汕頭大學
SHANTOU UNIVERSITY

硕 士 学 位 论 文

题　　目： 约束进化算法的研究及应用

英文题目： **Research and Application of Constrained Evolutionary Algorithm**

姓　　名 方毅　**学　　号** 11509034

所在学院 工学院　**导师姓名** 范　衡

专　　业 信息与通信工程

入学日期 2015 年 9 月　**答辩日期** 2018 年 6 月

学位论文原创性声明

本论文是我个人在导师指导下进行的工作研究及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其他机构已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在论文中以明确方式标明。本人完全意识到本声明的法律责任由本人承担。

作者签名：方毅　　日期：2018 年 6 月 4 日

学位论文使用授权声明

本人授权汕头大学保存本学位论文的电子和纸质文档，允许论文被查阅和借阅；学校可将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存和汇编论文；学校可以向国家有关部门或机构送交论文并授权其保存、借阅或上网公布本学位论文的全部或部分内容。对于保密的论文，按照保密的有关规定和程序处理。

作者签名：方毅　　导师签名：

日期：2018 年 6 月 4 日　　日期：2018 年 6 月 4 日

摘 要

进化算法是一类应用于求解复杂优化问题的基于种群搜索的启发式算法，其具有并行性，全局性，鲁棒性和非线性可求解等特点，适用于复杂数值优化求解及工程应用优化上，在近年来受到国内外专家学者广泛关注。一般来说，约束进化算法可以认为是集合了约束处理机制的可以解决约束优化问题的进化算法。很多情况下，连续数值优化问题及现实工程优化问题同时也是约束优化问题，根据优化问题的目标数目，通常这些问题可以划分为单目标约束优化问题、多目标约束优化问题及高维目标约束优化问题。本文主要针对前两类约束优化问题，分别提出对应的基于进化算法的约束处理机制，并应用于经典的测试问题集和现实工程应用上，具体如下：

1、针对约束单目标优化问题，本文在一种自适应更新算法参数的差分进化算法 LSHADE44 框架上，提出一种改进的ϵ约束处理机制 IEpsilon。IEpsilon 在经典的ϵ约束处理机制上，根据进化过程中种群可行比例情况，自适应调节ϵ数值。实验结果表明，在 CEC2017 单目标约束优化测试问题集及两个实际工程优化问题上，LSHADE44-IEpsilon 都显著优于 LSHADE44 及结合了原始ϵ约束处理机制的 LSHADE44-Epsilon。

2、针对约束多目标优化问题，设计了一种应用于多目标进化算法 MOEA/D 框架上的基于角度的约束支配处理机制。这种机制在进化过程中，通过解间的角度信息，维持种群多样性，并加强种群的收敛性。实验结果表明，在 LIR-CMOP 测试问题集及 I 型悬臂梁优化问题上，结合该机制的 MOEA/D 都显著优于其他 4 种经典约束多目标进化算法。

关键词：进化算法，单目标优化，多目标优化，约束处理机制

Abstract

Evolutionary algorithms (EAs) are a kind of population-based heuristic algorithms, which can be used to solve some complex optimization problems. The properties of EAs include parallel computing, globality, robustness, and nonlinear solvability. EAs are generally applied to complex numeric optimization problems and engineering application optimization problems, which have attracted a lot of attention of experts and scholars at home and abroad. In general, constrained evolutionary algorithms (CEAs) can be regarded as a kind of EAs, which apply different constraint-handling techniques to solve constrained optimization problems (COPs). According to the number of optimized objectives, COPs can be classified into constrained single-objective optimization problems (CSOPs), constrained multi-objective optimization problems (CMOPs) and constrained many-objective optimization problems (CMaOPs). This paper focused on the study of CSOPs and CMOPs, and it proposes three EA-based constraint handing techniques benchmarks and real-world engineering application, which are detailed as follows:

1) The paper proposes an improved version of ϵ constraint handing technique IEpsilon, which are embedded in the framework of the adaptive differential evolution(DE) algorithm LSHADE44, to solve CSOPs. Based on the rules of ϵ constraint handing technique, the IEpsilon can adjust the value of ϵ adaptively according to the proportion of feasible solutions in the current generation during the evolutionary process. The experimental results show that LSHADE44 outperforms LSHADE44 and LSHADE-Epsilon which is combined with the original ϵ method on CEC2017 benchmarks and two real-world engineering optimization problems.

2) The paper proposes an angle-based constrained domination principle (ACDP), which is applied to the framework of the multi-objective optimization evolutionary algorithm MOEA/D, to solve CMOPs. The principle uses the angle information of solutions to maintain the diversity of the population and enhance the convergence of the population. The experimental results manifest that the MOEA/D with ACDP

outperform 4 other classical constrained multi-objective optimization evolutionary algorithms on the LIR-CMOP benchmarks and an I-beam optimization problem.

Keywords: evolutionary algorithm, single-objective optimization, multi-objective optimization, constraint-handling technique

目　录

摘　要.....I
Abstract.....II
目　录.....IV
第 1 章　绪　论.....1
1.1 课题研究背景与意义.....1
1.2 进化算法发展简述.....4
1.3 约束处理机制国内外研究现状.....5
1.3.1 传统约束处理机制.....5
1.3.2 基于进化算法的约束处理机制研究现状.....6
1.3 主要研究内容及结构安排.....11
第 2 章　自适应差分进化算法 LSHADE44.....13
2.1 差分进化算法.....13
2.1.1 差分进化算法操作算子.....14
2.1.2 自适应差分进化算法研究简述.....16
2.2 LSHADE44 算法.....17
2.2.1 基于成功历史存档的参数设置方法.....17
2.2.2 线性种群规模缩减.....19
2.2.3 差分策略竞赛选择机制.....19
2.2.4 LSHADE44 算法框架.....20
2.3 本章小结.....22
第 3 章　基于改进 Epsilon 约束处理机制的 LSHADE44 算法.....23
3.1 LSHADE44-IEpsilon 算法.....23
3.1.1 Epsilon 约束处理机制.....23
3.1.2 改进的 Epsilon 约束处理机制 IEpsilon.....24
3.1.3 结合两种 Epsilon 约束处理机制的 LSHADE44.....25
3.2 实验研究.....26
3.2.1 单目标约束优化测试问题集.....26
3.2.2 工程优化测试问题.....27

3.2.3 实验参数设置及评估指标 30
3.2.4 实验结果及分析 31
3.3 本章小结 42
第 4 章 多目标进化算法 44
4.1 多目标进化算法基础 44
4.1.1 基于 Pareto 规则的多目标最优解集 44
4.1.2 多目标进化算法研究概述 45
4.1.3 多目标进化算法性能评估指标 47
4.2 基于分解的多目标进化算法 MOEA/D 49
4.2.1 MOEA/D 分解方法 49
4.2.2 MOEA/D 邻居机制 52
4.3 本章小结 52
第 5 章 基于角度约束支配规则的 MOEA/D 算法 53
5.1 约束多目标进化算法 MOEA/D-ACDP 53
5.1.1 基于角度的约束支配规则 ACDP 53
5.1.2 ACDP 机制进化过程 56
5.1.3 ACDP 阈值参数设置 58
5.1.4 MOEA/D-ACDP 算法流程 59
5.2 多目标约束优化测试问题集 60
5.3 工程优化问题：I 型焊接梁优化问题 63
5.4 实验研究 65
5.4.1 几种经典的基于分解的约束多目标进化算法 65
5.4.2 实验参数设置 66
5.5 实验分析及结论 67
5.5.1 LIR-CMOP 测试问题实验结果 67
5.5.2 I-型焊接臂优化问题实验结果 72
5.6 本章小结 74
总结与展望 76
参考文献 78

致　谢 87
附　录 88
攻读硕士学位期间主要的工作成果 95

第 1 章 绪 论

“物竞天择，适者生存”是达尔文《物种起源》中进化论的核心思想[1]。自然界总是变幻莫测的，根据达尔文的理论，随着自然环境的变化，自然界中生存的各种物种不会一直保持不变，物种中的部分个体会跟随外界变化产生相应的变异。最终通过环境的选择及淘汰，适应性强的变异基因能够在物种中保留下来。所以物种总是在不断进化的，而进化的过程总是低级通往高级，简单通往复杂的过程。

为了求解复杂的数值优化问题，受到进化论的启发，研究人员设计出了一类模仿自然界生物演化行为的算法——进化算法（Evolutionary Algorithms, EAs）。进化算法模拟生物遗传进化中自学习、自适应、自解决问题的过程，将生物进化的四个基本操作：繁殖（Reproduction）、重组（Recombination）、竞争（Competition）和选择（Selection）纳进到算法的框架中，是一种模仿生物演化行为的随机搜索优化算法。

随着进化算法不断被研究开发，各类数值优化求解及工程应用优化问题开始广泛应用进化算法，并取得了非常良好的效果。但是，由于此类问题通常带有约束[2]，所以进化算法方向的研究也必然引申到对相关约束处理机制的研究。

1.1 课题研究背景与意义

通常而言，最优化问题即是求最小化（或最大化）问题，为了方便叙述，本文统一用最小化问题来表示最优化问题。

一个最优化问题可以描述为以下形式：

$$\begin{aligned} &\text{minimize} \quad \vec{F}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x})) \\ &\text{subject to} \quad g_i(\vec{x}) \le 0, \quad i = 1, ..., p; \\ &\qquad\qquad h_j(\vec{x}) = 0, \quad j = 1, ..., q. \end{aligned} \tag{1-1}$$

其中，$\vec{F}(\vec{x})$ 即为目标函数，而 $\vec{x} = (x_1, x_2, ..., x_D \in R)$ 就是决策变量。当 $m = 1$，这是一个单目标优化问题，当 $2 \le m \le 3$，这是多目标优化问题。$R = \prod_{k=1}^{D} [L_k, U_k]$ 代表 D 维决策空间，U_i、L_i 分别代表决策向量上第 i 位的取值上下界。最优化问题

的约束一般由等式约束和不等式约束组成，式子(1-1)中 $g_i(\vec{x}) \le 0$ 是第 i 个不等式约束，而 $h_j(\vec{x}) = 0$ 是第 j 个等式约束。

根据文献[3]，规定 $G_i(\vec{x}) = |\max(g_i(\vec{x}), 0)|$ 为第 i 个不等式约束的约束违反值，$H_j(\vec{x}) = |h_j(\vec{x}) - \delta|$ 为等式约束违反值。δ 为等式约束违反容忍因子，本文 δ 均取为 0.0001。

同时，规定个体 $\vec{x}$ 的总约束违反程度为 $\phi(\vec{x})$，定义如下：

$$\phi(\vec{x}) = \sum_{i=1}^{p} G_i(\vec{x}) + \sum_{j=1}^{q} H_j(\vec{x}) \tag{1-2}$$

对于个体 $\vec{x}$，如果 $\phi(\vec{x}) = 0$，则 $\vec{x}$ 被称为可行解。如果 $\phi(\vec{x}) > 0$，则 $\vec{x}$ 被称为不可行解。而决策空间中所有可行解组成的区域就被称为可行区 S，所有不可行解组成的区域就被称为不可行区 S 。

一般地，根据约束的数目及目标的个数，最优化问题可以分为以下 6 类：（1）单目标无约束优化问题（ $m=1$； $p, q=0$ ）；（2）单目标约束优化问题（ $m=1$； $p+q\neq 0$ ）；（3）多目标无约束优化问题（ $2\le m\le 3$； $p, q=0$ ）；（4）多目标约束优化问题（ $2\le m\le 3$； $p+q\neq 0$ ）；（5）高维无约束优化问题（ $m>3$； $p, q=0$ ）；（6）高维约束优化问题（ $m>3; p+q\neq 0$ ）。本文的主要研究对象就是第 2 类和第 4 类问题。

为了求解最优化问题，一些经典的优化搜索算法相继被提出，其中包括梯度下降法[4]、牛顿法[5]、禁忌搜索（Tabu Search, TS）[6]、模拟退火（Simulate Annealing, SA）[7]等等。

梯度下降法又名最速下降法，其原理是将解的反梯度作为搜索方向，反复搜索迭代，从而逼近最优解。梯度下降法要求所求目标函数是可知的，同时是可导，所以此类方法不适用于黑箱问题。同时梯度下降法要求目标函数是凸的，这样才能保证求解出来的解是全局最优解，否则只能使求到的最终解陷入局部最优，所以对应这种方法，初值的设置十分重要。牛顿法可以在实数域和复数域上进行优化搜索的二阶收敛算法，收敛速度比梯度下降法快。缺点是这种方法需要求解目

标函数的 Hessian 矩阵的逆矩阵，计算复杂，并且需要同时知道目标函数的一阶导数和二阶导数。禁忌搜索是一种贪婪式的搜索方法，它基于领域局部搜索，引入禁忌表存储搜索过的局部最优解，防止当前解反复进入局部最优，同时这种算法也会在一定情况下放松禁忌。模拟退火法模拟金属退火原理，其原理可简述为在解的邻域生成子解后，根据当前温度使用 Metropolis 准则，选择是否接受子解为新解。该概率接受的子解有可能是差解，所以这种方法可以避免优化结果陷入局部最优。但模拟退火法参数难以预先设定，需通过反复试验确定，所以并不算是一种高效的优化方法。

可见上述方法都有各自各样的缺点，主要表现在全局性较差、可解决的问题条件严苛以及无法解决现实常见的黑箱问题等等。

进化算法相比其它经典算法，由于其并行性计算的特点，全局性搜索更为良好，同时也具有更好的鲁棒性和高效性。进化算法在解决多峰优化问题上，不易陷入局部最优。同时，由于进化算法在求解过程中不需要额外的梯度信息及其他先验信息，不受限于所求函数的凸性和连续性，所以进化算法可以广泛应用于复杂黑箱问题及非线性问题求解。进化算法适用于有限步骤内难以短时间求得最优解的问题，尤其是 NP 难问题，所以进化算法尤其适合于现实中经常出现的复杂优化问题。

原理上，进化算法并非由从单点搜索全局最优解，而是一种通过种群多点并行搜索的机制。一般地，进化算法初始化生成一个种群，通过种群的个体交叉变异后生成新的子解，并通过一定的评价标准对种群进行筛选，不断更新种群，如此反复迭代直到最终满足终止条件，得到最优解（或满意解）。一般的进化算法流程可表示为图 1-1。

在现实生活及工程应用中，诸如复杂机电集成系统，人工智能设计，送餐规划，出行路线规划，车间资源调度，及电网电力调度等等，都能归结为最优化问题。它们覆盖了我们生活大大小小的各个方面。同时，这类问题也通常都是单目标或多目标约束优化问题，这些类型的问题通常是普通的算法难以有效解决。基于此应用背景，本文开展对进化算法约束处理机制的研究，对于进一步拓展进化算法的适用性，从而解决现实生活的各类应用优化问题具有非常重要的意义。

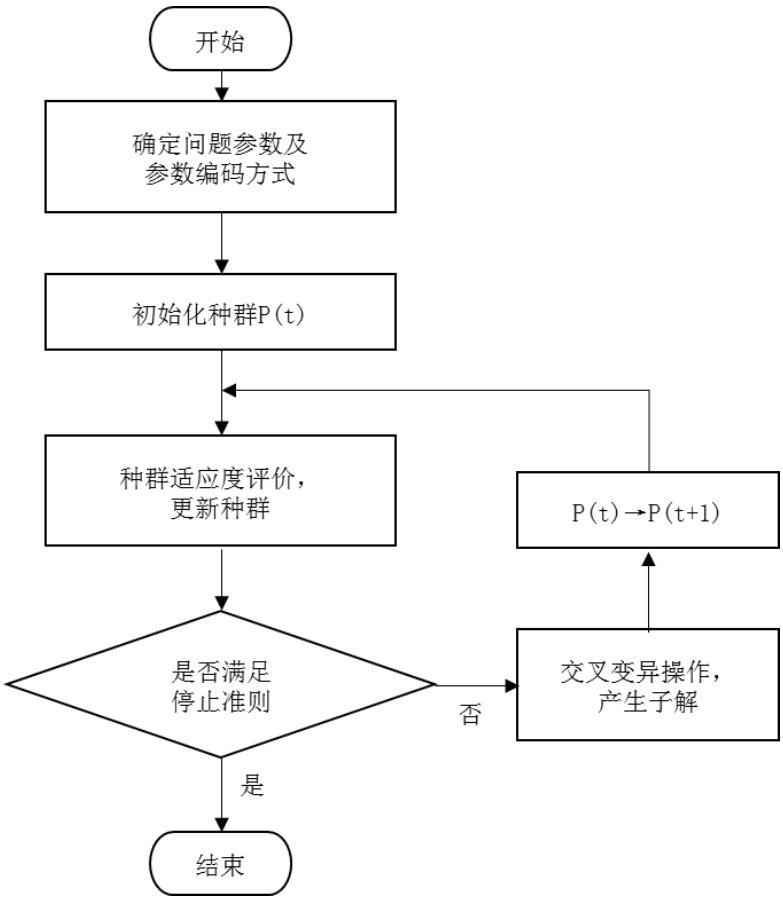


图 1-1 进化算法基本流程框图

1.2 进化算法发展简述

进化算法和其他的研究学科一样，都经过了长期的成长摸索与发展。

早在 20 世纪 50 年代后期，开始有生物遗传学家借用计算机的 0-1 编码方式，对生物遗传系统进行人工标识，这意味着遗传算法的编码雏形开始出现。1965 年德国的 Rechenberg 等人提出了进化策略（Evolutionary Strategy, ES）[8]的概念，同年美国的 Fogel 提出了进化规划（Evolutionary Programming, EP）[9]。1968 年 Holland 正式提出了遗传算法（Genetics Algorithm, GA）[10]的基础概念。20 世纪 90 年代，美国的 Koza 等人提出了遗传程序设计（Genetics Programming, GP）[11]。至此，ES、EP、GA 和 GP 正式成为进化算法的四大经典分支[12]，并在随后 20 多年里得到了快速发展。

如今进化算法已经逐渐成为了计算机科学领域内广受关注的学科，并且拥有众多的研究学者投入研究。种类众多的进化算法相继被提出，这些算法包括蚁群算法[13]、粒子群算法[14]、差分进化算法[15]、文化基因算法[16]、人工免疫算法[17]

等等。

根据进化算法所优化的问题目标数目，一般进化算法可分类为单目标进化算法、多目标进化算法和高维目标进化算法。早期研究的进化算法大多都是单目标进化算法，而近些年来多目标进化算法[18][19]、高维目标进化算法[20]也开始得到广泛关注和发展。

1.3 约束处理机制国内外研究现状

事实上处理约束优化问题是非常困难的，同时这个难度随着问题的目标函数数目、约束条件（尤其是等式约束）数目、决策变量个数的增加，也会显著提升。所以一个优异的约束处理机制在解决约束优化问题时起到非常重要的作用。

本小节将会分别对已有的传统约束处理机制及应用在进化算法上的约束处理机制国内外研究现状作简要概述。

1.3.1 传统约束处理机制

本文主要列举两种传统的约束处理方法[21]。

1）拉格朗日法

这种方法将约束问题转化为无约束拉格朗日方程。拉格朗日方程的一般形式如下式：

$$l(\vec{x},\vec{\lambda},\vec{\mu}) = f(\vec{x}) + \vec{\lambda}^T \vec{h}(\vec{x}) + \vec{\mu}^T \vec{g}(\vec{x}) \tag{1-3}$$

这里 $\vec{h}(\vec{x}) = [h_1(\vec{x}), h_2(\vec{x}), ..., h_q(\vec{x})]$ 及 $\vec{g}(\vec{x}) = [g_1(\vec{x}), g_2(\vec{x}), ..., g_p(\vec{x})]$ 为等式及不等式约束向量。$\vec{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_q]$ 和 $\vec{\mu} = [\mu_1, \mu_2, ...\mu_p]$ 都是方程的乘子。

当优化问题存在不等式约束时，只有满足 Karush–Kuhn–Tucker（KKT）条件时，求得的结果才是最优解。KKT 条件如下：（1）$l(\vec{x},\vec{\lambda},\vec{\mu})$ 对 $\vec{x}$ 求导为 0；（2）所有的 $\vec{h}(\vec{x}) = \vec{0}$；（3）$\vec{\mu}^T \vec{g}(\vec{x}) = 0$。

2）投影法

这种方法类似于梯度下降法，其思想是把每一个不可行解转化为离它最近的可行解。例如，这里对一个当前解 $\vec{x}(t)$ 根据一定步长 α_t 和搜索方向 $\vec{d_t}$ 进行逐步搜

索，即下一个搜索的解 $\vec{x}(t)+\alpha_t\vec{d_t}$。然后寻找离 $\vec{x}(t)+\alpha_t\vec{d_t}$ 最近的可行解作为解的下一个位置 $\vec{x}(t+1)$。

这两种方法显然不适合于求解一些复杂的约束优化问题或者黑箱问题。首先拉格朗日法需要满足严苛的 KKT 条件，才能求得最优解。而投影法则需要额外的函数梯度信息，并要求能够在优化过程中较易地搜索到可行替代解，只有这样才能使优化处理机制有效运行。所以在解决求解约束优化问题时，本文只考虑基于进化算法的约束处理机制。

1.3.2 基于进化算法的约束处理机制研究现状

伴随着进化算法的应用，基于进化算法的约束处理机制得到了国内外研究学者的深入研究[22]。根据文献[23]，应用于进化算法的约束处理机制可以分为四类：（1）罚函数法；（2）修复方法；（3）混合方法；（4）分离方法。

1.3.2.1 罚函数法

这类方法都是通过对约束优化问题的目标函数添加上一个数值（即惩罚项）转化为无约束优化问题。而这个数值一般是基于违反约束的数目或数值乘以惩罚因子确定的。

一般的罚函数的表达形式如下：

$$\min\gamma(\vec{x},M)=f(\vec{x})+\sum_{i=1}^{p}C_iG_i(x)+\sum_{j=1}^{q}K_jH_j(x) \tag{1-4}$$

当各惩罚因子 C_i 和 K_j 越大时，我们可以认为罚函数对约束的惩罚程度将越大。此时在罚函数的优化过程中，算法会更偏重于寻找可行解，同时对于目标函数数值优化的偏重程度会略弱些，反之则相反。所以罚函数法对各惩罚因子的设置是十分敏感的，事实上应用这一类方法，进化算法在优化过程中难以处理好目标函数优化及约束处理的偏重。

根据如何确定惩罚因子和构造罚函数，众多研究学者提出了几种不同的罚函数方法。

1）静态罚函数法[24]

这种方法使用公式(1-4)中函数构造形式，同时各个惩罚因子根据预设的惩罚

级别分别设定好，并保持固定不变。但这种方法参数设置十分依赖于实际优化问题，并且由于设值单一，实际效果并不佳。

2）动态罚函数法[25]

这种方法随着迭代次数的增加，逐渐增加惩罚项的值。具体的惩罚函数构造形式如下式：

$$\min \gamma(\vec{x}) = f(\vec{x}) + (C \cdot t)^{\alpha} \cdot SVC(\beta, \vec{x}) \tag{1-5}$$

式中α，β和C为预设参数，t为当前优化代数。

同时，

$$SVC(\beta, \vec{x}) = \sum_{i=1}^{p} (G_i(\vec{x}))^{\beta} + \sum_{j=1}^{q} H_j(\vec{x}) \tag{1-6}$$

这种方法前期比较注重对目标的优化求解，后期注重搜索可行解。

3）自适应罚函数法[26]

这种方法根据搜索进程的反馈来自适应修改惩罚因子。具体的惩罚函数构造形式如下式所示：

$$\min \gamma(\vec{x}, M) = f(\vec{x}) + \lambda(t) \cdot (\sum_{i=1}^{p} (G_i(x))^2 + \sum_{j=1}^{q} H_j(x)) \tag{1-7}$$

惩罚因子$\lambda(t)$下一时刻根据下列规则进行更新

$$\lambda(t+1) = \begin{cases} \lambda(t)/\beta_1 & \text{如果近}K\text{次迭代种群最优解都是可行解} \\ \beta_2 \cdot \lambda(t) & \text{如果近}K\text{次迭代种群最优解都是不可行解} \\ \lambda(t) & \text{其它情况} \end{cases} \tag{1-8}$$

其中β_1和β_2都是给定参数且满足$\beta_1 > \beta_2 > 1$。在这种机制下，当种群主要搜索可行区域时，增强它在不可行区域的搜索，反之则增强它在可行区域的搜索。

4）退火罚函数法[27]

这种方法应用模拟退火的机制，构造惩罚函数如下形式：

$$\min \gamma(\vec{x}) = A \cdot f(\vec{x}) \tag{1-9}$$

其中A的定义如下

$$A = e^{-M/T} \tag{1-10}$$

其中M指当前个体$\vec{x}$违反的约束的数目，T为随迭代代数t增加而下降的温度，

其定义如下：

$$T = \frac{1}{\sqrt{t}} \tag{1-11}$$

在这种方法下，随着迭代次数增加，T 逐渐变小，所以 A 值受到约束违反的个数的影响越来越大，从而约束对目标函数的惩罚逐渐增加，种群逐渐趋向搜索可行解。

5）共进化罚函数法[28]

这种方法将种群分为两个规模为 M_1 和 M_2 的子种群，其中一个子种群对解进行优化，而另一个字种群对惩罚因子 W_1 和 W_2 进行优化。惩罚函数的构造形式如下所示：

$$\min \gamma(\vec{x}) = f(\vec{x}) + W_1 \cdot (\sum_{i=1}^{p} G_i(x) + \sum_{j=1}^{q} H_j(x)) + W_2 \cdot N \tag{1-12}$$

其中 N 指的是个体解 $\vec{x}$ 的约束违反数目。

6）死罚函数法[29]

这种方法直接将惩罚因子设为无穷大（即将惩罚项设为无穷大）。所以在这种方法下所有搜索过程中遇到的不可行解都被直接舍弃，而种群只会搜索可行解。

小结：罚函数法有着很明显的缺点。首先，罚函数重构了目标函数，但算法在优化过程中仍然以目标最低的路径进行搜索，这使得种群的搜索空间被改变，尤其当约束条件是非线性、离散等复杂情况时，重构函数的搜索空间可能变得更为复杂，使求得的解是较差的解。其次从原理上罚函数兼顾了目标优化及约束处理的折衷，但是在两者的偏倚上，上述各种方法包括自适应法依然很难做到平衡。

1.3.2.2 修复方法

这类方法[30][31]在每个迭代循环中，将所有的不可行解通过一定的修复方法转化为可行解。修复后得到的解（1）可以直接替代原先解，或者（2）不替代原先解，只是用于原先解的近似评估。

小结：修复方法局限性十分明显，只适用于易于寻找到可行解的优化问题。这使得这种方法在大多数情况不会作为优先考虑的约束处理方法。

1.3.2.3 混合方法

文献[37]中，进化混合使用了模糊逻辑控制的方法来处理约束。然而这种方法类似于罚函数方法，很难处理好目标函数优化及约束处理的偏重。

目标函数被重构为如下形式：

$$\gamma(\vec{x}) = f(\vec{x}) \cdot \min(\mu G_1(\vec{x}),...,\mu G_p(\vec{x}),\mu H_1(\vec{x}),...,\mu H_q(\vec{x})) \tag{1-13}$$

使用模糊函数对不等式约束及等式约束进行模糊处理：

$$\mu G_i(\vec{x}) = \mu_\sigma(a_i, s_i)(G_i(\vec{x})), \quad i = 1,2,...,p \tag{1-14}$$

$$\mu H_j(\vec{x}) = \mu_\sigma(a_j, s_j)(H_j(\vec{x})), \quad j = 1,2,...,q \tag{1-15}$$

其中 p 和 q 代表不等式约束和等式约束的数目，模糊函数定义如下

$$\mu_\sigma(a,s)(X) = \begin{cases} 1 & \text{如果} X < a \\ \dfrac{\exp(-(p+q)((X-a)/s)^2) - \exp(-(p+q))}{1-\exp(-(p+q))} & \text{如果} a < X \le a+s \\ 0 & \text{如果} X > a \end{cases} \tag{1-16}$$

其中 a 和 s 为需提前设置的参数因子。

小结：这类方法使用其他学科的方法辅助解决进化算法进程中对约束条件的处理，是一种不错的思路，但是这种方法目前研究较少。同时上述提到结合模糊逻辑控制的混合方法与罚函数法同样重构了目标函数，重构后的目标函数搜索空间可能变得十分复杂，使求得的解是较差的解，本文中不作进一步讨论。

1.3.2.4 分离方法

这类方法是进化算法约束处理机制中当前应用最广泛的一类方法，其核心思想就是将目标及约束分开处理。既避免了构造新的罚函数的参数设置问题，又能同时考虑到目标函数优化及约束处理。

1）多目标分离法[32]

这种方法将原先优化问题的约束条件直接也当成目标进行优化，从而将约束优化问题目标升维后转化为无约束优化问题。其目标函数构造方法如下所示：

$$\vec{F}^*(\vec{x}) = (\vec{F}(\vec{x}), f_{m+1}(\vec{x}),... f_{m+p}(\vec{x}), f_{m+p+1}(\vec{x}),..., f_{m+p+q}(\vec{x})) \tag{1-17}$$

其中，$\vec{F}(\vec{x})$ 代表原 m 维目标函数，$f_{m+1}(\vec{x}),... f_{m+p}(\vec{x})$ 代表 p 个不等式约束违反值，

$f_{m+p+1}(\vec{x}),...,f_{m+p+q}(\vec{x})$代表$q$个等式约束违反值。

2）共进化分离法[33]

这种方法将种群分为两个子种群，一个子种群优化解，而另一个子种群基于约束进行优化。每个子种群里的解都来自另一个子种群的筛选，如此反复迭代进行优化。

3）约束支配准则（Constrained Dominance Principle, CDP）[34]

这是由 Deb 教授提出的最广泛应用的约束处理机制，其应用于个体解间的比较，具体可归结为以下三条规则：

a. 可行解优于不可行解。

b. 两个可行解相比，目标值小的解优于另一个解。

c. 两个不可行解相比，总约束违反值小的解优于另一个解。

其中规则 a 和规则 c 可合并为一条规则：两个可行解相比，总约束违反值小的解优于另一个解。

4）随机序值法（Stochastic Ranking, SR）[35]

这种方法预先设置一个阈值参数$r_f \in [0,1]$。每次当两个解进行比较时，产生一个 0~1 之间的随机数，（1）如果这个随机数小于r_f，则拥有更小的目标函数值的解优于另一个解；（2）如果这个随机数大于r_f，则根据 CDP 准则比较这两个解。实验显示r_f设置在 0.4~0.5 之间的效果最佳。

5）ϵ约束处理方法（Epsilon Constraint-handling Method）[36]

ϵ约束处理方法可以认为是一种改进的 CDP 规则，这种方法预先设定一个阈值ϵ，然后根据以下三条规则对任意两个解进行比较。

a. 若相比较的两个解的总约束违反值都小于ϵ时，目标值小的解优于另一个解。

b. 若相比较的两个解的总约束违反值相等时，目标值小的解优于另一个解。

c. 若至少有一个解的总约束违反值大于ϵ时，总约束违反值小的解优于另一个解。

当$\epsilon = 0$时，ϵ约束处理方法等价于 CDP 规则。一般地，ϵ的取值随着迭代次数的增加动态增长。这增加了种群前期对不可行的搜索，并随着代数增加逐渐增强

种群对可行解的搜索。

小结：多目标分离法从理论上是可行的，但是这种方法在将约束优化问题转化为无约束优化问题的同时，提升了目标函数的维度，这极大的增加问题的搜索空间，增加了问题求解的难度，特别当问题目标数量和约束条件都特别多时，这种难度的增长更加明显。共进化分离法更适合于目标与约束可分离的问题，适用性有限。

CDP 方法在解间比较时将总约束违反度作为首要考量因素，在处理简单的约束优化问题时效果非常好，是一种最常用的约束处理机制。SR 及ϵ约束处理方法都在一定程度上放松了 CDP 方法的严苛条件，平衡了种群搜索过程中可行解和不可行解的取舍，在解决一些复杂的约束优化问题时，性能表现更加良好。

