学士学位论文

基于参考向量的多目标约束演化算法

学　　号：　**20141440193**

姓　　名：　李超

学科专业：　通信工程

指导教师：　曾三友 教授

培养单位：　机械与电子信息学院

二〇一八年六月

中国地质大学（武汉）学士学位论文原创性声明

本人郑重声明：本人所呈交的学士学位论文《基于参考向量的多目标约束演化算法》，是本人在指导老师的指导下，在中国地质大学（武汉）攻读学士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果，对论文的完成提供过帮助的有关人员已在文中说明并致以谢意。

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

学位论文作者签名：_____

日　期：　　年　　月　　日

摘要

优化问题在科学和工程上有着广泛的应用。根据优化目标的个数，可以将优化问题简单的分为单目标优化和多目标优化，目标值只有一个的优化问题称之为单目标优化，目标值含有两个或者两个以上的优化问题称之为多目标优化。在多目标优化种，多个目标可能会相互冲突，因而很难同时是多个目标同时达到最优。这也是多目标优化问题较之单目标表优化问题更加的困难的地方。

为了解决多目标优化问题，很多研究学者提出了各种基于自然选择的演化算法。在用演化算法求解多目标优化问题时需要在种群的收敛性和多样性间平衡。RVEA(基于参考向量的演化算法)是一种基于参考向量的演化算法，该算法能动态的调整参考向量的分布来选择子代，因而在处理不同的多目标优化问题间有着良好的鲁棒性。大量的实验证明，RVEA 算法在较少演化代数后获得的种群能比较均匀的分布在其所求解的多目标优化问题的 PF（帕累托前沿）上。本文在 RVEA 算法的基础上做了以下几点工作：

1）采用差分进化策略替代原始 RVEA 算法的遗传进化策略。差分进化相对于遗传进化结构更加的简单，运行速度更快

2）原始 RVEA 算法处理约束条件时，违约值未归一化。本文用归一化违约值代替原始 RVEA 算法的违约值，提出了归一化违约值的 RVEA 算法。该算法在处理不同违约条件相差较大的约束多目标优化问题。

3）在归一化违约值的 RVEA 算法中，增加了种群决策向量多样性的 Niche-count 目标以及最小化归一化违约值的目标，并且结合ε可行解的动态约束边界的多目标优化问题。该算法由于增加了两个额外的优化目标，故可以求解单目标优化。本文采用该算法求解单目标优化问题，结果较佳。

关键字：多目标优化；演化算法 ；RVEA； 差分进化；约束优化

Abstract

Optimization problems have a wide range of applications in science and engineering. According to the number of optimization goals, the optimization problem can be simply divided into single-objective optimization and multi-objective optimization. The optimization problem with only one target value is called single-objective optimization. The target value contains two or more optimization problems. This is a multi-objective optimization. In multi-objective optimization, multiple goals may conflict with each other, making it difficult to achieve multiple goals at the same time. This is also where the multi-objective optimization problem is more difficult than the single-object optimization problem.

In order to solve the multi-objective optimization problem, many researchers have proposed various evolutionary algorithms based on natural selection. When solving multi-objective optimization problems with evolutionary algorithms, it is necessary to balance the convergence and diversity of the population. RVEA (based on the reference vector evolution algorithm) is an evolutionary algorithm based on the reference vector. The algorithm can dynamically adjust the distribution of the reference vector to select the children, so it has good robustness in dealing with different multi-objective optimization problems. A large number of experiments have proved that the population of RVEA algorithm obtained after less evolutionary algebra can be more evenly distributed on the PF (Pareto front) of the multi-objective optimization problem it solves. This article does the following on the basis of the RVEA algorithm:

1) Adopt a differential evolution strategy to replace the genetic evolutionary strategy of the original RVEA algorithm. Differential evolution is simpler and faster than genetic evolution.

2) When the original RVEA algorithm deals with constraint conditions, the default value is not normalized. In this paper, the default value of the normalized RVEA algorithm is used to replace the default value of the original RVEA algorithm, and a normalized default value RVEA algorithm is proposed. This algorithm deals with constrained multi-objective optimization problems with large differences in different breach conditions.

3) In the RVEA algorithm that normalizes the default value, the Niche-count goal of the diversity vector of the population decision-making and the goal of

minimizing the normalized default value are added, and the multi-objective optimization problem of the dynamic constraint boundary of the feasible solution is combined. Because this algorithm adds two additional optimization goals, single-objective optimization can be solved. This paper uses this algorithm to solve the single-objective optimization problem, but the result is very good.

Key words: multi-objective optimization; evolutionary algorithm; RVEA ; differential evolution constrained optimization

目录

第一章 绪论

1.1 研究背景

优化问题有着十分广泛的应用。我们见到优化问题常常只含有一个目标函数以及零个或者多个约束条件，这一类的优化问题我们称之为单目标优化。除此之外，还有一类优化问题我们偶尔也会遇到，有一个以上的目标函数以及零个或者多个约束条件的优化问题，我们称之为多目标优化。单目标优化得到的任何两个解都能比较孰优孰劣，因此单目标优化问题得到的是问题的解。多目标优化问题由于含有多个目标，某两个解可能存在在某一目标上解一优于解二，而在另一目标维度结果则相反，因此我们无法比较多个解之间的优劣性，因此多目标优化算法得到的问题的解是一个集合形式，我们称之为解集。

我们如何评价一个多目标优化问题解集的好坏呢？评价一个多目标优化问题解集的优劣常用经济学里面的帕累托积累来描述，帕累托积累其中一个重要的概念就是支配解和非支配解。如果一个解在所有的目标都劣于另外一个解，我们称之这样的解为支配解。显然支配的解对解集帕累托积累没有意义，所以我们需要去掉所有的支配解。为了具体描述各种算法的优劣，我们采取 HV 评价指标。

多目标优化问题求解算法有多种形式，大致可以分为两大类。一类是传统的算法，基于问题本身求解，这类算法针对具体问题或许会有很高的效率，然而不具有鲁棒性。另外一类是进化算法，这类算法不需要问题拥有特定的形式，鲁棒性较强，本文以后描述的所有相关算法都属于这一类算法。进化算法求解多目标优化也分为两大类，一本文采用的 RVEA（参考向量指导的进化算法）算法属于基于目标分解的算法。原始的 RVEA 算法处理均匀分布的帕累托面问题的效果比较好，然而帕累托面不规则以及约束条件较为复杂时，RVEA 算法处理起来比较棘手。

本文采用了两种方法处理约束多目标优化问题：1.动态边界约束：将一阶归一化违约条件和 Niche-count（描述解多样性的指标）当成额外的目标，并动态收缩约束边界的方法

2.一阶归一化违约条件：我们在原始的 RVEA 算法中加入归一化违约条件来选择子代。比较两种方法，实验结果显示方法 1 完全没效果，方法 2 效果还不错。在多目标优化问题中增加额外的目标的方法不适合 RVEA 算法，归一化违约条件有利于处理多个违约条件分布不均匀的问题。

1.2 约束多目标优化问题的研究现状

我们将所有非支配解组成的面称之为 PF（Pareto Front）。多目标优化算法的目的就是获得的解集更加接近于 PF。直观的讲，我们获得的解需要在 PF 上分布均匀。为了达到这一目的，大致有三类算法：1.基于收敛性增强的算法，收敛性增加算法直观的考虑就是修改优势关系增加对 PF 选择压力 ，这样能避免在传统多目标演化算法中因为收敛性选择压力的损失导致的典型优势无法区分。这一类算法主要包括 $\varepsilon-$ 占优[1]、 $L-$ 优化[2]、偏好排序[3]、模糊占优[4]等 2.基于目标分解，即将多目标问题分解为一系列题子问题并以协同的方式解决他们[5][6]。目标分解的方式也可以分为两类，一类是将多目标分解成为一系列的单目标问题，包括早期的的加权聚合方法[7][8]，以及最近引入子问题子问题之间解决方案更加明确协同的 MOEA(MOEA/d)算法的变种。第二类多目标分解的算法，多目标问题还是分解为多目标问题。例如 MOEA/D-M2M[9]是将整个 PF 划分成多个分段，并且每一个分段被视为一个新的子问题，NSGA-III 是预定义一系列的参考点来保证解的多样性以及收敛性[10]。大量的实验结果表明第二类分解策略比第一类分解策略更加的有效[11,12]3.基于评价指标。在求解多目标优化问题中，评价指标显示的结果越好，则算法的结果越好，因此就有了基于评价指标的优化算法，这一类算法主要有基于 S 度量选择的多目标进化算法[13]和基于 HV 动态邻域的多目标进化算法[14]以及基于快速 HV 的 HypE[15]。但是这一类算法有一个致命的缺点：当目标数较多时计算的时间复杂度较高，不适合目标数较多的多目标优化算法。

还有一些算法不属于其中的任何一类，比如一些人采用的基于参考点求解多目标优化问题[16]以及通过缩小目标求解多目标优化的算法[17] 。我们看到大多数算法都是收敛性增强和维持多样性的策略。我们也可以通过生成均匀分布的参考点或者参考向量，来作为选择策略，NSGA-III 既是一种基于目标分解的算法也是基于参考点选择的算法。与之类似 RVEA 是基于参考向量的择算法[18]。原始 RVEA 算法处理违约条件太过于粗糙，我们可以将一阶约束的违约值代替原始的违约值。这样处理违约条件可以避免某些违约条件下的违约值相差几个数量级情况下导致部分违约条件几乎失效。理论上结果应该会变好。对于一些等式约束较多的难约束的多目标优化问题，如果不缩放约束条件，则在选择过程中几乎只有违约值起作用（因为等式约束较多时随机初始化的种群几乎都是不可行解），这样 RVEA 算法里基于参考向量选择步骤形同虚设，这种情况原始算法效率较低。我们采用动态边界约束的方法处理这类问题结果应该会变好。

1.3 论文结构安排

论文结构安排如下:

第一章绪论，主要介绍多目标优化问题的研究背景与研究现状，最后给出论文的结构安排。

第二章介绍多目标优化问题的一些基本的概念，包括优化问题的定义、差分进化算法以及多目标优化问题的评价指标等。

第三章介绍 RVEA 算法的原理，并且给出了该算法在无约束多目标优化问题上的表现。

第四章介绍 RVEA 算法求解约束多目标优化问题，主要介绍了归一化一阶违约值和动态约束边界的两种 RVEA 算法的改进，并在文末给出了测试结果。

第二章 基本概念.

