学士学位论文

基于梯度的多因子进化算法求解多任务连续优化问题

学　　号：　　**20171002448**

姓　　名：　　李延炽

学科专业：　　计算机科学与技术

指导教师：　　龚文引 教授

培养单位：　　计算机学院

二〇二一年五月

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《基于梯度的多因子进化算法求解多任务连续优化问题》，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

近年来，随着云计算的普及，多任务处理的需求不断增加，出现了许多新的针对多任务优化问题的优化算法，它们利用了迁移学习的概念，被称为多因子进化算法。比如多因子遗传算法、多因子差分演化算法等，它们通过杂交的策略来迁移任务间的知识，用一个任务的知识去加速另一个任务的收敛，在处理多任务优化问题上相较于传统的优化算法有着显著的提升。

基于梯度和种群的进化算法吸收了梯度法和元启发式算法的优点，有着鲁棒性强、收敛速度快的特点，在解决连续优化问题上有着卓越的效果。

目前在多任务连续优化问题领域，传统的多因子进化算法没有针对此类问题作特定优化，并且目前还没有学者将多因子策略结合到基于梯度的进化算法中。本文旨在将多因子策略嵌入进基于梯度的进化算法中，创造出一种在解决多任务连续优化问题上性能卓越的新算法。

本文主要工作在于：

1. 介绍基于梯度的进化算法并与其他元启发式算法(遗传算法、差分演化算法、粒子群优化算法)比较，用了 6 个单峰函数和 7 和多峰函数验证其在连续优化问题上的性能；

2. 将多因子策略嵌入到基于梯度的进化算法中，构造出基于梯度的多因子进化算法；

3. 将构造的多因子进化算法与传统基于梯度的进化算法做对比，与其他多因子进化算法(多因子遗传算法、多因子差分演化算法、多因子粒子群优化算法)做对比，用了 3 个测试集(SOCO2016-2T、WCCI2020-Complex、WCCI2020-ManyTask)来验证基于梯度的多因子进化算法在处理多任务连续优化问题上的性能。

根据实验结果可以得出，基于梯度的多因子进化算法在处理一般的多任务连续优化问题上有着优异的性能，在处理多任务复杂问题和超多任务问题上性能和经典多因子算法相当。

关键词：优化算法；元启发式算法；多任务优化；多因子进化；梯度法；连续优化

Abstract

In recent years, with the popularization of cloud computing and the increasing demand of multi-task processing, many new multi-task optimization algorithms have emerged. They make use of the concept of transfer learning, called multi-factorial evolutionary algorithm. For example, multi-factorial genetic algorithm, multi-factorial differential evolution algorithm and so on. They use the strategy of crossover to transfer the knowledge between tasks, and use the knowledge of one task to accelerate the convergence of another task. Compared with the traditional optimization algorithm, they have an apparent improvement in dealing with multi-task optimization problem.

Gradient-based optimization algorithm taking the advantages of gradient method and metaheuristic algorithm, has strong robustness and fast convergence speed, and has excellent effect in solving continuous optimization problems.

At present, in the field of multi-task continuous optimization, the traditional multi-factorial evolutionary algorithm has not been specially optimized for this kind of problem, and no scholar has combined multi-factorial strategy into gradient-based evolutionary algorithm. The aim of this paper is to integrate multi-factorial strategy into gradient-based evolutionary algorithm to create a new algorithm with excellent performance in solving multi-task continuous optimization problems.

The main work of this paper is as follows:

1. Introduces a gradient-based evolutionary algorithm and compares it with other heuristic (GA, DE, PSO), six unimodal functions and seven and multi-modal functions are used to verify its performance in continuous optimization.

2. The gradient-based multi-factorial evolutionary algorithm is constructed by incorporating the multi-factorial strategy into the gradient-based optimization algorithm.

3. The proposed multi-factorial evolutionary algorithm is compared with the traditional gradient-based evolutionary algorithm and other multi-factorial evolutionary algorithms (MFGA, MFDE, MFPSO). Three test sets (SOCO2016-2T, WCCI2020-Complex, WCCI2020-ManyTask) are used to verify its performance in multi-task continuous optimization problems.

According to the experimental results, it can be concluded that the gradient-based multifactorial evolutionary algorithm has excellent performance in dealing with general

multi task continuous optimization problems, and the performance is equivalent to the classical multi factor algorithm in dealing with multi task complex problems and super multi task problems.

