematical Statistics Meeting Atlantic City, NJ, September 1957. Also, IEEE Trans Inform Theory (Special Issue on Quantization), 1982, vol. IT-28: 129 - 137.

[210] R. Cucchiara, Genetic algorithms for clustering in machine vision, Machine Vision Application, 1998, 11(1): 1 - 6.

[211] C.A. Murthy and N. Chowdhury, In search of optimal clusters using genetic algorithm, Pattern Recognition Letter, 1996, 17(8): 825 - 832.

[212] U. Maulik and S. Bandyopadhyay, Genetic Algorithm-based Clustering Technique, Pattern Recognition, 2000, 32: 1455 - 1465.

[213] W. Sheng, S. Swift, L. Zhang, and X. Liu, A Weighted Sum Validity Function for Clustering With a Hybrid Niching Genetic Algorithm, IEEE Transactions on Systems Man and Cybernetics, Part B: Cybernetics, 2005, 35(6): 1156 – 1167.

[214] S. Bandyopadhyay and S. Saha, GAPS: A clustering Method Using a New Point Symmetry-based Distance Measure, Pattern Recognition, 2007, 40: 3430 - 3451.

[215] P. Brucker, On the complexity of clustering problems, in Optimization and Operations Research, vol. 157, M. Beckmenn and H. P. Kunzi, Eds. Berlin, Germany: Springer-Verlag, 1978, pp. 45–54.

[216] G. Hamerly and C. Elkan, Learning the K in K-means, in Proc. NIPS, Dec. 8–13, 2003, pp. 281–288.

[217] Y. Kao, E. Zahara, and I. Kao, A hybridized approach to data clustering, Experts Systems with Applications, 34 (2008) 1754 – 1762.

[218] Y. Lu, S. Lu, F. Fotouhi, et al, FGKA: a Fast Genetic K-means Clustering Algorithm, Proceedings of the 2004 ACM symposium on Applied computing, 2004, 622 - 623.

[219] Z. Michalewicz, Genetic Algorithms + Data Structure =Evolution Programs. Springer, Berlin, 1992.

[220] E.R. Hruschka and N.F.F. Ebecken, A genetic algorithm for cluster analysis, Intelligent Data Analysis, 2003, 7: 15 – 25.

[221] M. Srinivas, L. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Trans. Syst. Man Cybern. 2004, 24(4): 656 – 667.

[222] M.-C. Su, C.-H. Chou, A modified version of the k-means algorithm with a distance based on cluster symmetry, IEEE Trans. Pattern Anal. Mach. Intell. 2001, 23(6): 674–680.

[223] C.H. Chou, M.C. Su, E. Lai, Symmetry as a new measure for cluster validity, in: Second WSEAS International Conference on Scientific Computation and Soft Computing, 2002, 209–213.

[224] K.-L. Chung and J.-S. Lin, Faster and more robust point symmetry-based K-means algorithm, Pattern Recognition, 2007, (40): 410 – 422.

[225] D.M. Mount, S. Arya, ANN: A Library for Approximate Nearest Neighbor Searching. Available online at: http://www.cs.umd.edu/~mount/ANN. 2005.

[226] S. Bandyopadhyay and S. Saha, A Point Symmetry-Based Clustering Technique for Automatic Evolution of Clusters, IEEE Transactions on Knowledge and Data Engineering, 2008, 20(11): 1441 - 1457.

[227] S. Bandyopadhyay and S.K. Pal, Classification and Learning Using Genetic Algorithms: Applications in Bioinformatics and Web Intelligence, Springer, Heidelberg, 2007.

[228] C. van Rijsbergen, Information Retrieval, 2nd edition. Butterworths, London, 1979.

[229] S. Bandyopadhyay and U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition, 2002, (35): 1197 – 1208.

[230] U. Maulik and S. Bandyopadhyay, Fuzzy Partitioning Using a Real-Coded Variable-Length Genetic Algorithm for Pixel Classification, IEEE Transactions on Geoscience and Remote Sensing, 2003, 41(5): 1075 - 1081.