2.1 优化问题的描述

2.1.1 单目标优化

单目标优化问题是指的优化目标只有一个，一般描述如公式：

$$\begin{aligned}&\min f(\mathbf{x})\\&st\begin{cases}\mathbf{g}(\mathbf{x})=(g_1(\mathbf{x}),g_2(\mathbf{x}),\cdots g_{m1}(\mathbf{x}))\le\mathbf{0}\\\mathbf{h}(\mathbf{x})=(h_1(\mathbf{x}),h_2(\mathbf{x}),\cdots h_{m2}(\mathbf{x}))=\mathbf{0}\end{cases}\\&\mathbf{x}\in\mathbf{X},\mathbf{X}=\{\mathbf{x}\mid\mathbf{l}<\mathbf{x}<\mathbf{u}\}\end{aligned}\tag{2.1}$$

其中$\mathbf{l}=(l_1,l_2,\cdots l_D)$，$\mathbf{u}=(u_1,u_2,\cdots u_D)$，$\mathbf{x}=(x_1,x_2,\cdots x_D)$，$D$为决策变量的个数。我们将$x$称之为决策向量，$\mathbf{X}$称之为决策空间，$l$和$u$分别称之为决策变量的下边界和上边界。$f(\mathbf{x})$是目标函数，$g(\mathbf{x})$和$h(\mathbf{x})$分别称之为不等式约束和等式约束，在实数范围内等式约束可以加上绝对值转换为不等式约束。如$h_1(\mathbf{x})=0$等价于$|h_1(\mathbf{x})|\le 0$。 为了讨论的方便我们今后优化问题统一写成不等式约束的形式。

在优化问题中我们根据是否含有约束条件$g(\mathbf{x})$将其分为无约束优化问题和约束优化问题。在约束优化问题中，满足约束条件的解称之为可行解，不满足约束条件的解称之为不可行解。

2.1.2 多目标优化

多目标优化问题描述和单目标优化问题表述类似，具体形式见公式(2.2)：

$$\begin{aligned}&\min\mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),f_2(\mathbf{x}),\cdots f_M(\mathbf{x}))\\&st\begin{cases}\mathbf{g}(\mathbf{x})=(g_1(\mathbf{x}),g_2(\mathbf{x}),\cdots g_{m1}(\mathbf{x}))\le\mathbf{0}\\\mathbf{h}(\mathbf{x})=(h_1(\mathbf{x}),h_2(\mathbf{x}),\cdots h_{m2}(\mathbf{x}))=\mathbf{0}\end{cases}\\&\mathbf{x}\in\mathbf{X},\mathbf{X}=\{\mathbf{x}\mid\mathbf{l}<\mathbf{x}<\mathbf{u}\}\end{aligned}\tag{2.2}$$

M表示优化目标的个数，其他符号的含义与单目标优化问题一致。我们可以看到单目标优化是多目标优化的一种特例。

2.2 差分进化算法

求解优化问题的算法传统算法有牛顿法，梯度下降法等。这一类算法一般来说效率比较高，但需要优化问题为凸的情况下才能使用。此外，这一类方法较为复杂，而且需要根据具体问题计算梯度，很多的优化问题梯度计算很复杂或者根本无法计算梯度，因此这一类算法不具有普适性。演化算法是一类更具自然界演化规律的算法，无需计算梯度，具有很好的普适性，并且易于大规模的并性计算。演化算法实质上是一种模仿自然规律的种群搜索的策略，将达尔文进化论的优胜劣汰作为子代选择的算法。常用的演化算法有遗传算法和差分进化算法。本文主要介绍差分进化算法。

2.2.1 差分进化算法概念

差分进化算法是为解决连续决策空间非线性及不可微（无法求梯度）优化问题提出来的一种启发性的算法。有大量的实验证明，对于连续决策空间优化问题差分进化算法比遗传算法更快、更精确的收敛。差分进化算法的流程图如图 2-1 所示。从图中可以看出差分进化算法和遗传算法大体框架类似，也分为变异、交叉、选择几个主要的步骤。关于差分进化算法具体的变异、交叉和选择的细节在下一节里面讲。

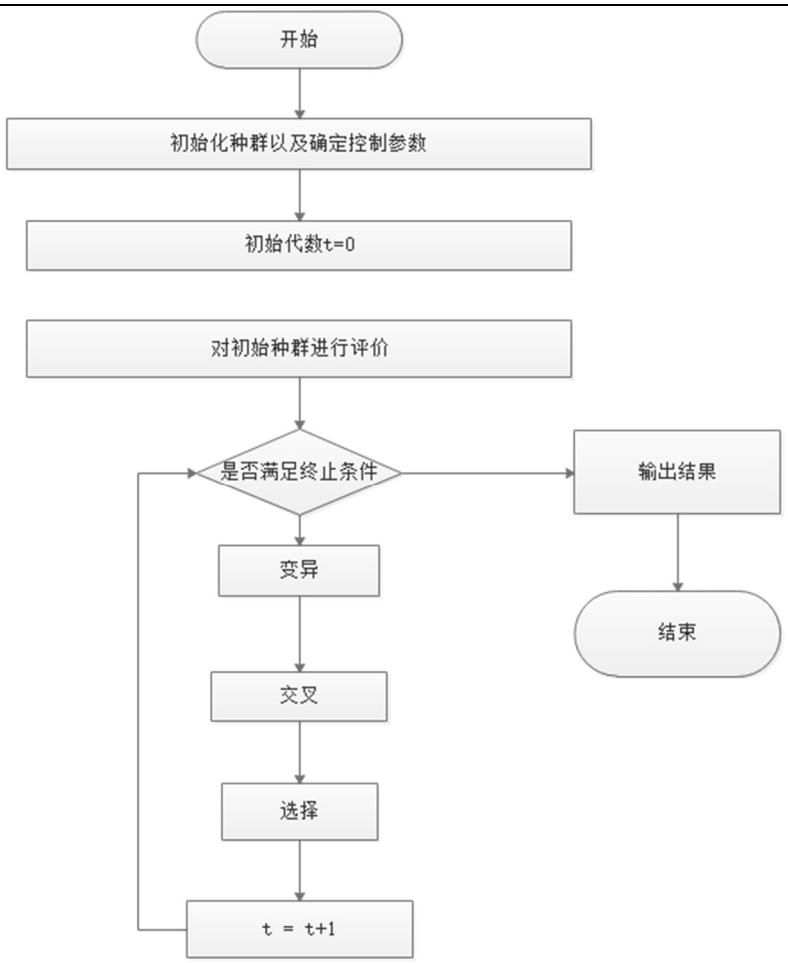

图 2-1 差分进化算法流程图

2.2.2 差分进化

1) 变异操作

对于每一个目标个体 F，差分变异向量 $v_i(g+1)$ 由以下公式产生:

$$\mathbf{v}_i(g+1)=\mathbf{x}_{r1}(g)+F\bullet(\mathbf{x}_{r2}(g)-\mathbf{x}_{r3}(g))\ \ i=1,2,3\cdots,NP \tag{2.3}$$

上述公式中 NP 表示种群的大小，$r1,r2,r3\in\{1,2,\cdots,NP\}$，$x_{r1}(g),x_{r2}(g),x_{r3}(g)$ g 表示第代随机采样的 3 个个体,缩放因子，$F\in(0,2)$，为一个实常数。

2) 交叉操作

为了增加种群的多样性，我们引入交叉操作。假设每个个体的基因数为,我们将父代个体 $x_i(g)$ 和 D 差分变异产生的个体,$v_i(g+1)$ 都表示为向量的形式，即有 $\mathbf{x}_i(g)=(x_{i1}(g),x_{i2}(g)\cdots,x_{iD}(g))$，$v_i(g+1)=(v_{i1}(g+1),v_{i2}(g+1)\cdots,v_{iD}(g+1))$，交叉操作由以下公式给出：

$$u_{ij}(g+1)=\begin{cases}v_{ij}(g+1)\ if\ rand(0,1)\le CR\\ x_{ij}(g)\quad otherwise\end{cases} \tag{2.4}$$

其中$i\in\{1,2,\cdots NP\}, j\in\{1,2,\cdots D\}$,$u_{ij}(g+1)$表示第$g+1$代的第$i$个个体的第$j$个基因。$rand(0,1)$表示随机产生的$(0,1)$区间的随机数，$CR$ 是交叉控制参数，$CR\in[0,1]$。CR 越小，子代和父代的相似性越高，极端的情形$CR=0$子代和父代的差异性为 0。为了保证子代的多样性，CR 尽量选择较接近于 1 的常数。图 2-2 解释了 7 个基因的交叉操作，其中$X_{i,G}$表示第G代第i个个体，$V_{i,G+1}$表示第$G+1$代第i个变异产生的个体，$U_{i,G+1}$表示第$G+1$代产生的第i个个体。

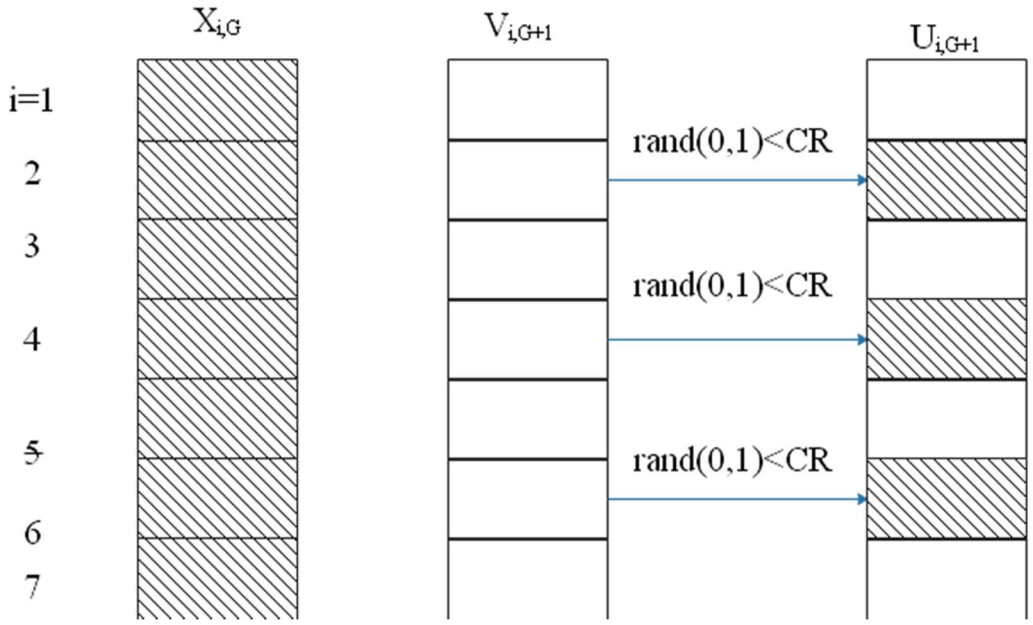

图 2-2　交叉操作概念图

3) 选择操作

差分进化算法选择的策略是一种贪心的策略：

$$\mathbf{x}_i(g+1)=\begin{cases}\mathbf{u}_i(g+1)\ if\ f(\mathbf{u}_i(g+1))\le f(\mathbf{x}_i(g))\\ \mathbf{x}_i(g)\quad otherwise\end{cases} \tag{2.5}$$

f 是最小化的目标函数，其他符号的意义于前文相同。

1） 差分算法的一个简单演示问题

$$\begin{aligned}&\min f(\mathbf{x})=\sum_{i=1}^{10}x_i^2\\ &st\ -10\le x_i\le 10\ i=1,2,3\cdots 10\end{aligned} \tag{2.6}$$

这一个问题相当简单，根据基本的数学知识我们知道$x_i=0$时，$f(x)$有最小值 0 。在这个问题上我们取$CR=0.8, F=0.5$,种群规模为 500 时做出每一代

$f(\mathbf{x})$ 的最优值如图 2-3 所示，我们从图可以看出差分进化算法收敛速度较快，100 代后基本收敛到了最优值。

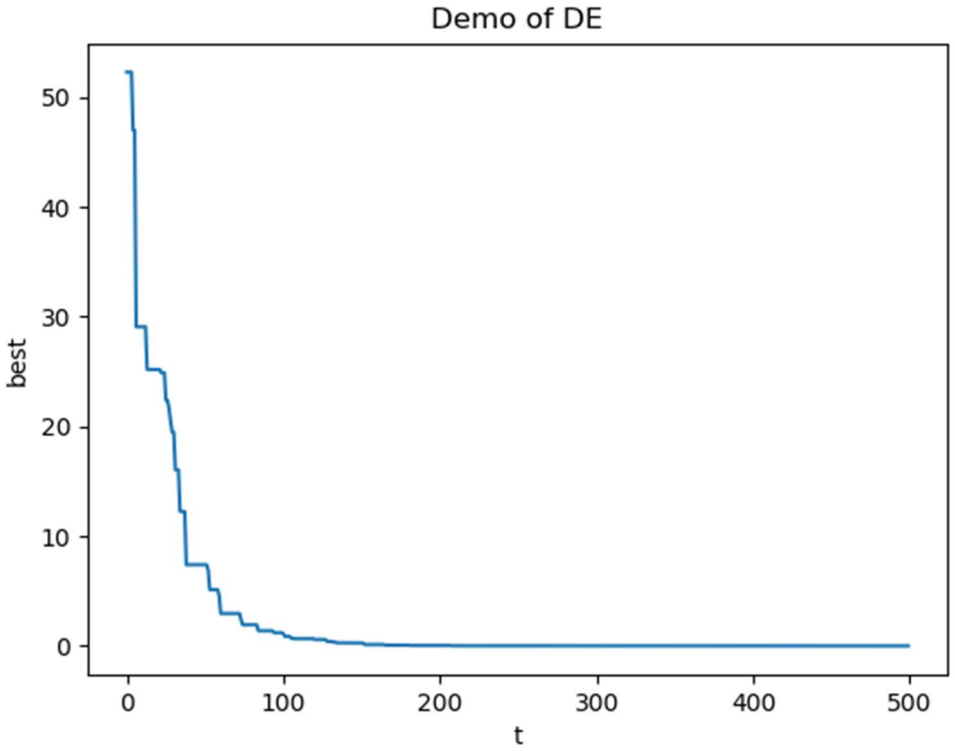

图 2-3 差分进化简单演示收敛曲线

2.2.3 RVEA 算法中的差分操作

在 RVEA 算法中，只用到了差分进化算法中的子代产生的方式。因，此 RVEA 的差分部分是在完整的差分进化算法中砍掉了选择操作的部分，我们不妨把 RVEA 算法中的差分操作称之为差分操作。差分操作的作用是产生子代，我们可以很轻易用其他产生子代的操作替换掉差分操作，比如说遗传算法、粒子群算法，因而我们的代码具有很好的扩展性。