1.3 主要研究内容及结构安排

本文主要对约束进化算法进行研究，针对单目标及多目标约束优化问题，分别设计相应的约束进化算法，并将它们应用到各种基准测试例子及现实应用问题中。根据“没有免费午餐”理论（No Free Lunch Theorem, NFL）[38]，并不会有适用于所有问题的算法存在。所以，设计的算法最好能够在进化过程中发掘问题的特征，自反馈调整参数，降低设计的算法对某一类特定问题的特殊敏感性，这有助于提高算法拥有更强的鲁棒性。基于这种思想，本文针对单目标约束优化问题提出一种改进ϵ处理方法，根据当前种群的情况对ϵ的取值进行自适应调节。另外针对多目标约束优化问题，本文结合种群的角度信息及可行解比例提出约束进化算法，在维持种群多样性的同时，平衡算法对可行区和不可行区的搜索，加速了种群的收敛。实验证明，本文提出的这些约束进化算法均适用于现实应用优化问题求解。

本文主要的结构及安排如下：

第一章是绪论部分。这部分主要介绍了本课题的研究背景，阐明了约束进化算法的研究意义，简要介绍了进化算法的定义和发展历程，并概要介绍了现有的传统约束处理机制及基于进化算法的约束处理机制。

第二章详细地介绍了一种应用于解决单目标优化问题的 DE 算法——LSHADE44[39]。本章首先对 DE 算法做概述，然后介绍了一种基于历史成功档集

自适应更新控制参数的改进 DE 算法 LSHADE[40]，这种改进算法可以根据以往成功更新解的控制参数对记录，自动生成算法下一代的控制参数对，同时随着迭代代数的增加，种群的规模线性减少，从而增加迭代的代数，有效地提高算法的性能。最后介绍了 LSHADE 的改进算法 LSHADE44，这种算法在 LSHADE 算法的基础上提供了试验向量产生机制的自适应选择策略，进一步提高了算法对不同问题的鲁棒性。本章为下一章应用于 LSHADE44 的两种约束处理机制研究提供了框架基础。

第三章提出了一种应用于 LSHADE44 框架上的约束处理机制 IEpsilon，并对本文所提出的 LSHADE44-IEpsilon 及 LSHADE44，LSHADE44-Epsilon 在 CEC2017 测试问题集[41]及两个现实工程优化问题进行测试。实验表明 LSHADE44-IEpsilon 均优于另外两种比较的约束进化算法。

第四章介绍目标进化算法的理论基础及发展历程，再介绍几种常见的评估多目标进化算法性能的评估指标。最后介绍一种经典的多目标进化算法—基于分解的多目标优化进化算法 MOEA/D[43]。

第五章提出了一种基于 MOEA/D 框架的基于角度约束支配规则 ACDP，所提出的 MOEA/D-ACDP 与其他四种基于分解的多目标约束进化算法在测试问题集 LIR-CMOP 及现实工程优化问题 I 型焊接梁优化问题[44]上进行测试并比较。实验表明应用了 ACDP 机制的 MOEA/D 明显优于其他四种经典算法。

最后，本文进行总结与展望。这部分既对本文所做的工作进行了总结，同时也对接下来可以开展的研究工作进行展望和说明。

第 2 章 自适应差分进化算法 LSHADE44

本文使用 LSHADE44[39]来求解单目标约束优化问题，本章将介绍 DE 相关基础知识及 LSHADE44 的详细算法框架。

2.1 差分进化算法

DE 算法在 1995 年由 Storn 和 Price 率先提出[15]，它是一种基于种群的优化搜索方法。与其他的进化算法类似，DE 同样具有交叉、变异、修复、选择等基本操作。不同的是，DE 算法使用几个个体的差分信息作为目标个体的扰动向量，这使得目标个体具有跳跃搜索的能力。在处理单目标无约束优化问题时，DE 算法统一使用如下框架：

输入：

1） 具体的优化问题以及算法终止条件。

2） N：进化种群的规模。

输出：最优解 $\vec{x}_{best}$。

步骤 1：初始化

随机产生一个落在搜索空间 R 内的种群 $P=\{\vec{x}_1,...,\vec{x}_N\}$，并对每个解的目标函数进行评估。

步骤 2：更新种群

对于种群 P 内每一个解 $\vec{x}_i, i=1,...,N$ 执行如下操作

a） 对 $\vec{x}_i$ 执行变异操作产生变异向量 $\vec{v}_i$。

b） 对 $\vec{x}_i$ 和 $\vec{v}_i$ 进行交叉操作产生试验向量（Trial Vector） $\vec{y}_i$。

c） **更新解：**如果 $f(\vec{y}_i)<f(\vec{x}_i)$，则用 $\vec{y}_i$ 替代种群中的 $\vec{x}_i$。

步骤 3：终止

如果终止条件满足，输出 P 中的最优解 $\vec{x}_{best}$；否则重新回到**步骤 2**。

根据上述框架可知，DE 算法其实非常简单。首先 DE 算法初始化一个规模为 N 的种群 P，然后在每一次迭代中，对种群中的每个个体 $\vec{x}_i, i=1,...,N$ 进行变异、交叉操作，产生一个试验向量 $\vec{y}_i$，如果 $\vec{y}_i$ 的目标值小于 $\vec{x}_i$ 的，则用 $\vec{y}_i$ 替代 $\vec{x}_i$，算法持续迭代直至终止条件满足为止。

自 DE 算法发明以来，DE 算法便引起了众多研究人员的浓厚兴趣，相比其他类型的进化算法，DE 算法具有以下特点[45]：

1）相比其他大多数进化算法，DE 算法更简单而且更易被实现。

2）研究表明 DE 在多种类型的问题上都具有广泛的适用性，表现出不错的性能，比如单峰，多峰，可分离，不可分离问题等等。

3）DE 算法的预设参数很少（经典的 DE 只有三个预设参数：缩放因子 F，交叉控制参数 CR 和种群规模 N）。

4）算法空间复杂度小于多数实参优化算法。

DE 算法广泛地应用于求解单目标优化问题上，并在各项国际会议竞赛上取得优异成绩，其中在 2017 年国际会议（IEEE Congress on Evolutionary Computation, CEC）连续单目标约束优化竞赛上，DE 算法 LSHADE44 获得了该项竞赛冠军。基于 DE 算法具有的各种优点，本文便在 LSHADE44 的框架上研究求解单目标约束问题的约束处理机制。

2.1.1 差分进化算法操作算子

2.1.1.1 变异算子

相比其他进化算法，DE 算法特点在于它的独特的变异算子。这类变异算子都是使用二个或以上个体的差分信息作为目标个体的扰动值，产生变异向量。以下介绍几种比较经典的 DE 变异算子:

1）DE/rand/1[46]:

$$\vec{v}_{i,G} = \vec{x}_{r_1,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) \tag{2-1}$$

2）DE/best/1[46]:

$$\vec{v}_{i,G} = \vec{x}_{best,G} + F \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_2,G}) \tag{2-2}$$

3）DE/current-to-best/1[46]:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{best,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_2,G}) \tag{2-3}$$

4）DE/current-to-pbest/1（不考虑外部存档 A）[47]:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{pbest,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_2,G}) \tag{2-4}$$

5）DE/current-to-pbest/1（考虑外部存档 A）[47]:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{pbest,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_4,G}) \tag{2-5}$$

6）DE/randr1/1[48]:

$$\vec{v}_{i,G} = \vec{x}_{r_1*,G} + F \cdot (\vec{x}_{r_2*,G} - \vec{x}_{r_3*,G}) \tag{2-6}$$

以上式子中，$\vec{x}_{i,G}$ 为当前种群 P_G 中第 i 个个体（$i = 1,2,...,N$），称为目标向量。$\vec{x}_{best,G}$ 为当前种群中的最优个体，$\vec{x}_{pbest,G}$ 为当前种群的 $100p\%$ 好的解，$\vec{v}_{i,G}$ 是进行变异操作后得到变异向量。r_1，r_2 和 r_3 是集合 $\{1,2...,N\} \setminus i$ 中不相同的三个整数，$\vec{x}_{r_4,G}$ 是集合 $P \cup A \setminus \{\vec{x}_{r_i,G}, \vec{x}_{r_1,G}\}$ 中随机取得的一个个体。$\vec{x}_{r_1*,G}$ 是个体 $\vec{x}_{r_1,G}$，$\vec{x}_{r_2,G}$ 和 $\vec{x}_{r_3,G}$ 中最好的解，而 $\vec{x}_{r_2*,G}$ 及 $\vec{x}_{r_3*,G}$ 表示其他两个个体。

F 称为缩放因子，它的取值决定了差分扰动对生成的试验向量的影响大小。一个变异算子可记做以下一般形式：DE/a/b。其中 a 表示试验向量的基向量选择方式，b 表示差分项的数目。

2.1.1.2 交叉算子

通常 DE 算法在产生变异向量 $\vec{v}_{i,G} = (v_{i,1,G},...,v_{i,D,G})$ 后，还需与目标向量 $\vec{x}_{i,G} = (x_{i,1,G},...,x_{i,D,G})$ 进行交叉操作后，最终产生试验向量 $\vec{y}_{i,G} = (y_{i,1,G},...,y_{i,D,G})$。目前有两种交叉算子被广泛认可及使用：

1）二项式交叉算子（Binomial Crossover, Bin）：

$$y_{i,j,G} = \begin{cases} v_{i,j,G} & \text{如果 } \mathrm{rand}_j(0,1) \le CR \text{ 或 } j = j_{rand} \\ x_{i,j,G} & \text{其他情况} \end{cases} \tag{2-7}$$

上式中，$\mathrm{rand}_j(0,1)$ 为在 $[0,1]$ 之间随机均匀取到的值，j_{rand} 为 $[1,D]$ 之间随机取到的值。

2）指数交叉算子（Exponential Crossover, Exp）[49]：

$$y_{i,j,G}=\begin{cases}v_{i,j,G} & \text{当 } j=\langle l\rangle_D,\langle l+1\rangle_D,...,\langle l+L\rangle_D \\ x_{i,j,G} & \text{其他情况}\end{cases} \tag{2-8}$$

上式中，$\langle\rangle_D$ 表示一个对 D 取模的函数。l 为 $[1,D]$ 区间内随机取到的整数，作为算子的起始标志。L 在区间 $[1,D]$ 上以概率 $P_r(L\geq v)=CR^{v-1}(v>0)$ 取到。

CR 称为交叉控制参数，取值范围在 0 到 1 之间。在 Bin 算子中，CR 直接作为个体交叉概率使用。特别地，当 $CR=1$ 时，试验变量 $\vec{y}_{i,G}=\vec{v}_{i,G}$。

一种明确的 DE 算法变异算子和交叉算子组合可以称为一种差分策略，可记作以下形式：DE/a/b/c。例如 DE/rand/1/Bin 就是使用了 DE/rand/1 变异算子及 Bin 交叉算子的一种差分策略。

2.1.1.3 修复算子

经过变异交叉操作后，DE 算法可以生成一个试验变量 $\vec{y}_{i,G}=(y_{i,1,G},...,y_{i,D,G})$。但这个变量不一定会遵循变量边界约束，即不一定落在搜索空间 R 内，此时就需要使用修复算子对试验变量进行修复。文献[50]提出一种简单的修复方法：对于试验向量 $\vec{y}_{i,G}=(y_{i,1,G},...,y_{i,D,G})$ 上的任意一位 $y_{i,j,G}$，当 $y_{i,j,G}<L_j$，我们修正 $y_{i,j,G}=L_j$，同理当 $y_{i,j,G}>U_j$，我们修正 $y_{i,j,G}=U_j$。

2.1.2 自适应差分进化算法研究简述

根据 2.1.1 小节可知，DE 算法主要受三个可调参数 F，CR 和 N 及使用的差分策略影响，基于这个背景，研究人员在 DE 算法做了很多相关的研究。

Storn 和 Price 建议初始设置 F 为 0.5，CR 设置成 0.1 或 0.9，N 取在 $5D$ 到 $10D$ 之间[14]。文献[51]通过实验研究，指出 F 应设置为 0.6，CR 取在 0.3 到 0.9 之间，N 应在 $3D$ 到 $8D$ 之间。文献[52]建议 F 应在 0.4 到 0.95 之间，CR 取为 0.9，N 应在 $2D$ 到 $4D$ 之间。CoDE 算法在 DE 算法中同时应用了多组 F 和 CR 参数对组合[53]。文献[54]提出了一种结合种群最优解信息指导的变异算子。文献[55]提出了一种 DE/current-to-best/1 的改进算子，强调算子领域局部搜索能力的同时，调整结合算子全局搜索的能力。

这些研究分析了参数及策略对 DE 算法性能的影响，但是所提出的都是静态的参数及策略设置方法，并没有利用到进化过程中的有用信息对算法进行反馈调整，所以这类研究在解决不同的问题时，参数及策略仍需要经过多次实验才能最终确定。基于这种情况，各种 DE 算法的自适应机制开始逐渐得到研究人员的关注。文献[56]提出 FADE 算法，它使用两代种群中的信息作为模糊逻辑控制的输入，从而产生 F 和 CR。jDE 算法在每一代以一定的概率随机更新参数 F 和 CR [57]。文献[47]提出了 JADE 算法，算法对每一代的每一个个体都产生一对 F 和 CR，而这对参数都是根据以往成功更新种群的 F 、CR 参数对记录进行更新的。文献[58]提出了另一种结合自适应参数调整机制的 Success-History based Adaptive DE（SHADE）算法，这种算法使用一个成功历史档集，保存一定数量的最近成功更新种群的 F 、CR 参数对信息。在每次更新个体解前，SHADE 随机选取记忆档集中的参数对信息，根据柯西分布及正态分布生成新的 F 、CR 参数对。SHADE with Linear Population Size Reduction（LSHADE）是 SHADE 的改进版本[59]，这种算法在 SHADE 框架基础上，随着目标函数评估次数（Objective Function Evalutations, FEs）的增加，线性减少种群规模 N 。

LSHADE44 是 LSHADE 算法的一种变种，它提出了一种差分策略竞赛机制可以自适应地选择差分策略，从而增强了算法对解决不同类型问题的鲁棒性[39]。算法框架中嵌入了四种不同的差分策略，根据以往成功更新种群的策略记录，算法自适应更新每种差分策略的选择概率，然后在每次生成试验向量前，根据概率选择差分策略。从 LSHADE44 算法机制上看，由于三个重要的 DE 算法参数及差分策略都可以在算法进程中自适应更新选择，所以算法对 DE 参数预设需求不多且不敏感，可以说 LSHADE44 是一种适应性很强的 DE 算法。

2.2 LSHADE44 算法

2.2.1 基于成功历史存档的参数设置方法

根据上文可知，DE 算法在进化过程中，对于种群 P 中的任意一个个体 $x_i, i = 1,2,...N$ ，都需要一对确定的参数对 $\{F_i, CR_i\}$ 从而生成试验向量。在 LSHADE44 中，参数 F_i 和 CR_i 分别有各自的长度为 H 的成功历史存档 M_F 和 M_{CR}。

每次在生成试验向量 y_i 前，如果当前存档 M_F 和 M_{CR} 是空的，参数 μ_F 及 μ_{CR} 会设为确定的初始值，否则算法会从存档中随机选取一对自适应数值 m_F 和 m_{CR} 作为 μ_F 及 μ_{CR}，然后如式子(2-9)和(2-10)所示，按照柯西分布及正态分布生成{ F_i, CR_i }。

$$F_i = \text{randc}_i(\mu_F, 0.1) \tag{2-9}$$

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1) \tag{2-10}$$

这里 $\text{randc}_i(\mu,\sigma)$ 表示一个由均值为 μ，方差为 σ 的柯西分布得到的值，而 $\text{randn}_i(\mu,\sigma)$ 表示一个由均值为 μ，方差为 σ 的正态分布得到的值。算法要 F_i 和 CR_i 都落在区间 $[0,1]$ 内，所以当 F_i 或 CR_i 落在 $[0,1]$ 外时，式子(2-9)或(2-10)会被反复执行直至满足要求为止。

每一次算法迭代结束后，如果有种群个体被更新，存档 M_F 和 M_{CR} 都会存储一对自适应数值，具体存档更新过程如下所示：

步骤 1：算法开始前初始化参数 $k = 0$。

步骤 2：在每一次迭代开始前初始化得到两个空集合 S_F 和 S_{CR}。

步骤 3：在当前迭代中，对于任意一个个体 x_i，如果它的试验向量 y_i 比 x_i 好，参数 F_i 和 CR_i 分别并入集合 S_F 和 S_{CR}，并根据式子(2-16)记录试验向量 y_i 与 x_i 的差异值。

步骤 4：在当时迭代结束时，当集合 S_F 和 S_{CR} 都是空集时，算法进入下一次迭代，回到**步骤 2**，否则 k 的值自加 1，如果此时 $k > H$，则重设 $k = 1$。

步骤 5：根据式子(2-11)-(2-16)计算出一对自适应数值 m_F 和 m_{CR}，并分别储存到存档 M_F 和 M_{CR} 的第 k 个位置中，算法进入下一次迭代，回到**步骤 2**。

$$m_F = \text{mean}_{WL}(S_F) \tag{2-11}$$

$$m_{CR} = \text{mean}_{WA}(S_{CR}) \tag{2-12}$$

$$\text{mean}_{WL}(S_F) = \frac{\sum_{t_1=1}^{|S_F|} \omega_{t_1} F_{t_1}^2}{\sum_{t_2=1}^{|S_F|} \omega_{t_2} F_{t_2}} \tag{2-13}$$

$$\text{mean}_{WA}(S_{CR}) = \sum_{t=1}^{|S_{CR}|} \omega_t CR_t \tag{2-14}$$

$$\omega_t = \frac{\Delta func_t}{\sum_{u=1}^{S_{CR}} \Delta func_u} \tag{2-15}$$

$$\Delta func_t = | \, func(\vec{x}_t) - func(\vec{y}_t) \, | \tag{2-16}$$

根据式子(1-1)和(1-2)，对于个体 x_i，当 $\phi(x_i) = \phi(y_i)$ 且 $f(x_i) > f(y_i)$，$func(\cdot)$ 为目标函数 $f(\cdot)$；当 $\phi(x_i) > \phi(y_i)$，$func(\cdot)$ 为总约束违反函数 $\phi(\cdot)$。

2.2.2 线性种群规模缩减

为了能提升 DE 算法，LSHADE44 应用了一种线性种群规模缩减机制（Linear Population Size Reduction, LPSR）。在这种机制下，种群规模 N 根据公式随着 FEs 的增加动态缩减。

$$N = \text{round}[N_{max} - \frac{FEs}{MaxFEs}(N_{max} - N_{min})] \tag{2-17}$$

式中 N_{max} 为初始种群规模，N_{min} 为最小种群规模，$MaxFEs$ 为最大评估次数。

2.2.3 差分策略竞赛选择机制

对于不同的优化问题，DE 算法应该采用相应适合的差分策略。然而在现实生活中，不少的优化问题都是黑箱问题。在面对这种缺乏对问题先验知识了解的情况，很难人工选择到一种适合的差分策略。于是我们可以在 DE 算法中同时应用多种差分策略，并根据进化过程中得到的反馈信息，自适应调整各种策略的选择概率，提升算法针对多种不同类型问题的鲁棒性，而这就是 LSHADE44 算法中提出的差分策略竞赛选择机制。LSHADE44 使用了四种不同的差分策略，赋予

每种差分策略一定的选择概率，在对于每个个体 $x_i, i=1,2,...N$ 生成相应的试验向量 y_i 根据概率选择一种差分策略，并当个体成功更新时，自适应更新各策略的选择概率，具体的机制流程如下所示：

步骤 1：在算法开始前，初始化 K 种差分策略的选择概率以相同的取值 $q_l = 1/K$（$l=1,2,...,K$）。

步骤 2：当一种差分策略成功更新个体时，所有差分策略的选择概率根据公式进行更新。

$$q_l = \frac{n_l + n_0}{\sum_{k=1}^{K}(n_k) + n_0} \tag{2-18}$$

这里 n_l 是第 l 种差分策略的累积成功更新次数。$n_0 > 0$ 是一个预设常数，它在公式中可以弱化一些偶然成功的策略对概率更新造成的影响。为了防止算法差分策略选择机制严重退化，这里预设一个参数值 δ，当任意一个策略的选择概率低于 δ 时，所有差分策略的选择概率 q_l（$l=1,2,...,K$）和其累积成功更新次数 n_l 都会被分别重新设置为 $1/K$ 和 0。

2.2.4 LSHADE44 算法框架

原始的 LSHADE44 算法框架基于上述的几种机制改进 DE 基本框架，并应用了 CDP 约束处理规则[34]解决单目标约束优化问题，LSHADE44 具体算法框架如下所示：

输入：

1）具体的优化问题以及算法终止条件。

2）N_{max}，N_{min}：进化种群的初始规模及最小规模。

3）H：成功历史存档长度。

4）n_0，δ：差分策略竞赛选择机制参数。

输出：最优解 $\vec{x}_{best}$。

步骤 1：初始化

a） 初始各差分策略选择概率 $q_l = 1/4$（$l = 1,2,3,4$）。

b） 初始化差分策略的累积成功更新次数 $n_l = 0$（$l = 1,2,3,4$）。

c） 随机产生一个落在搜索空间 R 内的种群 $P = \{\vec{x}_1,...,\vec{x}_N\}$，其中 $N = N_{max}$。

d） 并对种群内的每个解 $\vec{x}_i$ 的目标函数 $f(\vec{x}_i)$ 及其总约束违反值 $\phi(\vec{x}_i)$ 进行评估。

步骤 2：更新种群

对于种群 P 内每一个解 $\vec{x}_i, i = 1,...,N$ 执行如下操作

a） 根据概率 q_l（$l = 1,2,3,4$）选择第 l 个差分策略。

b） 根据成功历史存档 M_F 和 CR_i 产生 F_i 和 CR_i

c） 对 $\vec{x}_i$ 执行差分策略产生试验向量 $\vec{y}_i$。

d） 如果 $(\phi(x_i) = \phi(y_i) \wedge f(x_i) > f(y_i)) \vee (\phi(x_i) > \phi(y_i))$ 则

1） 用 $\vec{y}_i$ 替代种群中的 $\vec{x}_i$。

2） $n_l = n_l + 1$，根据公式(2-18)更新各策略的选择概率 $q_s, s = 1,2,3,4$。

3） 记录 F_i 和 C_i，并根据公式(2-16)记录 y_i 与 x_i 的差异值。

步骤 3：更新种群规模

根据公式(2-17)更新种群规模 N。

步骤 4：更新成功历史存档

根据公式(2-11)-(2-16)更新 M_F 和 M_{CR}。

步骤 5：终止

如果终止条件满足，输出 P 中的最优解 $\vec{x}_{best}$；否则重新回到**步骤 2**。

2.3 本章小结

本章主要介绍了一种自适应 DE 算法 LSHADE44 的相关知识，分别叙述了 DE 算法的相关理论知识和 LSHADE44 算法的具体框架。首先，本章介绍了进化算法领域中的非常经典的 DE 算法，DE 算法最独特的特性就是它的差分策略（变异、交叉算子）及其三个可控的算法参数 N ， F 和 CR 。其次本章研究了 DE 算法在自适应参数调节和策略选择上的研究及发展，并详细介绍了一种自适应 DE 算法 LSHADE44 的各项算法机制及具体框架。具体地说，LSHADE44 算法在 DE 算法的基础上，实现了动态、自适应算法参数控制及差分策略选择，是一种具有高度鲁棒性的自适应 DE 算法。因为 DE 算法具有广泛的适用性和简易可实现性等优点，并在解决单目标优化问题中表现出色，下文将选择基于 LSHADE44 的算法框架进行单目标约束优化问题约束处理机制研究。

第 3 章　基于改进 Epsilon 约束处理机制的 LSHADE44 算法

本章提出一种改进的ϵ约束处理机制（Improve Epsilon constraint-handling method, IEpsilon），同时将其应用到 LSHADE44 算法框架中，提出一种新型的算法 LSHADE44-IEpsilon，将其应用到单目标约束优化问题中，同 LSHADE44 算法及结合了原始ϵ约束处理机制的 LSHADE44 算法进行比较。

3.1 LSHADE44-IEpsilon 算法

3.1.1 Epsilon 约束处理机制

前文 1.3.2.4 小节中已经提到了ϵ约束处理机制是一种经典的分离类型约束处理机制，本小节将更进一步地介绍这种机制。

一般在单目标约束优化问题优化过程，在给定一个ϵ值后，当满足(3-1)中任意一种条件时，可以认为个体$\vec{x}_1$在ϵ水平下优于$\vec{x}_2$，记做$\vec{x}_1 <_\epsilon \vec{x}_2$。

$$\vec{x}_1 <_\epsilon \vec{x}_2 \Leftrightarrow \begin{cases} f(\vec{x}_1) < f(\vec{x}_2) & \text{如果} \quad \phi(\vec{x}_1), \phi(\vec{x}_1) \le \epsilon \\ f(\vec{x}_1) < f(\vec{x}_2) & \text{如果} \quad \phi(\vec{x}_1) = \phi(\vec{x}_2) \\ \phi(\vec{x}_1) < \phi(\vec{x}_2) & \text{其他情况} \end{cases} \tag{3-1}$$

可以说ϵ是一种约束松弛的尺度，可以说当ϵ的取值越大时，则算法对约束将更加松弛。ϵ约束处理机制在适当松弛约束的情况下保留了部分总约束违反值小于ϵ的不可行解，从而扩大种群的搜索空间，可以有效地加速收敛，避免陷入全局最优。但是ϵ的取值不能过大，也不能在进化迭代全程始终保持大于 0 的值，因为如果那样的话，虽然扩大了种群的搜索空间，但不利于种群往可行区域方向收敛，违背了约束处理机制的基本初衷：找到一个可行解。所以ϵ的参数设置是ϵ约束处理机制中最核心的部分。

在最早的ϵ约束处理机制版本中，ϵ值是随着迭代次数的增加而逐渐减小的，并在后期的一段时间内保持零值。这样在前期，ϵ约束处理机制可以有效扩大种群的搜索空间，而随着迭代的进行，逐渐将种群的搜索空间压缩，直至ϵ减少到

0，最终种群整体往可行区域收敛。

文献[60]给出了具体的ϵ参数调节方法。初始时按照公式(3-2)设置ϵ的初始值，其中$\vec{x}_\theta$为初始种群中第θ高总约束违反值的个体。当算法迭代至第k代时，$\epsilon(k)$的值根据公式(3-3)进行更新。

$$\epsilon(0)=\phi(\vec{x}_\theta) \tag{3-2}$$

$$\epsilon(k)=\begin{cases}\epsilon(k-1)(1-\dfrac{FEs}{T_c})^{cp} & \text{如果 } FEs<T_c\\ 0 & \text{如果 } FEs\geq T_c\end{cases} \tag{3-3}$$

其中FEs为当前的函数评估次数，T_c为截止更新ϵ的函数评估次数，而参数$cp=(-5-\log\epsilon(0))/\log(0.05)$。

3.1.2 改进的 Epsilon 约束处理机制 IEpsilon

原始的ϵ约束处理机制通过调整ϵ值，可以动态地调整种群的搜索空间。但是ϵ值只是根据进化代数的增加而逐渐减少，无法充分地利用当前种群中有用的信息。当种群可行比例太小时，过大的ϵ值会影响到种群向可行区域的收敛；当种群可行比例足够大时，适当地增加ϵ值可以扩大种群的搜索区域，有利于种群突破局部最优从而找到全局更优的解。所以设计一套可以根据当前种群可行比例自适应调整ϵ的方法，可以更有效地利用好ϵ约束处理机制的优势。

本小节提出了一种改进的ϵ约束处理机制 IEpsilon，根据当前第k代种群的可行比例r_k，自适应调整$\epsilon(k)$值。

$$\epsilon(k)=\begin{cases}\phi_\theta & \text{如果 } k=0\\ \epsilon(k-1)(1-\dfrac{FEs}{T_c})^{cp} & \text{如果 } r_k<\alpha \text{ and } FEs<T_c\\ (1+\tau)\phi_{max} & \text{如果 } r_k\geq\alpha \text{ and } FEs<T_c\\ 0 & \text{如果 } FEs\geq T_c\end{cases} \tag{3-4}$$

其中FEs为当前的函数评估次数，T_c为截止更新ϵ的函数评估次数，而参数cp和τ是预设参数，ϕ_{max}是历史最大总约束违反值，α是判断可行解比例大小的阈值。当可行比例大于α时，种群会认为当前保留了太多的可行解，可以尝试增加ϵ，

增加总群在不可行区域内搜索的机会。相反，当可行比例大于α时，种群会认为当前可行解太少，应该减少ϵ使种群更趋于搜索可行的解。

3.1.3 结合两种 Epsilon 约束处理机制的 LSHADE44

将两种 Epsilon 约束处理机制：ϵ约束处理机制及 IEpsilon 机制结合于 LSHADE44 算法框架，可以得到两种新的单目标约束进化算法 LSHADE44-Epsilon 和 LSHADE44-IEpsilon。这两种算法流程都具有类似的统一算法框架，具体如下所示。

LSHADE44-Epsilon / LSHADE44-IEpsilon 算法框架：

输入：

1）具体的优化问题以及算法终止条件。

2）LSHADE44 参数：N_{max}，N_{min}，H，n_0，δ。

3）ϵ 约束处理机制参数：T_c，θ / IEpsilon 机制参数：T_c，θ，α，cp。

输出：最优解$\vec{x}_{best}$。

步骤 1：初始化

1） 初始各差分策略选择概率$q_l = 1/4$（$l = 1,2,3,4$）。

2） 初始化差分策略的累积成功更新次数$n_l = 0$（$l = 1,2,3,4$）。

3）随机产生一个落在搜索空间R内的种群$P = \{\vec{x}_1, ..., \vec{x}_N\}$，其中$N = N_{max}$。

4）并对种群内的每个解$\vec{x}_i$的目标函数$f(\vec{x}_i)$及其总约束违反值$\phi(\vec{x}_i)$进行评估。

5）初始化$\epsilon = \phi(\vec{x}_\theta)$。

步骤 2：更新种群

对于种群P内每一个解$\vec{x}_i, i = 1, ..., N$执行如下操作

1） 根据概率q_l（$l = 1,2,3,4$）选择第l个差分策略。

2） 根据成功历史存档M_F和CR_i产生F_i和CR_i。

3） 对 $\vec{x}_i$ 执行差分策略产生试验向量 $\vec{y}_i$。

4）如果 $(\max(0,\phi(\vec{x}_i)-\epsilon)=\max(0,\phi(\vec{y}_i)-\epsilon)\wedge f(\vec{x}_i)>f(\vec{y}_i))$

或者 $(\max(0,\phi(\vec{x}_i)-\epsilon)>\max(0,\phi(\vec{y}_i)-\epsilon))$ **则**

a. 用 $\vec{y}_i$ 替代种群中的 $\vec{x}_i$。

b. $n_l=n_l+1$，根据公式(2-18)更新各策略的选择概率 $q_s, s=1,2,3,4$。

c. 记录 F_i 和 C_i，并根据公式(2-16)记录 y_i 与 x_i 的差异值。

步骤 3：更新种群规模

根据公式(2-17)更新种群规模 N。

步骤 4：更新成功历史存档

根据公式(2-11)-(2-16)更新 M_F 和 M_{CR}。

步骤 5：更新 ϵ **值**

根据公式(3-3) / (3-4)更新 ϵ **值。**

步骤 6：终止

如果终止条件满足，输出 P 中的最优解 $\vec{x}_{best}$；否则重新回到**步骤 2**。

3.2 实验研究

本节将基于单目标约束优化测试问题集及两个工程应用优化问题对本文提出的 LSHADE44-IEpsilon 同 LSHADE44 和 LSHADE44-Epsilon 分别独立运行 25 次进行比较，测试它们的性能。

3.2.1 单目标约束优化测试问题集

2017 年，IEEE 进化计算会议（Congress on Evolutionary Computation, CEC）公布了由 28 个连续实参单目标约束优化函数 C01-C28 组成的测试问题集（下简称 CEC2017 测试问题集）[41]。这些测试问题的具体目标函数及约束条件附于附录页附表 I 中，所有测试问题都带有一定数目的等式或不等式约束条件，每个目标函数及约束条件的类型包括了可分离/不可分离及旋转/非旋转等类型。同时

CEC2017 测试问题集还提供了不同的维度尺度，即变量维度 $D=10,30,50,100$ 共四种维度。在算法运行时，对于每种维度尺度 D，每次运行至函数评估次数达 $20000D$ 次停止。