Keywords： Optimization algorithm; Metaheuristic algorithm; Multi-task optimization; Multi-factories evolution; Gradient method; Continuous optimization

目　　录

摘　　要 ..... **I**

ABSTRACT ..... **II**

目　　录 ..... **IV**

第一章　　绪论 **1**

1.1　研究背景及国内外研究现状 1

1.2　研究目标与研究内容 2

1.3　论文的组织结构 3

第二章　　基于梯度的进化算法 **4**

2.1　引言 4

2.2　GBO 原理及方法论 4

2.3　实验结果与分析 10

2.4　本章小结 13

第三章　　基于梯度的多因子进化算法 **14**

3.1　引言 14

3.2　MFGBO 原理及方法论 14

3.3　实验结果与分析 17

3.4　本章小结 23

第四章　　总结与展望 **24**

4.1　创新点和优势 24

4.2　不足之处 24

4.3　展望 25

参考文献 **26**

致　　谢 **29**

附　　录 **30**

第一章 绪论

1.1 研究背景及国内外研究现状

1.1.1 元启发式算法 Metaheuristic Algorithm

在各类科学和工程领域中的许多实际应用可以转化为优化问题。然而，相关的问题往往是高维复杂的。虽然已经发展了各种各样的优化算法，但它们往往不能为这类具有挑战性的问题提供满意的结果，这就需要针对这些问题的新的优化方法。元启发式算法(Metaheuristic Algorithms, MAs)[1]多是基于一些物理学、群体智能和生物学的原理，擅长于全局优化问题，已成功地用于解决各种复杂现实中的优化问题[2]。

在过去的几十年里，不同的元启发式算法已经被大量地研发和应用。例如，遗传算法(Genetic Algorithm, GA)[3]就是从达尔文的进化论中衍生出来的。差分演化算法(Differential Evolution, DE)[4][5]采用了与遗传算法类似但具体过程不同的的算子(变异和交叉)，利用两个随机选择的向量之间的差来生成一个新的向量。粒子群优化算法(Particle Swarm Optimization, PSO)[6]是受到鸟群鱼群觅食的社会行为的启发。

上述提到的优化方法都是基于种群的概念，包含了一组解的优化过程。这种优化方法的搜索算子基于上述不同的原理。许多研究已经证明了这些方法在许多现实问题中的成功应用。一般来说，基于种群的优化算法不管本质如何，都会共享共同的信息。在这些算法中，搜索算子实现了两个步骤：探索(Exploration)和开采(Exploitation)。探索即为在整个搜索区域内基于现有位置探索新的位置，而开采的目的是找到附近的最优解的位置。单纯利用探索可能导致新的解精度不高。相反，仅仅利用开采容易陷入局部最优,许多研究强调了在元启发式算法中平衡探索和开采的重要性。因此，在这两个过程之间建立适当的平衡是至关重要的。

1.1.2 基于梯度的进化算法 Gradient-based Algorithm

常用的梯度搜索方法包括牛顿法[7]、拟牛顿法[8]和共轭梯度法[9]。这些方法已被应用于许多研究，以解决不同类型的优化问题。基于梯度的元启发式算法采用了梯度搜索方法和元启发式算法中种群和进化的概念，是一种新的元启发式算法。例如，Salejegheh 将拟牛顿法和粒子群优化算法结合起来[10]，以提高基本粒子群优化算法的性能和鲁棒性；Ibtissem 和 Nouredine 混合了差分演化算法和共轭梯度法[11]，以增强基本差分演化算法中的局部搜索能力；席红雷提出了基于梯度优化的自适应小生境遗传算法[12]；韩飞提出了一种改进的基于梯度搜索的粒子群优化算法[13]；

Iman Ahmadianfar 引入梯度搜索和局部逃逸概念，提出了一种新的基于梯度的进化算法(Gradient-Based Optimizer, GBO)[14]。这些研究表明了基于梯度的元启发式算法的重要作用。