在原始的差分进化算法中子代的种群数 NP 是固定的，而在 RVEA 算法中每一代选择的个体数目不是定值。在变异操作中，需要在 NP 个数中随机取三个数 $r1$，$r2$ ，$r3$ ，RVEA 算法种群数可能出现 $NP<3$,这样在其中随机取三个数就会出现异常。我们在处理这部分的时候用到了一个技巧，用洗牌算法生成了 $(1,2,3,\cdots NP)$ 个数的两个随机排列 $(a1,a2,a3,\cdots aNP)$,$(b1,b2,b3,\cdots bNP)$ 。差分变异操作随机取出的 $r1$ ，$r2$ ，$r3$ 由数对 (i,ai,bi) 代替，这样就不存在种群数目小于 2 的时候，差分变异操作异常。

2.3 多目标优化算法评价指标

如何评价两个不同算法的优劣程度呢？这是一个不可避免的问题。在单目标优化算法中，我们可以通过比较两个算法得到的最优值来评价解的优劣性。然而，在多目标优化问题中，评价连个算法的优劣性并没有那么的简单。我们首先介绍比较多目标优化算法的一些基本概念，然后给出 HV 指标（Hypervolume indicator）的基本概念与计算公式。

2.3.1 多目标优化算法评价的基本概念

1) 支配解与非支配解

假设 $\mathbf{x}_1=(x_{11},x_{12},\cdots x_{1D})$， $\mathbf{x}_2=(x_{21},x_{22},\cdots x_{2D})$ 计算的目标值 $\mathbf{f}_1$， $\mathbf{f}_2$ 满足:

$$\begin{aligned}&\mathbf{f}_1=(f_{11},f_{12},\cdots f_{1M});\mathbf{f}_2=(f_{21},f_{22},\cdots f_{2M})\\&f_{1j}<f_{2j},j\in(1,2,\cdots M)\end{aligned}\tag{2.7}$$

我们称之为 $\mathbf{x}_1$ 支配 $\mathbf{x}_2$， $\mathbf{x}_1$ 称之为非支配解， $\mathbf{x}_2$ 称之为支配解。

2) Pareto Front（帕累托前沿）

在多目标优化问题中有一个非常重要的概念就是 Pareto Front。Pareto Front 指的的是所有非支配解构成的的解集。评价两个多目标优化算法的一种可行的办法就是比较两个算法得到的解集与 Pareto Front 间的相似程度，越相似代表该算法越优秀。

2.3.2 HV 指标及计算算法

固然比较算法得到的解集与 Pareto Front 间的相似性，可以很好反映算法的好坏。但是这种相似性却不是那么好度量。在多目标优化问题上，我们常用的是 HV 评价指标[19]。HV 指的是解集的目标集合 S 中所有的非支配解与参考点间围成的超立方的体积。为了便于理解，我们画出二维情况下的 HV 计算的示意图。在图 2-4 中，$S=(\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3,\mathbf{f}_4)$，(1.0,1.0) 是参考点，根据 HV 的定义，图中阴影部分的面积代表 HV 的值。很明显的我们可以得出两个结论：1. 支配解对于 HV 值不起作用 2.HV 值越大，解集越好。所以我们可以用 HV 衡量算法的优劣。下面按照维度的不同把 HV 计算分为两种情形：

1) 低维 HV 的计算($M\le 5$)

HV 指标直观上看挺简单的，但是 HV 的计算并没有那么简单。低维空间 HV 指标计算概括就是将将种群目标值排序、分割种群并计算每一个个体对 HV 值的贡献、最后将所有的个体 HV 值相加[20]。由于 HV 计算不是本文的主要工作，而且这部分内容已经有较好的实现代码，故此不详细介绍。

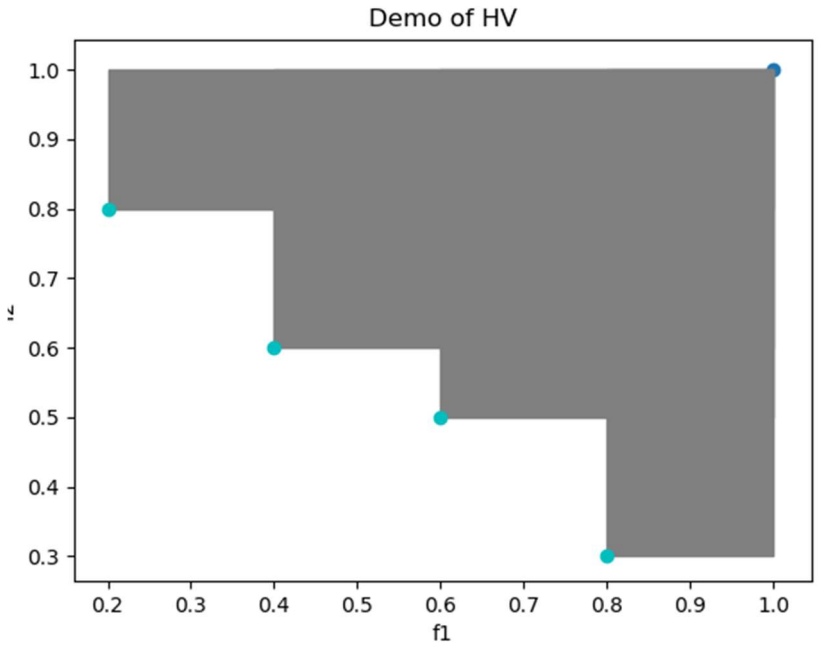

图 2-4 HV 概念解释

2) 高维 HV 的计算($M>5$)

HV 指标计算的一个重要的问题就是时间复杂度过高。HV 值计算的时间复杂度为$O(n^{M-2}log(n))$，M 表示目标维数，n 表示解集中非支配解的个数。显然当维度过高时计算确切的 HV 值是不划算的，我们可以采取蒙特卡罗方法计算近似解。HV 计算步骤由算法 1 给出。需要指出的是采用这种方法计算出来的 HV 是原始定义的 HV 与参考点与原点围成体积之比，所以由 HV$\in[0,1]$ 。为了统一 HV 计算值，我们把低维度的 HV 计算值也归一化。

算法 1 高维空间 HV 计算

输入：解集$S=\{S_1,S_2,\cdots S_{NP}\}$以及参考点$R$

输出：解集S的 HV 值

1. 在原点与参考点围成的超立方体间随机生成N个点，$V=\{\mathbf{v}_1,\mathbf{v}_2,\cdots\mathbf{v}_N\}$，并将这$N$个点的标志$q$设置为 false。即$q[\mathbf{v}_i]=\text{false}$，$v_i\in V$。
2. # 根据占优关系设置 flag
3. **For** S_i **in** S
4. **For** $\mathbf{v}_j$ **in** V
5. **If** S_i 占优V_j
6. $q[\mathbf{v}_j]=\text{true}$
7. **Endif**
8. **Endfor**
9. **Endfor**
10. cnt=0
11. # 统计N个点中 true 的个数

12. **For** $\mathbf{v}_i$ **in** V
13. **If** $q[\mathbf{v}_j] == \text{true}$
14. cnt+=1
15. **Endif**
16. **Endfor**
17. HV = cnt\N #计算 HV 值
18. **Return** HV

第三章 RVEA 求解无约束的多目标优化

3.1 RVEA 算法的背景知识

3.1.1 基于目标分解的多目标优化

在前面已经提到过基于目标分解的这一类多目标优化算法的研究概况，这里就不再赘述了。在这里我们主要谈谈目标分解的由来。

我们知道多目标优化难以求解的主要原因是多个目标值可能存在冲突，如果我们如果能把多个目标转化为一个目标问题就简单多了。一个直观的想法就是把多个目标累加得到新的目标，把问题转换为一个单目标优化的问题。然而如果多个目标在数量级上相差较大时会造成很大的问题，一个好的做法时计算多个目标的加权和。$\mathbf{f}=(f_1,f_2,\cdots f_M)$，加权目标和为$\sum_{i=1}^{M}v_i f_i$，我们把加权目标看成新的目标,新的约束问题变成：

$$\min \sum_{i=1}^{M}v_i f_i \quad v_i \text{为常数}$$

令$\mathbf{V}=(v_1,v_2,\cdots v_M)$,我们将$V$称之为权值向量。在这里，我们面临一个问题，我们如何选择权值的值。在多目标优化中问题中，我们求得的是一个解集而不是一个单独的解。因此我们可以考虑多个权值向量，每一个权值向量会构成一个新的单目标优化问题，这样我们就把多目标优化问题分解为一系列的单目标优化问题。当然这只是一个初步的雏形，多目标优化算法中还没有这么做的。

3.1.2 参考向量

在上文中，我们提到过权值向量权值向量也可以看成是一种参考向量，RVEA 算法就是一种基于参考向量的算法。参考向量在 RVEA 算法中的作用，我们后面会提到。这这里我们只介绍均匀参考向量的生成算法。不失一般性，我们在这里及以后提到的参考向量均值单位参考向量。为了生成均匀分布的参考向量，我们采用 R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff 提到的方法[21]。首先，我们根据以下公式在超平面上生成均匀分布的参考点：

$$\begin{aligned}&\mathbf{u}_i=(u_i^1,u_i^2,\cdots,u_i^M)\\&u_i^j\in(\frac{0}{H},\frac{1}{H},\cdots\frac{H}{H}),\sum_{j=1}^{M}u_i^j=1\end{aligned}\tag{3.1}$$

公式中$\mathbf{u}_i$是超平面上的参考点，$i=1,2,\cdots N$　N表示生成的参考向量的个数，

H 是我们选取的一个正整数。根据简单的排列组合知识，我们可以算出 N 的值，$N = C_{H+M-1}^{M-1}$。根据上述的公式生成了参考点后，生成参考向量的步骤就很简单了：

$$\mathbf{v}_i = \frac{\mathbf{u}_i}{\| \mathbf{u}_i \|} \tag{3.2}$$

$\mathbf{v}_i$ 是生成的参考向量，从公式可以看得出参考向量实际上是参考点与原点组成的归一化的向量。在 $H = 5, M = 3$ 情况下，生成均匀分布的参考点如图 3-1 所示，我们可以看到生成的参考点均匀的分布在同一超平面上。