3.2.2 工程优化测试问题

为了进一步测试提出的 LSHADE44-IEpsilon 算法的性能，我们将算法应用至两个经典的实际工程优化问题：压力容器优化问题及汽车侧面撞击问题。后续通过对这两个现实优化问题的求解，进一步测试 LSHADE44-IEpsilon 算法求解现实优化问题的性能。

3.2.2.1 压力容器优化问题

如图 3-1 所示，一个圆柱压力容器在其末端覆上一个半球形头部，要使设计出的组合构型花费费用最少，这就是压力容器优化问题[61]。在这个问题中需要优化的参数包括圆柱表层厚度 T_s，半球形头部厚度 T_h，共同的内径 R 及圆柱压力容器部分的长度 L，即决策向量为 $\vec{x}=[x_1,x_2,x_3,x_4]^T=[T_s,T_h,R,L]^T$。

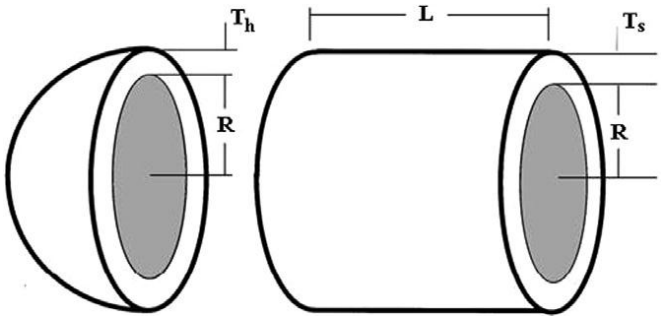


图 3-1　压力容器优化问题原理图

压力容器优化问题具体的数学表达形式如下所示：

$$\begin{cases} \text{minimize} & f(\vec{x})=0.6224T_sRL+1.7781T_hR^2+3.1661T_s^2L+19.84T_h^2R \\ \text{subject to} & g_1(\vec{x})=-T_s+0.0193R\le 0 \\ & g_2(\vec{x})=-T_h+0.00954R\le 0 \\ & g_3(\vec{x})=-\pi R^2L-\frac{4}{3}\pi R^3+750\times 1728\le 0 \\ & g_4(\vec{x})=L-240\le 0 \end{cases} \tag{3-5}$$

其中$1\times0.0625\le T_s,T_h\le99\times0.0625$，$10\le R\le200$，$10\le L\le240$。

3.2.2.2 汽车侧撞结构设计问题

图 3-2 展示了汽车侧撞结构设计的有限元模型示意图，文献[62]中使用了简易的回归模型构建了该问题的虚拟有限元模型。在这个问题中，需要优化的目标是最小化整车的总质量，需要优化的参数包括了内中立柱厚度（x_1）、加固中立柱厚度（x_2）、底部厚度（x_3）、横梁厚度（x_4）、车门防撞梁厚度（x_5）、车门腰线厚度（x_6）、车顶纵梁厚度（x_7）、内侧中立柱材料（x_8）、底部材料（x_9）、障碍物高度（x_{10}）及撞击位置（x_{11}）。

汽车侧撞结构设计问题具体的数学表达形式如下所示：

Minimize
$$f(\vec{x})=Weight \tag{3-6}$$

Subject to

$$g_1(x)=F_a\le1 \tag{3-7}$$

$$g_2(x)=VC_u\le0.32 \tag{3-8}$$

$$g_3(x)=VC_m\le0.32 \tag{3-9}$$

$$g_4(x)=VC_l\le0.32 \tag{3-10}$$

$$g_5(x)=\Delta ur\le32 \tag{3-11}$$

$$g_6(x)=\Delta mr\le32 \tag{3-12}$$

$$g_7(x)=\Delta lr\le32 \tag{3-13}$$

$$g_8(x)=F_p\le4 \tag{3-14}$$

$$g_9(x)=V_{MBP}\le9.9 \tag{3-15}$$

$$g_{10}(x)=V_{FD}\le15.7 \tag{3-16}$$

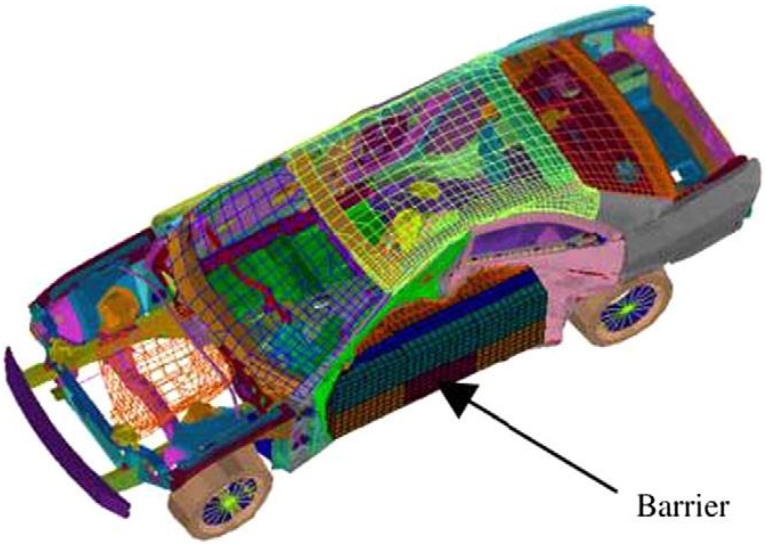


图 3-2　汽车侧撞结构设计有限元示意图

经过简易回归建模后，上述参数具体定义如下所示[63]。

$$Weight = 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 0.00001x_6 + 2.73x_7 \tag{3-17}$$

$$F_a = 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} \tag{3-18}$$

$$VC_u = 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10} + 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} \tag{3-19}$$

$$VC_m = 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7 + 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10} + 0.00121x_8x_{11} + 0.00184x_9x_{10} - 0.018x_2^2 \tag{3-20}$$

$$VC_l = 0.74 - 0.061x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_8 + 0.227x_2^2 \tag{3-21}$$

$$\Delta ur = 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10} \tag{3-22}$$

$$\Delta mr = 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11x_2x_8 - 0.0215x_5x_{10} - 9.98x_7x_8 + 22x_8x_9 \tag{3-23}$$

$$\Delta lr = 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} \tag{3-24}$$

$$F_p = 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^{\ 2} \tag{3-25}$$

$$V_{MBP}=10.58-0.674x_1x_2-1.95x_2x_8+0.02054x_3x_{10}-0.0198x_4x_{10}+0.028x_6x_{10} \tag{3-26}$$

$$V_{FD}=16.45-0.489x_3x_7-0.843x_5x_6+0.0432x_9x_{10}-0.0556x_9x_{11}-0.000786x_{11}^2 \tag{3-27}$$

其中各变量的取值范围如下所示：$0.5\le x_1,x_3,x_4,x_{10},x_{11}\le 1.5$，$0.45\le x_2\le 1.35$，$0.875\le x_5\le 2.625$，$0.4\le x_6,x_7\le 1.2$，$x_8,x_9\in\{0.192,0.345\}$。

3.2.3 实验参数设置及评估指标

3.2.3.1 算法参数设置

LSHADE44-IEpsilon 及 LSHADE44 和 LSHADE44-Epsilon 具体参数设置如下所示：

1）种群规模：最大种群规模 $N_{max}=10D$，最小种群规模 $N_{min}=5$。

2）历史成功档集长度：$H=10$。

3）差分策略竞赛选择机制：选用四种差分策略 DE/current-to-pbest/1/Bin；DE/current-to-pbest/1/Exp；DE/randr1/1/Bin；DE/randr1/1/Exp，$n_0=2$，$\delta=0.05$。

4）DE/current-to-pbest/1 参数：$p=0.2$。

5）IEpsilon 参数：$\alpha=0.5$，$\tau=0.1$。

6）评估次数：CEC2017 测试集 $MaxFEs=20000D$，工程应用 $MaxFEs=250D$。

7）其他参数：$\theta=20\%$，$T_c=0.8MaxFEs$。

3.2.3.2 实验评估指标

根据文献[41]，在比较两个约束单目标进化算法的性能时，可以使用以下两种指标进行比较：

1）基于均值的比较指标

a. 先根据多次独立实验得到可行解的比例对算法进行排序；

b. 再根据平均总约束违反值对算法进行排序；

c. 最后根据平均目标函数值对算法进行排序。

2）基于中位解的比较指标

a. 先根据中位解的可行性进行比较；

b. 两个可行中位解间根据它们的目标函数值进行比较；

c. 两个不可行中位解间根据它们的约束违反值进行比较。

3.2.4 实验结果及分析

3.2.4.1 CEC2017 测试问题集实验结果

对于 CEC2017 测试问题集中 28 个测试问题 C01-C28，应用 LSHADE44-Iepsilon 及 LSHADE44 及 LSHADE44-Epsilon 进行 25 次独立运行优化，所有实验结果列于表 3-1 到表 3-8 中。每个表中，‘mean’表示平均目标函数值，SR 表示 25 次独立实验得到可行解的比例，‘vio’表示平均总约束违反值，‘median’表示中值解的目标函数值，‘v’表示中位解的总约束违反值。根据基于均值及中值的两种算法性能比较指标，对于 LSHADE44-IEpsilon 与其他两种算法的性能比较情况列于表 3-9 及表 3-10 中。其中符号‘+’、‘-’及‘=’分别表示对应比较算法比 LSHADE44-IEpsilon 好、差及不好不差的测试问题数目。

根据表 3-9 及表 3-10 的比较结果可知，按照 2 种不同的比较指标，LSHADE44-IEpsilon 算法在所有变量维度（$D=10,30,50,100$）上表现更好的测试问题数目都比 LSHADE44 及 LSHADE44-Epsilon 多。这表明了 LSHADE44-IEpsilon 在 CEC2017 测试问题集上的表现比 2017 年 CEC 单目标约束竞赛冠军 LSHADE44 算法要优异，同时也表明了改进的 ϵ 机制 IEpsilon 在解决单目标约束优化问题比原始 ϵ 约束处理机制要好。

表 3-1　LSHADE44-IEpsilon 及其他两种算法在 C01-C14 上的优化结果（$D=10$）

	算法	mean	SR	vio	median	v
C01	LSHADE44-IEps	1.327E+01	100%	0.000E+00	3.787E-29	0.000E+00
	LSHADE44	3.220E+00	100%	0.000E+00	7.889E-29	0.000E+00
	LSHADE44-Eps	1.227E+00	100%	0.000E+00	2.524E-29	0.000E+00
C02	LSHADE44-IEps	1.472E-01	100%	0.000E+00	4.418E-29	0.000E+00
	LSHADE44	2.147E+00	100%	0.000E+00	7.573E-29	0.000E+00
	LSHADE44-Eps	3.873E-02	100%	0.000E+00	9.151E-29	0.000E+00
C03	LSHADE44-IEps	1.612E+02	100%	0.000E+00	9.260E+01	0.000E+00
	LSHADE44	1.030E+05	84%	1.267E-04	7.451E+04	0.000E+00

	LSHADE44-Eps	5.153E+02	100%	0.000E+00	2.652E+02	0.000E+00
C04	LSHADE44-IEps	2.730E+01	100%	0.000E+00	2.721E+01	0.000E+00
	LSHADE44	4.179E+01	100%	0.000E+00	4.078E+01	0.000E+00
	LSHADE44-Eps	2.870E+01	100%	0.000E+00	2.595E+01	0.000E+00
C05	LSHADE44-IEps	7.495E-01	100%	0.000E+00	2.194E-12	0.000E+00
	LSHADE44	7.881E-01	100%	0.000E+00	6.372E-08	0.000E+00
	LSHADE44-Eps	5.153E-01	100%	0.000E+00	1.900E-20	0.000E+00
C06	LSHADE44-IEps	4.230E+01	16%	2.482E-01	2.000E+00	2.805E-01
	LSHADE44	9.958E+02	24%	2.757E-01	5.692E+02	1.713E-01
	LSHADE44-Eps	4.618E+01	28%	2.785E-01	2.000E+00	2.805E-01
C07	LSHADE44-IEps	-3.053E+02	84%	1.427E+01	-3.139E+02	0.000E+00
	LSHADE44	2.001E+01	96%	5.011E-04	4.763E+01	0.000E+00
	LSHADE44-Eps	-2.488E+02	88%	7.692E+00	-2.539E+02	0.000E+00
C08	LSHADE44-IEps	-1.347E-03	100%	0.000E+00	-1.348E-03	0.000E+00
	LSHADE44	4.016E-02	96%	4.181E-02	-1.348E-03	0.000E+00
	LSHADE44-Eps	1.899E-02	96%	1.025E-02	-1.348E-03	0.000E+00
C09	LSHADE44-IEps	-4.975E-03	100%	0.000E+00	-4.975E-03	0.000E+00
	LSHADE44	1.077E+00	100%	0.000E+00	-4.975E-03	0.000E+00
	LSHADE44-Eps	-4.975E-03	100%	0.000E+00	-4.975E-03	0.000E+00
C10	LSHADE44-IEps	-5.096E-04	100%	0.000E+00	-5.096E-04	0.000E+00
	LSHADE44	-4.631E-04	100%	0.000E+00	-5.096E-04	0.000E+00
	LSHADE44-Eps	3.836E-02	96%	3.593E-01	-5.096E-04	0.000E+00
C11	LSHADE44-IEps	-3.481E+02	40%	7.466E+02	-5.920E+02	6.913E+02
	LSHADE44	2.372E+01	52%	4.438E+01	1.632E-02	0.000E+00
	LSHADE44-Eps	-2.598E+02	32%	3.883E+02	-1.973E+00	4.454E-08
C12	LSHADE44-IEps	4.348E+00	100%	0.000E+00	3.992E+00	0.000E+00
	LSHADE44	8.363E+00	96%	2.628E-02	3.988E+00	0.000E+00
	LSHADE44-Eps	1.294E+01	96%	1.464E+00	4.111E+00	0.000E+00
C13	LSHADE44-IEps	2.662E+00	100%	0.000E+00	2.050E-26	0.000E+00
	LSHADE44	6.891E+00	100%	0.000E+00	1.394E-13	0.000E+00
	LSHADE44-Eps	1.059E+04	96%	5.635E-01	2.189E-12	0.000E+00
C14	LSHADE44-IEps	3.825E+00	76%	2.924E+01	2.629E+00	0.000E+00
	LSHADE44	3.418E+00	100%	0.000E+00	3.449E+00	0.000E+00
	LSHADE44-Eps	4.220E+00	88%	2.976E+01	3.554E+00	0.000E+00

表 3-2　LSHADE44-IEpsilon 及其他两种算法在 C15-C28 上的优化结果（$D=10$）

	算法	mean	SR	vio	median	v
C15	LSHADE44-IEps	4.492E+00	84%	5.168E-05	2.356E+00	0.000E+00
	LSHADE44	1.857E+01	56%	4.671E-04	2.121E+01	0.000E+00
	LSHADE44-Eps	7.131E+00	84%	4.276E-05	5.498E+00	0.000E+00
C16	LSHADE44-IEps	6.283E-02	100%	0.000E+00	0.000E+00	0.000E+00
	LSHADE44	6.673E+01	96%	1.482E-06	6.911E+01	0.000E+00
	LSHADE44-Eps	6.472E+00	100%	0.000E+00	6.283E+00	0.000E+00

C17	LSHADE44-IEps	1.997E-01	0	5.220E+00	1.040E-01	5.500E+00
	LSHADE44	9.560E-01	0	5.500E+00	1.014E+00	5.500E+00
	LSHADE44-Eps	6.211E-01	0	5.460E+00	6.085E-01	5.500E+00
C18	LSHADE44-IEps	1.246E+02	56%	2.165E+01	4.794E+01	0.000E+00
	LSHADE44	5.280E+02	88%	2.608E+02	3.722E+01	0.000E+00
	LSHADE44-Eps	5.414E+01	84%	1.495E+02	3.660E+01	0.000E+00
C19	LSHADE44-IEps	3.345E+00	0	6.636E+03	3.049E-01	6.634E+03
	LSHADE44	7.971E-01	0	6.634E+03	0.000E+00	6.634E+03
	LSHADE44-Eps	1.137E+00	0	6.635E+03	0.000E+00	6.634E+03
C20	LSHADE44-IEps	9.138E-01	100%	0.000E+00	9.054E-01	0.000E+00
	LSHADE44	1.143E+00	100%	0.000E+00	1.205E+00	0.000E+00
	LSHADE44-Eps	8.512E-01	100%	0.000E+00	8.808E-01	0.000E+00
C21	LSHADE44-IEps	1.203E+01	100%	0.000E+00	4.329E+00	0.000E+00
	LSHADE44	7.626E+00	100%	0.000E+00	3.988E+00	0.000E+00
	LSHADE44-Eps	5.239E+00	100%	0.000E+00	3.988E+00	0.000E+00
C22	LSHADE44-IEps	4.397E+03	96%	1.790E-01	7.200E-11	0.000E+00
	LSHADE44	2.572E+05	84%	6.190E+00	3.987E+00	0.000E+00
	LSHADE44-Eps	1.151E+05	92%	4.264E+00	3.987E+00	0.000E+00
C23	LSHADE44-IEps	2.565E+00	100%	0.000E+00	2.629E+00	0.000E+00
	LSHADE44	3.496E+00	100%	0.000E+00	3.507E+00	0.000E+00
	LSHADE44-Eps	3.351E+00	96%	4.965E-01	3.388E+00	0.000E+00
C24	LSHADE44-IEps	3.613E+00	84%	7.552E-05	2.356E+00	0.000E+00
	LSHADE44	1.756E+01	96%	1.847E-07	1.806E+01	0.000E+00
	LSHADE44-Eps	6.503E+00	100%	0.000E+00	5.498E+00	0.000E+00
C25	LSHADE44-IEps	6.786E+00	100%	0.000E+00	6.283E+00	0.000E+00
	LSHADE44	7.219E+01	100%	0.000E+00	7.540E+01	0.000E+00
	LSHADE44-Eps	2.400E+01	100%	0.000E+00	1.885E+01	0.000E+00
C26	LSHADE44-IEps	2.435E-01	0	5.380E+00	2.529E-01	5.500E+00
	LSHADE44	1.012E+00	0	5.460E+00	8.645E-01	5.500E+00
	LSHADE44-Eps	9.170E-01	0	2.251E+01	1.010E+00	5.500E+00
C27	LSHADE44-IEps	7.715E+01	68%	6.471E+03	3.660E+01	0.000E+00
	LSHADE44	4.274E+03	68%	8.890E+04	3.852E+01	0.000E+00
	LSHADE44-Eps	3.889E+01	100%	0.000E+00	3.661E+01	0.000E+00
C28	LSHADE44-IEps	3.716E+01	0	6.657E+03	1.914E+01	6.658E+03
	LSHADE44	3.555E+01	0	6.658E+03	3.597E+01	6.659E+03
	LSHADE44-Eps	3.421E+01	0	6.658E+03	3.413E+01	6.659E+03

表 3-3　LSHADE44-IEpsilon 及其他两种算法在 C01-C14 上的优化结果（$D=30$）

	算法	mean	SR	vio	median	v
C01	LSHADE44-IEps	2.970E+02	100%	0.000E+00	2.497E-12	0.000E+00
	LSHADE44	4.662E+02	100%	0.000E+00	5.596E-09	0.000E+00
	LSHADE44-Eps	4.348E+02	100%	0.000E+00	4.313E-08	0.000E+00
C02	LSHADE44-IEps	1.478E+01	100%	0.000E+00	7.623E-08	0.000E+00

	LSHADE44	8.297E-01	100%	0.000E+00	4.706E-11	0.000E+00
	LSHADE44-Eps	5.933E+01	100%	0.000E+00	8.061E-08	0.000E+00
C03	LSHADE44-IEps	1.377E+04	100%	0.000E+00	1.148E+04	0.000E+00
	LSHADE44	1.219E+06	80%	1.342E-04	1.388E+06	0.000E+00
	LSHADE44-Eps	2.216E+04	96%	2.952E-05	2.228E+04	0.000E+00
C04	LSHADE44-IEps	1.498E+02	100%	0.000E+00	1.460E+02	0.000E+00
	LSHADE44	1.741E+02	100%	0.000E+00	1.666E+02	0.000E+00
	LSHADE44-Eps	1.539E+02	100%	0.000E+00	1.499E+02	0.000E+00
C05	LSHADE44-IEps	3.725E+00	100%	0.000E+00	3.346E-03	0.000E+00
	LSHADE44	1.024E+01	100%	0.000E+00	1.089E-02	0.000E+00
	LSHADE44-Eps	7.260E+00	100%	0.000E+00	2.008E+00	0.000E+00
C06	LSHADE44-IEps	3.151E+02	88%	1.022E-02	3.240E+02	0.000E+00
	LSHADE44	4.872E+03	88%	5.715E-02	4.904E+03	0.000E+00
	LSHADE44-Eps	4.116E+02	100%	0.000E+00	4.202E+02	0.000E+00
C07	LSHADE44-IEps	-4.305E+02	32%	1.813E+02	-4.581E+02	1.810E+02
	LSHADE44	3.030E-01	92%	1.163E+01	-2.015E+00	0.000E+00
	LSHADE44-Eps	-2.057E+02	92%	2.405E+01	-1.678E+02	0.000E+00
C08	LSHADE44-IEps	3.606E-01	96%	8.091E-01	6.886E-04	0.000E+00
	LSHADE44	2.841E+00	84%	5.666E+01	8.111E-04	0.000E+00
	LSHADE44-Eps	9.285E-01	96%	5.379E+00	6.362E-04	0.000E+00
C09	LSHADE44-IEps	1.570E-01	100%	0.000E+00	-2.666E-03	0.000E+00
	LSHADE44	1.088E+00	96%	1.156E-02	-2.666E-03	0.000E+00
	LSHADE44-Eps	2.718E-01	100%	0.000E+00	-2.666E-03	0.000E+00
C10	LSHADE44-IEps	1.334E-04	100%	0.000E+00	5.431E-05	0.000E+00
	LSHADE44	2.138E+00	88%	1.571E+02	1.656E-04	0.000E+00
	LSHADE44-Eps	2.141E-02	96%	1.874E-02	6.816E-05	0.000E+00
C11	LSHADE44-IEps	-1.464E+03	0	4.238E+02	-1.816E+03	2.019E+02
	LSHADE44	-9.707E+01	4%	4.629E+01	1.696E+02	1.827E+00
	LSHADE44-Eps	-1.147E+03	0	4.472E+02	-9.718E+02	1.288E+02
C12	LSHADE44-IEps	2.162E+01	80%	2.652E+00	1.574E+01	0.000E+00
	LSHADE44	1.150E+01	100%	0.000E+00	9.775E+00	0.000E+00
	LSHADE44-Eps	1.140E+01	100%	0.000E+00	9.775E+00	0.000E+00
C13	LSHADE44-IEps	4.951E+05	72%	1.246E+01	2.755E+02	0.000E+00
	LSHADE44	7.738E+05	56%	1.974E+01	3.976E+04	0.000E+00
	LSHADE44-Eps	8.164E+04	48%	7.144E+00	6.433E+04	1.450E-01
C14	LSHADE44-IEps	1.624E+01	16%	8.988E+03	1.944E+01	7.952E+03
	LSHADE44	2.037E+00	100%	0.000E+00	2.159E+00	0.000E+00
	LSHADE44-Eps	3.467E+00	92%	8.233E+02	2.165E+00	0.000E+00

表 3-4　LSHADE44-IEpsilon 及其他两种算法在 C15-C28 上的优化结果（$D=30$）

	算法	mean	SR	vio	median	v
C15	LSHADE44-IEps	1.367E+01	96%	2.781E-06	1.178E+01	0.000E+00
	LSHADE44	2.171E+01	80%	8.093E-05	2.121E+01	0.000E+00

	LSHADE44-Eps	1.618E+01	80%	8.926E-05	1.492E+01	0.000E+00
C16	LSHADE44-IEps	9.110E+00	100%	0.000E+00	6.283E+00	0.000E+00
	LSHADE44	2.135E+02	96%	8.981E-07	2.262E+02	0.000E+00
	LSHADE44-Eps	9.199E+01	100%	0.000E+00	8.325E+01	0.000E+00
C17	LSHADE44-IEps	7.716E-01	0	1.550E+01	6.082E-01	1.550E+01
	LSHADE44	1.026E+00	0	1.550E+01	1.029E+00	1.550E+01
	LSHADE44-Eps	1.023E+00	0	1.550E+01	1.030E+00	1.550E+01
C18	LSHADE44-IEps	8.197E+02	8%	4.595E+03	2.261E+02	1.082E+03
	LSHADE44	1.443E+02	84%	5.302E+00	3.822E+01	0.000E+00
	LSHADE44-Eps	6.787E+01	88%	8.575E-01	4.053E+01	0.000E+00
C19	LSHADE44-IEps	7.365E+00	0	2.139E+04	2.913E+00	2.138E+04
	LSHADE44	1.017E+01	0	2.139E+04	8.666E+00	2.139E+04
	LSHADE44-Eps	6.674E+00	0	2.138E+04	1.034E+00	2.138E+04
C20	LSHADE44-IEps	4.544E+00	100%	0.000E+00	4.557E+00	0.000E+00
	LSHADE44	5.399E+00	100%	0.000E+00	5.520E+00	0.000E+00
	LSHADE44-Eps	4.254E+00	100%	0.000E+00	4.341E+00	0.000E+00
C21	LSHADE44-IEps	9.090E+00	100%	0.000E+00	9.776E+00	0.000E+00
	LSHADE44	1.161E+01	100%	0.000E+00	9.795E+00	0.000E+00
	LSHADE44-Eps	1.119E+01	100%	0.000E+00	9.775E+00	0.000E+00
C22	LSHADE44-IEps	1.730E+06	20%	9.729E+01	2.494E+05	3.224E+01
	LSHADE44	5.335E+06	4%	1.283E+02	6.748E+05	6.834E+01
	LSHADE44-Eps	2.978E+06	12%	1.200E+02	7.929E+05	8.757E+01
C23	LSHADE44-IEps	2.327E+00	96%	9.670E+01	1.564E+00	0.000E+00
	LSHADE44	1.968E+00	100%	0.000E+00	2.088E+00	0.000E+00
	LSHADE44-Eps	1.950E+00	100%	0.000E+00	2.100E+00	0.000E+00
C24	LSHADE44-IEps	1.400E+01	96%	1.292E+02	1.178E+01	0.000E+00
	LSHADE44	2.058E+01	100%	0.000E+00	2.121E+01	0.000E+00
	LSHADE44-Eps	1.970E+01	100%	0.000E+00	1.806E+01	0.000E+00
C25	LSHADE44-IEps	9.946E+01	100%	0.000E+00	1.005E+02	0.000E+00
	LSHADE44	2.185E+02	100%	0.000E+00	2.199E+02	0.000E+00
	LSHADE44-Eps	1.954E+02	100%	0.000E+00	1.948E+02	0.000E+00
C26	LSHADE44-IEps	9.366E-01	0	1.931E+01	1.024E+00	1.550E+01
	LSHADE44	1.025E+00	0	1.550E+01	1.030E+00	1.550E+01
	LSHADE44-Eps	1.031E+00	0	3.001E+01	1.029E+00	1.550E+01
C27	LSHADE44-IEps	2.784E+02	44%	1.094E+04	7.118E+01	8.854E+00
	LSHADE44	8.443E+02	44%	2.439E+02	3.643E+01	6.784E-05
	LSHADE44-Eps	3.039E+02	64%	5.183E+02	5.299E+01	0.000E+00
C28	LSHADE44-IEps	1.682E+02	0	2.149E+04	1.666E+02	2.149E+04
	LSHADE44	1.652E+02	0	2.149E+04	1.588E+02	2.149E+04
	LSHADE44-Eps	1.648E+02	0	2.149E+04	1.506E+02	2.149E+04

表 3-5　LSHADE44-IEpsilon 及其他两种算法在 C01-C14 上的优化结果（$D=50$）

	算法	mean	SR	vio	median	v

C01	LSHADE44-IEps	6.887E+02	100%	0.000E+00	2.806E-01	0.000E+00
	LSHADE44	8.110E+02	100%	0.000E+00	4.574E-01	0.000E+00
	LSHADE44-Eps	1.904E+02	100%	0.000E+00	1.407E-01	0.000E+00
C02	LSHADE44-IEps	1.637E+03	100%	0.000E+00	5.727E-02	0.000E+00
	LSHADE44	2.237E+02	100%	0.000E+00	6.158E-03	0.000E+00
	LSHADE44-Eps	6.703E+02	100%	0.000E+00	2.177E-01	0.000E+00
C03	LSHADE44-IEps	3.385E+04	100%	0.000E+00	3.653E+04	0.000E+00
	LSHADE44	8.700E+06	76%	4.263E-05	1.744E+07	0.000E+00
	LSHADE44-Eps	8.162E+04	100%	0.000E+00	7.514E+04	0.000E+00
C04	LSHADE44-IEps	2.867E+02	100%	0.000E+00	2.863E+02	0.000E+00
	LSHADE44	3.173E+02	100%	0.000E+00	3.085E+02	0.000E+00
	LSHADE44-Eps	2.906E+02	100%	0.000E+00	2.904E+02	0.000E+00
C05	LSHADE44-IEps	1.142E+01	100%	0.000E+00	7.457E+00	0.000E+00
	LSHADE44	2.040E+01	100%	0.000E+00	1.722E+01	0.000E+00
	LSHADE44-Eps	1.370E+01	100%	0.000E+00	4.174E+00	0.000E+00
C06	LSHADE44-IEps	6.703E+02	96%	1.003E-02	6.371E+02	0.000E+00
	LSHADE44	8.254E+03	88%	1.970E-02	8.305E+03	0.000E+00
	LSHADE44-Eps	1.142E+03	96%	1.830E-04	1.040E+03	0.000E+00
C07	LSHADE44-IEps	-5.038E+02	44%	2.281E+02	-5.233E+02	1.508E+02
	LSHADE44	-8.224E+01	72%	4.071E+01	-9.311E+00	0.000E+00
	LSHADE44-Eps	-6.883E+01	76%	4.022E+01	2.733E+01	0.000E+00
C08	LSHADE44-IEps	4.121E-02	60%	5.264E-03	4.202E-03	0.000E+00
	LSHADE44	3.381E+00	52%	1.080E+02	5.131E-03	0.000E+00
	LSHADE44-Eps	1.455E-01	68%	8.685E-02	3.099E-03	0.000E+00
C09	LSHADE44-IEps	1.762E-01	100%	0.000E+00	-1.620E-03	0.000E+00
	LSHADE44	1.314E+00	100%	0.000E+00	-1.973E-03	0.000E+00
	LSHADE44-Eps	4.660E-01	96%	1.280E-02	-1.564E-03	0.000E+00
C10	LSHADE44-IEps	5.850E-04	100%	0.000E+00	5.471E-04	0.000E+00
	LSHADE44	3.164E+00	84%	1.115E+03	7.338E-04	0.000E+00
	LSHADE44-Eps	5.070E-01	92%	1.866E+02	6.067E-04	0.000E+00
C11	LSHADE44-IEps	-3.291E+03	0	5.250E+02	-3.012E+03	3.763E+02
	LSHADE44	4.435E+00	0	1.171E+02	-1.873E+01	9.304E-01
	LSHADE44-Eps	-1.821E+03	0	2.400E+02	-1.650E+03	7.250E+01
C12	LSHADE44-IEps	1.448E+01	84%	6.794E-01	1.460E+01	0.000E+00
	LSHADE44	1.951E+01	100%	0.000E+00	1.794E+01	0.000E+00
	LSHADE44-Eps	1.799E+01	100%	0.000E+00	1.794E+01	0.000E+00
C13	LSHADE44-IEps	4.902E+05	16%	4.488E+01	1.839E+05	2.769E+01
	LSHADE44	7.514E+05	4%	6.497E+01	3.092E+05	4.462E+01
	LSHADE44-Eps	1.071E+06	12%	5.884E+01	4.219E+05	5.554E+01
C14	LSHADE44-IEps	1.495E+01	24%	1.679E+04	1.959E+01	1.527E+04
	LSHADE44	1.515E+00	100%	0.000E+00	1.586E+00	0.000E+00
	LSHADE44-Eps	2.236E+00	96%	8.751E+02	1.498E+00	0.000E+00