[231] S. Das, A. Abraham, and A. Konar, Automatic kernel clustering with a Multi-Elitist Particle Swarm Optimization Algorithm, Pattern Recognition Letters, 2008, (29): 688–699.

[232] J. C. Dunn, Well separated clusters and optimal fuzzy partitions, J. Cybern., 1974, 4: 95–104.

[233] R.B. Calinski and J. Harabasz, A dendrite method for cluster analysis, Commun. Stat., 1974, 3(1): 1–27.

[234] D.L. Davies and D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell.,1979, 1(2): 224–227.

[235] M.K. Pakhira, S. Bandyopadhyay, and U. Maulik, Validity index for crisp and fuzzy clusters, Pattern Recognit. Lett., 2004, 37(3): 487–501.

[236] C. H. Chou, M. C. Su, and E. Lai, A new cluster validity measure and its application to image compression, Pattern Anal. Appl., 2004, 7(2): 205–220.

附录 1　无约束单目标优化函数

1. Sphere Model

$$f_{01}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2, \qquad -100 \le x_i \le 100$$

$\min(f_{01}(\mathbf{x})) = f_{01}(0, \mathbf{L}, 0) = 0.$

2. Schwefel's Problem 2.22

$$f_{02}(\mathbf{x}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|, \qquad -10 \le x_i \le 10$$

$\min(f_{02}(\mathbf{x})) = f_{02}(0, \mathbf{L}, 0) = 0.$

3. Schwefel's Problem 1.2

$$f_{03}(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2, \qquad -100 \le x_i \le 100$$

$\min(f_{03}(\mathbf{x})) = f_{03}(0, \mathbf{L}, 0) = 0.$

4. Schwefel's Problem 2.21

$$f_{04}(\mathbf{x}) = \max_i \{|x_i|, 1 \le i \le D\}, \qquad -100 \le x_i \le 100$$

$\min(f_{04}(\mathbf{x})) = f_{04}(0, \mathbf{L}, 0) = 0.$

5. Generalized Rosenbrock's Function

$$f_{05}(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right], \qquad -30 \le x_i \le 30$$

$\min(f_{05}(\mathbf{x})) = f_{05}(1, \mathbf{L}, 1) = 0.$

6. Step Function

$$f_{06}(\mathbf{x}) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2, \qquad -100 \le x_i \le 100$$

$\min(f_{06}(\mathbf{x})) = f_{06}(0, \mathbf{L}, 0) = 0.$

7. Quartic Function, i.e., Noise

$$f_{07}(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4 + random[0,1), \qquad -1.28 \le x_i \le 1.28$$

$\min(f_{07}(\mathbf{x})) = f_{07}(0, \mathbf{L}, 0) = 0.$

8. Generalized Schwefel's Problem 2.26

$$f_{08}(\mathbf{x}) = \sum_{i=1}^{D} \left(x_i \sin\left(\sqrt{|x_i|} \right) \right), \qquad -500 \le x_i \le 500$$

$\min(f_{08}(\mathbf{x})) = f_{08}(420.9687, \mathbf{L}, 420.9687) = -418.98289 \times D.$

9. Generalized Rastrigin's Function

$$f_{09}(\mathbf{x})=\sum_{i=1}^{D}\left[x_i^2-10\cos(2p\,x_i)+10\right],\qquad -5.12\le x_i\le 5.12$$

$\min(f_{09}(\mathbf{x}))=f_{09}(0,\mathbf{L},0)=0.$

10. Ackley's Function

$$f_{10}(\mathbf{x})=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2p\,x_i)\right)+20+\exp(1),\quad -32\le x_i\le 32$$

$\min(f_{10}(\mathbf{x}))=f_{10}(0,\mathbf{L},0)=0.$

11. Generalized Griewank Function

$$f_{11}(\mathbf{x})=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,\qquad -600\le x_i\le 600$$