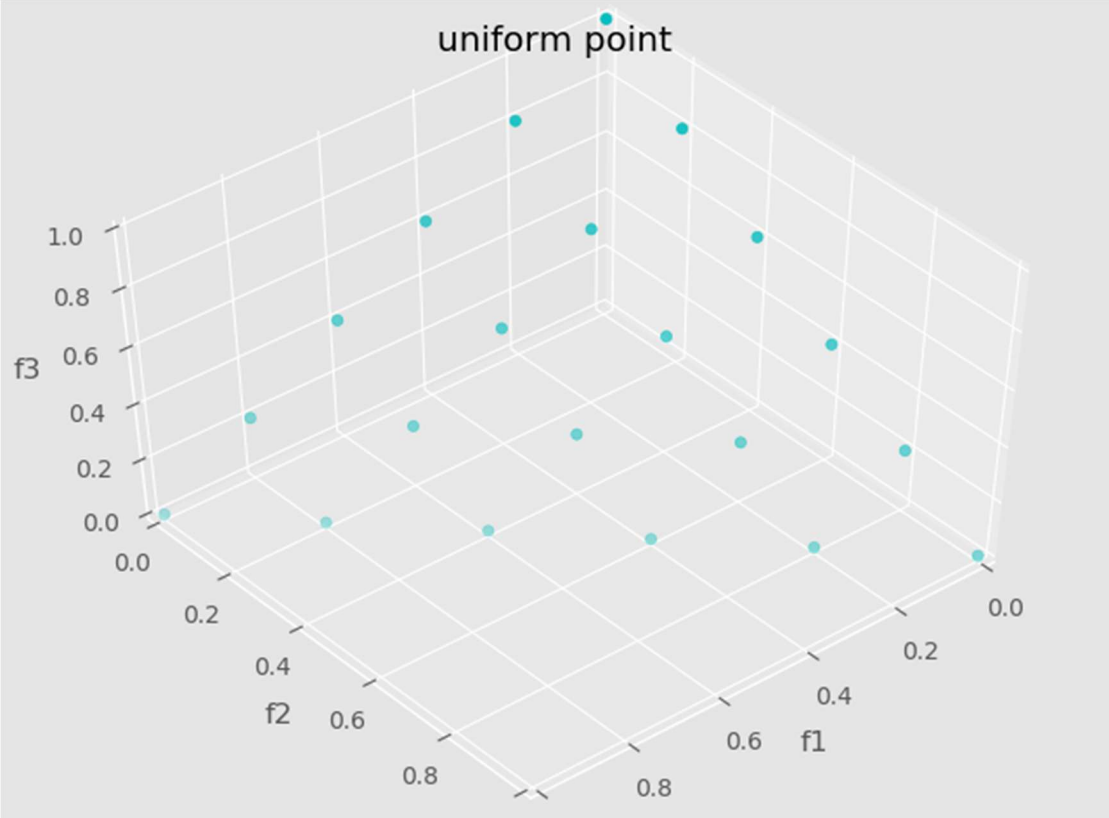

图 3-1 均匀分布的参考点图

关于参考向量生成部分的理论知识已经全部介绍完了，现在我们介绍一下参考向量生成的具体细节。令 $x_j = u_i^j H + 1$，则有：

$$x_1 + x_2 + \cdots x_M = H + M, x_j \in N^+ \tag{3.3}$$

我们求解出公式(3.3)的全部的解集，也就求出了所有的参考向量。由于 $x_i \geq 1$，所以我们可以把这个问题看成一个排列组合问题，把 $H + M$ 个人分成 M 份，求出所有的划分情形？为了直观的解释这一个问题，参见图 3-2。在图 3-2 中，正方形表示 $H + M = 7$ 个人， 其中 $M = 4$。3 个三角形把 7 个人分成了四个部分，其中的每一部分人的个数是相邻两个三角形之间正方形的个数。我们把正方形之间的空格依次编号为 $1, 2, \cdots H + M - 1$， $a_1, a_2, \cdots a_{M-1}$ 是 $1, 2, \cdots H + M - 1$ 取出 $M - 1$

个数的一种方式。显然每一组排列都对应正方形的的一组划分，即对应方程(3.3)的一组解。对于给定的一组排列方程的解由公式(3.4)给出：

$$x_i=\begin{cases}a_i & i=1\\ a_i-a_{i-1},1<i\le M-1\\ H+M-a_M\end{cases} \tag{3.4}$$

根据以上公式，只要能生成所有的排列数，就能生成均匀分布的参考点。排列数生成这一步可以调用库函数，这是一项相当简单的事情。算法 2 给出均匀分布参考点生成的算法。

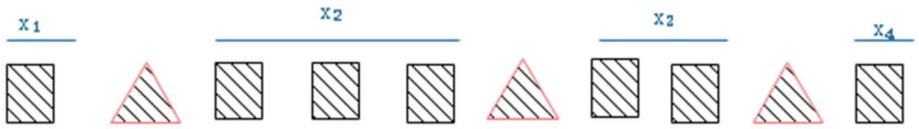

图 3-2 组合数与解的对应图

算法 2 均匀参考点生成算法

输入：目标个数 M 和控制参考点数目的 H
输出：均匀分布的参考点

1. 生成 $H+M-1$ 取 $M-1$ 点的全部组合方式集合 $\pi=\{\pi_1,\pi_2,\cdots\pi_N\}$，假设每一种组合数的排列方式都是从小到大排列的。
2. 令参考点集合为空集 $U=\varnothing$
3. **For** π_i **in** π
4. 令 $(a_1,a_2,\cdots a_{M-})=\pi_i$
5. 根据公式(3.4)计算 $(x_1,x_2,\cdots x_M)$
6. 根据 $u_i=\dfrac{x_i-1}{H}$ 计算一组解对应的一个参考点
7. $U=U\bigcup\{u\}$
8. **Endfor**
9. **Return** U

3.2 RVEA 算法

3.2.1 种群初始化

种群的初始化策略是在决策空间生成 NP 个随机分布的决策向量的， NP 表

示种群规模。初始化操作由以下公式给出：

$$x_{ij} = l_j + (u_j - l_j)rand(0,1), 1 \le i \le NP, 1 \le j \le D \tag{3.5}$$

x_{ij} 表示第 i 个个体的第 j 个决策变量的值，l_j，u_j 分别表示第 j 个决策变量的下界和上界，D 表示决策变量的数目，$rand(0,1)$ 表示生成的 $(0,1)$ 间分布的随机数。

3.2.2 子代的产生

我们在 RVEA 算法中，产生子代的操作就是差分操作。这一操作算法我们在上一章里面已经详细介绍过了，在此我们不再赘述。

3.2.3 基于参考向量的选择策略

基于参考向量的选择策略是 RVEA 算法的核心部分。我们可以把这一步骤分为 3 个小的步骤。

1) 目标值转化

在上一节中我们提到到参考向量都是分布在第一象限里面，为了与之保持一致，我们需要把目标值转化到第一象限中。令 $F_t = \{\mathbf{f}_{t1}, \mathbf{f}_{t2}, \cdots \mathbf{f}_{t|P_t|}\}$ 表示第 t 代目标值的集合，P_t 表示第 t 代种群。则目标转化可以表述为：

$$\begin{aligned} \mathbf{f}'_{t,i} &= \mathbf{f}_{t,i} - \mathbf{z}_t^{\min}, 1 \le i \le NP \\ \mathbf{z}_t^{\min} &= (z_{t,1}^{\min}, z_{t,2}^{\min}, \cdots z_{t,M}^{\min}) \end{aligned} \tag{3.6}$$

$z_{t,j}^{min}$ 代表第 t 代种群中第 j 个目标的最小值，$\mathbf{f}'_{t,i}$ 表示转换过后的目标值。显然经过转化过后的目标向量都分布在第一象限。

2) 种群的划分

目标值转换之后我们就需要把不同把种群进行划分。我们根据个体到参考向量的距离将种群划分成 N 个子种群 $\overline{P}_{t,1}\overline{P}_{t,2}, \cdots \overline{P}_{t,N}$，$N$ 表示参考向量的个数。个体中目标向量到参考向量之间的夹角可以表示为:

$$\cos\theta_{t,i,j} = \frac{\mathbf{f}'_{t,i} \cdot \mathbf{v}_{t,j}}{\| \mathbf{f}'_{t,i} \|} \tag{3.7}$$

$\theta_{t,i,j}$ 表示第 i 个个体 $I_{t,i}$ 转换的的参考向量 $f'_{t,i}$ 与参考向量 $v_{t,j}$ 之间的夹角。我们将第 t 代的个体 $I_{t,i}$ 划分到与它夹角最小的子种群中。我们把这一原则用具公式表述如下：

$$\overline{P}_{t,k} = \{I_{t,i} \mid k = \underset{j \in \{1,2,3,\cdots N\}}{argmax} \cos\ \theta_{t,i,j}\} \tag{3.8}$$

3) APD 选择

一旦种群划分成了不同的子种群，我们下一步的操作就是在每一个子种群中

选择最优的个体。选择子种群中最优个体需要兼顾两个方面：(1) 个体的转换后目标值尽可能小，我们将它称之为收敛性因子，为了简单起见我们不妨用$\|\mathbf{f}_{t,i}^{'}\|$表示。

(2)个体与参考向量的夹角$\theta_{t,ij}$尽可能的小。基于这两点考虑，我们不妨构建如下函数描述个体与参考向量间的距离:

$$d_{t,i,j}=(1+P(\theta_{t,i,j}))\cdot\|\mathbf{f}_{t,i}^{'}\|,I_{t,i}\in\overline{P}_{t,j} \tag{3.9}$$

$d_{t,i,j}$表示个体$I_{t,i}$到参考向量之间的距离，$P(\theta_{t,i,j})$是一个惩罚函数，与个体到参考向量之间的角度相关。惩罚函数的表达式由下面给出：

$$\begin{aligned}P(\theta_{t,i,j})&=M\cdot(\frac{t}{t_{max}})^{\alpha}\cdot\frac{\theta_{t,i,j}}{\gamma_{v_{t,j}}}\\ \gamma_{v_{t,j}}&=\min_{i\in\{1,2,\cdots N\}\ i\neq j}\left\langle\mathbf{v}_{t,i},\mathbf{v}_{t,j}\right\rangle\end{aligned} \tag{3.10}$$

$\gamma_{\mathbf{v}_{t,j}}$表示与参考向量$\mathbf{v}_{t,j}$与其他参考向量间角度的最小值最小值，$t_{max}$表示最大的迭代代数，$M$表示目标的个数，$\alpha$是一个参数。构建上述形式的惩罚函数是基于以下几点考虑的：(1) 在多目标优化中，我们需要很好的平衡种群的收敛性与多样性，稀疏性和收敛性在迭代的过程中保持固定的优先级时不明智的。因此在初期收敛性因子（$\|\mathbf{f}_{t,i}^{'}\|$）拥有较高的选择优先级，渐渐的个体与参考向量之间的夹角在选择的过程中发挥了越来越重要的作用。特别的，$t=0$时，只有$\|\mathbf{f}_{t,i}^{'}\|$对选择起作用（2）由于候选解的稀疏性依赖于目标的维数M，所以构建惩罚因子的时候加上目标个数M这个因子 （3）考虑到$\theta_{t,i,j}$受到参考向量数目影响较大，所以我们用$\gamma_{\mathbf{v}_{t,i,j}}$归一化角度因子。

前面就是我们介绍的基于参考向量选择策略的全部内容。大家可能会产生一个疑问，目标值转换一步为什么不是归一化？在大多数情况下，归一化能很好的消除不同目标量纲的影响。然而，实际的经验证明，归一化并不适合 RVEA 算法框架。基于参考向量选择部分的算法如下：

算法 3 基于参考向量的选择算法

输入：第t代种群P_t和单位参考向量集合$V_t=\{\mathbf{v}_{t,1},\mathbf{v}_{t,2},\cdots\mathbf{v}_{t,N}\}$

输出：第$t+1$代选择的种群P_{t+1}

1. # 目标值转化
2. 计算第t代每一个目标的最小值$\mathbf{z}_t^{\min}$
3. **For** i = 1 **to** $|P_t|$
4. $\quad\mathbf{f}_{t,i}^{'}=\mathbf{f}_{t,i}-\mathbf{z}_t^{\min}$
5. **Endfor**
6. # 种群划分

7. **For** i = 1 **to** $|P_t|$
8. **For** j = 1 **to** N
9. $\cos\theta_{t,i,j} = \frac{\mathbf{f}_{t,i}^{'} \cdot \mathbf{v}_{t,j}}{\| \mathbf{f}_{t,i}^{'} \|}$
10. **Endfor**
11. **Endfor**
12. **For** i =1 **to** $|P_t|$
13. $k = \underset{j\in\{1,2,3,\cdots N\}}{argmax} \cos\ \theta_{t,i,j}$
14. $\overline{P_{t,k}} = \overline{P_{t,k}} \bigcup \{\mathrm{I}_{t,k}\}$
15. **Endfor**
16. # APD 计算
17. **For** i = 1 **to**
18. **For** j = 1 **to** N
19. $d_{t,i,j} = (1 + P(\theta_{t,i,j})) \cdot \| \mathbf{f}_{t,i}^{'} \|, I_{t,i} \in \overline{P}_{t,j}$
20. **Endfor**
21. **Endfor**
22. # 选择 APD 最小的个体
23. $P_{t+1} = \varnothing$
24. **For** j= 1 **to** N
25. $k = \underset{i\in\{1,2,\cdots \overline{P}_{t,j}\}}{argmin}\ d_{t,i,j}$
26. $P_{t+1} = P_{t+1} \bigcup \{I_{t,k}\}$
27. **Endfor**
28. **Return** P_{t+1}