表 3-6　LSHADE44-IEpsilon 及其他两种算法在 C15-C28 上的优化结果（$D=50$）

	算法	mean	SR	vio	median	v
C15	LSHADE44-IEps	1.643E+01	96%	1.209E-05	1.492E+01	0.000E+00
	LSHADE44	2.648E+01	80%	2.495E-05	2.435E+01	0.000E+00
	LSHADE44-Eps	2.271E+01	92%	9.110E-06	2.121E+01	0.000E+00
C16	LSHADE44-IEps	4.028E+01	100%	0.000E+00	1.885E+01	0.000E+00
	LSHADE44	3.549E+02	100%	0.000E+00	3.644E+02	0.000E+00
	LSHADE44-Eps	2.395E+02	100%	0.000E+00	2.262E+02	0.000E+00
C17	LSHADE44-IEps	9.923E-01	0	2.550E+01	8.710E-01	2.550E+01
	LSHADE44	1.048E+00	0	2.550E+01	1.050E+00	2.550E+01
	LSHADE44-Eps	1.049E+00	0	2.550E+01	1.050E+00	2.550E+01
C18	LSHADE44-IEps	1.165E+03	16%	4.094E+03	1.225E+02	5.844E+02
	LSHADE44	8.927E+01	84%	9.211E-01	3.822E+01	0.000E+00
	LSHADE44-Eps	4.448E+01	92%	8.478E-01	3.723E+01	0.000E+00
C19	LSHADE44-IEps	1.517E+01	0	3.614E+04	5.537E+00	3.613E+04
	LSHADE44	1.242E+01	0	3.613E+04	1.592E+01	3.613E+04
	LSHADE44-Eps	1.218E+01	0	3.613E+04	9.250E+00	3.613E+04
C20	LSHADE44-IEps	9.107E+00	100%	0.000E+00	9.009E+00	0.000E+00
	LSHADE44	1.074E+01	100%	0.000E+00	1.093E+01	0.000E+00
	LSHADE44-Eps	8.376E+00	100%	0.000E+00	8.281E+00	0.000E+00
C21	LSHADE44-IEps	3.623E+01	96%	2.269E+00	7.076E+00	0.000E+00
	LSHADE44	1.478E+01	100%	0.000E+00	1.462E+01	0.000E+00
	LSHADE44-Eps	9.935E+00	100%	0.000E+00	7.075E+00	0.000E+00
C22	LSHADE44-IEps	4.046E+06	0	2.105E+02	1.733E+06	1.373E+02
	LSHADE44	3.980E+06	0	2.005E+02	1.591E+06	1.668E+02
	LSHADE44-Eps	1.629E+06	4%	1.270E+02	6.637E+05	9.435E+01
C23	LSHADE44-IEps	1.191E+00	100%	0.000E+00	1.130E+00	0.000E+00
	LSHADE44	1.349E+00	100%	0.000E+00	1.308E+00	0.000E+00
	LSHADE44-Eps	1.386E+00	100%	0.000E+00	1.321E+00	0.000E+00
C24	LSHADE44-IEps	1.681E+01	100%	0.000E+00	1.806E+01	0.000E+00
	LSHADE44	2.108E+01	100%	0.000E+00	2.121E+01	0.000E+00
	LSHADE44-Eps	2.171E+01	100%	0.000E+00	2.121E+01	0.000E+00
C25	LSHADE44-IEps	1.859E+02	100%	0.000E+00	1.759E+02	0.000E+00
	LSHADE44	3.599E+02	100%	0.000E+00	3.581E+02	0.000E+00
	LSHADE44-Eps	3.637E+02	100%	0.000E+00	3.707E+02	0.000E+00
C26	LSHADE44-IEps	1.041E+00	0	2.550E+01	1.050E+00	2.550E+01
	LSHADE44	1.048E+00	0	2.550E+01	1.050E+00	2.550E+01
	LSHADE44-Eps	1.049E+00	0	2.550E+01	1.050E+00	2.550E+01
C27	LSHADE44-IEps	2.512E+02	44%	4.684E+03	3.419E+01	5.166E-03
	LSHADE44	5.912E+02	20%	9.248E+01	3.367E+01	7.762E-03
	LSHADE44-Eps	9.136E+01	32%	3.288E+01	3.517E+01	1.853E-03
C28	LSHADE44-IEps	2.863E+02	0	3.632E+04	2.575E+02	3.632E+04

	LSHADE44	2.784E+02	0	3.632E+04	2.743E+02	3.632E+04
	LSHADE44-Eps	2.868E+02	0	3.632E+04	2.775E+02	3.632E+04

表 3-7　LSHADE44-IEpsilon 及其他两种算法在 C01-C14 上的优化结果（$D=100$）

	算法	mean	SR	vio	median	v
C01	LSHADE44-IEps	3.355E+03	100%	0.000E+00	1.409E+03	0.000E+00
	LSHADE44	6.273E+03	100%	0.000E+00	9.014E+02	0.000E+00
	LSHADE44-Eps	3.626E+03	100%	0.000E+00	2.274E+03	0.000E+00
C02	LSHADE44-IEps	1.528E+03	100%	0.000E+00	7.409E+02	0.000E+00
	LSHADE44	5.088E+02	100%	0.000E+00	3.178E+02	0.000E+00
	LSHADE44-Eps	9.367E+02	100%	0.000E+00	4.671E+02	0.000E+00
C03	LSHADE44-IEps	1.488E+05	100%	0.000E+00	1.402E+05	0.000E+00
	LSHADE44	2.036E+07	96%	3.046E-06	1.089E+07	0.000E+00
	LSHADE44-Eps	3.262E+05	100%	0.000E+00	3.152E+05	0.000E+00
C04	LSHADE44-IEps	5.965E+02	100%	0.000E+00	5.962E+02	0.000E+00
	LSHADE44	6.464E+02	100%	0.000E+00	6.467E+02	0.000E+00
	LSHADE44-Eps	5.904E+02	100%	0.000E+00	5.836E+02	0.000E+00
C05	LSHADE44-IEps	6.867E+01	100%	0.000E+00	6.751E+01	0.000E+00
	LSHADE44	6.984E+01	100%	0.000E+00	6.788E+01	0.000E+00
	LSHADE44-Eps	7.654E+01	100%	0.000E+00	6.904E+01	0.000E+00
C06	LSHADE44-IEps	1.821E+03	100%	0.000E+00	1.814E+03	0.000E+00
	LSHADE44	1.625E+04	96%	7.662E-04	1.589E+04	0.000E+00
	LSHADE44-Eps	3.512E+03	100%	0.000E+00	3.453E+03	0.000E+00
C07	LSHADE44-IEps	-4.959E+02	28%	7.135E+02	-2.900E+02	6.653E+02
	LSHADE44	4.681E+01	56%	1.530E+02	2.772E+02	0.000E+00
	LSHADE44-Eps	-1.388E+02	56%	1.900E+02	2.568E+01	0.000E+00
C08	LSHADE44-IEps	9.544E+00	0	1.367E+02	6.130E+00	4.443E+01
	LSHADE44	1.604E+01	0	5.456E+02	1.009E+01	1.006E+02
	LSHADE44-Eps	1.289E+01	0	3.610E+02	9.060E+00	5.220E+01
C09	LSHADE44-IEps	5.218E+00	96%	1.271E-02	4.021E+00	0.000E+00
	LSHADE44	6.183E+00	96%	1.271E-02	4.528E+00	0.000E+00
	LSHADE44-Eps	7.536E+00	88%	3.814E-02	4.330E+00	0.000E+00
C10	LSHADE44-IEps	8.346E-01	0	1.254E+01	2.579E-01	1.905E+00
	LSHADE44	6.096E-01	0	1.245E+01	1.619E-01	5.182E-01
	LSHADE44-Eps	3.632E-01	0	1.928E+00	6.760E-01	8.299E-01
C11	LSHADE44-IEps	-7.775E+03	0	6.843E+02	-7.630E+03	5.073E+02
	LSHADE44	2.576E+01	0	2.805E+01	-2.964E+02	7.397E+00
	LSHADE44-Eps	-2.803E+03	0	7.428E+01	-2.503E+03	3.714E+01
C12	LSHADE44-IEps	1.338E+01	100%	0.000E+00	9.999E+00	0.000E+00
	LSHADE44	1.697E+01	100%	0.000E+00	1.886E+01	0.000E+00
	LSHADE44-Eps	1.829E+01	100%	0.000E+00	1.886E+01	0.000E+00
C13	LSHADE44-IEps	3.228E+06	0	2.212E+02	2.301E+06	2.059E+02
	LSHADE44	2.064E+06	0	1.668E+02	7.911E+05	1.434E+02

	LSHADE44-Eps	3.016E+06	0	2.224E+02	1.335E+06	1.982E+02
C14	LSHADE44-IEps	9.468E+00	52%	1.872E+04	9.176E-01	0.000E+00
	LSHADE44	9.740E-01	100%	0.000E+00	1.008E+00	0.000E+00
	LSHADE44-Eps	9.342E-01	100%	0.000E+00	9.243E-01	0.000E+00

表 3-8　LSHADE44-IEpsilon 及其他两种算法在 C15-C28 上的优化结果（$D=100$）

	算法	mean	SR	vio	median	v
C15	LSHADE44-IEps	2.435E+01	100%	0.000E+00	2.121E+01	0.000E+00
	LSHADE44	2.623E+01	96%	8.225E-06	2.435E+01	0.000E+00
	LSHADE44-Eps	2.875E+01	96%	3.596E-05	2.749E+01	0.000E+00
C16	LSHADE44-IEps	1.947E+02	100%	0.000E+00	1.948E+02	0.000E+00
	LSHADE44	7.016E+02	100%	0.000E+00	7.100E+02	0.000E+00
	LSHADE44-Eps	6.825E+02	100%	0.000E+00	6.990E+02	0.000E+00
C17	LSHADE44-IEps	1.097E+00	0	5.050E+01	1.100E+00	5.050E+01
	LSHADE44	1.098E+00	0	5.050E+01	1.100E+00	5.050E+01
	LSHADE44-Eps	1.092E+00	0	5.050E+01	1.100E+00	5.050E+01
C18	LSHADE44-IEps	9.438E+01	48%	2.577E+01	3.597E+01	3.887E-04
	LSHADE44	5.918E+01	68%	3.390E+00	4.896E+01	0.000E+00
	LSHADE44-Eps	6.648E+01	76%	5.080E+00	4.746E+01	0.000E+00
C19	LSHADE44-IEps	2.797E+01	0	7.300E+04	2.513E+01	7.300E+04
	LSHADE44	2.792E+01	0	7.300E+04	2.203E+01	7.300E+04
	LSHADE44-Eps	3.212E+01	0	7.301E+04	3.285E+01	7.301E+04
C20	LSHADE44-IEps	2.233E+01	100%	0.000E+00	2.235E+01	0.000E+00
	LSHADE44	2.584E+01	100%	0.000E+00	2.602E+01	0.000E+00
	LSHADE44-Eps	2.059E+01	100%	0.000E+00	2.008E+01	0.000E+00
C21	LSHADE44-IEps	7.713E+00	100%	0.000E+00	5.040E+00	0.000E+00
	LSHADE44	9.608E+00	100%	0.000E+00	1.000E+01	0.000E+00
	LSHADE44-Eps	1.090E+01	100%	0.000E+00	1.002E+01	0.000E+00
C22	LSHADE44-IEps	1.316E+07	0	4.811E+02	2.952E+06	3.076E+02
	LSHADE44	1.431E+07	0	6.091E+02	6.384E+06	4.865E+02
	LSHADE44-Eps	1.505E+07	0	6.066E+02	8.169E+06	4.615E+02
C23	LSHADE44-IEps	8.413E-01	100%	0.000E+00	8.002E-01	0.000E+00
	LSHADE44	9.621E-01	100%	0.000E+00	9.502E-01	0.000E+00
	LSHADE44-Eps	9.104E-01	100%	0.000E+00	8.987E-01	0.000E+00
C24	LSHADE44-IEps	2.083E+01	100%	0.000E+00	2.121E+01	0.000E+00
	LSHADE44	2.359E+01	100%	0.000E+00	2.435E+01	0.000E+00
	LSHADE44-Eps	2.322E+01	100%	0.000E+00	2.435E+01	0.000E+00
C25	LSHADE44-IEps	5.583E+02	100%	0.000E+00	5.592E+02	0.000E+00
	LSHADE44	7.451E+02	100%	0.000E+00	7.493E+02	0.000E+00
	LSHADE44-Eps	7.424E+02	100%	0.000E+00	7.430E+02	0.000E+00
C26	LSHADE44-IEps	1.099E+00	0	5.050E+01	1.100E+00	5.050E+01
	LSHADE44	1.099E+00	0	5.050E+01	1.100E+00	5.050E+01
	LSHADE44-Eps	1.099E+00	0	5.050E+01	1.100E+00	5.050E+01

C27	LSHADE44-IEps	1.223E+03	20%	1.398E+02	1.204E+02	8.688E+01
	LSHADE44	6.983E+03	0	1.461E+03	1.018E+04	3.435E+02
	LSHADE44-Eps	3.373E+03	8%	7.344E+02	1.205E+03	2.016E+02
C28	LSHADE44-IEps	6.326E+02	0	7.341E+04	6.357E+02	7.341E+04
	LSHADE44	6.319E+02	0	7.341E+04	6.533E+02	7.341E+04
	LSHADE44-Eps	6.045E+02	0	7.341E+04	6.179E+02	7.341E+04

表 3-9　LSHADE44-IEpsilon 基于均值与其他两种算法的比较情况

vs. LSHADE44-IEpsilon	符号	$D=10$	$D=30$	$D=50$	$D=100$
LSHADE44	+	9	11	10	10
	-	19	17	18	18
	=	0	0	0	0
LSHADE44-Epsilon	+	13	11	14	11
	-	14	17	14	17
	=	1	0	0	0

表 3-10　LSHADE44-IEpsilon 基于中位解与其他两种算法的比较情况

vs. LSHADE44-IEpsilon	符号	$D=10$	$D=30$	$D=50$	$D=100$
LSHADE44	+	6	6	6	6
	-	21	21	22	22
	=	1	1	0	0
LSHADE44-Epsilon	+	11	9	9	6
	-	17	18	19	22
	=	0	1	0	0

3.2.4.2 工程实例优化问题实验结果

对于两个工程实例优化问题压力容器优化问题及汽车侧撞结构设计问题，应用 LSHADE44-IEpsilon 及 LSHADE44 及 LSHADE44-Epsilon 进行 25 次独立运行优化，所有实验结果列于表 3-11 到表 3-12 中。每个表中，‘mean’表示平均目标函数值，SR 表示 25 次独立实验得到可行解的比例，‘vio’表示平均总约束违反值，‘Best’、‘worst’、‘median’表示最优、最差及中值解的目标函数值，‘v’表示中位解的总约束违反值，‘std’表示目标值标准差。粗体表示性能最佳。由表 3-11 及表 3-12 的实验结果可知，在求解这两个实际工程优化问题时，LSHADE44-IEpsilon 无论从均值或者中值上都是表现最好的算法。这表明了 LSHADE44-IEpsilon 在解决单目标约束工程实例优化问题时，同样是一种性能优越的算法。

表 3-11　三种算法在压力容器优化问题上 25 次独立运行优化结果

	LSHADE44-IEpsilon	LSHADE44	LSHADE44-Epsilon
Best	5.795E+03	5.795E+03	5.795E+03
mean	**5.898E+03**	6.009E+03	6.192E+03
worst	6.491E+03	6.439E+03	8.055E+03
std	1.997E+02	2.789E+02	5.508E+02
SR	100%	100%	100%
vio	0	0	0
median	**5.799E+03**	5.829E+03	5.900E+03
v	0	0	0

表 3-12　三种算法在汽车侧撞结构设计问题上 25 次独立运行优化结果

	LSHADE44-IEpsilon	LSHADE44	LSHADE44-Epsilon
Best	2.356E+01	2.356E+01	2.357E+01
mean	**2.362E+01**	2.370E+01	2.389E+01
worst	2.456E+01	2.440E+01	2.781E+01
std	1.976E-01	2.586E-01	8.440E-01
SR	100%	100%	100%
vio	0	0	0
median	**2.358E+01**	2.359E+01	2.363E+01
v	0	0	0

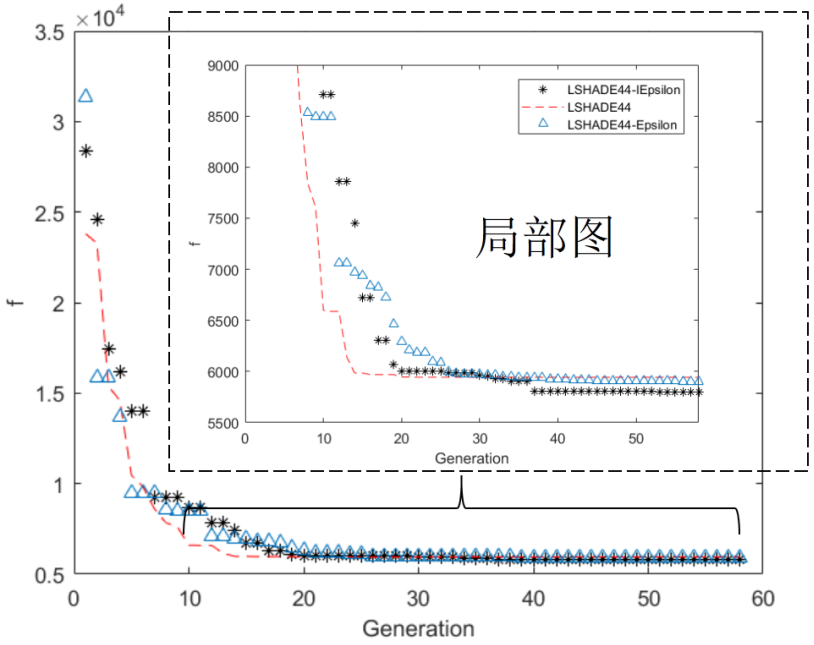


图 3-3　压力容器优化问题中三种算法的中位种群的收敛过程示意图

图 3-3 和图 3-4 显示了三种算法对这两个问题的中位种群的收敛过程。对于

压力容器优化问题，LSHADE44 算法虽然前期收敛最快，但在第 10 代后开始陷入局部收敛之后最优解基本保持不变，而 LSHADE44-Epsilon 算法速度收敛最慢，同时在第 30 代后基本进入局部收敛保持不变而，LSHADE44-IEpsilon 算法是唯一一个在 30 代后突破局部收敛的算法。对于汽车侧撞结构设计问题，LSHADE44-IEpsilon 算法是最快搜索到最优解附近的算法，LSHADE44 次之，而 LSHADE44-Epsilon 依然是收敛最慢的算法，并随着代数的增加缓慢逼近 LSHADE44-IEpsilon 算法搜索到的局部最优解。这表明了 LSHADE44-IEpsilon 比其他两种算法更易突破局部最优，并且在收敛速度上明显快于 LSHADE44-Epsilon。

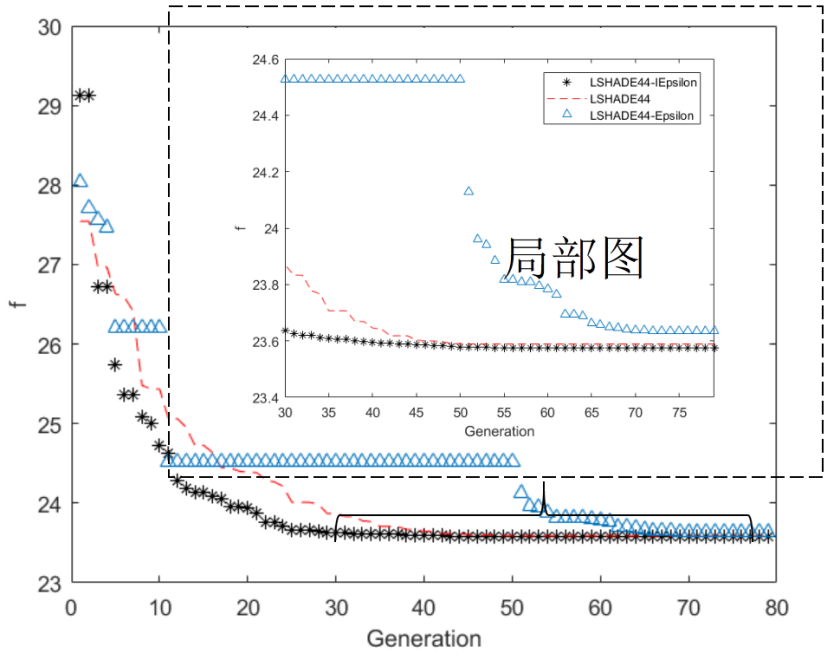


图 3-4　汽车侧撞结构设计问题中三种算法的中位种群的收敛过程示意图

3.3 本章小结

本章提出了一种改进的自适应 ϵ 约束处理机制 IEpsilon。同时文中将这种新型的机制应用到取得 2017 年国际会议 CEC 单目标约束优化竞赛中夺冠的 LSHADE44 算法中，提出了一种新的算法 LSHADE44-IEpsilon，并与 LSHADE44 及结合了 ϵ 约束处理机制的 LSHADE44-Epsilon 进行比较。

本章使用了 CEC2017 测试问题集及两个现实工程优化问题对算法性能进行测试。实验证明 LSHADE44-IEpsilon 在求解单目标约束优化问题时明显优于

LSHADE44 及结合原始ϵ约束处理机制的 LSHADE44-Epsilon，侧面说明了 IEpsilon 机制同样是一种极具竞争力的约束处理机制。总体而言，LSHADE44-IEpsilon 具有以下优点：

1）通过改进的ϵ约束处理机制，动态平衡了种群在可行区域在不可行区域里的搜索。

2）可以自适应根据当前种群可行比例自适应调节ϵ阈值，使种群满足搜索可行解的需求，同时增强了种群突破局部最优的能力，加快了收敛的速度。

第 4 章 多目标进化算法

4.1 多目标进化算法基础

优化问题根据目标函数数目的多少，可以划分为单目标，多目标及高维目标优化问题，其中用于解决解决多目标优化问题（Multi-objective Optimization Problems, MOPs）的进化算法，被称为多目标进化算法（Multi-objective Evolutionary Algorithms, MOEAs）。

4.1.1 基于 Pareto 规则的多目标最优解集

不同于单目标优化问题，多目标优化问题求解出来的目标函数是一个向量形式，而非标量形式，同时由于多目标优化问题的各个目标之间往往存在冲突，所以多目标优化问题往往是不存在绝对的或唯一的最优解。为了得到多目标优化问题的综合最优解，通常需要对各个目标进行折衷（Tradeoffs）考虑，因此在求解多目标优化问题时，我们通常会引入 Pareto 规则，从而得到了一个折衷的多目标最优解集。

定义 4.1：Pareto 支配

对于决策空间 R 内的任意两个解 $\vec{x}_a$ 和 $\vec{x}_b$，若满足以下 Pareto 支配规则，可认为 $\vec{x}_a$ 优于 $\vec{x}_b$，或称 $\vec{x}_a$ 支配 $\vec{x}_b$，记做 $\vec{x}_a \prec \vec{x}_b$：

1）对于所有的目标函数，$\vec{x}_a$ 不比 $\vec{x}_b$ 差。

2）至少存在一个目标函数，$\vec{x}_a$ 比 $\vec{x}_b$ 好。

即

$$\forall i\ f_i(\vec{x}_a) \le f_i(\vec{x}_b) \wedge \exists j\ f_j(\vec{x}_a) < f_j(\vec{x}_b) \tag{4-1}$$

其中 $i, j \in \{1,2,...,m\}$，m 为目标函数的数目。

定义 4.2：Pareto 解集

如果在对于决策空间 R 内存在一个解 $\vec{x}^*$，不被 R 内其他任意一个解支配，那么 $\vec{x}^*$ 被称为一个 Pareto 最优解。R 内的所有 Pareto 最优解组成的集合，称为 Pareto

解集（Pareto Set, PS）。

定义 4.3：Pareto 前沿

PS 在目标空间上的映射向量集合称为 Pareto 前沿（Pareto Front, PF）。可定义为以下形式：$PF = \{\vec{F}(\vec{x}) \mid \vec{x} \in PS\}$。

一般地，双目标优化问题的 PF 通常在目标空间表现为曲线形式，三目标优化问题的 PF 表现为曲面形式，而高维目标优化问题的 PF 则呈超曲面的形态。如图 4-1 所示，黑色实线是一个双目标优化问题的 PF，其中实心点 a，b，c，d，e 为 PF 上的个体，是映射在目标空间上 Pareto 最优解，它们不被目标空间搜索区域的任意一个个体支配，被称为非支配解（Non-dominated Solutions）。而空心点 f，g，h，i，j 都不在 PF 上面，在搜索区域内必然存在至少一个可以支配（优于）它们的解，所以它们被称为支配解（Dominated Solutions）。

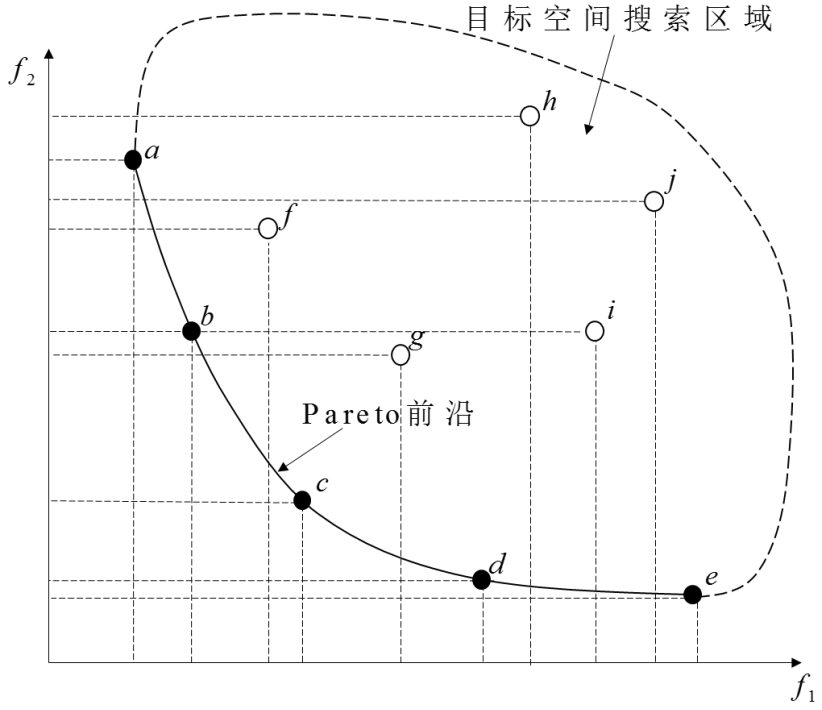


图 4-1 Pareto 支配示意图

4.1.2 多目标进化算法研究概述

多目标进化算法是组成约束多目标进化算法的基础，如同其他经典的智能算法研究一样，多目标进化算法也有着一段漫长的发展历程。

1967 年，Rosenberg 首次提出使用进化搜索的方法来解决多目标优化问题，

但最终并没有实现[64]。1984 年，David Schaffer 成功实现向量评估遗传算法（Vector Evaluated Genetic Algorithm，VEGA）并应用于机器学习中[65]。1989 年，David Goldberg 提出了一套完整的多目标进化算法技术理论，对多目标进化算法之后的发展方向有着重要的指导意义[66]。1990 年以后，多种优秀的多目标进化算法被相继提出，其中最具代表性的算法包括 Multi-objective Genetic Algorithm（MOGA）[67], Non-Dominated Sorting Genetic Algorithm（NSGA）[68]，和 Niched Pareto Genetic Algorithm（NPGA）[69]。这类算法融入了 Pareto 最优的概念，采用了非支配排序的机制。1999 年，Eckart Zitzler 等人提出了 Strength Pareto Evolutionary Algorithm（SPEA）[70]，第一次提出了精英保留策略，直接将当前种群中优秀的个体保留到下一代中。基于精英保留思想，更多优异的算法被提出。其中就有 Pareto Envelope-Based Selection Algorithm（PESA）[71]、Pareto Archived Evolution Strategy（PAES）[72]、SPEA2[73]、NSGA-II[74]以及 NPGA2[75]等。

2004 年以后，更多类型的机制被应用到多目标进化计算领域。2004 年, Eckart Zitzler 提出了基于指标的 Indicator-based Evolutionary Algorithm（IBEA）[76]，使用当前种群的性能评估指标来作为推动种群进化的选择压力。2005 年，Coello Coello 等人提出了 Muti-objective Particle Swarm Optimization（MOPSO）[77]，将原本只应用于单目标优化问题的粒子群算法拓展到多目标优化领域。

2007 年，国内的张青富教授首次提出了一种基于分解的多目标进化算法 MOEA Based on Decomposition（MOEA/D）[78]，这种算法利用目标空间中均匀分布的权重向量，运用一类传统运筹学的方法，将多目标优化问题分解为多个单目标优化子问题，然后同时求解各个子问题，避免了多目标 Pareto 比较频繁出现的不可比较性（个体间互不支配）。由于各个子问题关联的子解都通过与其最近的邻居解变异交叉操作后实现更新，所以 MOEA/D 最终得到的解可以很好地平衡进化种群的收敛性与多样性。基于 MOEA/D 求解的优异特性，自其提出之日起便受到了研究学者的广泛关注，各种 MOEA/D 的改进版本被相继提出，如 MOEA/D-DE[79]，MOEA/D-M2M[80]，EAG-MOEA/D[81]，MOEA/D-SAS[82]等，同时一类解决约束优化问题的改进 MOEA/D 也被相继提出，如 C-MOEA/D[83]，

MOEA/D-CDP[84]，MOEA/D-Epsilon[85]及 MOEA/D-SR[84]等。

4.1.3 多目标进化算法性能评估指标

一般而言，要评估一个算法的性能，需要一类可数值化的性能评估指标来直观表现。目前，国内外研究学者针对 MOEA 提出了许多评价工具，主要评估 MOEA 的两类性能：收敛性和分布性（或称多样性）。对 MOEA 进行评估时，两个参数十分重要，分别是已得的 Pareto 前沿 PF_{known} 及实际的 Pareto 前沿 PF_{true}。

基于收敛性进行考量，我们希望：

1） PF_{known} 尽可能趋近于 PF_{true}。

2） PF_{known} 尽可能完整布满整个 PF_{true}。

基于分布性进行考量，我们希望：

1） PF_{known} 尽可能地均匀分布。

2） PF_{known} 尽可能拥有较大的分布广度。

本文主要介绍三种最常用到的 MOEA 性能评估指标。

1）世代距离（Generational Distance, GD）指标

GD 指标表示从 PF_{known} 到 PF_{true} 的平均距离，能够用来评估 MOEA 的算法收敛性。一般地，GD 指标越小，说明算法的收敛性越好。

$$\begin{cases} GD(A,P^*)=\dfrac{\sqrt{\sum_{v\in A}(d(v,P^*))^2}}{|A|} \\ d(v,P^*)=\min\limits_{x^*\in P^*}\left\{\sqrt{\sum_{i=1}^{m}(f_i(v)-f_i(x^*))^2}\right\} \end{cases} \tag{4-2}$$

式中 A 表示 PF_{known} 上的解集，而 P^* 表示 PF_{true} 上均匀分布的采样个体。$d(v,P^*)$ 表示 PF_{known} 的个体 v 到 P^* 的最小欧氏距离。

2）反世代距离（Inverted Generational Distance, IGD）指标

IGD 指标表示从 PF_{true} 到 PF_{known} 的平均距离，能够同时评估 MOEA 的算法收敛性和分布性。一般地，IGD 指标越小，说明算法的收敛性和分布性越好。

$$\begin{cases} IGD(P^*, A) = \dfrac{\sum_{y^* \in P^*} d(y^*, A)}{|P^*|} \\ d(y^*, A) = \min\limits_{y \in A} \left\{ \sqrt{\sum_{i=1}^{m} (y_i^* - y_i)^2} \right\} \end{cases} \tag{4-3}$$