1.1.3 多因子进化 Multifactorial Evolution

近年来，元启发式进化算法已成功地应用于科学、运筹学和工程领域的各种优化问题。这些问题一般可以分为三类：a)单目标优化问题(Single-Objective Optimization, SOO)[15]，其中搜索空间中的每一点映射到一个标量目标值；b)多目标优化问题(Multi-Objective Optimization, MOO)[16]，其中搜索空间中的每一点映射到一个向量值目标函数；c)多因子优化(Multifactorial Optimization, MFO)[17]，其中搜索空间中的每一点映射到不同任务的标量目标值。

随着近年来“云计算”的飞速发展，多个用户会同时发出请求，云服务器需要同时处理多个优化任务，如何高效快速地同时解决多个任务成为了一个亟待解决的问题。多因子优化算法就是在此基础上产生的，多因子优化是一种进化多任务(Evolutionary Multitasking)概念，不同任务具有不同的的搜索空间(可能相互依赖，也可能不相互依赖)，每个搜索空间对应相互之间独立的函数，每一项任务都有一个影响单一个体种群进化的独特因子。常规的元启发式算法的设计通常集中在有效地一次解决一个优化问题上，而基于种群的搜索方法具有隐并行性，在多任务优化领域利用好这种隐并行性是高效解决问题的关键，近年来已经有部分研究者提出了不同的多因子进化算法。例如，Abhishek Gupta 利用遗传算法的杂交操作在不同因子间传递信息[17]，提出了一种可以在任务间迁移知识的多因子进化范式(Multifactorial Evolutionary Algorithm, MFEA)；Liang Feng 在 MFEA 的基础上相继提出多因子粒子群算法(Multifactorial PSO)、多因子差分演化算法(Multifactorial DE)[18]，并在之后提出了一种基于显式自动编码(Explicit Autoencoding)的进化多任务算法[19]；Zedong Tang 提出了一种任务间坐标系自适应映射的算法框架[20]；么双双提出了一种基于分解的多目标多因子进化算法[21]；尚青霞提出了一种基于降噪自动编码器的多任务优化算法[22]。由此可见，近年来解决各类多任务优化问题的多因子进化算法正如雨后春笋般涌现。

1.2 研究目标与研究内容

1.2.1 研究目标

基于 No Free Lunch(NFL)原则[23]，一种特定优化算法不能在所有问题上都取得非常好的效果，这表明一种特定的算法可以为某些问题提供非常好的结果，但是

对于其他的问题集可能效率较低，这促进了多年以来新的元启发式优化算法的发展。

基于梯度的进化算法因其鲁棒性好及收敛快速的特点，在高维单峰和多峰的连续优化问题上取得了较好的成果；多因子进化策略是在 2016 年提出的[17]，在近年涌现了大量基于此的算法解决各类多任务优化问题。而在多任务连续优化问题领域，现有的算法多是基于传统的元启发式算法(GA, DE, PSO…)，没有针对这一类问题的特性去做改进，且收敛性一般较差，能有效地处理多任务连续优化问题，此外，还没有研究者将最新的基于梯度的进化算法引入其中。

本文旨在吸收基于梯度的进化算法在处理连续优化问题上的优点，以及利用能够高效处理多任务优化问题的多因子策略，提出基于梯度的多因子进化算法来高效解决多任务连续优化问题。

1.2.2 研究内容

首先介绍基于梯度的进化算法(Gradient-Based Optimizer, GBO)的原理和方法论，详述梯度搜索规则(Gradient Search Rule, GSR)和局部逃逸算子(Local Escaping Operator, LEO)。通过 6 个高维单峰(Unimodal)问题和 9 个高维多峰(Multimodal)问题[14]来验证 GBO 在连续优化问题上的先进性。

然后将多因子策略引入 GBO 来构建基于梯度的多因子进化算法 MFGBO，详述多因子策略在 GBO 中的实现。通过 9 个 2-Task 多任务单目标优化问题[24]、10 个 2-Task 多任务单目标复杂优化问题(WCCI 2020)和 10 个 10-Task 超多任务单目标优化问题(WCCI 2020)来验证 GBO 算法在多任务连续优化问题上的先进性。