3.2.4 参考向量适应

给出均与分布的参考向量，最终得到的解也是均匀分布的。这一愿望看似挺美好的，然而这只对均匀分布的 PF 有效。实际上我们会遇到一些多目标优化问题的 PF 是非均匀分布的。对于非均匀分布的 PF，我们不得不把目标归一化。上文中，我们已经提到过归一化的方法并不适合 RVEA 算法框架。因此，我们考虑调整参考向量。调整参考向量，我们采取的策略是根据目标值的范围缩放参考向量。用公式描述如下：

$$\begin{aligned} &\mathbf{u}_{t,i} = (u_{t,i,1}, u_{t,i,2}, \ldots u_{t,i,M}) \\ &u_{t+1,i,j} = v_{0,i}(z_{t,i,j}^{\max} - z_{t,i,j}^{\min}), 1 \le i \le N, 1 \le j \le M \\ &\mathbf{v}_{t+1,i} = \frac{\mathbf{u}_{t+1,i}}{\| \mathbf{u}_{t+1,i} \|} \end{aligned} \tag{3.11}$$

式中$\mathbf{u}_{t+1,i}$代表第$t+1$代第i个未归一化参考向量，N代表参考向量的个数，M代表目标的个数。为了说明这种归一化的策略的确效，我们给出目标个数为 2 的情形的示意图。在图 3-3 中，实线代表问题 PF，带有箭头的虚线代表参考向量，实心的红点代表基于算法求解得到的帕累托优化解。第一幅图是均匀的参考向量求解的帕累托解，第二幅图是均匀分布的参考向量加入参考向量的适应求解出的帕累托解。从图中我们可以很明显的看出，加入了参考向量适应得到的解分布更加的均匀，从而说明我们的参考向量适应策略是有效的。

然而根据 Giagkiozis 的实验结论[22]，参考向量不应该更新太过于频繁，否则算法收敛不稳定。因此我们可以加一个控制参考数fr 控制参考向量更新的频率。参考向量更新算法比较简单，我们就不给出伪代码了。至此，基本的 RVEA 算法已经全部介绍完了。

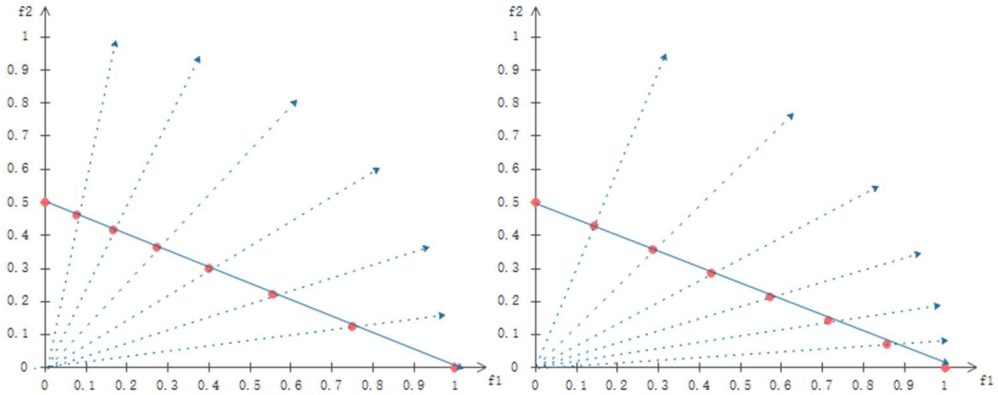

图 3-3 均匀分布参考向量和自适应参考向量对比图

3.3 RVEA 算法求解多目标优化问题

3.3.1 RVEA 正确性验证以及参数设置

1) 算法正确性验证

为了验证算法的正确性，我们以 DTLZ2 问题为例。为了方便描述，我们给出 DTLZ2 问题的一般表达式[23]：

$$\begin{cases}\min f_1(\mathbf{x}) = (1+g(\mathbf{x}_{\mathrm{M}}))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{\mathrm{M-2}}\pi/2)\cos(x_{\mathrm{M-1}}-\pi/2)\\ \min f_2(\mathbf{x}) = (1+g(\mathbf{x}_{\mathrm{M}}))\cos(x_1\pi/2)\cos(x_2\pi/2)\cdots\cos(x_{\mathrm{M-2}}\pi/2)\sin(x_{\mathrm{M-1}}-\pi/2)\\ \vdots\\ \min f_M(\mathbf{x}) = (1+g(\mathbf{x}_{\mathrm{M}}))\sin(x_1\pi/2)\end{cases}$$

其中$0<x_i<1, i=1,\cdots D$

$$g(\mathbf{x}_{\mathrm{M}})=\sum_{x_i\in x_M}(x_i-0.5)^2$$

式中f_1，f_2，f_M表示M 个目标值。D表示决策变量的个数为了统一标准，我们采用与文献相同的表述方式。$\mathbf{x}_M=\{x_M,x_{M+1},\cdots x_D\}$

在测试该问题时，我们取$D=12$，$M=3$。RVEA 算法中参数选取$\alpha=2$，最大迭代次数$t_{max}=500$，参考向量个数$N=105$,初始种群规模$NP=105$。我们画出 RVEA 解决 3 目标 DTLZ2 求解演化图见图 3 -4。图中红色的点代表 DTLZ2 理论上的 PF 值，蓝色的点代表 RVEA 算法求解得到的种群目标值。从图中可以看出开始时种群目标值几乎随机分布在整个空间，后来慢慢靠近 PF。第 60 代，几乎所有的点都分布在 PF 上，但分布不均匀。到了 100 代以后，所有的点都比较均匀的分布在 PF 上。根据文献[23]描述，DTLZ2 的 PF 为一个球面，与我们得到的结果类似。由此可以证明我们的 RVEA 算法是正确的。

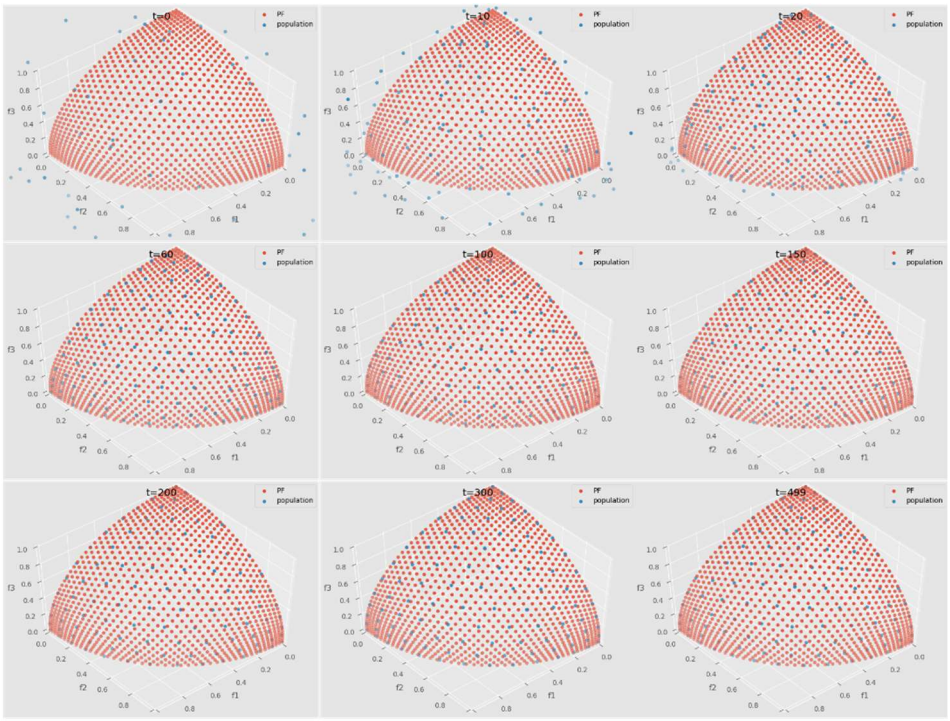

图 3 -4 RVEA 算法在 DTLZ2 上的演化图

2) 超参数设置

RVEA 算法的参数可以分为差分操作部分的参数和算法本身的参数。差分操

作部分的参数我们与上一章保持一致：即 $CR=0.8, F=0.5$。自身的参数有 α，最大迭代次数 t_{max}，参考向量的数目 N,参考向量更新频率 fr 还是以上述的 DTLZ2 问题为例。根据文献[18]，我们设置参考参考向量的数目 $N=105$，$NP=105$，$\alpha=2, fr=0.1$。在计算 HV 时，我们取参考点为 $(2.0,2.0,2.0)$ 画出进化 1000 代时，HV 值随迭代次数间的变化曲线如图 3-5，在图中，我们可以看出迭代大概进化 200 代后 HV 的值提升很缓慢，由于 DTLZ2 问题本身比较简单，故收敛比较快，其他的一些比较复杂的多目标优化问题可能需要更多的迭代次数才能收敛。考虑到演化代数的增加能提升 HV 同时也能增加计算成本，我们不妨将迭代次数稍微提升一些：即将最大迭代次数设置为 500。所以如不特殊说明，我们采用 $t_{\max}=500$。在 RVEA 算法中 α 和 fr 参数主要影响算法的演化速率和结果的稳定性。大量的实验显示设置 $\alpha=2$ 和 $fr=0.1$ 效果较好，所以不加说明，以后 RVEA 算法设置的参数 $\alpha=2, fr=0.1, t_{max}=500$。

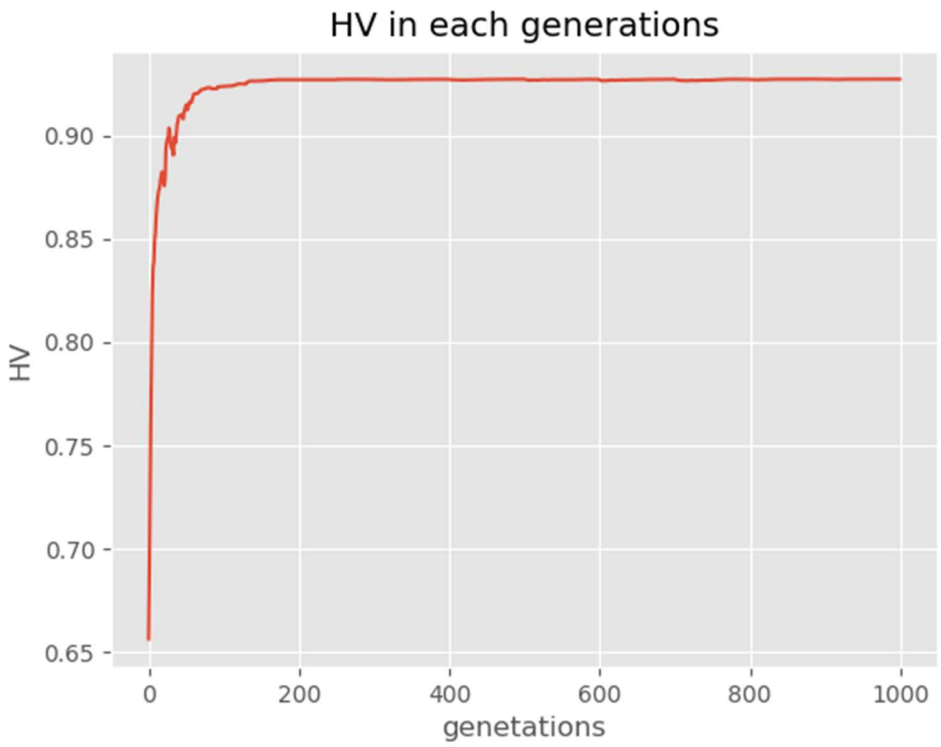

图 3-5 HV 随着迭代次数变化曲线

3.3.2 RVEA 在多目标优化问题中的表现

我们选择测试的问题时 DTLZ 系列问题[23]，评价指标选择的是 HV 指标。根据 DTLZ 问题测试建议：决策变量个数 $D=M+K-1$，其中 M 为目标数，对于 DTLZ1 问题 $K=5$，对于 DTLZ2，DTLZ3，DTLZ4 问题，$K=10$。对于 DTLZ1 问题我们选择的参考点为 $(1.5,1.5,\cdots 1.5)$，对于 DTLZ2、DTLZ3、DTLZ4 我们的