式中 A 表示 PF_{known} 上的解集，而 P^* 表示 PF_{true} 上均匀分布的采样个体。本文中，在优化双目标优化问题上，P^* 在 PF_{true} 上均匀采样 1000 个个体，在优化三目标优化问题上，P^* 在 PF_{true} 上均匀采样 10000 个个体。$d(y, P^*)$ 表示 PF_{true} 的个体 y 到 A 的最小欧氏距离。

3）超体积（Hypervolume, HV）指标

HV 指标可以反映 PF_{true} 及 PF_{known} 之间的紧密程度，能够同时评估 MOEA 的算法收敛性和分布性。一般地，HV 指标越小，说明算法的收敛性和分布性越好。

$$HV(S) = VOL\left(\bigcup_{x \in S} [f_1(x), z_1^r] \times ... [f_m(x), z_m^r] \right) \tag{4-4}$$

式中 $VOL(\cdot)$ 表示一个 Lebesgue 积分函数，S 表示 PF_{known} 上的解集。$\vec{z}^r = (z_1^r, z_2^r, ..., z_m^r)^T$ 为目标空间中参考向量，本文统一设置参考向量值为极端点的 1.4 倍。其中 $\vec{z}^n = (z_1^n, z_2^n, ..., z_m^n)^T$ 为极端点，且 $z_i^n = \max\left\{ f_i\left(\vec{x}\right) \middle| \vec{x} \in R \right\}, i = 1, 2, ..., m$。HV 指标实际反映的是参考点与 PF_{known} 间围成的超体积大小。以双目标优化为例，如图 4-2 所示，黑色实点为 PF_{known} 的解集，空心点为参考向量，蓝色区域大小即为超体积指标的大小。

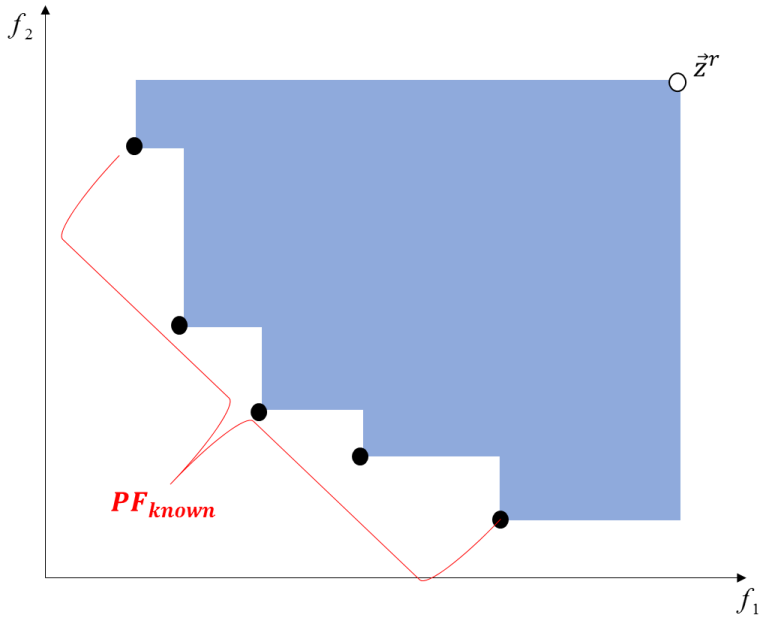


图 4-2 HV 指标原理示意图

4.2 基于分解的多目标进化算法 MOEA/D

MOEA/D 作为一种经典的解决多目标优化问题的算法，其核心思想是将一个多目标优化问题分解为多个单目标优化子问题[78]，逐一求解最终得到 PF 上的各个 Pareto 最优解。MOEA/D 的一般算法流程如图 4-3 所示。

4.2.1 MOEA/D 分解方法

一般地，在 MOEA/D 中，需要先初始化得到 N 个均匀分布的权重向量 $\lambda=(\vec{\lambda}^1,...,\vec{\lambda}^N)$，通过它们将一个 m 目标优化问题分解出 N 个子问题。其中的任意一个权重向量 $\vec{\lambda}^i=(\lambda_1^i,...,\lambda_m^i)^T, i=1,2,...,N$ 需满足以下条件：

$$\sum_{k=1}^{m}\lambda_k^i=1 \text{ 且 } \lambda_k^i\geq 0, k\in\{1,2,...,m\} \tag{4-5}$$

MOEA/D 中，主要有三种常用的分解子问题的方法：加权求和法、切比雪夫法、边界交叉法，以下对这三种方法作详细介绍：

1）加权求和法（Weighted Sum Approach）

对于一个多目标优化问题，可通过将各目标函数通过加权求和的方式，将这个问题转化为多个单目标优化子问题。$\vec{\lambda}^i$ 对应分解得到的第 i 个子问题的数学表

达形式如下所示：

$$\begin{cases} \text{minimize } g^{ws}\left(\vec{x}\middle|\vec{\lambda}^{i}\right)=\sum_{k=1}^{m}\lambda_k{}^{i} f_k(x) \\ \text{subject to } \vec{x}\in R \end{cases} \tag{4-6}$$

式中 $g^{ws}\left(\vec{x}\middle|\vec{\lambda}^{i}\right)$ 为第 i 个子问题的聚合函数，$\vec{x}$ 为决策变量，R 为决策变量搜索空间。加权求和法只适用于求解凸的优化问题，所以局限性较大，一般不采用这种方法。

2）切比雪夫法（Tchebycheff Approach）

通过切比雪夫方法，可将一个多目标优化问题转化为多个单目标优化子问题。$\vec{\lambda}^{i}$ 对应分解得到的第 i 个单目标子问题的数学表达形式如下所示：

$$\begin{cases} \text{minimize } g^{te}\left(\vec{x}\middle|\vec{\lambda}^{i},\vec{z}^{*}\right)=\max\limits_{1\le k\le m}\left\{(1/\lambda_k^{i})\cdot\left|f_k\left(\vec{x}\right)-z_k^{*}\right|\right\} \\ \text{subject to } \vec{x}\in R \end{cases} \tag{4-7}$$

式中 $g^{te}\left(\vec{x}\middle|\vec{\lambda}^{i}\right)$ 为第 i 个子问题的聚合函数，$\vec{x}$ 为决策变量，R 为决策变量搜索空间。其中 $\vec{z}^{*}=(z_1^{*},z_2^{*},...,z_m^{*})^{T}$ 为理想点，且 $z_i^{*}=\min\left\{f_i\left(\vec{x}\right)\middle|\vec{x}\in R\right\}$，$i=1,2,...,m$。

3）边界交叉法（Penalty-based Boundary Intersection Approach, PBI）

边界交叉方法也是一种经典的分解方法。使用这方法，$\vec{\lambda}^{i}$ 对应分解得到的第 i 个单目标子问题的数学表达形式如下所示：

$$\begin{cases} \text{minimize } g^{bip}\left(\vec{x}\middle|\vec{\lambda}^{i},\vec{z}^{*}\right)=d_1+\theta d_2 \\ \text{subject to } \vec{F}\left(\vec{x}\right)-\vec{z}^{*}=d_1\vec{\lambda}^{i} \\ \qquad\qquad d_2=\|\vec{F}\left(\vec{x}\right)-\vec{z}^{*}-d_1\vec{\lambda}^{i}\| \\ \qquad\qquad \vec{x}\in R \end{cases} \tag{4-8}$$

式中 $g^{bip}\left(\vec{x}\middle|\vec{\lambda}^{i}\right)$ 为第 i 个子问题的聚合函数，$\vec{x}$ 为决策变量，R 为决策变量搜索空间。其中 $\vec{z}^{*}=(z_1^{*},z_2^{*},...,z_m^{*})^{T}$ 为理想点，且 $z_i^{*}=\min\left\{f_i\left(\vec{x}\right)\middle|\vec{x}\in R\right\}$，$i=1,2,...,m$。如图 4-4 所示，$d_1$ 为理想点沿权重向量到 $\vec{F}\left(\vec{x}\right)$ 的距离，d_2 为到 $\vec{F}\left(\vec{x}\right)$ 权重向量的距离。PBI 方法跟切比雪夫方法一样可以搜索任意特性的优化问题，但由于多了一个需要调节的惩罚参数 θ，其值的确定比较麻烦，所以本文的 MOEA/D 统一采

用切比雪夫方法。

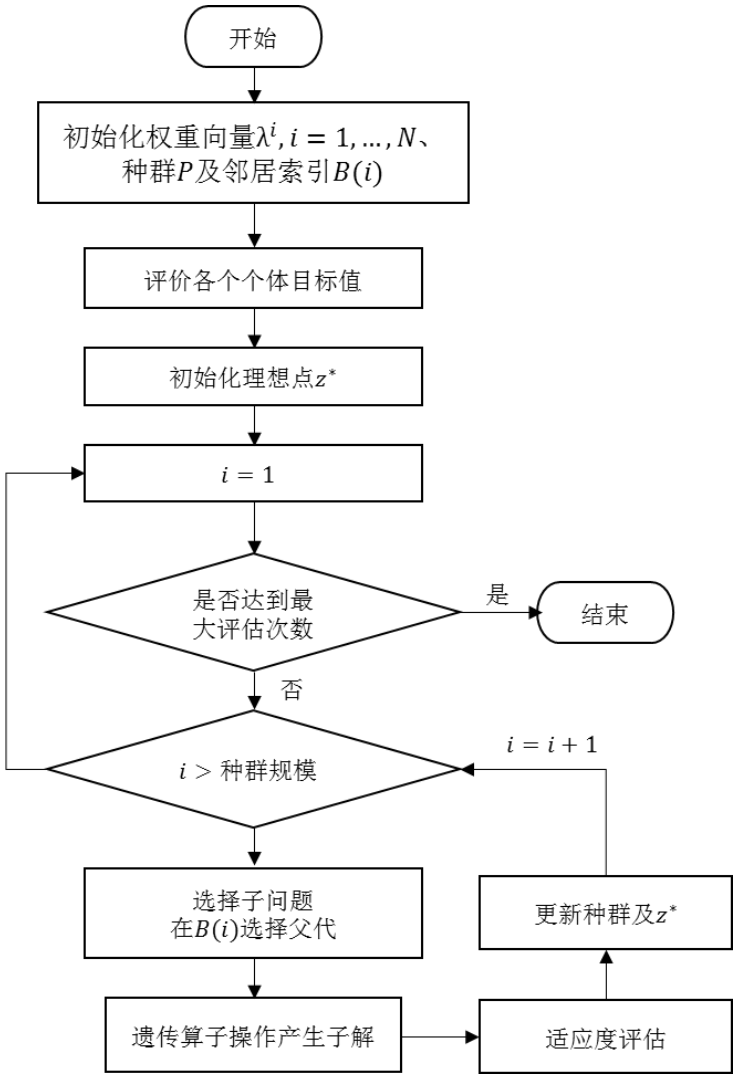


图 4-3 MOEA/D 算法流程图

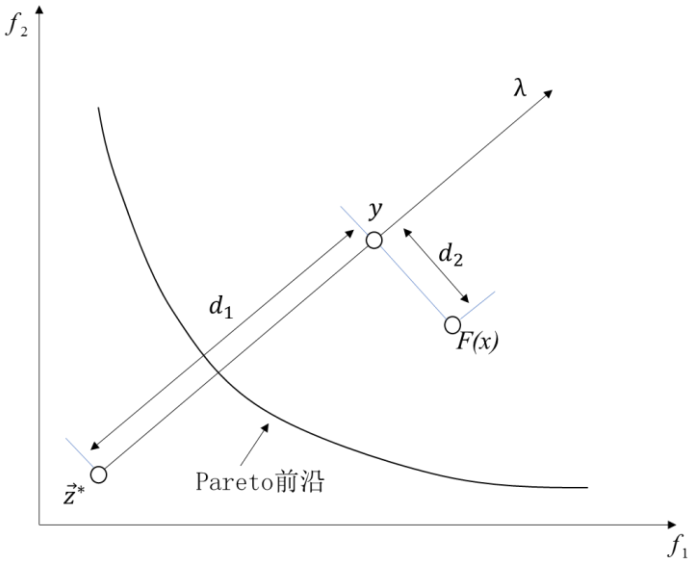


图 4-4 PBI 分解方法示意图

4.2.2 MOEA/D 邻居机制

在 MOEA/D 算法中，首创性地引入了邻居的机制。根据权重向量在目标空间的欧氏距离，可以确定权重向量间的间隔距离，MOEA/D 定义每个权重向量 $\vec{\lambda}^i, i=1,2,...,N$ 的邻居向量为距离 $\vec{\lambda}^i$ 最近的 T 个权重向量。假设 $\vec{\lambda}^{i_1}, \vec{\lambda}^{i_2}, \cdots, \vec{\lambda}^{i_T}$ 为 $\vec{\lambda}^i$ 的邻居向量，则 $B(i)=(i_1,i_2,...,i_T)\in(1,2,...,N)$ 称为 $\vec{\lambda}^i$ 的邻居索引。

每个权重向量 $\vec{\lambda}^i$ 关联一个解 $\vec{x}_i$，MOEA/D 通过随机选择 $\vec{\lambda}^i$ 的两个邻居向量关联的解作为父解，然后通过交叉变异操作产生子解 $\vec{y}_i$，从而根据聚合函数值更新 $\vec{\lambda}^i$ 上及其邻居向量上关联的解。一般地，以切比雪夫方法为例，对于一个个体解 $\vec{x}^i$，互为邻居的两个权重向量 $\vec{\lambda}^j$ 和 $\vec{\lambda}^k$（$j,k\in B(i)$）所构成的聚合函数的值 $g^{te}\left(\vec{x}_i\middle|\vec{\lambda}^j,\vec{z}^*\right)$ 和 $g^{te}\left(\vec{x}_i\middle|\vec{\lambda}^k,\vec{z}^*\right)$ 是比较接近的。因此使用邻居机制进行遗传操作父代选择及更新种群，一方面加强了种群的局部搜索能力，一方面也节省了计算资源。

4.3 本章小结

本章主要介绍了多目标进化算法基础知识，作为之后研究的基础。基于多目标比较中常出现的冲突情况，本章首先介绍了根据 Pareto 规则的支配关系，引出了 Pareto 最优解集、Pareto 前沿及非支配解等多目标优化领域的基本概念。其次本章介绍了多目标进化算法的发展历史，并介绍了多目标进化计算领域最常用的三个算法性能评价指标：GD、IGD 和 HV，为本文后续进行算法性能评估提供了工具。最后本章介绍了经典的多目标进化算法 MOEA/D，因其独特的分解机制，在算法进化过程中具有多样性保持及局部搜索能力强等特点，适用于解决各类型的优化问题。后续的约束多目标进化算法研究将基于 MOEA/D 开展。

第 5 章 基于角度约束支配规则的 MOEA/D 算法

本章我们将提出一种基于角度的约束支配规则（Angle-based Constrained Dominance Principle, ACDP），并应用于 MOEA/D 框架中。提出的新算法 MOEA/D-ACDP 将在约束多目标测试问题集及实际工程应用问题上，与其他四种同样基于分解的经典的约束多目标进化法进行测试及比较，并进行分析。

5.1 约束多目标进化算法 MOEA/D-ACDP

5.1.1 基于角度的约束支配规则 ACDP

在经典的 CDP 方法[36]中，它的三项基本规则规定了总约束违反值在进化过程中是最重要的考虑因素。应用 CDP 方法，当进化种群的可行解比例足够大时，不可行解往往更趋向于被直接舍弃，一些不可行区域里有用信息在优化的过程中有时候很容易被忽略，从而使种群陷入局部可行区域中。反正如果能够在求解复杂约束优化问题时，根据种群中的有用信息以一定的机制适当保留不可行解可以有效地提升最终求解的质量。

种群多样性保持在多目标进化算法中非常重要，一些早期的算法研究例如 NSGA-II 中，通常使用欧式距离来作为种群聚集密度的计算基准，从而表示种群的多样性尺度。而在近期，文献[86][87]中阐述了角度信息能比距离更有效地表示多样性尺度。任意两个解在目标空间上的角度信息能有效地反映出两个解的相似程度，因此可以作为反映种群多样性的辅助信息。

本小节依据个体间存在角度信息设计出一种新的约束处理机制 ACDP。在进化过程中，ACDP 依据解的角度信息，适当地保留一些优异的不可行解，在维持种群的多样性的当时，加速种群收敛性。对于 ACDP，我们给出相关定义如下所示：

定义 5.1 解间的角度

假设存在任意的两个解 $\vec{x}_1 \in R$ 和 $\vec{x}_2 \in R$，则在这两个解在目标空间中的角度如下式表示：

$$angle(\vec{x}_1,\vec{x}_2,\vec{z}^*)=\arccos\left(\frac{(\vec{F}(\vec{x}_1)-\vec{z}^*)^T\cdot(\vec{F}(\vec{x}_2)-\vec{z}^*)}{\|\vec{F}(\vec{x}_1)-\vec{z}^*\|\cdot\|\vec{F}(\vec{x}_2)-\vec{z}^*\|}\right) \tag{5-1}$$

其中 $\vec{z}^*$ 为理想点，而 $\|\cdot\|$ 表示一个向量的二范数。

依旧定义，两个解的角度即为它们在目标空间中基于参考点的夹角。如图 5-1 所示，在目标空间中，$\vec{x}_1$ 和 $\vec{x}_2$ 的角度为 θ_1^2。显然目标空间中任意两个解的角度大小会落在区间 $[0,\pi/2]$ 上。

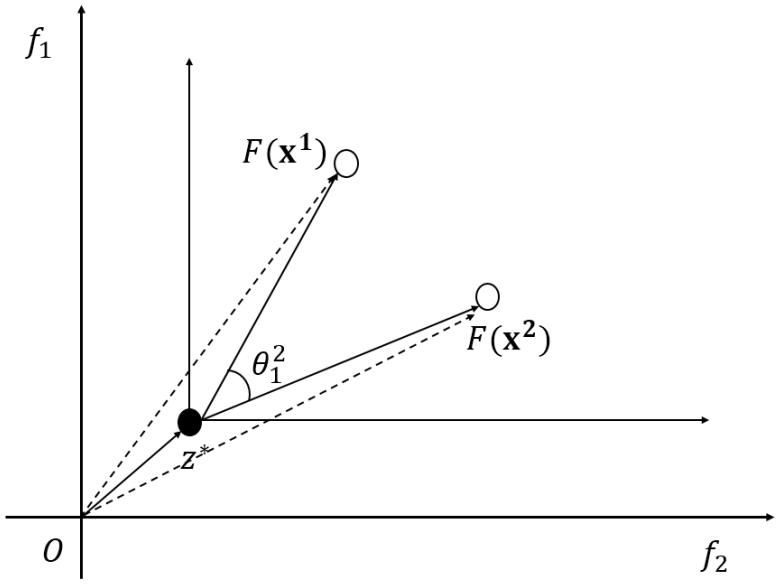


图 5-1　解 $\vec{x}_1$ 和 $\vec{x}_2$ 角度示意图

定义 5.2 可行解比例

对于一个给定种群，已知种群规模为 N，当前种群中可行解数目为 num_{fea}。则定义可行解比例 r 如下定义：

$$r=\frac{num_{fea}}{N} \tag{5-2}$$

定义 5.3 基于角度的约束支配规则 ACDP

存在任意两个解 $\vec{x}_1\in R$ 和 $\vec{x}_2\in R$，假如给定一个阈值参数 θ，则根据这两个解的比较情况，ACDP 规则定义如下：

1）如果 $\vec{x}_1$ 和 $\vec{x}_2$ 都为可行解，则根据 Pareto 规则被支配的那个解更差。

2）如果这两个解中至少存在一个不可行解，且 $angle(\vec{x}_1,\vec{x}_2,\vec{z}^*)\le\theta$，则总约

束违反值大的解更差。

3）如果这两个解中至少存在一个不可行解，且 $angle(\vec{x}_1,\vec{x}_2,\vec{z}^*)>\theta$，则当 $rand()<r$，则根据 Pareto 规则被支配的那个解更差。

除了上述三种情况外，$\vec{x}_1$ 和 $\vec{x}_2$ 互不可比较。其中 $rand()$ 为一个[0,1]之间的一个随机值。

MOEA/D 作为一种使用分解函数来更新其邻居的算法，直接应用 ACDP 规则是不可行的，这里我们提供一种适用于 MOEA/D 更新邻居的 ACDP 修正版本。

给定一个由权重向量 $\vec{\lambda}$ 分解得到的子问题 sp，对于解 $\vec{x}_1$ 和 $\vec{x}_2$，它们的分解函数值是 ϕ_1 和 ϕ_2，分解函数值为 $g^{te}(\vec{x}_1|\vec{\lambda},\vec{z}^*)$ 和 $g^{te}(\vec{x}_2|\vec{\lambda},\vec{z}^*)$。我们规定当满足以下 3 条适用于 MOEA/D 的 ACDP 规则时，$\vec{x}_1$ 比 $\vec{x}_2$ 好，记做 $\vec{x}_1 \prec_\theta \vec{x}_2$

$$\vec{x}_1 \prec_\theta \vec{x}_2 \Leftrightarrow \begin{cases} \textbf{Rule1 if } \phi^1=0,\phi^2=0: \\ \quad g^{te}(\vec{x}_1|\vec{\lambda},\vec{z}^*)<g^{te}(\vec{x}_1|\vec{\lambda},\vec{z}^*); \\ \textbf{Rule2 if } \phi^1\neq\phi^2: \\ \quad angle(\vec{x}_1,\vec{x}_2,\vec{z}^*)\le\theta \\ \quad \phi^1<\phi^2; \\ \textbf{Rule3 if } \phi^1\neq\phi^2: \\ \quad angle(\vec{x}_1,\vec{x}_2,\vec{z}^*)>\theta, rand()<r, \\ \quad g^{te}(\vec{x}_1|\vec{\lambda},\vec{z}^*)<g^{te}(\vec{x}_2|\vec{\lambda},\vec{z}^*). \end{cases} \tag{5-3}$$

式中 θ 是一个阈值参数，需要由规则使用者预先设定。在公式(5-3)中，当 $\theta\ge\pi/2$ 时，ACDP 等同于 CDP。因为当 $\theta\ge\pi/2$ 时，$angle(\vec{x}_1,\vec{x}_2,\vec{z}^*)\le\theta$ 恒成立。则此时 ACDP 规则在忽略了角度比较的情况下只考虑约束违反值的比较及分解函数值的比较，可等同于 CDP 三条基本规则。那么本文以下的讨论将基于阈值 $\theta<\pi/2$ 的情况下进行讨论。

首先，在 ACDP 的第一条规则中，当 $\vec{x}_1$ 和 $\vec{x}_2$ 都为可行解时，ACDP 根据两个解的分解函数值进行比较比较，这与 CDP 的第一条规则完全相同。

而当 $\vec{x}_1$ 和 $\vec{x}_2$ 至少有一个是不可行解时，ACDP 和 CDP 规则区别就十分明显了。CDP 只利用两个解的总约束违反值进行比较，从而决定它们的取舍，然而只

根据这样的规则更新留下来的种群很难继续维持种群的多样性，尤其当当前种群中大多数个体都是不可行解时。而 ACDP 则引入了其它两个额外的信息来进行辅助对这两个解进行比较。其一是解之间的信息。其二是当前种群的可行比例。

在 ACDP 第二条规则中，当$\vec{x}_1$和$\vec{x}_2$的角度小于θ时，ACDP 认为这两个解是相似的，基于 MOEA/D 的节本框架，这两个解被认为与当前子问题 *sp* 的关联性相似，此时，如果只使用总约束违反值来比较这两个解，不会引起种群多样性的缺失。

在 ACDP 第三条规则中，当$\vec{x}_1$和$\vec{x}_2$的角度大于θ时，ACDP 认为这两个解是不相似的，此时如果直接根据总约束违反值来进行比较，可能会使一个新颖的解有用信息丢失。所以 ACDP 中以一定概率 r 认为，在这两个解中具有更小的分解函数值的解更好。通过这一规则，ACDP 使得一些新颖的拥有较好的分解函数值的不可行解有机会保留在种群中，这样一个机制既能补充种群的多样性，也能有效地加速种群的收敛，避免种群陷入局部最优。

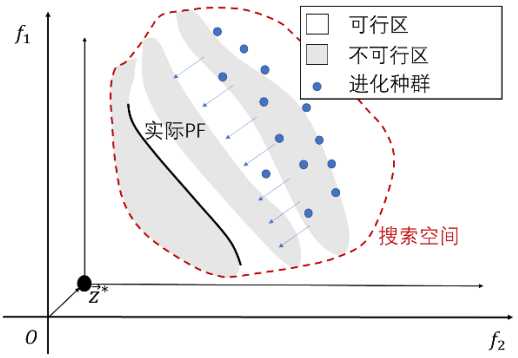
需要补充一点是，在第三条规则中，概率 r 的值被设置为当前种群的可行比例。这是为了维持种群在可行区和不可行区间的搜索的动态平衡。当可行比例较大时，一些新颖的拥有较好的分解函数值的不可行解更可能被留下来，而当可行比例较小时，一些新颖的不可行更可能被舍弃。

5.1.2 ACDP 机制进化过程

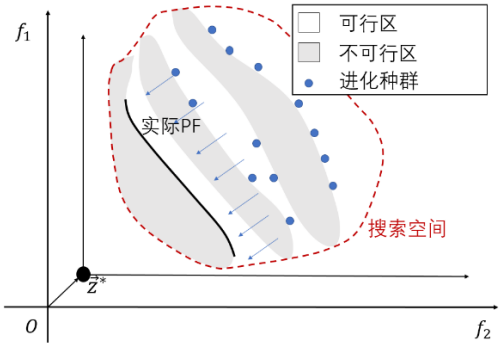
在本小节中，我们将基于 MOEA/D 的算法框架，对经典约束处理机制 CDP 和我们提出的 ACDP 在处理约束多目标优化问时是的机制运作过程进行比较分析。进化过程中，我们一般可以将进化种群的进化过程分为三种阶段。

第一阶段，如图 5-2（a）和图 5-3（a）所示，算法会初始随机生成一个种群，此时种群中的多数个体离实际 PF 较远。第二阶段，种群开始在搜索空间内进行探索。如图 5-2（b）所示，当 MOEA/D 使用了 CDP 机制，种群将会快速地被吸引到可行区域中，然而由于个体难以进入不可行区域，之后种群将很难穿过不可行区。当 MOEA/D 中使用 ACDP 机制时，如图 5-3（b）所示，算法将根据角度信息维持进化种群多样性，同时，一部分种群个体进入到不可行区域，这将有效

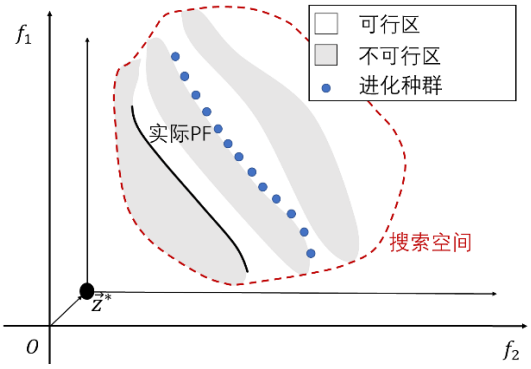
地帮助种群穿越不可行区域。此外，ACDP 还利用了当前种群的可行比例，平衡了种群在可行区和不可行区之间的搜索，这不仅避免了种群过多地落入不可行区从而搜索不到可行解，同时也避免了种群大量地留在可行区而停滞搜索。第三阶段，进化种群开始收敛至当前种群附近的可行区域内。如图 5-2（c），如果使用 CDP 机制，种群因为种群难以进入并穿越不可行区域，所以有可能收敛到局部最优的位置。而如图 5-3（c）所示，如果使用 ACDP 机制，由于进化过程中种群同时平衡搜索了可行区域和不可行区域，并始终基于角度维持着种群多样性，所以进化种群最终能更完整地收敛至实际的 PF 上。



（a）第一阶段

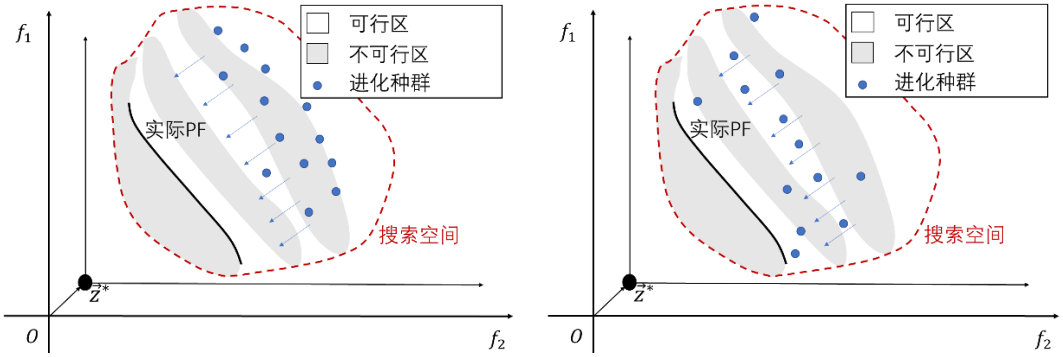


（b）第二阶段

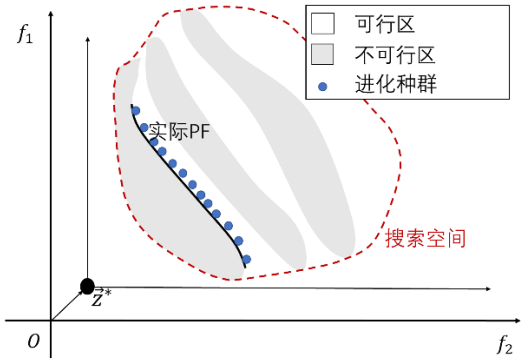


（c）第三阶段

图 5-2 基于 CDP 机制的 MOEA/D 进化过程



（a）第一阶段　　（b）第二阶段



（c）第三阶段

图 5-3 基于 ACDP 机制的 MOEA/D 进化过程

5.1.3 ACDP 阈值参数设置

在早期进化过程中，种群通常离实际 PF 较远。为了防止种群过早陷入局部最优，种群中应该尽量多搜索一些新颖的拥有较好分解函数值的解，以维持种群的多样性。随着进化进程的推进，算法则应逐步加强。种群收敛至可行区域。基于上述讨论，阈值参数 $\theta(k)$ 应随着进化代数 k 的增加而逐渐增加。本文提出一种 $\theta(k)$ 的设置方法如下式所示：

$$\theta(k)=\begin{cases}\theta_0\left(1+\dfrac{k}{T_{max}}\right)^{cp}, & 1\le k\le T_c\\ \dfrac{\pi}{2}, & T_c<k\le T_{max}\end{cases} \tag{5-4}$$

其中 θ_0 为一个初始阈值，这里设置为 $\pi/2N$ 。N 是种群规模，T_{max} 是最大进化代数，$T_c=\alpha T_{max}$ 是阈值调整的终止进化代数。cp 初始化设置为 $\log(N)/\log(1+\alpha)$，而参数 $\alpha=0.8$ 。

根据公式(5-4)，$\theta(k)$ 由 $\pi/2N$ 随着 k 的增加逐渐增加，当 $k=T_c$ 时，$\theta(k)=\pi/2$ 随后保持不变。已知两个均匀分布的权重向量间的最大夹角为 $\pi/2$，则每个相邻的权重向量的平均角度为 $\pi/2N$，所以 θ_0 初始设置为 $\pi/2N$ 。初始时，当两解角度小于这个值，则认为这两个解关联于同一个权重向量分解得到的子问

题，随后阈值逐渐缩小，判断两解相似的条件将会越来越放松，直至 $\theta = \pi / 2$，ACDP 等价于 CDP，总约束违反值成为两解比较的首要考虑因素，不再考虑两解间的角度信息对多样性的维持，而种群将趋向于收敛至可行区域。以图 5-4 为例，假设种群规模 $N = 300$，最大进化代数 $T_{max} = 500$，θ 初始设置为 $\pi / 2N$。在早期进化过程中，θ 持续但非常缓慢地增加，使得种群较好地维持多样性。随着进化代数 k 越来越接近 T_c，θ 增长越来越快，这使得种群逐渐加速收敛到可行区域内。当 k 到达 T_c 时，$\theta = \pi / 2$ 并持续维持此值直至 $k = T_{max}$。