1.3 论文的组织结构

本文结构分为以下四个章节：

第一章，介绍基于梯度的进化算法和多因子进化的概念及研究现状，引出本文的研究目标和内容；

第二章，详细介绍基于梯度的进化算法(GBO)的原理，并通过实验来验证算法在连续优化问题上的性能；

第三章，详细介绍在 GBO 算法的基础上加入多因子策略，改进为基于梯度的多因子进化算法(MFGBO)，并通过实验来验证算法在多任务连续优化问题上的性能；

第四章，总结本文内容，并指出现有算法的优势、不足之处及展望。

第二章 基于梯度的进化算法

2.1 引言

本章主要介绍基于梯度的进化算法(Gradient-Based Optimizer, GBO)的原理及实验验证其在连续优化问题上的先进性。在本章没有引入多任务和多因子概念，主要目的是单独探讨基础 GBO 算法处理连续优化问题的能力。

2.2 GBO 原理及方法论

2.2.1 牛顿法 Newton's method

牛顿法是一种强有力的数值求解方程根的方法[7]，该方法是一种利用泰勒级数初始项的求根算法。这种方法从一个点x_0开始，然后使用点x_0评估的泰勒级数来估计解附近的另一个点。这个过程一直持续到得到最终解为止。泰勒级数$f(x)$可以表示为：

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \frac{f^{(3)}(x_0)(x - x_0)^3}{3!} + \cdots \quad (2.1)$$

将泰勒级数截断，可以得到一个线性近似的$f(x)$，具体如下：

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) \quad (2.2)$$

令$f(x)$为 0，则有：

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \quad (2.3)$$

因此，给定x_n^m，下一代x_n^{m+1}可以表示为：

$$x_n^{m+1} = x_n^m - \frac{f(x_n^m)}{f'(x_n^m)} \quad (2.4)$$

牛顿法使用了一个迭代过程，经过数次迭代最终得到近似解。

2.2.2 种群初始化 Initialization

连续优化问题包括一组决策变量、约束条件和一个目标函数。GBO 的控制参数包括从勘探到开采的过渡参数α和逃逸概率pr。根据问题的复杂度和规模来确定迭代次数和种群大小。在 GBO 算法中，种群的每个个体被称为向量。因此，GBO 在一个 D 维搜索空间中包含 N 个向量，向量可以表示为：

$$X_{n,d} = [X_{n,1}, X_{n,2}, \ldots, X_{n,D}], \qquad n = 1,2,\ldots,N, \qquad d = 1,2,\ldots,D$$

通常，GBO 的初始向量是在 D 维搜索空间中随机生成，可以定义为：

$$X_n = X_{min} + \text{rand}(0,1) \times (X_{max} - X_{min})$$

其中，X_{max}和X_{min}是决策变量X的上下界；$\text{rand}(0,1)$是服从$U(0,1)$的均匀分布随机数。

2.2.3 梯度搜索规则 Gradient Search Rule

在 GBO 算法的梯度搜索规则中，通过控制向量的运动方向，可以在可行域中进行更好的搜索以到达更好的位置。为了提高 GBO 的勘探能力，加快 GBO 的收敛速度，在牛顿法的基础上提出了梯度搜索规则 GSR。由于许多优化问题是不可微的，故采用数值梯度来代替函数的直接导数。通常，牛顿法从一个预测的初始解开始，沿着指定的梯度方向向下一个位置移动。为了实现 GSR，要利用泰勒级数计算函数的一阶导数。$f(x+\Delta x)$和$f(x-\Delta x)$的泰勒级数可以分别表示为：

$$f(x+\Delta x) = f(x) + f'(x_0)\Delta x + \frac{f''(x_0)\Delta x^2}{2!} + \frac{f^{(3)}(x_0)\Delta x^3}{3!} + \cdots \tag{2.5}$$

$$f(x-\Delta x) = f(x) - f'(x_0)\Delta x + \frac{f''(x_0)\Delta x^2}{2!} - \frac{f^{(3)}(x_0)\Delta x^3}{3!} + \cdots \tag{2.6}$$

根据式(2.5)和式(2.6)，函数的一阶导数可以表示为：

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \tag{2.7}$$

根据式(2.4)和式(2.7)，x_n^{m+1}可以定义为：

$$x_n^{m+1} = x_n^m - \frac{2\Delta x \times f(x_n^m)}{f(x_n^m + \Delta x) - f(x_n^m - \Delta x)} \tag{2.8}$$