参考点选择为(2.0,2.0,…2.0)。在本文中，测试问题目标数分别选择 3、6、8、10，其对应的种群规模分别为 105、132、156、275。由于 DTLZ1 问题收敛较慢，我们设置 DTLZ1 最大迭代次数为 1000。RVEA 算法在 DTLZ1-4 问题中的表现参见表 3-1。表中 RVEA 算法的结果是运行 10 次的平均值，NSGIII 和 MOEA/DD 算法的结果是摘自参考文献[18]。从表格可以看出，RVEA 算法略差于 MOEA/DD 算法，且几乎在所有的测试问题都优于 NSGAIII。

表 3-1RVEA 算法与其他多目标优化算法 HV 比较表

算法 / 问题	目标数 M	RVEA	NSGAIII	MOEA/DD
DTLZ1	3	0.992305	0.992275	**0.992321**
	6	0.999850	0.999951	**0.999965**
	8	0.999950	0.999993	**0.999984**
	10	0.999960	0.999992	**0.999995**
DTLZ2	3	0.927039	0.927012	**0.927292**
	6	0.995210	0.995689	**0.996096**
	8	0.998800	0.998223	**0.999330**
	10	0.999220	0.999804	0.999920
DTLZ3	3	0.926633	0.95650	**0.926921**
	6	0.993860	0.768089	**0.995848**
	8	**0.999690**	0.683011	0.999338
	10	0.997100	0.588230	**0.999914**
DTLZ4	3	0.927026	0.926947	**0.927295**
	6	**0.996160**	0.993353	0.995972
	8	0.993300	0.999016	**0.999374**
	10	**0.999940**	0.999844	0.999916

第四章 RVEA 求解约束多目标优化问题

在求解约束多目标优化问题时，我们不得不考虑一个问题：我们应该如何处理违约条件？常用的处理约束优化问题的方法可以分为可行解规则，随机排序，ε 约束方法，构建惩罚函数，以及将将违约条件转换为额外的目标方法等[24]。这些方法都不是孤立的，可以结合使用。在这一章中，我们主要介绍两个相关算法：1) 归一化一阶违约值处理的 RVEA 算法 2）基于动态边界的 RVEA 算法。

4.1 归一化违约值处理的 RVEA 算法

在第二章中，我们看到了等式约束和不等式约束可以转换成相同的形式，在后文中。我们的多目标优化问题统一采取下面形式表述：

$$\begin{aligned}&\min \mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),f_2(\mathbf{x}),\cdots f_M(\mathbf{x}))\\&\mathbf{g}(\mathbf{x})=(g_1(\mathbf{x}),g_2(\mathbf{x}),\cdots g_m(\mathbf{x}))\le\mathbf{0}\\&\mathbf{X}=\{\mathbf{x}\,|\,\mathbf{l}<\mathbf{x}<\mathbf{u}\}\end{aligned}\tag{4.1}$$

式中 m 表示约束条件的个数。我们构建一阶违约值如下：

$$\begin{aligned}&G_i(\mathbf{x})=\max\{g_i(\mathbf{x}),0\},i=0,1,\cdots m\\&cv(\mathbf{x})=\frac{1}{m}\sum_{i=1}^{m}\frac{G_i(\mathbf{x})}{\max\limits_{x\in P(0)}\{G_i(\mathbf{x})\}}\end{aligned}\tag{4.2}$$

$cv(\mathbf{x})$ 是归一化后一阶违约值，$P(0)$ 是初始化种群。我们采取初始化种群的最大违约值归一化，这样做能保证在进化过程中 $cv(\mathbf{x})$ 值收敛。在原始的 RVEA 算法处理约束时，违约值的处理未加归一化处理。这种处理违约条件的方法，在不同约束条件下违约值的数量级相同或者相近的情况下才能成立。而我们所采用的归一化违约值处理的方法可以保证，我们计算的 $cv(\mathbf{x})$ 不受违约值数量级的影响。

下面，我们介绍如何在 RVEA 算法种加入归一化一阶违约值。毫无疑问，违约值的处理应当加在 RVEA 算法的选择策略种。这样我们就有两种添加违约值的方法： 一种就是在种群划分之前加上违约值处理，另一种就是在种群划分以后再加违约值处理。由于 RVEA 算法是基于参考向量的选择算法，所以在种群划分之前加入违约值处理，只能采取固定选择子代数目的做法：当可行解的数目超过需要选择子代数目时，按照无约束 RVEA 算法在可行解种进行选择；当可行解

的数目小于需要选择的子代时，选择所有的可行解，并在不可行解中选择 $cv(x)$ 小的个体。这样做的坏处时，当不可行解较多时，RVEA 算法基本是按照违约值进行选择，参考向量不起任何的作用。因此我们采用的算法是在划分种群后加入违约值。具体的做法是：在种群划分子种群 $\overline{P}_{t,i}$ 中，如果存在可行解，则在所有的可行解中按照 APD 策略选择子代个体；如果所有的解都是不可行的，则选择 $cv(\mathbf{x})$ 值最小的个体作为子代。具体算法参见算法 4。

算法 4 约束 APD 选择规则

输入：已经划分好的一个子种群 $\overline{P}_{t,j}$ 以及其一阶违约值 $cv(\mathbf{x})$

输出：选择的下一代个体 $I_{t+1,j}$

1. # $I_{t,k}$ 表示第 t 代的第 k 个个体， $x_{t,k}$ 是 $I_{t,k}$ 对应的决策向量
2. $S=\varnothing$ #表示可行解集
3. **For** $I_{t,k}$ **in** $\overline{P}_{t,j}$
4. **If** $cv(\mathbf{x}_{t,k})=0$
5. $S=S\bigcup\{I_{t,k}\}$
6. **Endif**
7. **Endfor**
8. **If** $S=\varnothing$ #全部都是不可行解
9. $k=\underset{I_{t,i}\in\overline{P}_{t,j}}{argmin}\ cv(\mathbf{x}_{t,i})$
10. **Else**: #存在可行解
11. $k=\underset{I_{t,i}\in S}{argmin}\ d_{t,i,j}$ # $d_{t,i,j}$ 计算参见算法 3
12. **Endif**
13. $I_{t+1,j}=I_{t,k}$
14. **Return** $I_{t+1,j}$

4.2 动态约束边界的 RVEA 算法

考虑到一些约束优化问题含有等式约束条件，这类优化问题通常难以约束。这时候如果采用上一节中那种固定模式处理约束，则再整个迭代的过程中。主要

是违约值起作用，APD 选择不起任何作用。因此我们有了这一节中的动态约束边界的 RVEA 算法。

4.2.1 ε 可行解

在处理难约束优化问题时，不妨松弛约束约束条件。松弛后的问题转化为：

$$\begin{aligned}&\min \mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),f_2(\mathbf{x}),\cdots f_M(\mathbf{x}))\\&\mathbf{g}(\mathbf{x})=(g_1(\mathbf{x}),g_2(\mathbf{x}),\cdots g_m(\mathbf{x}))\le\boldsymbol{\varepsilon}\\&\mathbf{X}=\{\mathbf{x}\mid\mathbf{l}<\mathbf{x}<\mathbf{u}\}\\&\boldsymbol{\varepsilon}=(\varepsilon_1,\varepsilon_2,\cdots\varepsilon_m)\end{aligned}\tag{4.3}$$

其中 $\varepsilon_i>0,i=1,2,\cdots m$，我们将 ε 称之为松弛因子。满足公式(4.3)约束条件的解我们称之为 ε 可行解。

4.2.2 Niche-count 计算

RVEA 算法是根据参考向量进行目标值进行选择后代的，从种群的目标值维度上能一定程度的保证种群的多样性。然而在解空间上，决策向量的多样性却并没有得到保证。根据 Goldberg 和 Richardson 的论文，Niche-count 在阻止陷入局部最优和保证陷入局部最优有着重要的作用[25]。Niche-count 定义:

$$\begin{aligned}&nc(\mathbf{x}\mid\sigma)=\sum_{\mathbf{x}_i\in P_t,\mathbf{x}_i\neq\mathbf{x}}sh(\mathbf{x},\mathbf{x}_i)\\&\mathbf{x}\in P_t\end{aligned}\tag{4.4}$$

$nc(x\mid\sigma)$ 表示决策向量 x 的 Niche-count，P_t 表示第 t 代种群。$sh(x_i,x_j)$ 表示共享函数，定义为：

$$sh(\mathbf{x}_1,\mathbf{x}_2)=\begin{cases}1-\left(\dfrac{d(\mathbf{x}_1,\mathbf{x}_2)}{\sigma}\right), d(x_1,x_2)<\sigma\\0\qquad\qquad\quad ,otherwise\end{cases}\tag{4.5}$$

其中 $d(\mathbf{x}_1,\mathbf{x}_2)$ 表示个体 $\mathbf{x}_1,\mathbf{x}_2$ 的欧式距离。我们根据 Niche-count 的定义式可以看出：个体分布越稀疏，Niche-count 越小，因此 Niche-count 能在一定程度上反映种群决策向量的多样性。

4.2.3 动态约束边界多目标优化问题

我们综合 ε 可行解，Niche-count，还有上一节提到的归一化违约值的条件。可以构造：

$$\begin{aligned}&\min \mathbf{f}(\mathbf{x})=(f_1(\mathbf{x}),f_2(\mathbf{x}),\cdots f_M(\mathbf{x}),cv(\mathbf{x}),nc(\mathbf{x}\mid\sigma^{(s)}))\\&\mathbf{g}(\mathbf{x})=(g_1(\mathbf{x}),g_2(\mathbf{x}),\cdots g_m(\mathbf{x}))\le\boldsymbol{\varepsilon}^{(s)}\\&\mathbf{X}=\{\mathbf{x}\mid\mathbf{l}<\mathbf{x}<\mathbf{u}\}\end{aligned}\tag{4.6}$$

我们将公式(4.6)称之为动态约束多目标优化问题。式中 $\boldsymbol{\varepsilon}^{(s)}=(\varepsilon_1^{(s)},\varepsilon_2^{(s)},\cdots\varepsilon_m^{(s)})$，表示动态约束边界，满足 $\boldsymbol{\varepsilon}^{(0)}>\boldsymbol{\varepsilon}^{(1)}\cdots>\boldsymbol{\varepsilon}^{(S)}=\mathbf{0}$。$\sigma^{(s)}$ 表示动态 Niche-count 半径，

满足$\sigma^{(0)} > \sigma^{(1)} \cdots > \sigma^{(S)} = 0$。$s$ 表示状态变量，$s = 1,2,\cdots S$，S 表示最大状态改变次数。

比较原始多目标优化问题和动态边界约束多目标优化问题，可以轻易的看出当$cv(\mathbf{x}) = 0$和$nc(\mathbf{x} \mid \sigma^{(S)}) = 0$时两个问题等价。

4.2.4 动态约束边界选择规则

1) 动态约束边界 RVEA 选择规则

考虑动态约束边界的多目标优化问题比原始问题多了 Niche-count 和违约值两个目标，而且 Niche-count 半径和违约值边界都在收缩，所以在迭代的后期原始 RVEA 算法基于参考向量的选择规则选择的个体较少，导致优化结果较差。为了避免选择个体数较少，可以采用帕累托占优分层的思想将种群进行非支配排序，分为第一层、第二层等等，依次从每一层中选择全部的个体直到选择的个体数大于或等于需要选择的个体数。如果选择的个体数恰好是所需选择个体数，则选择步骤完成；如果选择的个体数大于所需选择的个体数，则在最后一层用基于密度的 RVEA 选择规则选择一定数量的个体（前面各层个体全部选择）使得总选择个体数与所需选择的个体数一致。算法描述见算法 5。