$\min(f_{11}(\mathbf{x}))=f_{11}(0,\mathbf{L},0)=0.$

12. Generalized Penalized Functions

$$f_{12}(\mathbf{x})=\frac{p}{D}\left\{10\sin^2(p\,y_1)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(p\,y_{i+1})]+(y_D-1)^2\right\}+\sum_{i=1}^{D}u(x_i,10,100,4),\quad -50\le x_i\le 50$$

$\min(f_{12}(\mathbf{x}))=f_{12}(1,\mathbf{L},1)=0.$

$$f_{13}(\mathbf{x})=0.1\left\{\sin^2(3p\,x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3p\,x_{i+1})]+(x_D-1)^2[1+\sin^2(2p\,x_D)]\right\}$$
$$+\sum_{i=1}^{D}u(x_i,5,100,4),\quad -50\le x_i\le 50$$

$\min(f_{13}(\mathbf{x}))=f_{13}(1,\mathbf{L},1)=0.$

其中:

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a,\\ 0, & -a\le x_i\le a,\\ k(-x_i-a)^m, & x_i<a.\end{cases}$$

$$y_i=1+\frac{1}{4}(x_i+1).$$

13. Shekel's Foxholes Function

$$f_{14}(\mathbf{x})=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{i,j})^6}\right]^{-1},\qquad -65.536\le x_i\le 65.536$$

$\min(f_{14}(\mathbf{x}))=f_{14}(-32,-32)\approx 1.$

其中:

$$a_{i,j}=\begin{pmatrix}-32,-16,0,16,32,-32,\mathbf{L},0,16,32\\ -32,-32,-32,-32,-32,-16,\mathbf{L},32,32,32\end{pmatrix}.$$

14. Kowalik's Function

$$f_{15}(\mathbf{x})=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)^2}{b_i^2+b_ix_3+x_4}\right],\qquad -5\le x_i\le 5$$

$\min(f_{15}(\mathbf{x}))\approx f_{15}(0.1928,0.1908,0.1231,0.1358)\approx 0.0003075$. 其中 a_i,b_i 如表 F1-1 所示.

表 F1-1 Kowalik's Function f_{15} 的参数

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
b_i^{-1}	0.25	0.50	1	2	4	6	8	10	12	14	16

16. Six-Hump Camel-Back Function

$$f_{16}(\mathbf{x})=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4,\qquad -5\le x_i\le 5$$

$\mathbf{x}_{\min}=(0.08983,-0.7126),(-0.08983,0.7216)$, $\min(f_{16}(\mathbf{x}))=f_{16}(\mathbf{x}_{\min})=-1.0316285.$

17. Branin Function

$$f_{17}(\mathbf{x})=\left(x_2-\frac{5.1}{4p^2}x_1^2-6\right)^2+10\left(1-\frac{1}{8p}\right)\cos(x_1)+10,\qquad -5\le x_1\le 10,\ 0\le x_2\le 15$$

$\mathbf{x}_{\min}=(-3.142,12.275),(3.142,2.275),(9.425,2.425)$, $\min(f_{17}(\mathbf{x}))=f_{17}(\mathbf{x}_{\min})=0.398.$

18. Goldstein-Price Function

$$f_{18}(\mathbf{x})=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^3)]$$
$$[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)],\quad -2\le x_i\le 2$$

$\min(f_{18}(\mathbf{x}))=f_{18}(0,-1)=3.$

19. Hartman's Family

$$f(\mathbf{x})=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{D}a_{i,j}(x_j-p_{i,j})^2\right]$$

其 $D=3,6$ 对 $f_{19}(\mathbf{x})$ 和 $f_{20}(\mathbf{x})$，$0\le x_j\le 1$. c_i，$a_{i,j}$ 和 $p_{i,j}$ 分别见表 F1-2 和 F1-3. $\min(f_{19}(\mathbf{x}))=-3.86$，

$\min(f_{20}(\mathbf{x}))=-3.32.$

表 F1-2 Hartman Function f_{19} 的参数

i	$a_{i,j},j=1,2,3$			c_i	$p_{i,j},j=1,2,3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