梯度搜索规则 GSR 是 GBO 算法的核心，需要在牛顿法的基础上修改为基于种群的搜索规则。根据式(2.8)，x_n^m的相邻位置是$x_n^m + \Delta x$和$x_n^m - \Delta x$。在 GBO 算法中，这些相邻位置被替换为种群中的其他两个位置：$x_n^m + \Delta x$被替换为x_{best}，即最优解位置；$x_n^m - \Delta x$被替换为x_{worst}，即最差解位置。此外，由于个体的适应度计算时间较长，因此 GBO 算法采用解x_n^m代替适应度$f(x_n^m)$。GSR 定义如下：

$$GSR = randn \times \frac{2\Delta x \times x_n^m}{(x_{worst} - x_{best} + \varepsilon)} \tag{2.9}$$

其中，$randn$是服从$N(0,1)$正态分布的随机数；ε是$[0, 0.1]$范围内的一个小数；x_{best}和 x_{worst}是在优化过程中得到的最优解和最差解。式(2.9)可以协助当前的个体更新位置。为了提高 GBO 的搜索能力，平衡全局勘探和局部开采能力，通过在 GBO 中引入自适应参数ρ_1对式(2.9)进行改进：

$$\rho_1 = 2 \times rand \times \alpha - \alpha \tag{2.10}$$

$$\alpha = |\beta \times \sin(\frac{3\pi}{2} + \sin(\beta \times \frac{3\pi}{2}))| \tag{2.11}$$

$$\beta = \beta_{min} + (\beta_{max} - \beta_{min}) \times (1 - (\frac{m}{M})^3)^2 \tag{2.12}$$

其中，β_{min}和β_{max}分别为 0.2 和 1.2；m是当前迭代次数；M是迭代总次数。为了平衡勘探和开发过程，参数ρ_1随正弦函数α变化。α随着迭代次数增加而变化。

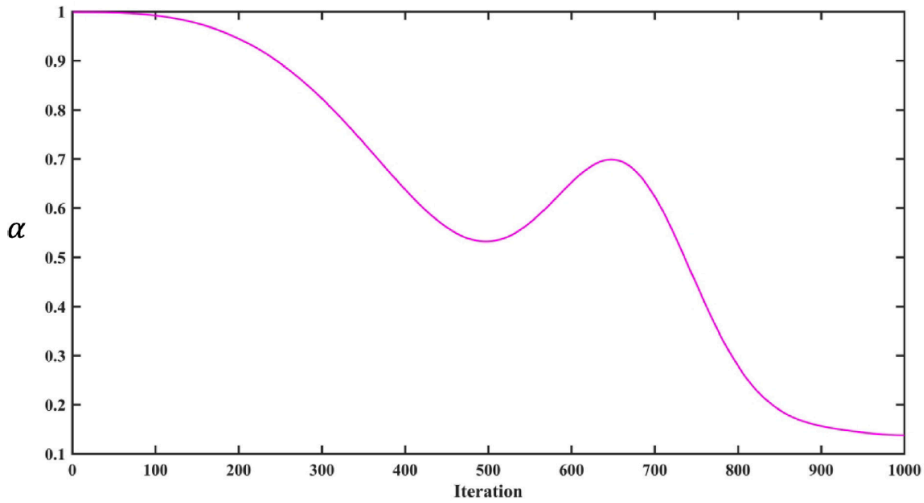

图2.1 α随迭代次数的变化

图 2.1 展示了α的变化趋势，其中最大迭代次数为 100，这个参数可以在具体问题中更改。α在早期迭代时有较大的值，以增强种群的多样性，随着迭代次数的增加，其值逐渐减小，以加速收敛。此外，产生的解应当能够探索最优解周围的空间，为此，α在从 550 到 750 代左右会增加，这有助于提升算法逃脱局部最优的能力。因此，式(2.9)可以改写为：

$$GSR = randn \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(x_{worst} - x_{best} + \varepsilon)} \tag{2.13}$$

其中Δx是根据最优解x_{best}和随机个体x_{r1}^m来确定的(见式(2.14)、式(2.15)和式(2.16))。为了确保Δx在每次迭代中都发生变化，引入参数δ(见式(2.16))。此外，为了增强勘探能力，在式(2.16)中引入随机数$rand$。

$$\Delta x = rand(1{:}N) \times | \ step \ | \tag{2.14}$$

$$step = \frac{(x_{best} - x_{r1}^m) + \delta}{2} \tag{2.15}$$

$$\delta = 2 \times rand \times (|\frac{x_{r1}^m + x_{r2}^m + x_{r3}^m + x_{r4}^m}{4} - x_n^m|) \tag{2.16}$$