算法 5 动态约束边界 RVEA 选择规则

输入：种群P，所需选择的个体数NP

输出：选择的下一代种群$\overline{\mathrm{P}}$

```
1   P̄ = ∅ ， i = 1
2   非支配排序将种群 P 分层得到(F_1, F_2, ⋯)
3   While |P̄| + |F_i| ≤ NP
4           P̄ = P̄ ⋃ F_i
5           i = i + 1
6   Endwhile
7   l = i
8   If |P̄| < NP
9           K = NP − |P̄|
10          #参见算法 7 和算法 8
            根据基于密度的 RVEA 选择规则在 F_l 层选择 K 个个体构成 S_l
11          P̄ = P̄ ⋃ S_l
12  Endif
13  Retutn P̄
```

2) 非支配排序

在给出非支配排序前，先给出ε可行解下的非支配解的定义：

a) 如果解S_1是ε可行解，如果在集合中存在解$S_2(S_2 \neq S_1)$为ε可行解且S_2的每一个目标都优于S_1，则S_1是支配解，反之则是非支配解。

b) 如果解S_1是ε不可行解，如果存在$S_2(S_2 \neq S_1)$是可行解或者$S_2(S_2 \neq S_1)$是不可行解且S_2的一阶违约值小于S_1，则S_1为支配解，反之S_1是非支配解。

非支配排序就是递归寻找所有非支配解集将原始种群分层的算法，算法描述见算法 6。

算法 6 非支配排序算法

输入：种群P

输出：非支配排序分层结果$(F_1, F_2, \cdots)$

1. $S = P$， $i = 1$
2. **While** $S \neq \varnothing$
3. $F_i = \varnothing$
4. **For** s **in** S #对于解集中的每一个解
5. **If** s 是非支配解
6. $F_i = F_i \bigcup \{s\}$
7. **Endif**
8. **Endfor** $S = S / F_i$
9. $i = i + 1$
10. **Endwhile**
11. **Return** $(F_1, F_2, \cdot)$

3) 基于密度的 RVEA 选择规则

动态约束边界优先选择占优关系较佳的个体，只在F_l层根据参考向量选择一定数量的个体。如果采用原始 RVEA 算法的选择规则在F_l层选择K个个体，这样前面$l-1$层选择的个体信息未被利用，导致种群多样性不佳。将前面$l-1$层个体与参考向量关联，有的参考向量与一个个体关联，有的与多个个体关联，有的与零个个体关联。不妨将参考向量i与之关联的个体数记作ρ_i，称之为参考向量的密度。显然在F_l根据原始 RVEA 选择的个体数目大于所需选择的个体数时，优先考虑选择与个体关联参考向量密度小的个体。计算参考向量密度的算法见算法 7，根据密度选择的算法算法 8。

算法 7 计算参考向量密度

输入：前$l-1$层构成的种群P和单位参考向量集合$V=\{\mathbf{v}_1,\mathbf{v}_2,\cdots\mathbf{v}_N\}$

输出：参考向量的密度$(\rho_1,\rho_2,\cdots,\rho_N)$

1. 计算种群每一个目标的最小值$Z^{\min}$
2. **For** i = 1 **to** $|P|$
3. $f_i^{'}=f_i-z_t$
4. **Endfor**
5. # 种群划分
6. **For** i = 1 **to** $|P|$
7. **For** j = 1 **to** N
8. $\cos\theta_{i,j}=\dfrac{f_i^{'}\cdot v_j}{\|f_i^{'}\|}$
9. **Endfor**
10. **Endfor**
11. $\rho_1=\rho_2=\cdots=\rho_N=0$
12. **For** i =1 **to** $|P|$
13. $k=\underset{j\in\{1,2,3,\cdots N\}}{argmax}\cos\ \theta_{i,j}$
14. $\rho_k=\rho_k+1$
15. **Endfor**
16. **Return** $(\rho_1,\rho_2,\cdots,\rho_N)$

算法 8 基于密度的选择规则

输入：第l代种群F_l，单位参考向量集合$V=\{v_1,v_2,\cdots v_N\}$，需要选择的个体数K和参考向量的密度$(\rho_1,\rho_2,\cdots,\rho_N)$

输出：第l层选择的K个个体S_l

目标值转化

1. 计算F_l每一个目标的最小值$Z^{\min}$
2. **For** i = 1 **to** $|F_l|$
3. $\mathbf{f}_i^{'}=\mathbf{f}_i-\mathbf{z}^{\min}$
4. **Endfor**
5. # 种群划分
6. **For** i = 1 **to** $|P_t|$
7. **For** j = 1 **to** N
8. $\cos\theta_{i,j}=\dfrac{f_i^{'}\cdot v_j}{\|f_i^{'}\|}$
9. **Endfor**
10. **Endfor**

11. $\overline{P_1} = \overline{P_2} = \cdots \overline{P_N} = \varnothing$ #子种群集
12. **For** i =1 **to** $|F_l|$
13. $k = \underset{j\in\{1,2,3,\cdots N\}}{argmax} \cos\ \theta_{i,j}$
14. $\overline{P_k} = \overline{P_k} \bigcup \{\mathrm{I}_k\}$
15. **Endfor**
16. # APD 计算
17. **For** i = 1 **to** $|F_l|$
18. **For** j = 1 **to** N
19. $d_{t,i,j} = (1 + P(\theta_{i,j})) \cdot \| f_i^{'} \|, I_{t,i} \in \overline{P}_{t,j}$
20. **Endfor**
21. **Endfor**
22. # 选择 APD 最小的个体
23. $S = \varnothing$
24. $\phi = \varnothing$ #记录参考向量密度信息
25. **For** j= 1 **to** N
26. $k = \underset{i\in\{1,2,\cdots \overline{P}_{t,j}\}}{argmin}\ d_{t,i,j}$
27. $S = S \bigcup \{I_{t,k}\}$
28. $\phi = \phi \bigcup \rho_k$
29. **Endfor**
30. **If** $|S| = K$
31. $S_l = S$
32. **Elseif** $|S| > K$
33. 将种群 S 按照参考向量密度排序选择密度小的 K 个个体构成 S_l
34. **Else**
35. 选择全部的 S 以及在 F_l 中未被选择的个体中选择一定数目的个体构成 S_l
36. **Endif**
37. **Return** S_l

4.2.5 动态约束边界的 RVEA 算法

1) $\boldsymbol{\varepsilon}^{(s)}$，$\sigma^{(s)}$ 更新规则

受模拟退火算法的启示，我们采用衰减的指数函数最为 $\varepsilon^{(s)}$ 和 $\sigma^{(s)}$ 的更新规则，表达式为：

$$\begin{aligned}\varepsilon_i^{(s)} &= A_i e^{-(\frac{s}{B_i})^2} - \delta, i = 1,2,\cdots m \\ \sigma^{(s)} &= Ce^{-(\frac{s}{D})^2} - \delta\end{aligned} \tag{4.7}$$

A_i，B_i，C，D 都是参数，δ 为常数，在本文中，$\delta = 1e-8$。为了让初始化种群中的所有个体都满足约束条件，我们将初始化种群中最大违约值作为约束边界，即 $\varepsilon_i^{(0)} = \max_{x \in P(0)} G_i(x)$。最终的收缩边界 $\varepsilon_i^{(S)} = 0$，将 $s = 0$ 和 $s = S$ 代入公式(4.7)：

$$\begin{cases}\varepsilon_i^{(0)} = A_i - \delta, s = 0 \\ 0 = A_i e^{-(\frac{s}{B_i})^2} - \delta, s = S\end{cases} \tag{4.8}$$

其中 $i = 1,2,\cdots m$，求解公式(4.8)得到 A_i，B_i 的值：

$$\begin{cases}A_i = \varepsilon_i^{(0)} + \delta \\ B_i = \dfrac{S}{\sqrt{\ln(\frac{\varepsilon_i^{(0)} + \delta}{\delta})}}\end{cases} \tag{4.9}$$

同理，在已知 $\sigma^{(0)}$ 的情况下，C 和 D 的表达式由公式(4.10)给出：

$$\begin{cases}C = \sigma^{(0)} + \delta \\ D = \dfrac{S}{\sqrt{\ln(\frac{\sigma^{(0)} + \delta}{\delta})}}\end{cases} \tag{4.10}$$

初始化 Niche-count 半径由公式(4.11) 给出：

$$\sigma^{(0)} = \frac{1}{2}\sqrt[D]{(2D\prod_{i=1}^{D}(u_i - l_i)) / L\pi}, i = 1,2,\cdots D \tag{4.11}$$

式中，L 表示解的个数，l_i、u_i 分别表示决策变量的下确界和上确界，D 表示决策向量的个数。

2) 动态边界 RVEA 算法框架

动态边界收缩 RVEA 是基于 ε 可行规则收缩约束边界的，如果种群所有的解都是 ε 可行解，则 $s = s + 1$，约束边界收缩，Niche-count 半径 σ 也收缩，反之，则不收缩。直到达到最大的状态数 S 或者最大迭代次数 $t_{\max}$。动态 RVEA 算法的伪代码见算法 9。

算法 9 动态边界 RVEA 算法框架

输入：最大的迭代次数 $t_{\max}$，最大状态数 S

输出：最后一代的种群 P_{last}

初始化：创建初始化种群 P_0，设置初始化约束边界 $\varepsilon=\varepsilon^{(0)}$、初始化 Niche-count 半径 $\sigma=\sigma^{(0)}$、$t=0$ 和 $s=0$

1. **While** $t<t_{\max}$
2. **If** $\forall I_{t,k}\in P_t, g(\mathbf{x}_{t,k})\le\varepsilon$ #种群个体都满足 ε 可行
3. $s=s+1$
4. $\boldsymbol{\varepsilon}=\boldsymbol{\varepsilon}^{(s)}$
5. $\sigma=\sigma^{(s)}$
6. **Endif**
7. 由 P_t 作为父代产生子代种群 Q_t
8. 根据动态 RVEA 选择规则在 $P_t\bigcup Q_t$ 中选择下一代种 群 P_{t+1} #参见算法 5
9. $t=t+1$
10. **If** $s\ge S$
11. **Break**
12. **Endif**
13. **Endwhile**
14. $P_{last}=P_t$
15. **Return** P_{last}

4.3 约束优化问题测试

4.3.1 归一化违约值 RVEA 算法测试

我们以归一化一阶违约值的 RVEA 算法作为测试算法。测试问题我们选择 CDTLZ 系列问题，参数设置同前一章。

归一化一阶违约值 RVEA 算法在 3 维 C2DTLZ2 问题上得到的种群见图 4-1，图中蓝色的点是生成的种群的目标值，红色的点是半径为 1 的 1/8 球面上均匀分布的点。从图中可以看出蓝色的点比较均匀的分布在 1/8 球面上的四个区域，其中一个是 1/8 球面的面心附近，另外三个是 1/8 球面与坐标轴围成的角落里，这与 C2DTLZ2 问题的 PF 十分接近，足以证明归一化一阶违约值的 RVEA 算法的有效性。其他问题的测试结果见表 4-1，表格中 cRVEA 测试数据来源于文献[18]，从表格可以看出归一化一阶违约值的 RVEA 算法在部分测试问题优于原始的约束 RVEA 算法(cRVEA)。如果我们测试的问题违约值不是同一数量级，则这种差