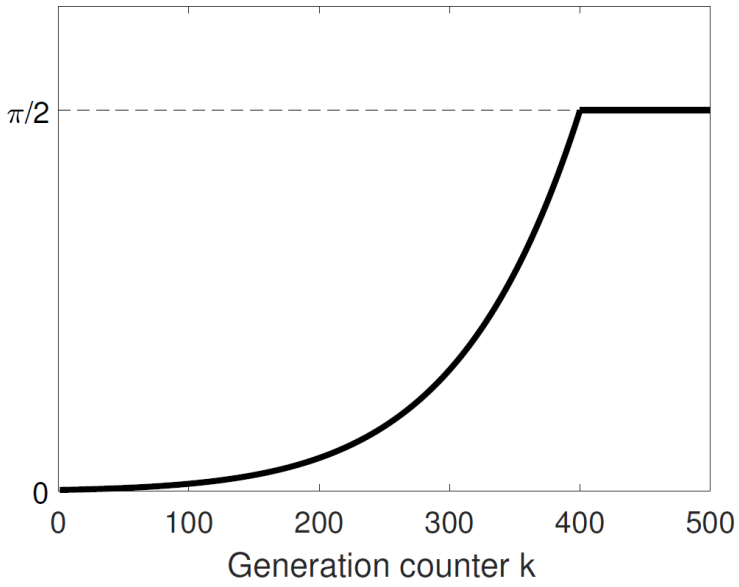


图 5-4　阈值 θ 变化趋势

5.1.4 MOEA/D-ACDP 算法流程

本文结合 MOEA/D 的算法框架及基于角度的约束支配规则 ACDP，提出了一种解决多目标约束优化问题的约束多目标进化算法 MOEA/D-ACDP，它的具体算法框架如下所示：

输入：

N：种群规模/权重向量数目/子问题数目。N 个权重向量：$\lambda = (\vec{\lambda}^1, ..., \vec{\lambda}^N)$。

T：邻居数目。δ：邻居索引选择概率。n_r：子解更新最大数目

α：ACDP 算法参数。 θ_0：初始阈值参数。

输出： NS：可行的非支配解集。

步骤 1：初始化

a） 将一个约束多目标优化问题分解成与 $\vec{\lambda}^1,...,\vec{\lambda}^N$ 相关联的 N 个子问题

b） 随机初始一个种群 $P=\{\vec{x}_1,...,\vec{x}_N\}$。

c） 参数阈值 $\theta=\theta_0$，计算参数 $cp=\log(N)/\log(1+\alpha)$。

d） 初始化理想点 $\vec{z}^*=(z_1,...,z_m)$。

e） 对于所有子问题 sp_i， $i=1,...,N$，计算它们的邻居索引 $B(i)=\{i_1,...,i_T\}$。

f） 计算当前种群可行比例 r。

步骤 2：更新种群

对于每一个子问题 sp_i， $i=1,...,N$，执行如下操作

a） 产生一个 $[0,1]$ 之间的随机数 rp。

b） 如果 $\delta<rp$，则更新索引 $S_l=B(i)$，否则更新索引 $S_l=\{1,...,N\}$。

c） 执行遗传算子产生子解 $\vec{y}_i$。

d） **更新解：**随机从 S_l 选出 n_r 个索引 $j_1,...,j_p,....,j_{n_r}\in S_l$，对于每一个选出的索引对应的解 $\vec{x}_{j_p}$， $j_p\in(j_1,...,j_{n_r})$，如果 $\vec{y}_i\prec_\theta\vec{x}_{j_p}$，则用 $\vec{y}_i$ 替代种群中的 $\vec{x}_{j_p}$。

步骤 3：更新阈值参数

更新种群可行比例 r，根据公式(5-4)更新 θ。

步骤 4：终止

如果终止条件满足，输出 P 中的最优解 $\vec{x}_{best}$；否则重新回到**步骤 2**。

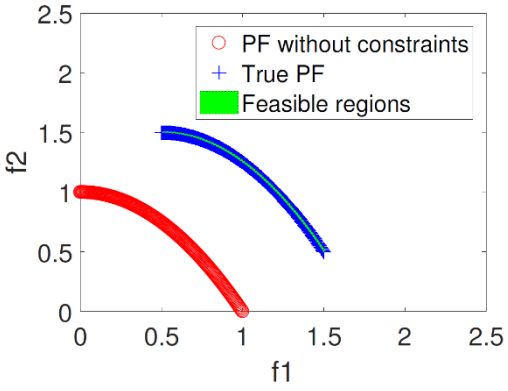
5.2 多目标约束优化测试问题集

为了测试约束多目标进化算法的性能和效率，我们可以用测试问题集对算法进行测试。这类测试问题一般会提供已知的 PF 信息，结合性能评估指标可

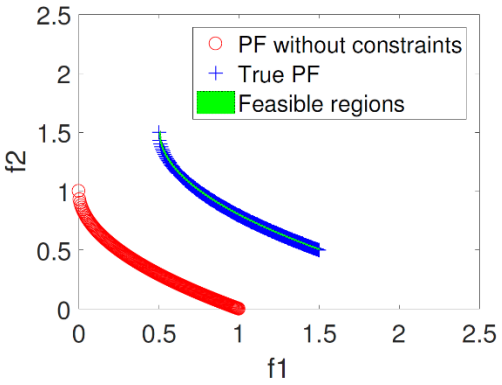
以评定算法的性能，方便算法的调试与比较。可以说选择好一个合适的测试问题，对于研究约束多目标进化算法非常重要。

目前最常用的多目标约束优化测试问题集主要有两类：CTP 测试问题[88]和 CF 测试问题[89]。这两类测试的共同特点是它们在目标空间里都有较大的可行区域。然而当种群落在可行区域中时，约束处理机制是不起作用的，所以这两类测试问题实际上并不十分适合用于测试约束处理机制的性能。

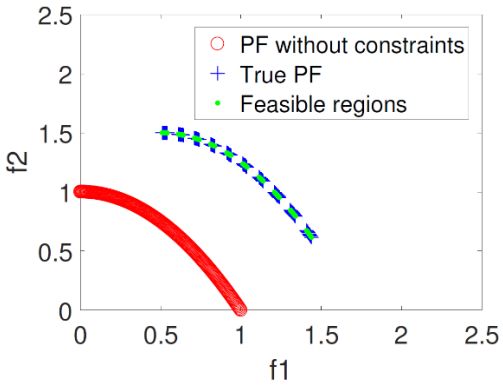
本文中使用一类测试问题集 LIR-CMOP[90][91]对算法性能进行测试，其具体的目标函数及约束条件附于附录页附表 II 中。这类测试问题的一般特性是它们都带有大范围的不可行区域，搜索空间被众多的不可行区域分隔开，使得普通的约束处理机制很难快速找到实际的 PF。这类测试问题目标函数及约束条件是由多组可控的形状函数和距离函数[92]组成。具体而言，形状函数是用来调整 PF 形状为凸的或凹的，而距离函数用来调整问题对于算法的收敛难度。下图分别画出 LIR-CMOP 第 1 到第 14 个测试问题在目标空间上的可行区域及不可行区域的分布情况。



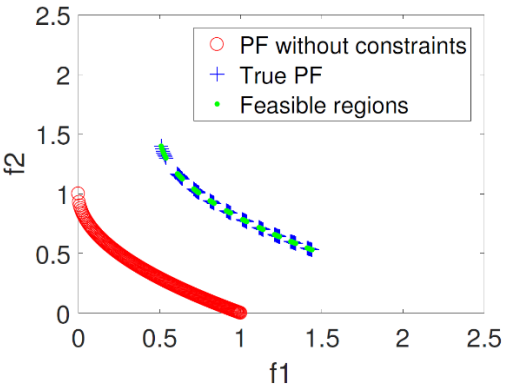
a）LIR-CMOP1



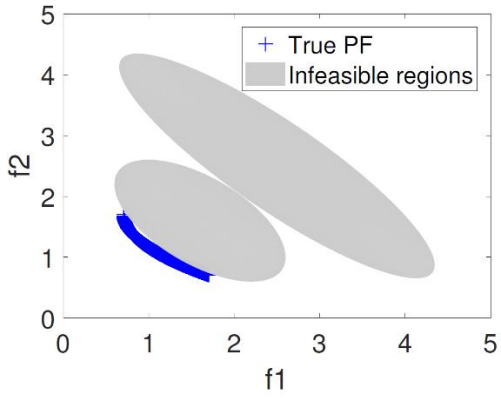
b）LIR-CMOP2



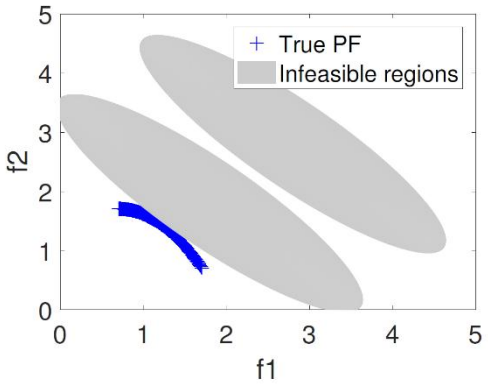
c）LIR-CMOP3



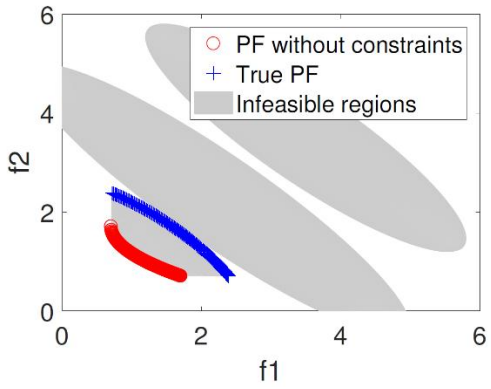
d）LIR-CMOP4



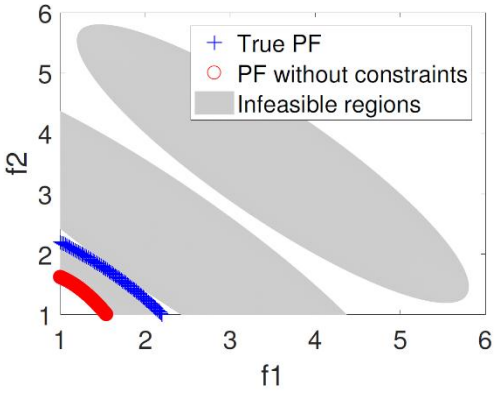
e）LIR-CMOP5



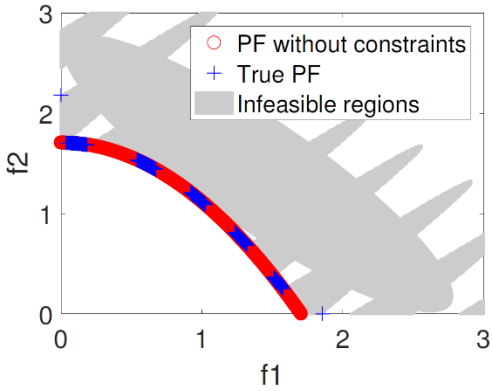
f）LIR-CMOP6



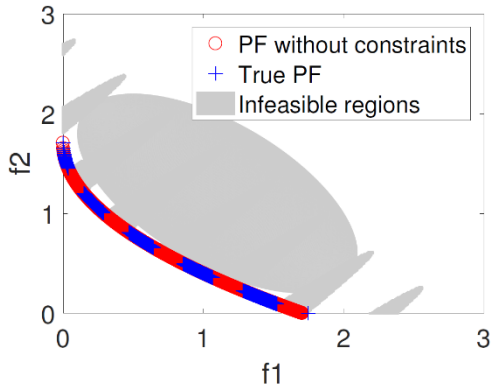
h）LIR-CMOP7



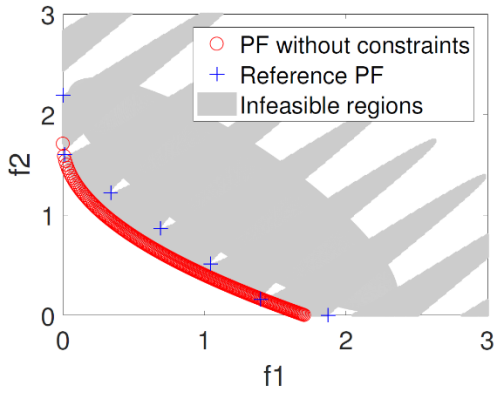
i）LIR-CMOP8



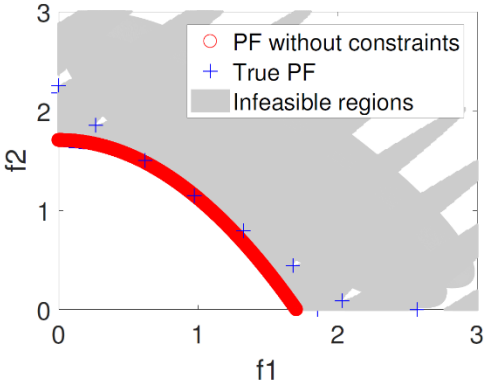
j）LIR-CMOP9



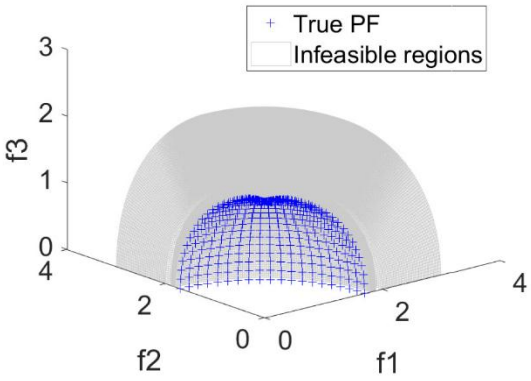
k）LIR-CMOP10



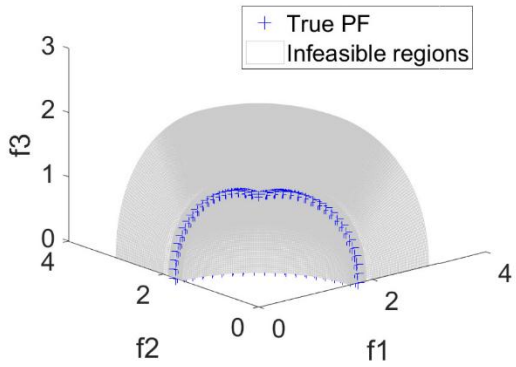
l）LIR-CMOP11



m）LIR-CMOP12



o）LIR-CMOP1



p）LIR-CMOP2

图 5-5　测试问题 LIR-CMOP1-14 在目标空间上可行区域及不可行区域的分布情况

如图 5-5 所示，LIR-CMOP1-14 在目标空间中都有着大范围的不可行区域，实际 PF 就被隐藏在这些不可行区域中，这对于测试约束进化算法的性能更优实际的意义。

5.3 工程优化问题：I 型焊接梁优化问题

为了研究 MOEA/D 解决现实优化问题的性能，本小节介绍一个工程优化问题：I 型焊接梁（I-beam）优化问题[44]。这是一个双目标约束优化问题，在这个问题中有两个需要同时最小化的相互冲突的目标：

1）焊接梁横断面面积；

2）施力 P 使焊接梁产生位移的静挠度。

I 型焊接梁的几何示意图如图 5-6 所示，其需要优化的决策变量向量为 $\vec{x}=[x_1,x_2,x_3,x_4]^T$，这些变量都以厘米为单位，每个变量的决策范围如下所示：

$10 \le \vec{x}_1 \le 80$，$10 \le \vec{x}_2 \le 50$，$0.9 \le \vec{x}_3 \le 5$，$0.9 \le \vec{x}_4 \le 5$。

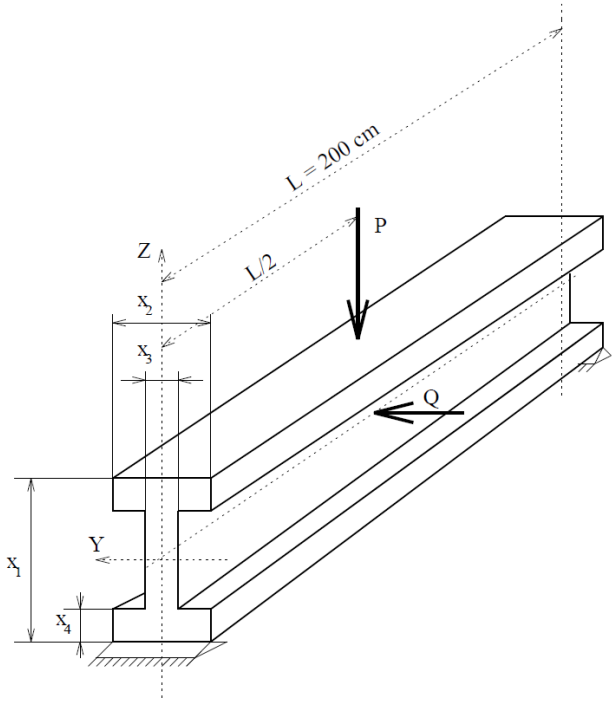


图 5-6　I 型焊接梁的几何示意图

一些已知的给定参数如下所示：

1）焊接梁材料允许弯曲压力：$k_g = 1.6kN/cm^2$；

2）杨氏弹性模量：$E = 2 \times 10^4 kN/cm^2$；

3）最大弯曲力：$P = 600kN$，$Q = 50kN$；

4）I 型焊接梁长度：$l = 200cm$。

I 型焊接梁优化问题具体数学定义如下：

$$\begin{cases} \text{minimize} \quad f_1(\vec{x}) = 2x_2x_4 + x_3(x_1 - 2x_4) \\ \text{minimize} \quad f_2(\vec{x}) = \dfrac{Pl^3}{48EI} \\ \text{subject to} \quad g(\vec{x}) = k_g - \dfrac{M_y}{W_y} - \dfrac{M_z}{W_z} > 0 \end{cases} \tag{5-5}$$

其中 $M_y = 30000kN \cdot cm$ 而 $M_z = 2500kN \cdot cm$。

I 为惯性系数，具体可由下式求得：

$$I = \frac{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]}{12} \tag{5-6}$$

断面系数W_y和W_z可由公式(5-7)及(5-8)求得

$$W_y = \frac{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]}{6x_1} \tag{5-7}$$

$$W_z = \frac{(x_1 - 2x_4)x_3^3 + 2x_4x_2^3}{6x_2} \tag{5-8}$$

为了方便研究 I 型焊接梁优化问题在目标空间中解的分布情况。我们在目标空间中随机采样 850000 个解，并加上在 MOEA/D-ACDP 进化过程中产生的 150000 个解，利用这 1000000 个解绘画出了这个问题在目标空间中解的大致分布情况。具体分布情况如图 5-7 所示，其中红点表示可行解，蓝点表示不可行解，可见这个约束优化问题的可行区域非常狭长且小，需要应用合适的约束处理机制才能求解此类问题。

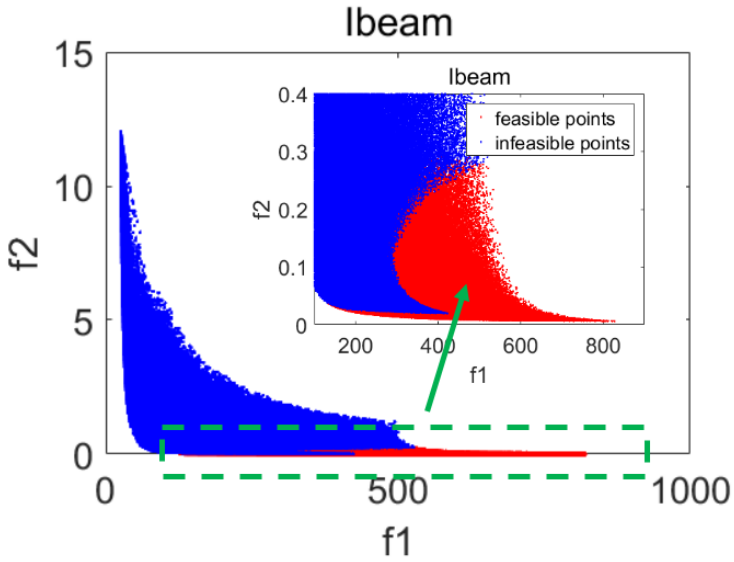


图 5-7　I 型焊接梁优化问题目标空间分布情况

5.4 实验研究

5.4.1 几种经典的基于分解的约束多目标进化算法

为了评估本文提出的 MOEA/D-ACDP 算法的性能，这里我们提供了 4 类基

于分解的约束多目标进化算法对 LIR-CMOP 测试问题集及 I 型焊接梁优化问题进行测试比较，这 4 类算法分别是 C-MOEA/D[83]，MOEA/D-CDP[84]，MOEA/D-Epsilon[85]及 MOEA/D-SR[84]。

1) C-MOEA/D

C-MOEA/D 使用了一种新型的ϵ约束处理机制。算法中ϵ的是当前种群的约束违反值和当前可行解比例来调整。当两个解进行比较时，如果它们的约束违反值都小于等于ϵ，则根据它们的分解函数进行比较，否则根据它们的约束违反值进行比较。

2) MOEA/D-CDP

MOEA/D-CDP 基于 MOEA/D 的框架，在种群更新的过程中，使用 CDP 规则[34]来判断任意两个比较的解的支配关系。在解决一些简单的约束优化问题时，有着不错的性能。

3) MOEA/D-Epsilon

MOEA/D-Epsilon 使用原始的ϵ约束处理机制，ϵ随着进化代数的增加而动态减小，其具体的设置方式参考文献[36]所示。

4) MOEA/D-SR

MOEA/D-SR 在 MOEA/D 框架里嵌入了 SR 约束处理机制[35]。算法中设置了一个$[0,1]$之间的阈值r_f。在种群更新过程中比较两个解时，如果一个 0-1 之间随机数小于r_f，则根据它们的分解函数进行比较，否则根据它们的约束违反值进行比较。当$r_f = 0$时，MOEA/D-SR 等同于 MOEA/D-CDP。

5.4.2 实验参数设置

为了保证实验比较的公平性，MOEA/D-ACDP 及四种比较的约束多目标进化算法都采用了 DE 遗传算子，而其他可以统一的参数都会相同设置，具体参数设置如下所示：

1) 变异概率$P_m = 1/D$（D为决策变量的数目），分布指数$di = 20$。DE 算子中$CR = 1.0$，$F = 0.5$

2) 种群规模$N = 300$，邻居规模$T = 30$，邻居索引选择概率$\delta = 0.9$，最大邻居

更新次数 $n_r = 2$。

3) 终止条件：每种算法独立运行 30 次，每次运行达到 150000 次函数评估次数。

4) MOEA/D-ACDP 参数设置：$\alpha = 0.8$，$\theta_0 = \pi / 2N$。

5) MOEA/D-Epsilon 参数设置：$T_c = 400$，$cp = 2$，$\theta = 0.05N$。

6) MOEA/D-Epsilon 参数设置：$r_f = 0.01$。

对于 LIR-CMOP 测试问题集，使用 IGD 及 HV 作为性能评估指标，对于 I 性焊接梁优化问题，由于实际的 PF 未可得知，所以不适合使用 IGD 指标，只使用 HV 指标进行比较。

5.5 实验分析及结论

5.5.1 LIR-CMOP 测试问题实验结果

5 种约束多目标进化算法 MOEA/D-ACDP、C-MOEA/D，MOEA/D-CDP，MOEA/D-Epsilon 及 MOEA/D-SR 对测试问题 LIR-CMOP1-14 优化得到的 IGD 及 HV 指标结果列于表 5-1 及表 5-2 中，其中‘mean’表示 30 次优化的平均指标，‘std’表示 30 次优化的指标标准差。加黑体的数值表示该算法的 IGD（或 HV）指标性能显著优于其它算法。

基于 IGD 指标进行比较，对于问题 LIR-CMOP3-14，本文提出的 MOEA/D-ACDP 都显著优于其他 4 种比较的算法。而对于 LIR-CMOP1-2，MOEA/D-ACDP 略差于 MOEA/D-SR，而显著优于其他三种算法。

基于 HV 进行比较，则对于问题 LIR-CMOP3-14，LIR-CMOP3-14 显著优于其他 4 种比较的算法。对于 LIR-CMOP1，MOEA/D-ACDP 略差于 MOEA/D-SR，而显著优于其他三种算法。对于 LIR-CMOP2，MOEA/D-ACDP 与 MOEA/D-SR 十分接近，而显著优于其他三种算法。

图 5-8 至图 5-11 给出了 5 种算法在 30 次独立运行优化中，针对问题 LIR-CMOP3、LIR-CMOP5、LIR-CMOP10 及 LIR-CMOP11，取得中位 IGD 值时的非支配解集获得情况。图中蓝点表示真实 PF 而红点表示这些非支配解。

图 5-8 中可以明显看出 MOEA/D-ACDP 对 LIR-CMOP3 获得的非支配解集几乎覆盖整个实际 PF，同时在五种算法中拥有最好的分布性。图 5-9 中，MOEA/D-ACDP 对 LIR-CMOP5 获得的非支配解集覆盖了整个实际 PF，而其他算法全都陷入了局部最优中。图 5-10 明显可以看出 MOEA/D-ACDP 对 LIR-CMOP5 获得的非支配解集相对于其他四种比较算法拥有更好的收敛性。图 5-11 中，MOEA/D-ACDP 可以获得 PF 上的大部分非支配解，而其他算法只能搜索到实际 PF 少部分非支配解。

综合 IGD 及 HV 指标结果及上述部分优化结果，在 LIR-CMOP 测试问题集上，MOEA/D-ACDP 显然要比其他四种经典的约束多目标优化算法好，而且值得一提的是，LIR-CMOP13 及 LIR-CMOP14 作为三目标约束优化问题，MOEA/D-ACDP 同样是五种算法中表现最好的算法。

如我们前面所提到，LIR-CMOP1-14 都有着大范围的不可行区域。上述的实验结果表明了 ACDP 机制能够通过利用种群角度信息，很好地处理约束多目标优化问题。

表 5-1　MOEA/D-ACDP 及其他四种比较的算法在测试问题 LIR-CMOP1-14 上的 IGD 结果

LIR-CMOP		MOEA/D-ACDP	C-MOEA/D	MOEA/D-CDP	MOEA/D-Epsilon	MOEA/D-SR
1	mean	5.159E-02	1.591E-01	1.348E-01	8.234E-02	**4.406E-02**
	std	1.815E-02	3.534E-02	5.996E-02	5.321E-02	3.360E-02
2	mean	2.269E-02	1.462E-01	1.549E-01	4.708E-02	**2.057E-02**
	std	9.418E-03	4.141E-02	2.966E-02	1.339E-02	1.072E-02
3	mean	**4.659E-02**	2.309E-01	2.268E-01	7.858E-02	1.529E-01
	std	1.850E-02	4.135E-02	4.403E-02	2.978E-02	7.688E-02
4	mean	**2.784E-02**	2.080E-01	2.188E-01	5.662E-02	2.038E-01
	std	1.477E-02	4.197E-02	3.766E-02	3.366E-02	7.907E-02
5	mean	**1.771E-02**	1.162E+00	1.207E+00	1.201E+00	1.123E+00
	std	2.965E-02	2.180E-01	1.660E-02	1.963E-02	2.842E-01
6	mean	**1.757E-01**	1.265E+00	1.303E+00	1.231E+00	1.175E+00
	std	4.129E-02	3.067E-01	2.319E-01	3.602E-01	3.967E-01
7	mean	**1.408E-01**	1.620E+00	1.623E+00	1.568E+00	1.136E+00
	std	4.385E-02	3.036E-01	2.905E-01	4.101E-01	7.315E-01
8	mean	**1.812E-01**	1.607E+00	1.631E+00	1.577E+00	1.369E+00
	std	4.854E-02	2.680E-01	2.464E-01	3.767E-01	5.735E-01
9	mean	**3.595E-01**	4.981E-01	4.868E-01	4.962E-01	4.813E-01
	std	5.345E-02	6.991E-02	5.372E-02	6.987E-02	4.571E-02

10	mean	**1.388E-01**	3.775E-01	3.774E-01	3.257E-01	2.821E-01
	std	1.148E-01	7.446E-02	6.858E-02	9.833E-02	1.135E-01
11	mean	**1.318E-01**	4.422E-01	4.662E-01	4.154E-01	3.489E-01
	std	4.487E-02	1.759E-01	1.439E-01	1.508E-01	1.129E-01
12	mean	**1.497E-01**	3.597E-01	3.236E-01	3.680E-01	3.012E-01
	std	9.985E-03	1.074E-01	1.023E-01	8.664E-02	8.989E-02
13	mean	**7.414E-02**	1.266E+00	1.289E+00	1.183E+00	1.093E+00
	std	2.727E-03	2.173E-01	6.321E-02	3.456E-01	4.269E-01
14	mean	**6.732E-02**	1.235E+00	1.103E+00	1.127E+00	1.143E+00
	std	1.918E-03	1.209E-01	3.857E-01	3.329E-01	3.002E-01

表 5-2 MOEA/D-ACDP 及其他四种比较的算法在测试问题 LIR-CMOP1-14 上的 HV 结果

LIR-CMOP		MOEA/D-ACDP	C-MOEA/D	MOEA/D-CDP	MOEA/D-Epsilon	MOEA/D-SR
1	mean	1.365E+00	9.499E-01	1.009E+00	1.353E+00	**1.376E+00**
	std	2.493E-02	7.038E-02	1.298E-01	4.417E-02	3.974E-02
2	mean	1.737E+01	1.395E+01	1.374E+01	1.705E+01	**1.736E+01**
	std	1.306E-02	8.154E-02	6.160E-02	1.693E-02	1.890E-02
3	mean	**1.188E+00**	7.558E-01	7.600E-01	1.184E+00	9.313E-01
	std	4.929E-02	5.730E-02	5.809E-02	2.898E-02	1.620E-01
4	mean	**1.421E+00**	1.069E+00	1.051E+00	1.390E+00	1.089E+00
	std	1.946E-02	6.952E-02	5.462E-02	4.405E-02	1.360E-01
5	mean	**1.903E+00**	1.192E-01	5.805E-02	5.829E-02	1.707E-01
	std	5.658E-02	3.352E-01	4.042E-04	2.022E-04	4.442E-01
6	mean	**1.280E+00**	7.863E-02	4.312E-02	1.325E-01	1.682E-01
	std	4.613E-02	3.011E-01	2.362E-01	4.251E-01	4.061E-01
7	mean	**3.408E+00**	2.990E-01	2.886E-01	4.055E-01	1.313E+00
	std	1.409E-01	6.927E-01	6.348E-01	8.879E-01	1.567E+00
8	mean	**3.330E+00**	3.246E-01	2.695E-01	3.859E-01	8.287E-01
	std	1.461E-01	5.878E-01	5.297E-01	8.166E-01	1.244E+00
9	mean	**4.080E+00**	3.715E+00	3.755E+00	3.724E+00	3.752E+00
	std	9.501E-02	2.079E-01	1.600E-01	2.033E-01	1.142E-01
10	mean	**3.755E+00**	3.274E+00	3.268E+00	3.385E+00	3.477E+00
	std	2.208E-01	1.623E-01	1.416E-01	2.122E-01	2.383E-01
11	mean	**5.004E+00**	3.937E+00	3.842E+00	4.038E+00	4.274E+00
	std	1.564E-01	6.479E-01	5.507E-01	5.727E-01	4.463E-01
12	mean	**6.713E+00**	5.977E+00	6.134E+00	6.010E+00	6.240E+00
	std	5.874E-02	3.855E-01	3.617E-01	3.074E-01	2.950E-01
13	mean	**7.897E+00**	6.444E-01	4.728E-01	1.092E+00	1.513E+00
	std	2.943E-02	1.317E+00	2.689E-01	2.052E+00	2.422E+00
14	mean	**8.641E+00**	7.766E-01	1.627E+00	1.430E+00	1.269E+00
	std	1.546E-02	6.140E-01	2.473E+00	2.095E+00	1.919E+00