其中，$rand(1:N)$是一个N维的随机数；$r1, r2, r3, r4$是从[1, N]中随机选取的不同整数；$step$是由x_{best}和x_{r1}^m决定的步长。因此，式(2.8)可以改写为：

$$x_n^{m+1} = x_n^m - \mathrm{GSR} \tag{2.17}$$

2.2.4 个体移动方向 Direction of Movement

个体移动方向 DM 是 GBO 算法的另一个重要组成部分，是为了更好地开采x_n附近的区域，DM 使当前个体x_n朝着$(x_{best} - x_n)$方向移动，能够增强局部开采能力以及提升收敛速度。DM 定义如下：

$$DM = rand \times \rho_2 \times (x_{best} - x_n^m) \tag{2.18}$$

其中，rand是服从$U(0,1)$的均匀分布随机数；ρ_2与ρ_1定义相同，在迭代初始阶段步长在大范围内随机选取，随着迭代的增加，步长会在小范围内随机选取，在$[0.5, 0.7] \times$ M范围内将随机步长范围扩大来增强脱离局部最优的能力。ρ_2定义如下：

$$\rho_2 = 2 \times rand \times \alpha - \alpha \tag{2.19}$$

2.2.5 生成子代个体

子代个体x_n^{m+1}由多个中间解向量构成，包含$X1_n^m, X2_n^m, X3_n^m$。

$X1_n^m$适合于全局勘探，定义如下：

$$X1_n^m = x_n^m - GSR + DM \tag{2.20}$$

$$X1_n^m = x_n^m - randn \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(x_{worst} - x_{best} + \varepsilon)} + rand \times \rho_2 \times (x_{best} - x_n^m) \tag{2.21}$$

$X2_n^m$适合于局部开采，定义如下：

$$X2_n^m = x_{best} - randn \times \rho_1 \times \frac{2\Delta x \times x_n^m}{(x_{worst} - x_{best} + \varepsilon)} + rand \times \rho_2 \times (x_{r1}^m - x_n^m) \tag{2.22}$$

$X3_n^m$定义如下：

$$X3_n^m = X_n^m - \rho_1 \times (X2_n^m - X1_n^m) \tag{2.23}$$

x_n^{m+1}定义如下：

$$x_n^{m+1} = r_a \times (r_b \times X1_n^m + (1 - r_b) \times X2_n^m) + (1 - r_a) \times X3_n^m \tag{2.24}$$