异应该更加的明显。

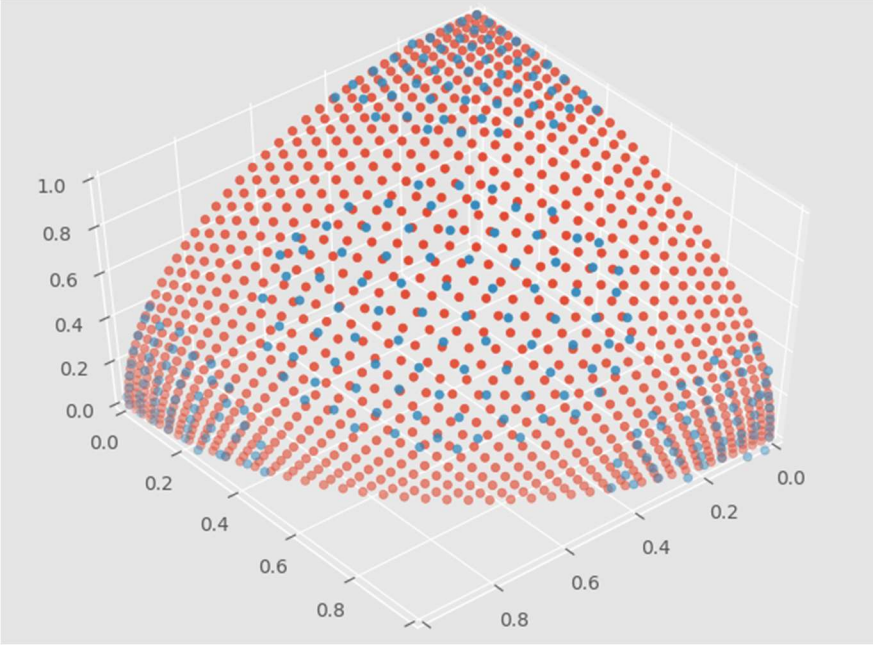

图 4-1 归一化违约值 RVEA 算法在 C2DTLZ2 问题生成的结果

表 4-1 归一化 RVEA 算法与 cRVEA 算法比较

算法 / 问题	目标数 M	归一化违约值 RVEA 算法	cRVEA
C1DTLZ1	3	**0.992300**	0.990636
	6	**0.999820**	0.998719
	8	**0.999880**	0.998628
	10	0.999910	0.998613
C2DTLZ2	3	**0.918659**	0.91574
	6	0.993780	**0.993848**
	8	**0.998870**	0.998513
	10	0.999620	**0.999746**
C3DTLZ4	3	0.918302	**0.918766**
	6	0.99326	**0.996745**
	8	**0.999530**	0.999456
	10	0.999572	**0.999971**

4.3.2 动态约束边界 RVEA 算法测试

因为动态边界 RVEA 算法能 Niche-count 和违约值 两个额外的目标，因此我们可以将单目标优化问题转换成多目标优化问题，这样就可以用动态边界 RVEA 算法求解了。在测试动态约束边界 RVEA 算法时，我们采用 IEEE 2006 CEC 问题集[26]，设置最大环境数 $S=2000$ ，最大迭代次数 $t_{max}=10000$,RVEA 算

法其他参数的设置同上一章。

25 次运行统计结果参见表 4-2，从表格可以看出除了 g08 和 g23 问题的最坏结果较差外其他的最优结果均与已知最优结果十分接近，而且方差值比较小(g20 问题未发现有效解)。由此可以说明动态约束边界 RVEA 算法将 Niche-count 和一阶违约值做为额外的目标是有效的。

表 4-2 动态约束边界 RVEA 算法

问题	已知最优	动态约束边界 RVEA 算法			
		最坏结果	最优结果	平均值	方差
g01	-15	-15	15	-15	1.05E-09
g02	-0.8036191	-0.803619	-0.8036191	-0.8036191	2.00E-08
g03	-1.0005001	-1.0005001	-1.0005001	-1.0005001	5.54E-10
g04	-30665.539	-30665.539	-30665.539	-30665.539	3.83E-08
g05	5126.4967	5126.49671	5126.49671	5126.49671	1.31E-10
g06	-6961.81387	-6961.8139	-6961.8139	-6961.8139	8.90E-08
g07	24.3062	24.3062092	24.3062092	24.3062483	7.34E-05
g08	-0.09582504	-0.0258123	-0.095825	0.03143565	3.14E-02
g09	680.63	680.630057	680.630057	680.630057	1.34E-10
g10	7049.248	7049.25777	7049.24811	7049.25099	2.85E-03
g11	0.7499	0.7499	0.7499	0.7499	1.90E-12
g12	-1	-1	-1	-1	0
g13	0.05394115	0.05394151	0.05394151	0.05394151	8.49E-13
g14	-47.764888	-47.764743	-47.764888	-47.764877	2.85E-05
g15	961.715	967.999886	961.715022	962.510081	1.98E+00
g16	-1.9051553	-1.9051553	-1.9051553	-1.9051553	8.99E-12
g17	8853.53967	8853.53387	8853.53387	8853.53387	2.43E-09
g18	-0.866025	-0.8660239	-0.8660254	-0.8660253	3.07E-07
g19	32.655593	32.6561999	32.6555936	32.6557002	1.73E-04
g20	-	-	-	-	-
g21	193.72451	193.724512	193.724511	193.724511	3.32E-07
g22	236.430976	655.330567	237.776938	281.382133	7.84E+01
g23	-400.0551	-330.99218	-400.05509	-392.03783	17.29018
g24	-5.5080133	-5.5080133	-5.5080133	-5.5080133	4.59E-11

第五章 总结与展望

5.1 总结

在这次毕业设计中我学到了以下知识：1)系统的学习了差分进化算法 2）学习了多目标优化算法的基本知识以及评价指标 3) 学习了常见的多目标优化算法并且用 Python 实现了 RVEA 算法 4） 在 RVEA 算法的基础上提出了两种不同的改进算法，而且都具有一定的实用性。5）加深了对 Python Numpy 库和数值优化的理解

5.2 展望

5.2.1 工程化的改进

由于时间的关系，我的 RVEA 算法没有实现 GUI 界面。只要将原始代码进行稍许的修改，用上 PyQt5 界面库，我相信不需要费多少精力就可以实现一个简单的用户界面，这样也方面我们理解算法的执行过程以及直观的比较不同算法的优劣性。

5.2.2 算法的改进

在 RVEA 中差分操作只是一种常见的生成子代的操作，我们完全可以采用遗传操作、粒子群操作来取代差分操作，比较不同生成子代的操作对 RVEA 算法的影响。此外，APD 选择和违约值计算公式都可以对其进行改进。

在处理难以约束的优化问题时，将违约值等条件当成额外的优化目标这种方法局限性在于，转化后的优化问题和原始问题不完全等价。所有动态约束边界的 RVEA 算法并不一定是最好的方法，以后可以尝试结合其他的处理约束的算法来改进该算法。

致谢

四年寒暑，如白驹过隙，转瞬间到了毕业季。毕业季也是收获季，在此我向所有帮助过我的老师、同学表示由衷的感谢。

首先，要感谢我的毕业设计指导老师曾三友老师以及焦如旺学长和呼彩娥学姐。他们在我论文的选题、以及相关知识的学习与研究和论文的写作上给予了我极大的帮助。没有了他们，毕设我不可能保质保量的完成。另外还要感谢陈士其同学，他和我的毕设题目类似，我们相互学习，一起讨论那些论文里不太容易理解的部分。

其次，要感谢所有一起做毕设的同学。虽然，他们并没有在毕设上给予我直接的帮助。然而，是他们的奋斗让我看见了希望，是他们的拼搏让我看到了曙光，是他们让我在毕设最困难的时候仍然坚持。

最后，我要感谢大学期间所有的任课老师。基础不牢，地动山摇。所有的任课老师都深入浅出的向我讲解了各自领域的专业知识，为我打下了良好的基础，让我一点点的成长、进步。

参考文献

[1] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, “Combining convergence and diversity in evolutionary multiobjective optimization,” *Evol.Comput.*, vol. 10, no. 3, pp. 263–282, 2002.

[2] X. Zou, Y. Chen, M. Liu, and L. Kang, “A new evolutionary algorithm for solving many-objective optimization problems,” IEEE Trans. Syst.,Man, Cybern. B, Cybern., vol. 38, no. 5, pp. 1402–1412, Oct. 2008

[3] F. di Pierro, S.-T. Khu, and D. A. Savic, “An investigation on preference order ranking scheme for multiobjective evolutionary optimization,”IEEE Trans. Evol. Comput., vol. 11, no. 1, pp. 17–45, Feb. 2007

[4] G. Wang and H. Jiang, “Fuzzy-dominance and its application in evolutionary many objective optimization,” in Proc. Int. Conf. Comput. Intell.Security Workshops, Harbin, China, 2007, pp. 195–198.

[5] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” IEEE Trans. Evol. Comput., vol. 11, no. 6,pp. 712–731, Dec. 2007.

[6] H.-L. Liu, F. Gu, and Q. Zhang, “Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems,” IEEE Trans. Evol. Comput., vol. 18, no. 3, pp. 450–455,Jun. 2014.

[7] Y. Jin, M. Olhofer, and B. Sendhoff, “Dynamic weighted aggregation fore-volutionary multi-objective optimization: Why does it work and how?”in Proc. Genet. Evol. Comput. Conf., San Francisco, CA, USA, 2001,pp. 1042–1049.

[8] T. Murata, H. Ishibuchi, and M. Gen, “Specification of genetic search directions in cellular multi-objective genetic algorithms,” in Proc. Int. Conf. Evol. Multi-Criterion Optim., Zürich, Switzerland, 2001,pp. 82–95.

[9] H.-L. Liu, F. Gu, and Q. Zhang, “Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems,” IEEE Trans. Evol. Comput., vol. 18, no. 3, pp. 450–455,Jun. 2014.

[10] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach,part I: Solving problems with box constraints,” IEEE Trans. Evol.Comput., vol. 18, no. 4, pp. 577–601, Aug. 2014.

[11] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, “A multiobjectiveevolutionary

algorithm using Gaussian process-based inverse modeling,”IEEE Trans. Evol. Comput., vol. 19, no. 6, pp. 838–856, Dec. 2015.

[12] K. Li, K. Deb, Q. Zhang, and S. Kwong, “An evolutionary manyobjective optimization algorithm based on dominance and decomposition,” IEEE Trans. Evol. Comput., vol. 19, no. 5, pp. 694–716,Oct. 2015.

[13] N. Beume, B. Naujoks, and M. Emmerich, “SMS-EMOA: Multiobjective selection based on dominated hypervolume,” Eur. J. Oper. Res., vol. 181,no. 3, pp. 1653–1669, 2007

[14] K. Li et al., “Achieving balance between proximity and diversity in multi-objective evolutionary algorithm,” Inf. Sci., vol. 182, no. 1,pp. 220–242, 2012.

[15] J. Bader and E. Zitzler, “HypE: An algorithm for fast hypervolume-based many-objective optimization,” Evol. Comput., vol. 19, no. 1, pp. 45–76,2011

[16] R. Wang, R. C. Purshouse, and P. J. Fleming, “Preference-inspired coevolutionary algorithms for many-objective optimization,” IEEE Trans. Evol. Comput., vol. 7, no. 4, pp. 474–494, Aug. 2013.

[17] D. K. Saxena and K. Deb, “Dimensionality reduction of objectives and constraints in multi-objective optimization problems: A system designperspective,” in Proc. IEEE Congr. Evol. Comput., Hong Kong, 2008,pp. 3204–3211

[18] Cheng, Ran, et al. "A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization." IEEE Transactions on Evolutionary Computation 20.5(2016):773-791.

[19] L. While, P. Hingston, L. Barone, and S. Huband, “A faster algorithm for calculating hypervolume,” IEEE Trans. Evol. Comput., vol. 10, no. 1,pp. 29–38, Feb. 2006.

[20] Fonseca, C. M, L. Paquete, and M. Lopez-Ibanez. "An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator." Evolutionary Computation, 2006. CEC 2006. IEEE Congress on IEEE, 2006:1157-1163.

[21] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, “A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling,”IEEE Trans. Evol. Comput., vol. 19, no. 6, pp. 838–856, Dec. 2015.

[22] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, “Towards understanding the cost of adaptation in decomposition-based optimization algorithms,” in Proc. IEEE Int. Conf. Syst. Man Cybern., Manchester,U.K., 2013, pp. 615–620

[23] Deb, Kalyanmoy, et al. Scalable Test Problems for Evolutionary Multiobjective

Optimization. Evolutionary Multiobjective Optimization. 2005:105-145.

[24] Zeng, Sanyou, et al. "A General Framework of Dynamic Constrained Multi-objective Evolutionary Algorithms for Constrained Optimization." IEEE transactions on cybernetics 47.9 (2017): 2678-2688.

[25] D. E. Goldberg and J. Richardson, “Genetic algorithms with sharing for multimodal function optimization,” in Proc. 2nd Int. Conf.Gen. Algorithms Gen. Algorithms Appl., Cambridge, MA, USA, 1987,pp. 41–49.

[26] Liang, J. J., et al. "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization." Journal of Applied Mechanics 41.8 (2006).