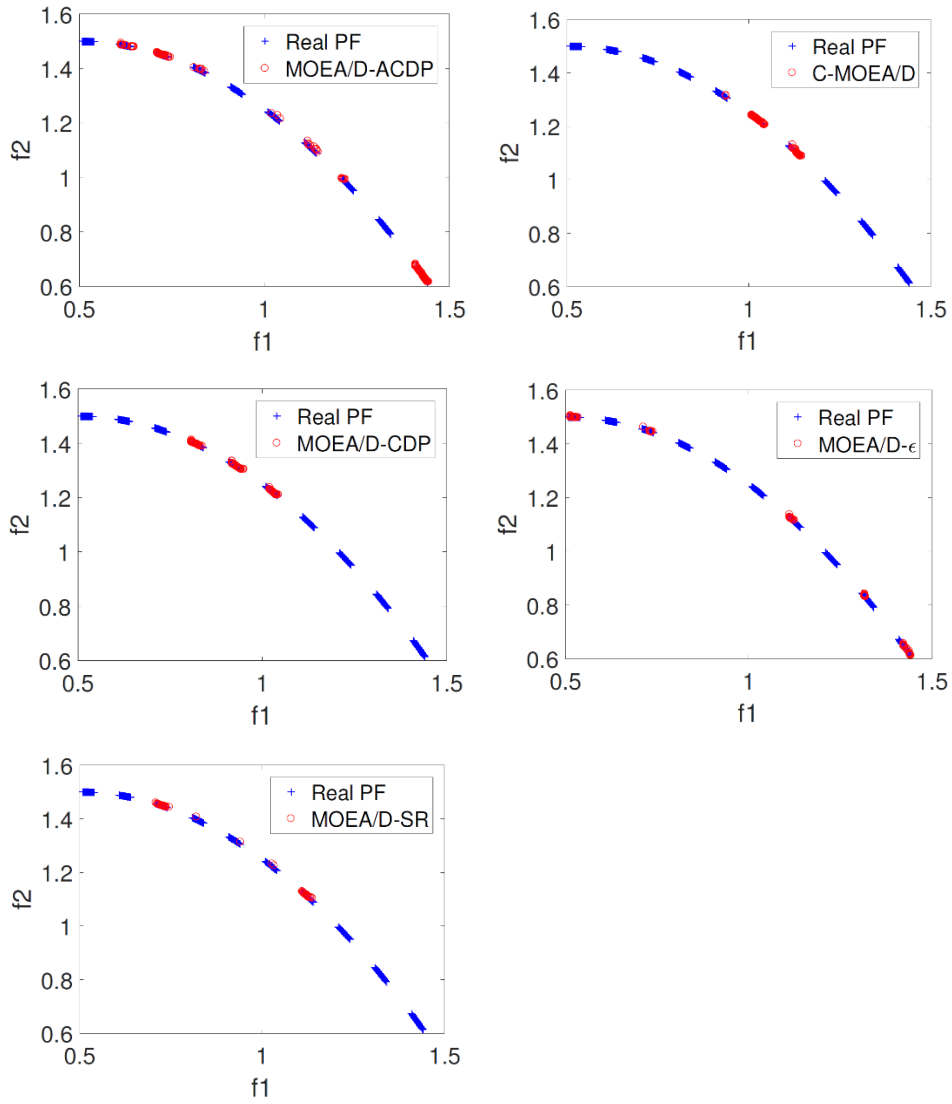
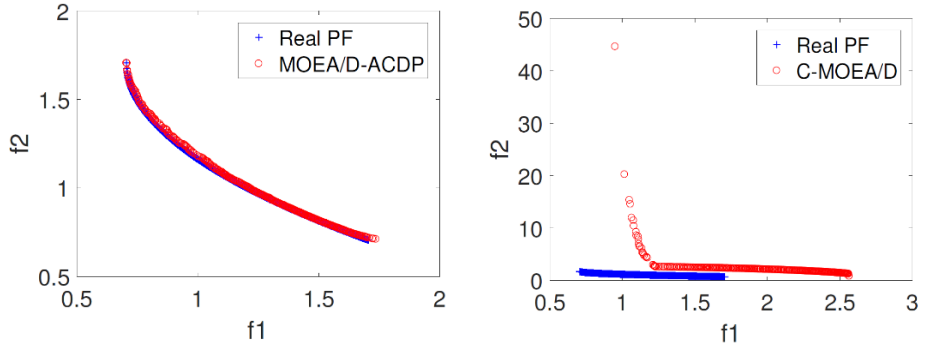


图 5-8　五种算法在 30 次独立运行中针对 LIR-CMOP3 取得中位 IGD 时的非支配集



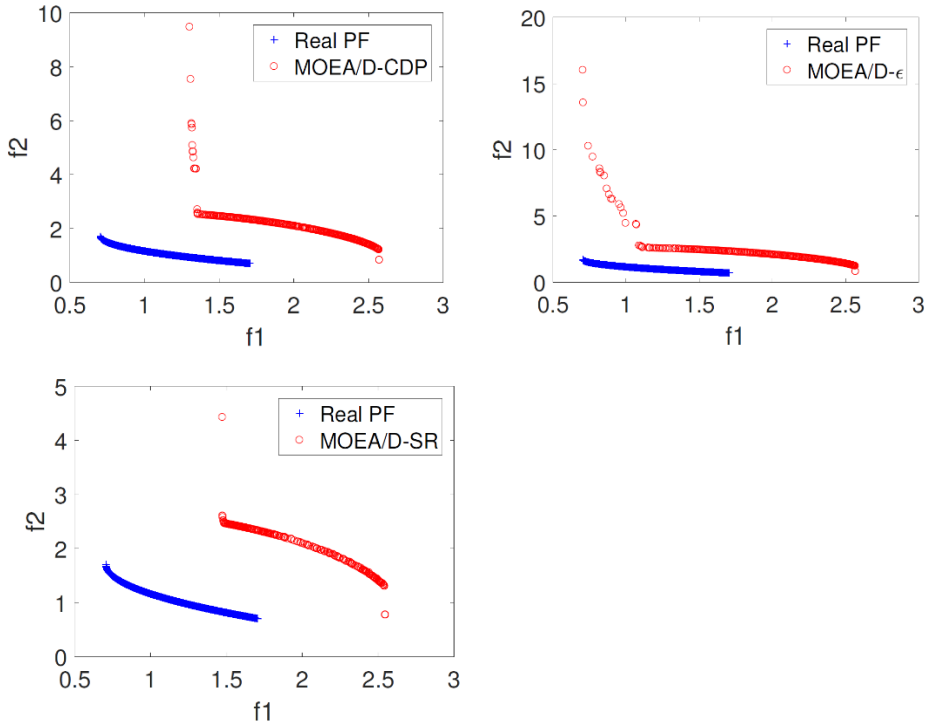
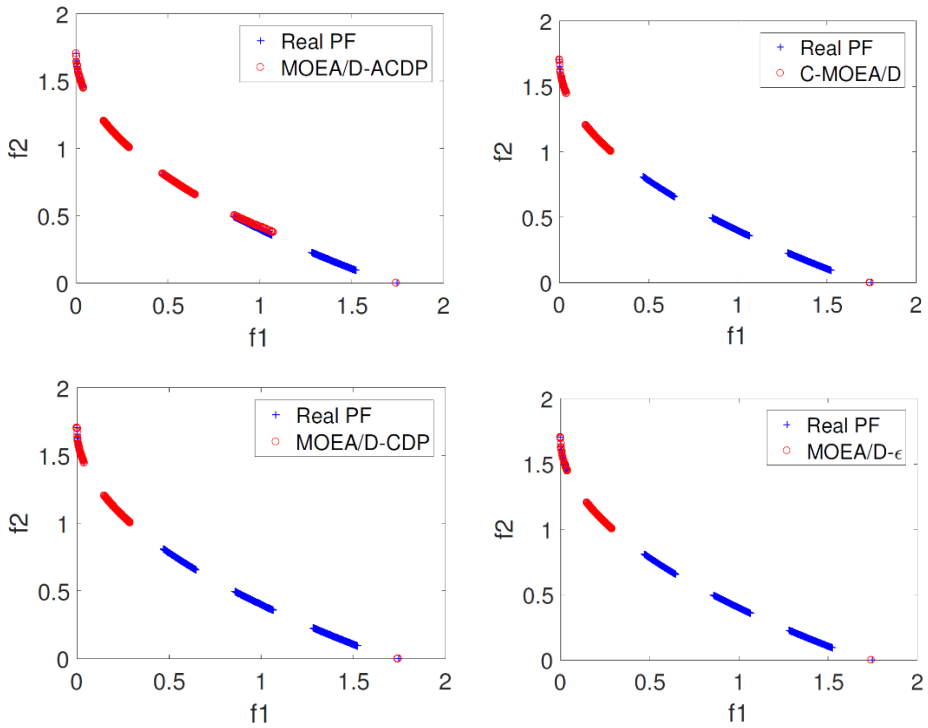


图 5-9 五种算法在 30 次独立运行中针对 LIR-CMOP5 取得中位 IGD 时的非支配解集



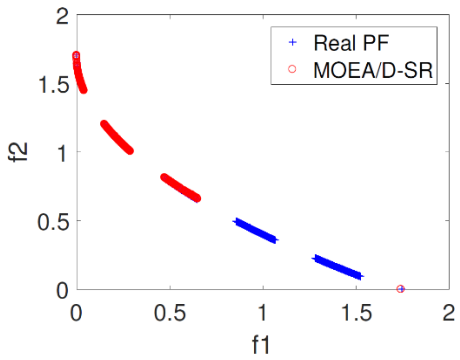


图 5-10　五种算法在 30 次独立运行中针对 LIR-CMOP10 取得中位 IGD 时的非支配解况集

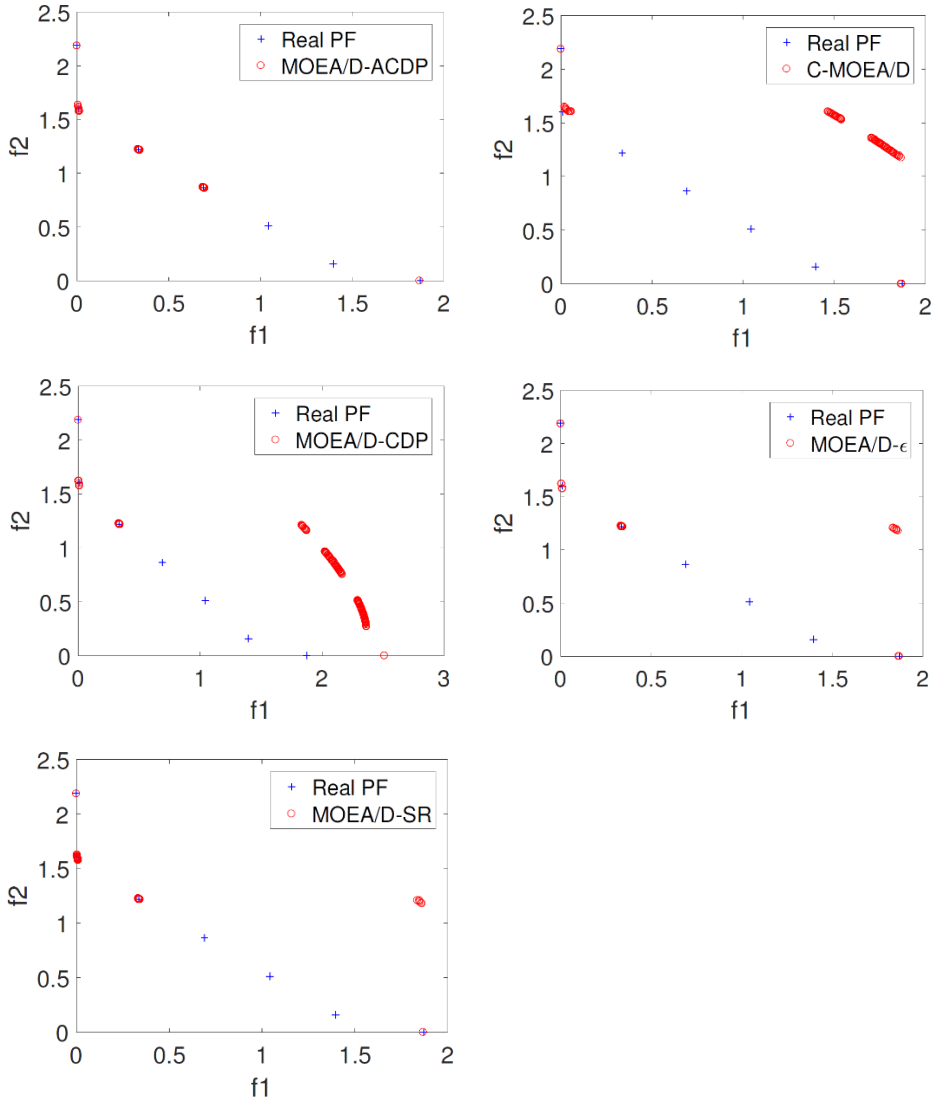


图 5-11　五种算法在 30 次独立运行中针对 LIR-CMO11 取得中位 IGD 时的非支配解集

5.5.2 I-型焊接臂优化问题实验结果

MOEA/D-ACDP 及其他四种比较的算法在 I 型焊接梁优化问题上的 HV 指标

结果列于表 5-3 中。其结果表明 MOEA/D-ACDP 显著优于其他四种约束多目标进化算法。

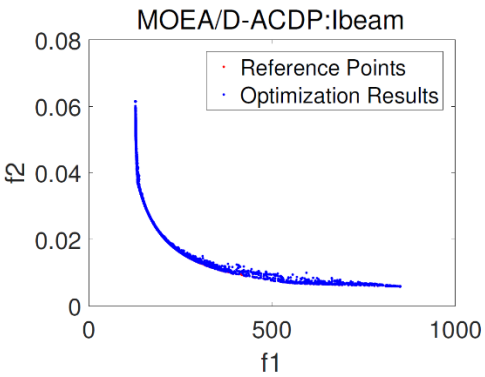
为了更进一步地显示 MOEA/D-ACDP 与其他算法相比的优越性，图 5-12 绘出五种算法 30 次独立优化后得到的所有解。图中这些解用蓝点表示，而红点是所有这些解的非支配解集，并作为这个优化问题的近似 PF，用参考点（Reference Points）表示。可以看出，MOEA/D-ACDP 得到的所有解具有最好的收敛性。

图 5-13 是五种算法在 I 型焊接梁优化问题上的 HV 指标箱形图，‘a’、‘b’、‘c’、‘d’、‘e’表示 MOEA/D-ACDP、C-MOEA/D，MOEA/D-CDP，MOEA/D-Epsilon 及 MOEA/D-SR。从箱形图也可以看出，MOEA/D-ACDP 在 HV 指标上明显优于其他算法。

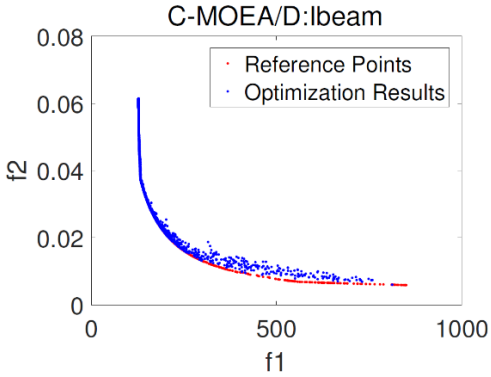
综合上述实验结果表明，MOEA/D-ACDP 在 I 型焊接梁优化问题上性能明显优于其他四种比较算法，这侧面表明了 MOEA/D-ACDP 同样是也一种十分具有竞争力的实用于现实工程应用优化问题的约束多目标进化算法。

表 5-3　MOEA/D-ACDP 及其他四种比较的算法在 I 型焊接梁优化问题上的 HV 结果

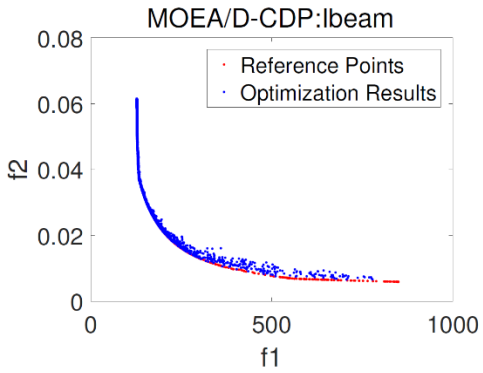
I 型焊接梁	MOEA/D-ACDP	C-MOEA/D	MOEA/D-CDP	MOEA/D-Epsilon	MOEA/D-SR
mean	**3.583E+01**	3.481E+01	3.518E+01	3.514E+03	3.477E+03
std	1.950E-01	2.968E-01	2.078E-01	2.034E-01	1.248E+00



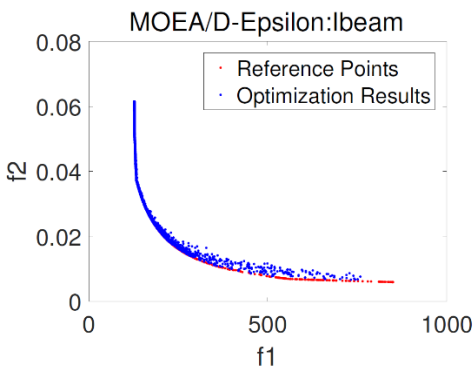
（a）MOEA/D-ACDP



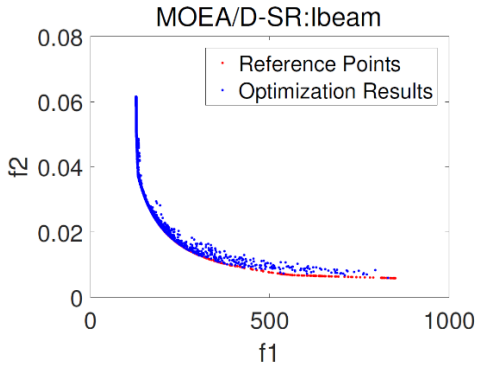
（b）C-MOEA/D



（c）MOEA/D-CDP



（d）MOEA/D-Epsilon



（e）MOEA/D-SR

图 5-12　五种算法 30 次独立运行后得到的所有 I 型焊接梁优化问题非支配解

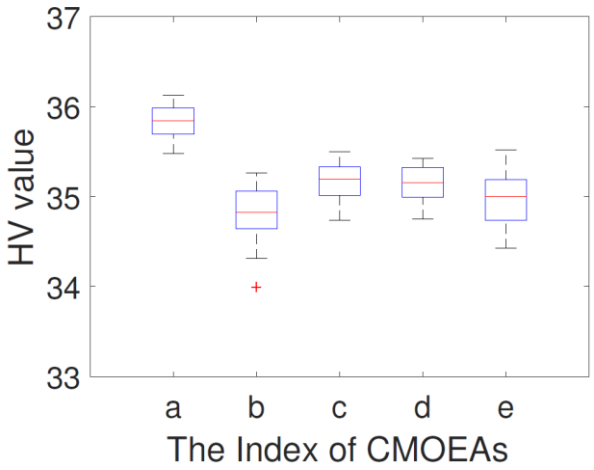


图 5-13　五种算法在 I 型焊接梁优化问题上的 HV 指标箱形图

5.6 本章小结

本章提出了一种名为 ACDP 的新型约束处理机制，它利用了种群的角度信息在进化过程中动态维持了种群多样性。同时 ACDP 被应用于多目标进化算法 MOEA/D 的框架中，构成了一种新的约束多目标进化算法 MOEA/D-ACDP，并

与其他四种经典的基于分解的约束多目标进化算法进行比较。一类拥有大范围不可行区域的测试问题集 LIR-CMOP 被用于测试算法性能。MOEA/D-ACDP 在测试中能使种群保持更好收敛性及分布性，并更加接近于实际的 PF，拥有最好的 IGD 及 HV 性能指标。同时在解决现实工程应用问题：I 行焊接梁优化问题时，MOEA/D-ACDP 依然表现出最好的性能。这表明了 MOEA/D-ACDP 比其他四种比较的经典算法都要明显优异，而且 ACDP 是一种非常有效的约束处理机制。总体而言，ACDP 有以下两点优点：

1）ACDP 利用了种群角度信息有效地维持了多目标优化问题中种群的多样性。

2）ACDP 通过平衡种群在可行区域及不可行区域的探索，从而避免浪费非支配解中的有用信息，增强了种群的收敛性。

总结与展望

总结

约束优化问题在现实生活及工程应用中极为常见，然而有时候对它们求解起来却非常困难。作为一项非常有意义的研究工作，约束优化方法一直以来都受到国内外专家学者的广泛关注。对于求解约束优化问题，传统经典的优化方法往往对问题类型有着严苛的要求，同时它们通常不具有较强的鲁棒性，并且很容易陷入局部最优。经过学界长时间的研究及发展，诞生了一类以进化算法为代表的启发式随机搜索算法，能在有限的计算资源内保持较强的全局搜索能力和鲁棒性，为约束优化方法的研究带来了新的机遇。常见的约束优化问题包括单目标约束优化问题和多目标约束优化问题，所以针对求解这两类问题的约束进化算法研究成为了学界经久不衰的研究热点。基于这样的背景，本文主要完成了以下两项工作：

第一，提出了一种解决单目标约束优化问题的进化算法 LSHADE44-IEpsilon，并将其应用于 CEC2017 测试问题集及两个经典的实际工程应用优化问题中。DE 算法是一种非常经典的可以解决单目标优化问题的进化算法，由于算法的简易性及较强的全局搜索能力，长期以来 DE 算法广受热捧。其中以 DE 算法为基础发展起来的单目标约束进化算法 LSHADE44，可以根据进化过程中的信息自适应调节算法参数，极大地增强了其解决不同类型问题的鲁棒性。优异的特性使它夺得了 2017 年 CEC 会议单目标约束优化竞赛的冠军，然而 LSHADE44 算法框架里最核心的约束处理机制只是传统的 CDP 规则，这使得种群进化的过程中丢失了不少不可行解的有用信息，从而可能导致种群在进化过程中陷入局部最优。所以本文在 LSHADE44 算法框架上应用了一种改进的ϵ约束处理机制 IEpsilon，这种机制既能有效地利用不可行解的有用信息，增加种群的搜索空间从而挣脱局部最优，同时又能根据种群当前可行比例自适应调节ϵ取值，动态调整种群进化过程中的约束松弛尺度。结合了 IEpsilon 的新算法 LSHADE44-IEpsilon 在求解 CEC2017 测试问题集和两个工程优化问题时，都明显优于 LSHADE44 及结合原始ϵ约束处理机制的 LSHADE44-Epsilon。这不仅说明了 LSHADE44-IEpsilon 是一种优异的约束进化算法，同时也说明了 IEpsilon 是一种极具竞争力的单目标进

化算法约束处理机制。

第二，提出了一种解决多目标约束优化问题的进化算法 MOEA/D-ACDP，并将其应用于约束多目标测试问题集及一项实际工程应用中。相比求解单目标约束优化问题，求解多目标约束优化问题是一件更加困难的任务。求解多目标优化问题时，由于目标可能存在冲突性，所以最终求得的最优解不止一个，而是一组 Pareto 解集。所以在求解约束多目标优化问题时，既要充分考虑种群的收敛，也要考虑种群的多样性。多目标进化算法 MOEA/D 基于分解的思想，在求解无约束多目标优化问题时，能很好地保持种群收敛性及多样性的平衡。本文基于 MOEA/D，提出一种基于角度的约束支配规则 ACDP，从而提出了一种新型多目标约束进化算法 MOEA/D-ACDP。在求解 LIR-CMOP 测试问题集和 I 型焊接梁优化问题时，MOEA/D-ACDP 都明显优于其他四种基于 MOEA/D 的约束多目标进化算法。这不仅说明了 MOEA/D-ACDP 是一种优异的约束多目标进化算法，同时也说明了 ACDP 在求解约束多目标优化问题时同样具有很强的效用。

论文的创新点在于：（1）针对单（及多）目标优化问题时，文中提出了两种新型约束处理机制，它们都能充分挖掘种群进化过程中的有用信息，从而自适应地调节机制的规则及参数，增强了约束处理机制在解决不同类型问题的鲁棒性；（2）结合了提出的约束处理机制构成了新的约束进化算法，并且在实验中也取得了不错的效果，为解决约束优化问题提供了新的算法工具。

展望

虽然本文在对约束进化算法的研究上取得了一定的成果，但是在后续的研究中还有很多问题需要解决，主要有以下几方面：

第一，本文提出了单目标及多目标优化的约束进化算法研究，但在高维目标约束优化方面的研究，仍然是一件非常具有挑战性的工作，需要进一步地跟进。

第二，本文提出了两种自适应的约束处理机制，但是对进化过程中的种群信息挖掘的方法还是比较简易粗浅，个人认为可以进一步应用上机器学习等一类工具结合上进化算法对种群信息进行更充分的挖掘。

第三，本文尚未考虑到对算法机制的复杂度进行优化，未来可以进行改善。

参考文献

[1] Darwin C. On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life[J]. London, 1859, 1: 859.

[2] Floudas C A, Pardalos P M. A collection of test problems for constrained global optimization algorithms[M]. Springer Science & Business Media, 1990.

[3] 李智勇, 黄滔, 陈少淼等. 约束优化进化算法综述[J]. Journal of Software, 2017, 28(6).

[4] 张纪元. 梯度下降法[C]// 中国数学会全国最优化数值方法学术会. 1987.

[5] 李庆扬, 王能超, 易大义. 数值分析[M]. 北京: 清华大学出版社, 2008.

[6] Glover F. Tabu Search: A Tutorial[J]. Interfaces, 1990, 20(4):74-94.

[7] Kirkpatrick S, Vecchi M P. Optimization by simulated annealing[M]// Readings in computer vision: issues, problems, principles, and paradigms. Morgan Kaufmann Publishers Inc. 1987:339-348.

[8] Rechenberg E. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 1973[J]. Frommann-Holzboog Verlag, Stuttgart.

[9] Fogel L J, Owens A J, Walsh M J. Artificial intelligence through simulated evolution[J]. 1966.

[10] Holland J H. Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence[J]. Quarterly Review of Biology, 1975, 6(2):126–137.

[11] Koza J R. Genetic programming: on the programming of computers by means of natural selection[M]. MIT Press, 1992.

[12] Yu X, Gen M. Introduction to evolutionary algorithms [M]. Springer Science & Business Media, 2010.

[13] Dorigo M. Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale[J]. Unpublished doctoral dissertation, Politecnico di Milano, Italy, 1992.

[14] Kennedy J, Eberhart R. Particle swarm optimization[J]. Proc. of 1995 IEEE Int. Conf. Neural Networks, (Perth, Australia), Nov. 27-Dec. 2011, 4(8):1942-1948 vol.4.

[15] Storn R, Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces[J]. Journal of Global Optimization, 1997, 11(4):341-359.

[16] Moscato P. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms[J]. Caltech Concurrent Computation Program, 1989.

[17] Castro L R D, Timmis J. Artificial Immune Systems: A New Computational Intelligence Paradigm[M]. Springer-Verlag New York, Inc. 2002.

[18] Coello C A C. Evolutionary multi-objective optimization: a historical view of the field[J]. IEEE computational intelligence magazine, 2006, 1(1): 28-36.

[19] Van Veldhuizen D A, Lamont G B. Multiobjective evolutionary algorithm research: A history and analysis[R]. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[20] Ishibuchi H, Tsukamoto N, Nojima Y. Evolutionary many-objective optimization: A short review[C]//Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on. IEEE, 2008: 2419-2426.

[21] Chong E K P, Zak S H. An introduction to optimization[M]. John Wiley & Sons, 2013.

[22] Michalewicz Z. A survey of constraint handling techniques in evolutionary computation methods[J]. Evolutionary Programming, 1995, 4: 135-155.

[23] Jordehi A R. A review on constraint handling strategies in particle swarm optimisation[J]. Neural Computing and Applications, 2015, 26(6): 1265-1275.

[24] Homaifar A, Qi C X, Lai S H. Constrained optimization via genetic algorithms[J]. Simulation, 1994, 62(4): 242-253.

[25] Joines J A, Houck C R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's[C]//Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. IEEE, 1994: 579-584.

[26] Ben Hadj-Alouane A, Bean J C. A genetic algorithm for the multiple-choice integer program[J]. Operations research, 1997, 45(1): 92-101.

[27] Carlson S E, Shonkwiler R. Annealing a genetic algorithm over constraints[C]//Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on. IEEE, 1998, 4: 3931-3936.

[28] Coello C A C. Use of a self-adaptive penalty approach for engineering optimization problems[J]. Computers in Industry, 2000, 41(2): 113-127.

[29] Hoffmeister F, Sprave J. Problem-independent handling of constraints by use of metric penalty functions[J]. 1996.

[30] Michalewicz Z, Nazhiyath G. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints[C]//Evolutionary Computation, 1995., IEEE International Conference on. IEEE, 1995, 2: 647-651.

[31] Xiao J, Michalewicz Z, Zhang L, et al. Adaptive evolutionary planner/navigator for mobile robots[J]. IEEE transactions on evolutionary computation, 1997, 1(1): 18-28.

[32] Surry P D, Radcliffe N J, Boyd I D. A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA Method[C]//AISB Workshop on Evolutionary Computing. Springer, Berlin, Heidelberg, 1995: 166-180.

[33] Paredis J. Co-evolutionary constraint satisfaction[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 46-55.

[34] Deb K. An efficient constraint handling method for genetic algorithms[J]. Computer methods in applied mechanics and engineering, 2000, 186(2-4): 311-338.

[35] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on evolutionary computation, 2000, 4(3): 284-294.

[36] Takahama T, Sakai S, Iwane N. Solving nonlinear constrained optimization problems by the ε constrained differential evolution[C]//Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on. IEEE, 2006, 3: 2322-2327.

[37] Zheng J, Wu Q, Song W. An improved particle swarm algorithm for solving nonlinear constrained optimization problems[C]//Natural Computation, 2007. ICNC 2007. Third International Conference on. IEEE, 2007, 4: 112-117.

[38] Wolpert D H, Macready W G. No free lunch theorems for optimization[J]. IEEE transactions on evolutionary computation, 1997, 1(1): 67-82.

[39] Poláková R. L-SHADE with competing strategies applied to constrained optimization[C]//Evolutionary Computation (CEC), 2017 IEEE Congress on. IEEE, 2017: 1683-1689.

[40] Tanabe R, Fukunaga A S. Improving the search performance of SHADE using linear population size reduction[C]//Evolutionary Computation (CEC), 2014 IEEE Congress on. IEEE, 2014: 1658-1665.

[41] Wu G, Mallipeddi R, Suganthan P N. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization[J]. National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report, 2016.

[42] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE transactions on evolutionary computation, 2002, 6(2): 182-197.

[43] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on evolutionary computation, 2007, 11(6): 712-731.

[44] Osyczka A. Multicriteria optimization for engineering design[M]//Design optimization. 1985: 193-227.

[45] Das S, Suganthan P N. Differential evolution: A survey of the state-of-the-art[J]. IEEE transactions on evolutionary computation, 2011, 15(1): 4-31.

[46] Price K, Storn R M, Lampinen J A. Differential evolution: a practical approach to global optimization[M]. Springer Science & Business Media, 2006.

[47] Zhang J, Sanderson A C. JADE: adaptive differential evolution with optional external archive[J]. IEEE Transactions on evolutionary computation, 2009, 13(5): 945-958.

[48] Tvrdık J. Competitive differential evolution[C]//MENDEL. 2006: 7-12.

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

[50] Liao T, Molina D, de Oca M A M, et al. A note on bound constraints handling for the IEEE CEC'05 benchmark function suite[J]. Evolutionary computation, 2014, 22(2): 351-359.

[51] Gämperle R, Müller S D, Koumoutsakos P. A parameter study for differential evolution[J]. Advances in intelligent systems, fuzzy systems, evolutionary computation, 2002, 10(10): 293-298.

[52] Ronkkonen J, Kukkonen S, Price K V. Real-parameter optimization with differential evolution[C]//Evolutionary Computation, 2005. The 2005 IEEE Congress on. IEEE, 2005, 1: 506-513.

[53] Wang Y, Cai Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 55-66.

[54] Mezura-Montes E, Velázquez-Reyes J, Coello C A C. Modified differential evolution for constrained optimization[C]//Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. IEEE, 2006: 25-32.

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

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

[57] Brest J, Greiner S, Boskovic B, et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems[J]. IEEE transactions on evolutionary computation, 2006, 10(6): 646-657.

[58] Tanabe R, Fukunaga A. Success-history based parameter adaptation for differential evolution[C]//Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 2013: 71-78.

[59] Tanabe R, Fukunaga A S. Improving the search performance of SHADE using linear population size reduction[C]//Evolutionary Computation (CEC), 2014 IEEE Congress on. IEEE, 2014: 1658-1665.