表 F1-3 Hartman Function f_{20} 的参数

i	$a_{i,j},j=1,\mathbf{L},6$						c_i	$p_{i,j},j=1,\mathbf{L},6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.8307	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8732	0.8732	0.5743	0.1091	0.0381

20. Shekel's Family

$$f(\mathbf{x}) = -\sum_{i=1}^{m}\left[(\mathbf{x}-a_i)(\mathbf{x}-a_i)^T + c_i\right]^{-1}$$

其中对函数 $f_{21}(\mathbf{x}), f_{22}(\mathbf{x}), f_{23}(\mathbf{x})$ 的 m 分别为 $m = 5,7,10.$ $0 \le x_j \le 10$. 这些函数分别有 5, 7 和 10 个局部最优解. $\mathbf{x}_{\text{local\_opt}} \approx a_i$, $f(\mathbf{x}_{\text{local\_opt}}) \approx 1/c_i$, 对 $1 \le i \le m$. 相关系数在表 F1-4 中给出.

表 F1-4 Shekel's Famaily 的参数

i	1	2	3	4	5	6	7	8	9	10
$a_{i,1}$	4	1	8	6	3	2	5	8	6	7
$a_{i,2}$	4	1	8	6	7	9	5	1	2	3.6
$a_{i,3}$	4	1	8	6	3	2	3	8	6	7
$a_{i,4}$	4	1	8	6	7	9	3	1	2	3.6
c_i	0.1	0.2	0.2	0.4	0.4	0.6	0.3	0.7	0.5	0.5

21. Neumaire 3 Function

$$f_{24}(\mathbf{x}) = \sum_{i=1}^{D}(x_i - 1)^2 + \sum_{i=2}^{D} x_i x_{i-1} + \frac{D(D+1)(D-1)}{6}, \qquad -D^2 \le x_i \le D^2$$

$\min(f_{24}(\mathbf{x})) = 0.$

22. Salomon Function

$$f_{25}(\mathbf{x}) = 1 - \cos(2p\,\|\mathbf{x}\|) + 0.1\|\mathbf{x}\|, \text{ where } \|\mathbf{x}\| = \sum_{i=1}^{D} x_i, \qquad -100 \le x_i \le 100$$

$\min(f_{25}(\mathbf{x})) = 0.$

23. Alpine Function

$$f_{26}(\mathbf{x}) = \sum_{i=1}^{D}\left|x_i \sin(x_i) + 0.1x_i\right|, \qquad -10 \le x_i \le 10$$

$\min(f_{26}(\mathbf{x})) = 0.$

附录 2　多目标优化函数

§F2.1　无约束多目标测试函数

1. ZDT1 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = x_1,\\
&f_2(\mathbf{x}) = g(\mathbf{x})[1-(x_1 / g(\mathbf{x}))^{0.5}],\\
&g(\mathbf{x}) = 1+9(\textstyle\sum_{i=2}^{n} x_i)/(n-1).
\end{aligned}
$$

$n = 30, x_i \in [0,1].$

2. ZDT2 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = x_1,\\
&f_2(\mathbf{x}) = g(\mathbf{x})[1-(x_1 / g(\mathbf{x}))^{2}],\\
&g(\mathbf{x}) = 1+9(\textstyle\sum_{i=2}^{n} x_i)/(n-1).
\end{aligned}
$$

$n = 30, x_i \in [0,1].$

3. ZDT3 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = x_1,\\
&f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{x_1/g(\mathbf{x})}-x_1\sin(10\pi x_1)/g(\mathbf{x})],\\
&g(\mathbf{x}) = 1+9(\textstyle\sum_{i=2}^{n} x_i)/(n-1).
\end{aligned}
$$