其中，r_a, r_b是服从$U(0,1)$的均匀分布随机数。

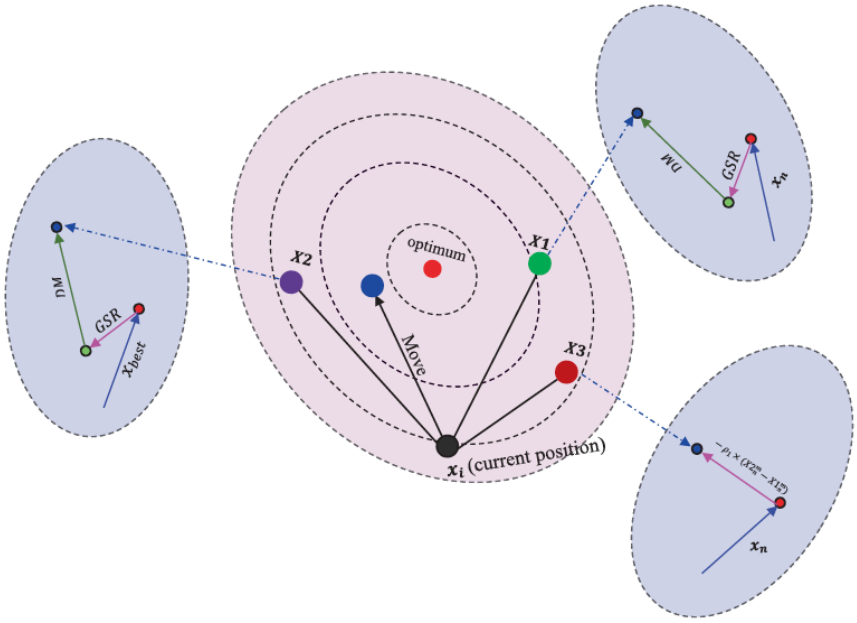

图2.2 x_n^{m+1}的生成图解

图 2.2 描绘了一个个体在二维搜索空间中更新位置的过程。

2.2.6 局部逃逸算子 Local Escaping Operator

为了提高 GBO 算法求解复杂问题的效率，引入了局部逃逸算子 LEO。这个算子可以显著改变x_n^{m+1}的位置。LEO 通过使用多个位置，包含$x_{best}, X1_n^m, X2_n^m, x_{r1}^m, x_{r2}^m$以及一个新的随机解$x_k^m$，生成了表现较优的解$X_{LEO}^m$。

表2.1 X_{LEO}^m生成过程伪代码

```
if rand < pr
    if rand < 0.5
        X_LEO^m = X_n^{m+1} + f_1 × (u_1 × x_best − u_2 × x_k^m) + (f_2 × ρ_1 × (u_3 × (X2_n^m − X1_n^m) + u_2 × (x_r1^m − x_r2^m))) / 2
        X_n^{m+1} = X_LEO^m
    else
        X_LEO^m = X_n^{m+1} + f_1 × (u_1 × x_best − u_2 × x_k^m) + (f_2 × ρ_1 × (u_3 × (X2_n^m − X1_n^m) + u_2 × (x_r1^m − x_r2^m))) / 2
        X_n^{m+1} = X_LEO^m
    endif
endif
```

表 2.1 展示了X_{LEO}^m的生成过程的伪代码，其中，f_1是服从$U(-1,1)$的均匀分布随机数；f_2是服从$N(0,1)$的均匀分布随机数；pr是逃逸概率；u_1, u_2, u_3是 3 个随机数，定义如下：

$$u_1 = \begin{cases} 2 \times rand & if\, \mu_1 < 0.5 \\ 1 & other \end{cases} \tag{2.25}$$

$$u_2 = \begin{cases} rand & if\mu_1 < 0.5 \\ 1 & other \end{cases} \tag{2.26}$$

$$u_3 = \begin{cases} rand & if\mu_1 < 0.5 \\ 1 & other \end{cases} \tag{2.27}$$

其中，rand, μ_1是服从$U(0,1)$的均匀分布随机数。

x_k^m定义如下：

$$x_k^m = \begin{cases} x_{rand} & if\mu_2 < 0.5 \\ x_p^m & other \end{cases} \tag{2.28}$$

$$x_{rand} = x_{min} + rand(0,1) \times (x_{max} - x_{min}) \tag{2.29}$$

其中，x_{rand}是新生成的随机解；x_p^m是在种群中随机选取的解；μ_2是服从$U(0,1)$的均匀分布随机数。

在选择u_1, u_2, u_3时的这种随机行为有助于增加种群的多样性以及避开局部最优解。

2.2.7 GBO 算法框架

GBO 算法的伪代码如表 2.2 所示。

表2.2 GBO 算法伪代码

步骤 1. 初始化
定义迭代次数M、种群数量N、解的维度D、逃逸概率pr
生成随机初始种群 X_0
评价每个个体的函数值 $f(X_0), n = 1, \dots, N$
确定 x_{best}, x_{worst}
步骤 2. 主循环
while $(m < M)$
  $for\ n = 1{:}N$
    $for\ i = 1{:}D$
      选取 $r1, r2, r3, r4$
      根据式(2.24)计算 x_n^{m+1}
    $endfor$
    $if\ rand < pr$
      根据表 2.1 计算 x_{LEO}^m
      $x_n^{m+1} = x_{LEO}^m$
    $endif$
    评价个体并进行 1 对 1 锦标赛选择
    更新x_{best}, x_{worst}
  $endfor$
endwhile
步骤 3. 返回x_{best}^m

2.3 实验结果与分析

2.3.1 实验设计

1) 测试集 Benchmark

使用 14 个数学函数来评价 GBO 算法在连续优化问题上的性能，这些数学函数在以前的研究中[14]得到了广泛的应用。这些测试函数可分为两种不同类型：单峰函数(f1-f6)、多峰函数(f7-f14)，每个函数在 2 维空间上的图像展示在附录一中。