[60] Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation[C]//Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE, 2010: 1-9.

[61] Gandomi A H, Yang X S, Alavi A H. Mixed variable structural optimization using Firefly Algorithm[J]. Computers & Structures, 2011, 89(23):2325-2336.

[62] Gu L, Yang R J, Tho C H, et al. Optimisation and robustness for crashworthiness of side impact[J]. International Journal of Vehicle Design, 2001, 26(4):348-360(13).

[63] Deb K, Gupta S, Daum D, et al. Reliability-based optimization using evolutionary algorithms[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 1054-1074.

[64] Rosenberg R. Simulation of genetic populations with biochemical properties [M]. 1967.

[65] Schaffer J D. Multiple objective optimization with vector evaluated genetic algorithms [C]// Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985. 1985: 93-100.

[66] Goldberg D E. Genetic algorithms in search, optimization, and machine learning [J]. Boston: Addison-Wesley, 1989.

[67] Fonseca C M, Fleming P J. Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization [C]// ICGA. 1993, 93: 416-423.

[68] Srinivas N, Deb K. Muiltiobjective optimization using nondominated sorting in genetic algorithms [J]. Evolutionary computation, 1994, 2(3): 221-248.

[69] Horn J, Nafpliotis N, Goldberg D E. A niched Pareto genetic algorithm for multiobjective optimization [C]// Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. 1994: 82-87.

[70] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach [J]. Evolutionary computation, IEEE transactions on, 1999, 3(4): 257-271.

[71] Corne D W, Knowles J D, Oates M J. The Pareto envelope-based selection algorithm for multiobjective optimization [C]// Parallel Problem Solving from Nature PPSN VI. Springer Berlin Heidelberg, 2000: 839-848.

[72] Knowles J D, Corne D W. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy[M]. MIT Press, 2000.

[73] Zitzler E, Laumanns M. SPEA2: Improving the strength Pareto evolutionary algorithm [J]. 2001.

[74] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. Evolutionary Computation, IEEE Transactions on, 2002, 6(2): 182-197.

[75] Erickson M, Mayer A, Horn J. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems [C]// Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, 2001: 681-695.

[76] Zitzler E, Künzli S. Indicator-based selection in multiobjective search[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 2004: 832-842.

[77] Sierra M R, Coello C A C. Improving PSO-based multi-objective optimization using crowding, mutation and ϵ -dominance [C]// Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, 2005: 505-519.

[78] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition [J]. Evolutionary Computation, IEEE Transactions on, 2007, 11(6): 712-731.

[79] Li H, Zhang Q. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II[J]. IEEE Transactions on evolutionary computation, 2009, 13(2): 284-302.

[80] Liu H L, Gu F, Zhang Q. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 450-455.

[81] Cai X, Li Y, Fan Z, et al. An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(4): 508-523.

[82] Cai X, Yang Z, Fan Z, et al. Decomposition-based-sorting and angle-based-selection for evolutionary multiobjective and many-objective optimization[J]. IEEE transactions on cybernetics, 2017, 47(9): 2824-2837.

[83] Asafuddoula M, Ray T, Sarker R, et al. An adaptive constraint handling approach embedded MOEA/D[C]//Evolutionary Computation (CEC), 2012 IEEE Congress on. IEEE, 2012: 1-8.

[84] Jan M A, Khanum R A. A study of two penalty-parameterless constraint handling techniques in the framework of MOEA/D[J]. Applied Soft Computing, 2013, 13(1): 128-148.

[85] Yang Z, Cai X, Fan Z. Epsilon constrained method for constrained multiobjective optimization problems: some preliminary results[C]//Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM, 2014: 1181-1186.

[86] Cheng R, Jin Y, Olhofer M, et al. A reference vector guided evolutionary algorithm for many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2016, 20(5): 773-791.

[87] Xiang Y, Zhou Y, Li M, et al. A vector angle-based evolutionary algorithm for unconstrained many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(1): 131-152.

[88] Deb K, Pratap A, Meyarivan T. Constrained test problems for multi-objective evolutionary optimization[C]//Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, 2001: 284-298.

[89] Zhang Q, Zhou A, Zhao S, et al. Multiobjective optimization test instances for the CEC 2009 special session and competition[J]. University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report, 2008, 264.

[90] Fan Z, Li W, Cai X, et al. Difficulty Adjustable and Scalable Constrained Multi-objective Test Problem Toolkit[J]. arXiv preprint arXiv:1612.07603, 2016.

[91] Fan Z, Li W, Cai X, et al. An improved epsilon constraint-handling method in MOEA/D for cmops with large infeasible regions[J]. arXiv preprint arXiv:1707.08767, 2017.

[92] Huband S, Hingston P, Barone L, et al. A review of multiobjective test problems and a scalable test problem toolkit[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(5): 477-506.

致 谢

时光穿梭，转眼间已是我在汕头大学待过的第七个年头了。此时的校园正值万物欣然，金凤花开的时刻，正预示着我在这里三年的研究生生涯即将走向结束。回望过往，我欣喜于我在汕头大学得到了许多学问、见识及成长，而这些都得益于我的母校、我的导师、我的同学还有其他所有给予过我帮助的人。我竭诚感恩，因为有了你们一直以来的支持、鼓励与陪伴，使我受益良多，我莫不敢忘。

首先我要感谢我的导师，尊敬的范衡教授。学术上您是一位学识渊博、治学严谨、视野开拓的学者，是您在学业上对我进行了悉心的指导，也是您引领着我走进神圣的科研殿堂。工作中您求真务实，踏实勤奋，您的作风深深地感染了我，您是我学习的榜样。生活中您不仅是一名良师，同时也是一位乐于关心倾听学生的益友，您常常以您丰富的人生经验给我带来许多宝贵的意见和帮助，使我受益匪浅。没有您无私的付出，也不会有我这三年里这么快速的成长。在此我由衷地感谢您，我的导师范衡老师。

其次我要感谢在汕头大学学习的这段时间里为我传道授业的崔岩老师、陈耀文老师、姜永权老师、陈力老师、李旭涛老师、闫敬文老师、庄哲民老师、唐雅娟老师、魏楚亮老师、蔡泽民老师、赖李洋老师、袁野老师和其他电子系的老师们，谢谢您们教授了我专业方面的知识，并给予了我许多生活上的关心和帮助。

然后我要感谢人工智能与机器人实验室的李文姬博士和朱贵杰博士，谢谢你们在科研上学术调研、论文撰写、专利申请等的指导，也谢谢你们在生活中给我带来了关心，你们始终都是值得我学习的兄长。感谢李中兴、陈燊、王宇鹏、李冲、姚利、王继彪、赖夏智、谢婷婷、赵善民同学，你们不仅在学习上给我带来了帮助，也是我生活中的挚友。我还要感谢游煜根、莫嘉杰、卢杰威、袁振国、邹婉欣、陈宏达、张绮婷、王宏志、刘义南、王诏君、袁宇彤等可爱的师弟师妹们，学习上我们不仅经常在一起进行讨论，生活中我们也成为了无话不谈的朋友。

最后，我要感谢我的家人。这么多年来我的父母含辛茹苦地抚育了我，我的姐姐鼓励并资助着我的学业。您们始终以我为骄傲，并用最无私的爱默默地关怀支持着我，让我能够毫无后顾之忧地完成我的硕士学业。所以在这里对于我的家人，我由衷地谢谢您们。

附 录

附表 I 测试问题 C01-C28 目标函数及其约束条件

编号	目标函数	约束条件
1	$\begin{cases} f(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}z_j)^2 \\ z=x-o \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}[z_i^{\,2}-5000\cos(0.1\pi z_i)-4000]\le 0 \\ x\in[-100,100]^D \end{cases}$
2	$\begin{cases} f(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}z_j)^2 \\ z=x-o, y=M\cdot z \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}[y_i^{\,2}-5000\cos(0.1\pi y_i)-4000]\le 0 \\ x\in[-100,100]^D \end{cases}$
3	$\begin{cases} f(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}z_j)^2 \\ z=x-o \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}[z_i^{\,2}-5000\cos(0.1\pi z_i)-4000]\le 0 \\ h(x)=-\sum_{i=1}^{D}z_i\sin(0.1\pi z_i)=0 \\ x\in[-100,100]^D \end{cases}$
4	$\begin{cases} f(x)=\sum_{i=1}^{D}[z_i^{\,2}-10\cos(2\pi z_i)+10] \\ z=x-o \end{cases}$	$\begin{cases} g_1(x)=-\sum_{i=1}^{D}z_i\sin(2z_i)\le 0 \\ g_2(x)=\sum_{i=1}^{D}z_i\sin(z_i)\le 0 \\ x\in[-10,10]^D \end{cases}$
5	$\begin{cases} f(x)=\sum_{i=1}^{D-1}(100(z_i^2-z_{i+1})^2+(z_i-1)^2) \\ z=x-o, y=M_1\cdot z, w=M_2\cdot z \end{cases}$	$\begin{cases} g_1(x)=\sum_{i=1}^{D}[y_i^2-50\cos(2\pi y_i)-40]\le 0 \\ g_2(x)=\sum_{i=1}^{D}[w_i^2-50\cos(2\pi w_i)-40]\le 0 \\ x\in[-10,10]^D \end{cases}$

6	$\begin{cases} f(x)=\sum_{i=1}^{D}[z_i^{\,2}-10\cos(2\pi z_i)+10] \\ z=x-o \end{cases}$	$\begin{cases} h_1(x)=-\sum_{i=1}^{D} z_i\sin(z_i)=0 \\ h_2(x)=\sum_{i=1}^{D} z_i\sin(\pi z_i)=0 \\ h_3(x)=-\sum_{i=1}^{D} z_i\cos(z_i)=0 \\ h_4(x)=\sum_{i=1}^{D} z_i\cos(\pi z_i)=0 \\ h_5(x)=\sum_{i=1}^{D}(z_i\sin(2\sqrt{\lvert z_i\rvert})=0 \\ h_6(x)=-\sum_{i=1}^{D}(z_i\sin(2\sqrt{\lvert z_i\rvert})=0 \\ x\in[-20,20]^D \end{cases}$
7	$\begin{cases} f(x)=\sum_{i=1}^{D}(z_i^{\,2}\sin(z_i)) \\ z=x-o \end{cases}$	$\begin{cases} h_1(x)=\sum_{i=1}^{D}[z_i-100\cos(0.5z_i)+100]=0 \\ h_2(x)=-\sum_{i=1}^{D}[z_i-100\cos(0.5z_i)+100]=0 \\ x\in[-50,50]^D \end{cases}$
8	$\begin{cases} f(x)=\max(z) \\ z=x-o, y_l=z_{(2l-1)}, w_l=z_{(2l)} \\ where\ \ l=1,...,D/2 \end{cases}$	$\begin{cases} h_1(x)=\sum_{i=1}^{D/2}(\sum_{j=1}^{i} y_j)^2=0 \\ h_2(x)=\sum_{i=1}^{D/2}(\sum_{j=1}^{i} w_j)^2=0 \\ x\in[-100,100]^D \end{cases}$
9	$\begin{cases} f(x)=\max(z) \\ z=x-o, y_l=z_{(2l-1)}, w_l=z_{(2l)} \\ where\ \ l=1,...,D/2 \end{cases}$	$\begin{cases} g(x)=\prod_{i=1}^{D/2} w_i\le 0 \\ h(x)=\sum_{i=1}^{D/2-1}(y_i^2-y_{i+1})^2=0 \\ x\in[-10,10]^D \end{cases}$
10	$\begin{cases} f(x)=\max(z) \\ z=x-o \end{cases}$	$\begin{cases} h_1(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2=0 \\ h_2(x)=\sum_{i=1}^{D-1}(z_i-z_{i+1})^2=0 \\ x\in[-100,100]^D \end{cases}$
11	$\begin{cases} f(x)=\sum_{i=1}^{D}(z_i) \\ z=x-o \end{cases}$	$\begin{cases} g(x)=\prod_{i=1}^{D} z_i\le 0 \\ h(x)=\sum_{i=1}^{D-1}(z_i-z_{i+1})^2=0 \\ x\in[-100,100]^D \end{cases}$

12	$\begin{cases} f(x)=\sum_{i=1}^{D}[y_i^{\,2}-10\cos(2\pi y_i)+10] \\ y=x-o \end{cases}$	$\begin{cases} g_1(x)=4-\sum_{i=1}^{D}\lvert y_i^2\rvert\le 0 \\ g_2(x)=\sum_{i=1}^{D}y_i^2-4\le 0 \\ x\in[-100,100]^D \end{cases}$
13	$\begin{cases} f(x)=\sum_{i=1}^{D-1}(100(y_i^2-y_{i+1})^2+(y_i-1)^2) \\ y=x-o \end{cases}$	$\begin{cases} g_1(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10)-100\le 0 \\ g_2(x)=\sum_{i=1}^{D}y_i-2D\le 0 \\ g_3(x)=5-\sum_{i=1}^{D}y_i^2\le 0 \\ x\in[-100,100]^D \end{cases}$
14	$\begin{cases} f(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}y_i^2})+20 \\ \qquad -\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi y_i))+e \\ y=x-o \end{cases}$	$\begin{cases} g(x)=\sum_{i=2}^{D}y_i^2+1-\lvert y_1\rvert\le 0 \\ h(x)=\sum_{i=1}^{D}y_i^2-4=0 \\ x\in[-100,100]^D \end{cases}$
15	$\begin{cases} f(x)=\max\{\lvert y_i\rvert,1\le i\le D\} \\ y=x-o \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}y_i^2-100D\le 0 \\ h(x)=\cos f(x)+\sin f(x)=0 \\ x\in[-100,100]^D \end{cases}$
16	$\begin{cases} f(x)=\sum_{i=1}^{D}\lvert y_i\rvert \\ y=x-o \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}y_i^2-100D\le 0 \\ h(x)=(\cos f(x)+\sin f(x))^2-\exp(\cos f(x) \\ \qquad +\sin f(x))-1+\exp(1)=0 \\ x\in[-100,100]^D \end{cases}$
17	$\begin{cases} f(x)=\frac{1}{4000}\sum_{i=1}^{D}y_i^{\,2}+1-\prod_{i=1}^{D}\cos(\frac{y_i}{\sqrt{i}}) \\ y=x-o \end{cases}$	$\begin{cases} g(x)=1-\sum_{i=1}^{D}\mathrm{sgn}(\lvert y_i\rvert-\sum_{j=1,\ldots,D,j\ne i}^{D}y_j^2-1)\le 0 \\ h(x)=\sum_{i=1}^{D}y_i^2-4D=0 \\ x\in[-100,100]^D \end{cases}$
18	$\begin{cases} f(x)=\sum_{i=1}^{D}z_i^{\,2}-10\cos(2\pi z_i)+10 \\ y=x-o \\ z_i=\begin{cases} y_i, if\ \lvert y_i\rvert\le 0.5 \\ 0.5round(2y_i), otherwise \end{cases} \end{cases}$	$\begin{cases} g_1(x)=1-\sum_{i=1}^{D}\lvert y_i\rvert\le 0 \\ g_2(x)=\sum_{i=1}^{D}y_i^2-100D\le 0 \\ h(x)=\sum_{i=1}^{D}100(y_i^2-y_{i+1})^2+\prod_{i=1}^{D}\sin^2(y_i-1)\pi=0 \\ x\in[-100,100]^D \end{cases}$

19	$\begin{cases} f(x)=\sum_{i=1}^{D}(\lvert y_i\rvert^{0.5}+2\sin y_i^3) \\ y=x-o \end{cases}$	$\begin{cases} g_1(x)=\sum_{i=1}^{D-1}(-10\exp(-0.2\sqrt{y_i^2+y_{i+1}^2})) \\ \quad +(D-1)\cdot 10/\exp(-5)\le 0 \\ g_2(x)=\sum_{i=1}^{D}\sin^2(2y_i)-0.5D\le 0 \\ x\in[-50,50]^D \end{cases}$
20	$\begin{cases} f(x)=\sum_{i=1}^{D-1}g(y_i,y_{i+1})+g(y_D,y_1) \\ y=x-o \\ g(y_i,y_{i+1})=0.5+\frac{(sin^2(\sqrt{y_i^2+y_{i+1}^2})-0.5)}{(1+0.001(\sqrt{y_i^2+y_{i+1}^2})^2)^2} \end{cases}$	$\begin{cases} g_1(x)=\cos^2(\sum_{i=1}^{D}y_i)-0.25\cos(\sum_{i=1}^{D}y_i)-0.125\le 0 \\ g_2(x)=\exp(\cos(\sum_{i=1}^{D}y_i))-\exp(0.25)\le 0 \\ x\in[-100,100]^D \end{cases}$
21	$\begin{cases} f(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10) \\ z=M(x-o) \end{cases}$	$\begin{cases} g_1(x)=4-\sum_{i=1}^{D}\lvert z_i^2\rvert\le 0 \\ g_2(x)=\sum_{i=1}^{D}z_i^2-4\le 0 \\ x\in[-100,100]^D \end{cases}$
22	$\begin{cases} f(x)=\sum_{i=1}^{D}(100(z_i^2-x_{i+1})^2+(z_i-1)^2) \\ z=M(x-o) \end{cases}$	$\begin{cases} g_1(x)=\sum_{i=1}^{D}(z_i^2-10\cos(2\pi z_i)+10)-100\le 0 \\ g_2(x)=\sum_{i=1}^{D}z_i-2D\le 0 \\ g_3(x)=5-\sum_{i=1}^{D}z_i^2\le 0 \\ x\in[-100,100]^D \end{cases}$
23	$\begin{cases} f(x)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_i^2})+20 \\ \quad -\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i))+e \\ z=M(x-o) \end{cases}$	$\begin{cases} g(x)=\sum_{i=2}^{D}z_i^2+1-\lvert z_1\rvert\le 0 \\ h(x)=\sum_{i=1}^{D}z_i^2-4=0 \\ x\in[-100,100]^D \end{cases}$
24	$\begin{cases} f(x)=\max\{\lvert z_i\rvert,1\le i\le D\} \\ z=M(x-o) \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}z_i^2-100D\le 0 \\ h(x)=\cos f(z)+\sin f(z)=0 \\ x\in[-100,100]^D \end{cases}$
25	$\begin{cases} f(x)=\sum_{i=1}^{D}\lvert z_i\rvert \\ y=M(x-o) \end{cases}$	$\begin{cases} g(x)=\sum_{i=1}^{D}z_i^2-100D\le 0 \\ h(x)=(\cos f(z)+\sin f(z))^2-\exp(\cos f(z) \\ \quad +\sin f(z))-1+\exp(1)=0 \\ x\in[-100,100]^D \end{cases}$

26	$\begin{cases} f(x)=\frac{1}{4000}\sum_{i=1}^{D} y_i^{\,2}+1-\prod_{i=1}^{D}\cos(\frac{y_i}{\sqrt{i}}) \\ z=M(x-o) \end{cases}$	$\begin{cases} g(x)=1-\sum_{i=1}^{D}\mathrm{sgn}(\lvert z_i\rvert-\sum_{j=1,\ldots,D,\,j\neq i}^{D} z_j^2-1)\le 0 \\ h(x)=\sum_{i=1}^{D} z_i^2-4D=0 \\ x\in[-100,100]^D \end{cases}$
27	$\begin{cases} f(x)=\sum_{i=1}^{D} z_i^{\,2}-10\cos(2\pi z_i)+10 \\ z=M(x-o) \\ z_i=\begin{cases} y_i, if\ \lvert y_i\rvert\le 0.5 \\ 0.5round(2y_i), otherwise \end{cases} \end{cases}$	$\begin{cases} g_1(x)=1-\sum_{i=1}^{D}\lvert y_i\rvert\le 0 \\ g_2(x)=\sum_{i=1}^{D} y_i^2-100D\le 0 \\ h(x)=\sum_{i=1}^{D}100(y_i^2-y_{i+1})^2+\prod_{i=1}^{D}\sin^2(y_i-1)\pi=0 \\ x\in[-100,100]^D \end{cases}$
28	$\begin{cases} f(x)=\sum_{i=1}^{D}(\lvert z_i\rvert^{0.5}+2\sin z_i^3) \\ z=M(x-o) \end{cases}$	$\begin{cases} g_1(x)=\sum_{i=1}^{D-1}(-10\exp(-0.2\sqrt{z_i^2+z_{i+1}^2})) \\ \qquad +(D-1)\cdot 10/\exp(-5)\le 0 \\ g_2(x)=\sum_{i=1}^{D}\sin^2(2z_i)-0.5D\le 0 \\ x\in[-50,50]^D \end{cases}$

附表 II 测试问题 LIR-CMOP1-14 目标函数及其约束条件

编号	目标函数	约束条件
1	$\begin{cases} f_1(x)=x_1+g_1(x) \\ f_2(x)=1-x_1^2+g_2(x) \\ g_1(x)=\sum_{j\in J_1}(x_j-sin(0.5\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-cos(0.5\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=(a-g_1(x))(g_1(x)-b)\ge 0 \\ c_2(x)=(a-g_2(x))(g_2(x)-b)\ge 0 \\ a=0.51, b=0.5 \\ x\in[0,1]^{30} \end{cases}$
2	$\begin{cases} f_1(x)=x_1+g_1(x) \\ f_2(x)=1-\sqrt{x_1}+g_2(x) \\ g_1(x)=\sum_{j\in J_1}(x_j-sin(0.5\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-cos(0.5\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=(a-g_1(x))(g_1(x)-b)\ge 0 \\ c_2(x)=(a-g_2(x))(g_2(x)-b)\ge 0 \\ a=0.51, b=0.5 \\ x\in[0,1]^{30} \end{cases}$

3	$\begin{cases} f_1(x)=x_1+g_1(x) \\ f_2(x)=1-x_1^2+g_2(x) \\ g_1(x)=\sum_{j\in J_1}(x_j-sin(0.5\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-cos(0.5\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=(a-g_1(x))(g_1(x)-b)\geq 0 \\ c_2(x)=(a-g_2(x))(g_2(x)-b)\geq 0 \\ c_3(x)=sin(c\pi x_1)-0.5\geq 0 \\ a=0.51, b=0.5, c=20 \\ x\in[0,1]^{30} \end{cases}$
4	$\begin{cases} f_1(x)=x_1+g_1(x) \\ f_2(x)=1-\sqrt{x_1}+g_2(x) \\ g_1(x)=\sum_{j\in J_1}(x_j-sin(0.5\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-cos(0.5\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=(a-g_1(x))(g_1(x)-b)\geq 0 \\ c_2(x)=(a-g_2(x))(g_2(x)-b)\geq 0 \\ c_3(x)=sin(c\pi x_1)-0.5\geq 0 \\ a=0.51, b=0.5, c=20 \\ x\in[0,1]^{30} \end{cases}$
5	$\begin{cases} f_1(x)=x_1+10g_1(x)+0.7057 \\ f_2(x)=1-\sqrt{x_1}+10g_2(x)+0.7057 \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_k(x)=((f_1-p_k)\cos\theta_k-(f_2-q_k)\sin\theta_k)^2/a_k^2 \\ +((f_1-p_k)\sin\theta_k+(f_2-q_k)\cos\theta_k)^2/b_k^2\geq r \\ p_k=[1.6,2.5], q_k=[1.6,2.5] \\ a_k=[2,2], b_k=[4,8] \\ r=0.1, \theta_k=-0.25\pi \\ x\in[0,1]^{30}, k=1,2 \end{cases}$
6	$\begin{cases} f_1(x)=x_1+10g_1(x)+0.7057 \\ f_2(x)=1-x_1^2+10g_2(x)+0.7057 \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_k(x)=((f_1-p_k)\cos\theta_k-(f_2-q_k)\sin\theta_k)^2/a_k^2 \\ +((f_1-p_k)\sin\theta_k+(f_2-q_k)\cos\theta_k)^2/b_k^2\geq r \\ p_k=[1.8,2.8], q_k=[1.8,2.8] \\ a_k=[2,2], b_k=[8,8] \\ r=0.1, \theta_k=-0.25\pi \\ x\in[0,1]^{30}, k=1,2 \end{cases}$
7	$\begin{cases} f_1(x)=x_1+10g_1(x)+0.7057 \\ f_2(x)=1-\sqrt{x_1}+10g_2(x)+0.7057 \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_k(x)=((f_1-p_k)\cos\theta_k-(f_2-q_k)\sin\theta_k)^2/a_k^2 \\ +((f_1-p_k)\sin\theta_k+(f_2-q_k)\cos\theta_k)^2/b_k^2\geq r \\ p_k=[1.2,2.25,3.5], q_k=[1.2,2.25,3.5] \\ a_k=[2,2.5,2.5], b_k=[6,12,10] \\ r=0.1, \theta_k=-0.25\pi \\ x\in[0,1]^{30}, k=1,2,3 \end{cases}$
8	$\begin{cases} f_1(x)=x_1+10g_1(x)+0.7057 \\ f_2(x)=1-x_1^2+10g_2(x)+0.7057 \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_k(x)=((f_1-p_k)\cos\theta_k-(f_2-q_k)\sin\theta_k)^2/a_k^2 \\ +((f_1-p_k)\sin\theta_k+(f_2-q_k)\cos\theta_k)^2/b_k^2\geq r \\ p_k=[1.2,2.25,3.5], q_k=[1.2,2.25,3.5] \\ a_k=[2,2.5,2.5], b_k=[6,12,10] \\ r=0.1, \theta_k=-0.25\pi \\ x\in[0,1]^{30}, k=1,2,3 \end{cases}$

9	$\begin{cases} f_1(x)=1.7057x_1(10g_1(x)+1) \\ f_2(x)=1.7057(1-x_1^2)(10g_2(x)+1) \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=((f_1-p_1)\cos\theta_1-(f_2-q_1)\sin\theta_1)^2/a_1^2 \\ +((f_1-p_1)\sin\theta_1+(f_2-q_1)\cos\theta_1)^2/b_1^2\geq r \\ c_2(x)=f_1\sin\alpha+f_2\cos\alpha \\ -\sin(4\pi(f_1\cos\alpha-f_2\sin\alpha))-2\geq 0 \\ p_1=1.4, q_1=1.4, a_1=1.5, b_1=6.0 \\ r=0.1, \alpha=0.25\pi, \theta_1=-0.25\pi \\ x\in[0,1]^{30} \end{cases}$
10	$\begin{cases} f_1(x)=1.7057x_1(10g_1(x)+1) \\ f_2(x)=1.7057(1-\sqrt{x_1})(10g_2(x)+1) \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=((f_1-p_1)\cos\theta_1-(f_2-q_1)\sin\theta_1)^2/a_1^2 \\ +((f_1-p_1)\sin\theta_1+(f_2-q_1)\cos\theta_1)^2/b_1^2\geq r \\ c_2(x)=f_1\sin\alpha+f_2\cos\alpha \\ -\sin(4\pi(f_1\cos\alpha-f_2\sin\alpha))-1\geq 0 \\ p_1=1.1, q_1=1.2, a_1=2.0, b_1=4.0 \\ r=0.1, \alpha=0.25\pi, \theta_1=-0.25\pi \\ x\in[0,1]^{30} \end{cases}$
11	$\begin{cases} f_1(x)=1.7057x_1(10g_1(x)+1) \\ f_2(x)=1.7057(1-\sqrt{x_1})(10g_2(x)+1) \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=((f_1-p_1)\cos\theta_1-(f_2-q_1)\sin\theta_1)^2/a_1^2 \\ +((f_1-p_1)\sin\theta_1+(f_2-q_1)\cos\theta_1)^2/b_1^2\geq r \\ c_2(x)=f_1\sin\alpha+f_2\cos\alpha \\ -\sin(4\pi(f_1\cos\alpha-f_2\sin\alpha))-2.1\geq 0 \\ p_1=1.2, q_1=1.2, a_1=1.5, b_1=5.0 \\ r=0.1, \alpha=0.25\pi, \theta_1=-0.25\pi \\ x\in[0,1]^{30} \end{cases}$
12	$\begin{cases} f_1(x)=1.7057x_1(10g_1(x)+1) \\ f_2(x)=1.7057(1-x_1^2)(10g_2(x)+1) \\ g_1(x)=\sum_{i\in J_1}(x_i-\sin(\frac{0.5i}{30}\pi x_1))^2 \\ g_2(x)=\sum_{j\in J_2}(x_j-\cos(\frac{0.5j}{30}\pi x_1))^2 \\ J_1=\{3,5,\ldots,29\}, J_2=\{2,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=((f_1-p_1)\cos\theta_1-(f_2-q_1)\sin\theta_1)^2/a_1^2 \\ +((f_1-p_1)\sin\theta_1+(f_2-q_1)\cos\theta_1)^2/b_1^2\geq r \\ c_2(x)=f_1\sin\alpha+f_2\cos\alpha \\ -\sin(4\pi(f_1\cos\alpha-f_2\sin\alpha))-2.5\geq 0 \\ p_1=1.6, q_1=1.6, a_1=1.5, b_1=6.0 \\ r=0.1, \alpha=0.25\pi, \theta_1=-0.25\pi \\ x\in[0,1]^{30} \end{cases}$
13	$\begin{cases} f_1(x)=(1.7057+g_1)\cos(0.5\pi x_1)\cos(0.5\pi x_2) \\ f_2(x)=(1.7057+g_1)\cos(0.5\pi x_1)\sin(0.5\pi x_2) \\ f_3(x)=(1.7057+g_1)\sin(0.5\pi x_1) \\ g_1=\sum_{i\in J}10(x_i-0.5)^2 \\ J=\{3,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=(g(x)-9)(g(x)-4) \\ c_2(x)=(g(x)-3.61)(g(x)-3.24) \\ g(x)=f_1^2+f_2^2+f_3^2 \\ x\in[0,1]^{30} \end{cases}$
14	$\begin{cases} f_1(x)=(1.7057+g_1)\cos(0.5\pi x_1)\cos(0.5\pi x_2) \\ f_2(x)=(1.7057+g_1)\cos(0.5\pi x_1)\sin(0.5\pi x_2) \\ f_3(x)=(1.7057+g_1)\sin(0.5\pi x_1) \\ g_1=\sum_{i\in J}10(x_i-0.5)^2 \\ J=\{3,4,\ldots,30\} \end{cases}$	$\begin{cases} c_1(x)=(g(x)-9)(g(x)-4) \\ c_2(x)=(g(x)-3.61)(g(x)-3.24) \\ c_3(x)=(g(x)-3.0625)(g(x)-2.56) \\ g(x)=f_1^2+f_2^2+f_3^2 \\ x\in[0,1]^{30} \end{cases}$

攻读硕士学位期间主要的工作成果

已发表国际会议论文

1、Zhun Fan, **Yi Fang**, Wenji Li, Yutong Yuan, Zhaojun Wang and Xinchao Bian, LSHADE44 with an Improved Epsilom Constraint-handling Method for Solving Constrained Single-objective Optimization Problems[C]// Evolutionary Computation (CEC), 2018 IEEE Congress on. IEEE, 2018.

2、Zhun Fan, Wenji Li, Xinye Cai, **Yi Fang**, Jiewei Lu and Caimin Wei. A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems[C]//Evolutionary Computation (CEC), 2017 IEEE Congress on. IEEE, 2017: 209-216.

3、Jiajie Mo, Zhun Fan, Wenji Li, **Yi Fang**, Yugen You and Xinye Cai, Multi-Factorial Evolutionary Algorithm Based on M2M Decomposition[C]//Asia-Pacific Conference on Simulated Evolution and Learning. Springer, Cham, 2017: 134-144.

已投稿 SCI 期刊论文

1、Zhun Fan , **Yi Fang** , Wenji Li , Xinye Cai , Caimin Wei and Erik Goodman. MOEA/D with Angle-based Constrained Dominance Principle for Constrained Multi-objective Optimization Problems. Applied Soft Computing.

2、Zhun Fan, Wenji Li, Xinye Cai, Han Huang, **Yi Fang**, Yugen You, Jiajie Mo, Caimin Wei and Erik Goodman. An Improved Epsilon Constraint-handling Mehtod in MOEA/D for CMOPs with Large Infeasible Regions. Soft Computing.

攻读硕士期间获奖情况

1、第十二届中国研究生电子设计竞赛全国二等奖、华南赛区一等奖。

2、第十四届中国研究生数学建模竞赛全国三等奖。