$n = 30, x_i \in [0,1].$

4. ZDT4 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = x_1,\\
&f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{x_1 / g(\mathbf{x})}],\\
&g(\mathbf{x}) = 1+10(n-1)+\textstyle\sum_{i=2}^{n}[x_i^2-10\cos(4p\, x_i)].
\end{aligned}
$$

$n = 10, x_1 \in [0,1], x_i \in [-5,5].$

5. ZDT6 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = 1-\exp(-4x_1)\sin^6(6p\, x_1),\\
&f_2(\mathbf{x}) = g(\mathbf{x})[1-(f_1(\mathbf{x}) / g(\mathbf{x}))^{2}],\\
&g(\mathbf{x}) = 1+9[(\textstyle\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}.
\end{aligned}
$$

$n = 10, x_i \in [0,1].$

6. ZDT1.1 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = x_1,\\
&f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{x_1 / g(\mathbf{x})}],\\
&g(\mathbf{x}) = 1+9(\textstyle\sum_{i=2}^{n}(x_i-x_1)^2)/(n-1).
\end{aligned}
$$

$n=10, x_i \in [0,1].$

7. DTLZ1 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = 0.5x_1x_2(1+g(\mathbf{x}_M)),\\
&f_2(\mathbf{x}) = 0.5x_1(1-x_2)(1+g(\mathbf{x}_M)),\\
&f_3(\mathbf{x}) = 0.5(1-x_1)(1+g(\mathbf{x}_M)),\\
&g(x_M) = 100[|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M}(x_i-0.5)^2 - \cos(20p(x_i-0.5))].
\end{aligned}
$$

$n=12, x_i \in [0,1].$

8. DTLZ7 Problem

$$
\begin{aligned}
&f_1(\mathbf{x}) = x_1,\\
&f_2(\mathbf{x}) = x_2,\\
&f_3(\mathbf{x}) = (1+g(\mathbf{x}_M))h(f_1, f_2, g)\\
&g(\mathbf{x}_M) = 1+\frac{9}{|\mathbf{x}_M|}\sum_{x_i \in \mathbf{x}_M} x_i,\\
&h(f_1, f_2, g) = 3-\sum_{i=1}^{2}[\frac{f_i}{1+g}(1+\sin(3p\ f_i))].
\end{aligned}
$$

$n=22, x_i \in [0,1].$

§ F2.2 约束多目标测试函数

F2.2.1 标准测试函数

1. OSY Problem

$$
\min\begin{cases} f_1(\mathbf{x}) = -25\left[\sum_{i=1}^{2}(x_i-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2\right] \\ f_2(\mathbf{x}) = \sum_{i=1}^{6} x_i^2 \end{cases}
$$

subject to:

$$
\begin{cases}
e_1(\mathbf{x}) = x_1 + x_2 - 2 \ge 0\\
e_2(\mathbf{x}) = 6 - x_1 - x_2 \ge 0\\
e_1(\mathbf{x}) = 2 + x_1 - x_2 \ge 0\\
e_1(\mathbf{x}) = 2 - x_1 + 3x_2 \ge 0\\
e_1(\mathbf{x}) = 4 - (x_3-3)^2 - x_4 \ge 0\\
e_1(\mathbf{x}) = (x_5-3)^2 + x_6 - 4 \ge 0
\end{cases}
$$

$n=6, 0 \le x_1, x_2, x_6 \le 10, 1 \le x_3, x_5 \le 5, 0 \le x_4 \le 6.$

2. SRN Problem

$$
\min\begin{cases} f_1(\mathbf{x}) = 2 + (x_1-2)^2 + (x_2-1)^2 \\ f_2(\mathbf{x}) = 9x_1 - (x_2-1)^2 \end{cases}
$$

subject to:

$$\begin{cases} e_1(\mathbf{x}) = 225 - x_1^2 - x_2^2 \geq 0 \\ e_2(\mathbf{x}) = 10 - x_1 + 3x_2 \geq 0 \end{cases}$$

$n = 2, x_i \in [-20, 20].$

3. TNK Problem

$$\min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = x_2 \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x}) = x_1^2 + x_2^2 - 1 - 0.1\cos(16\arctan(x_1 / x_2)) \geq 0 \\ e_2(\mathbf{x}) = 0.5 - \sum_{i=1}^{2}(x_i - 0.5)^2 \geq 0 \end{cases}$$

$n = 2, x_i \in [0, p\,].$

4. Kita Problem

$$\min \begin{cases} f_1(\mathbf{x}) = x_2 - x_1^2 \\ f_2(\mathbf{x}) = 0.5x_1 + x_2 + 1 \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x}) = 6.5 - x_1 / 6 - x_2 \geq 0 \\ e_2(\mathbf{x}) = 7.5 - x_1 / 2 - x_2 \geq 0 \\ e_3(\mathbf{x}) = 30 - 5x_1 - x_2 \geq 0 \end{cases}$$

$n = 2, x_i \in [0, +\infty)$

5. Tamaki Problem

$$\min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = x_2 \\ f_3(\mathbf{x}) = x_3 \end{cases}$$

subject to:

$e_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 \geq 0$

$n = 3, x_i \in [0,1].$

6. DTLZ8 Problem

$$\min \begin{cases} f_1(\mathbf{x}) = \dfrac{1}{10}\sum_{i=1}^{10} x_i \\ f_2(\mathbf{x}) = \dfrac{1}{10}\sum_{i=11}^{20} x_i \\ f_3(\mathbf{x}) = \dfrac{1}{10}\sum_{i=21}^{30} x_i \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x}) = f_3(\mathbf{x}) + 4f_1(\mathbf{x}) - 1 \geq 0 \\ e_2(\mathbf{x}) = f_3(\mathbf{x}) + 4f_2(\mathbf{x}) - 1 \geq 0 \\ e_3(\mathbf{x}) = 2f_3(\mathbf{x}) + f_1(\mathbf{x}) + f_2(\mathbf{x}) - 1 \geq 0 \end{cases}$$

$n = 30, x_i \in [0,1].$

7. DTLZ9 Problem

$$\min\begin{cases} f_1(\mathbf{x})=\dfrac{1}{10}\sum_{i=1}^{10}x_i^{0.1} \\ f_2(\mathbf{x})=\dfrac{1}{10}\sum_{i=11}^{20}x_i^{0.1} \\ f_3(\mathbf{x})=\dfrac{1}{10}\sum_{i=21}^{30}x_i^{0.1} \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x})=f_3^2(\mathbf{x})+f_1^2(\mathbf{x})-1\ge 0 \\ e_2(\mathbf{x})=f_3^2(\mathbf{x})+f_2^2(\mathbf{x})-1\ge 0 \end{cases}$$

$n=30, x_i\in[0,1].$

F2.2.2 工程优化实例

1. Two-bar truss design

$$\min\begin{cases} f_{\text{volume}}=f_1(\mathbf{x})=x_1(16+y^2)^{0.5}+x_2(1+y^2)^{0.5} \\ f_{\text{stress,AC}}=f_2(\mathbf{x})=20(16+y^2)^{0.5}/(x_1y) \end{cases}$$

subject to:

$$\begin{cases} f_{\text{volume}}\le 0.1 \\ f_{\text{stress,AC}}\le 100{,}000 \\ f_{\text{stress,BC}}=80(1+y^2)^{0.5}/(x_2y)\le 100{,}000 \end{cases}, \quad 1\le y\le 3, x_1, x_2\ge 0.$$

2. Welded beam design

$$\min\begin{cases} f_1(\mathbf{x})=1.10471h^2l+0.04811tb(14.0+t) \\ f_2(\mathbf{x})=d(\mathbf{x})=2.1952/(t^3b)\%\dfrac{2.1952}{t^3b} \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x})=13{,}600-t(\mathbf{x})\ge 0 \\ e_2(\mathbf{x})=30{,}000-s(\mathbf{x})\ge 0 \\ e_3(\mathbf{x})=b-h\ge 0 \\ e_4(\mathbf{x})=\text{Pc}(\mathbf{x})-6{,}000\ge 0 \end{cases}$$

where:

$$\begin{cases} t(\mathbf{x})=\sqrt{(t')^2+(t'')^2+\dfrac{lt't''}{\sqrt{0.25(l^2+(h+t)^2)}}} \\ t'=6{,}000/(\sqrt{2}hl) \\ t''=\dfrac{6{,}000(14.0+0.5l)\sqrt{0.25(l^2+(h+t)^2)}}{2[0.707hl(l^2/12+0.25(l^2+(h+t)^2))]} \\ s(\mathbf{x})=504{,}000/(t^2b) \\ \text{Pc}(\mathbf{x})=64{,}764.022(1-0.0282346t)tb^3 \\ 0.125\le h,b\le 5.0, \text{and} 0.1\le l,t\le 10.0 \end{cases}$$

3. Speed reducer design

$$\min\begin{cases} f_{weight}=f_1(\mathbf{x})=0.7854x_1x_2^2(10x_3^2/3+14.933x_3-43.0934)-1.508x_1(x_6^2+x_7^2) \\ \qquad +7.477(x_6^3+x_7^3)+0.7854(x_4x_6^2+x_5x_7^2) \\ f_{stress}=f_2(\mathbf{x})=\dfrac{\sqrt{(\frac{745x_4}{x_2x_3})^2+1.69\times10^7}}{0.1x_6^3} \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x})=1/27-1/(x_1x_2^2x_3)\geq0 \\ e_2(\mathbf{x})=1/397.5-1/(x_1x_2^2x_3^2)\geq0 \\ e_3(\mathbf{x})=1/1.93-x_4^3/(x_2x_3x_6^4)\geq0 \\ e_4(\mathbf{x})=1/1.93-x_5^3/(x_2x_3x_7^4)\geq0 \\ e_5(\mathbf{x})=40-x_2x-3\geq0 \\ e_6(\mathbf{x})=12-x_1/x_2\geq0 \\ e_7(\mathbf{x})=x_1/x_2-5\geq0 \\ e_8(\mathbf{x})=x_4-1.5x_6-1.9\geq0 \\ e_9(\mathbf{x})=x_5-1.1x_7-1.9\geq0 \\ e_{10}(\mathbf{x})=1{,}300-f_2(\mathbf{x})\geq0 \\ e_{11}(\mathbf{x})=1{,}100-\dfrac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2+1.275\times10^{-8}}}{0.1x_7^3}\geq0 \end{cases}$$

$2.6\leq x_1\leq3.6, 0.7\leq x_2\leq0.8, 17\leq x_3\leq28, 7.3\leq x_4, x_5\leq8.3, 2.9\leq x_6\leq3.9, 5.0\leq x_7\leq5.5.$

4. Disc brake design

$$\min\begin{cases} f_1(\mathbf{x})=4.9\times10^{-5}(x_2^2-x_1^2)(x_4-1) \\ f_2(\mathbf{x})=\dfrac{9.82\times10^6(x_2^2-x_1^2)}{x_3x_4(x_2^3-x_1^3)} \end{cases}$$

subject to:

$$\begin{cases} e_1(\mathbf{x})=(x_2-x_1)-20\geq0 \\ e_2(\mathbf{x})=30-2.5(x_4+1)\geq0 \\ e_3(\mathbf{x})=0.4-x_3/(3.14(x_2^2-x_1^2))\geq0 \\ e_4(\mathbf{x})=1-\dfrac{2.22\times10^{-3}x_3(x_2^3-x_1^3)}{(x_2^2-x_1^2)^2}\geq0 \\ e_5(\mathbf{x})=\dfrac{2.66\times10^{-2}x_3x_4(x_2^3-x_1^3)}{(x_2^2-x_1^2)}-900\geq0 \end{cases}$$

$55\leq x_1\leq80, 75\leq x_2\leq110, 1000\leq x_3\leq3000, 2\leq x_4\leq20.$