2) 对比算法

遗传算法(Genetic Algorithm, GA)[3]；

差分演化算法(Differential Evolution, DE)[4]；

粒子群算法(Particle Swarm Optimization, PSO)[6]。

3) 参数设置

种群数量(N)：*50*；

问题维度(D)：*30*；

迭代次数(M)：*500*；

独立运行次数：*30*；

GBO：逃逸概率 *pr=0.5*；

GA：基因长度 *lenChrome=5,* 交叉概率 *pc=0.6,* 变异概率 *pm=0.05*；

DE：交叉概率 *pc=0.5,* 缩放因子 *F=0.5*；

PSO：*w* 初始值 *wInital=0.9, w* 结束值 *wEnd=0.4, c1=0.5, c2=0.5*。

4) 评价指标

平均最优解 Average Best Cost，每次独立运行后算法取得最优解的平均值；

标准差 Standard Deviation，每次独立运行后算法取得最优解的标准差；

收敛速度 Convergence Speed，收敛图中曲线下降的越快说明收敛速度越高。

2.3.2 实验结果与分析

表 2.3 展示了各函数上的 GBO、GA、DE 和 PSO 算法的平均最优解，其中每个测试问题上的最优算法对应的数据标绿；表 2.4 展示了标准差，最优标绿；图 2.3 展示了各算法的收敛曲线。

1) 开采能力 Exploitation

通常用单峰函数来评价优化算法的开采能力。这些测试函数只有一个全局最优，没有局部最优。单峰函数对应表 2.3 和表 2.4 中第一列为 U 的前 6 个函数。

在平均最优解、最优解、标准差的单峰函数数据中，可以看到 GBO 算法相较于 GA、DE、PSO，在所有问题上都是最优结果，除了 f4 没有达到全局最优，其

他所有问题都到达了全局最优。GBO 算法对单峰函数具有非常好的准确度，说明 GBO 算法比其他 3 种优化算法具有更好的开采能力。

表2.3 平均最优解

Function		GBO	GA	DE	PSO
U	1	0.00E+00	1.18E+07	2.34E+01	2.78E+06
	2	0.00E+00	9.33E+11	0.00E+00	4.72E+32
	3	0.00E+00	1.39E+02	2.03E+03	1.02E+03
	4	2.30E+01	1.81E+04	5.17E+01	8.21E+03
	5	0.00E+00	1.13E+02	0.00E+00	2.89E+03
	6	0.00E+00	5.61E+04	3.30E-02	1.86E+06
M	7	0.00E+00	3.77E+00	1.07E+01	8.47E+00
	8	0.00E+00	1.33E+01	4.00E-03	3.02E+02
	9	0.00E+00	1.96E+00	0.00E+00	1.23E+01
	10	0.00E+00	1.62E+00	2.00E-03	3.52E+00
	11	0.00E+00	5.20E+00	2.13E+01	3.63E+01
	12	5.98E-01	1.35E+00	5.72E-01	1.05E+01
	13	4.84E-01	1.61E+01	4.63E-01	5.46E+02
	14	0.00E+00	3.36E+00	5.18E+00	3.99E+01

表2.4 标准差

Function		GBO	GA	DE	PSO
U	1	0.00E+00	1.04E+14	1.34E+02	3.42E+12
	2	0.00E+00	1.59E+25	0.00E+00	5.53E+66
	3	0.00E+00	7.00E+03	3.69E+05	3.44E+05
	4	1.54E+00	3.59E+08	1.14E+03	3.13E+08
	5	0.00E+00	1.65E+04	0.00E+00	1.25E+06
	6	0.00E+00	1.23E+10	0.00E+00	1.38E+12
M	7	0.00E+00	4.11E-01	1.63E-01	1.21E+00
	8	0.00E+00	1.36E+02	0.00E+00	2.92E+04
	9	0.00E+00	2.04E+00	0.00E+00	8.88E+01
	10	0.00E+00	4.72E-01	0.00E+00	8.52E-01
	11	0.00E+00	1.69E-01	2.45E+00	1.33E+01
	12	1.50E-02	9.59E-01	6.00E-03	8.26E+01
	13	1.00E-03	2.16E+02	2.10E-02	3.07E+05
	14	0.00E+00	1.30E+00	1.27E+01	1.29E+02

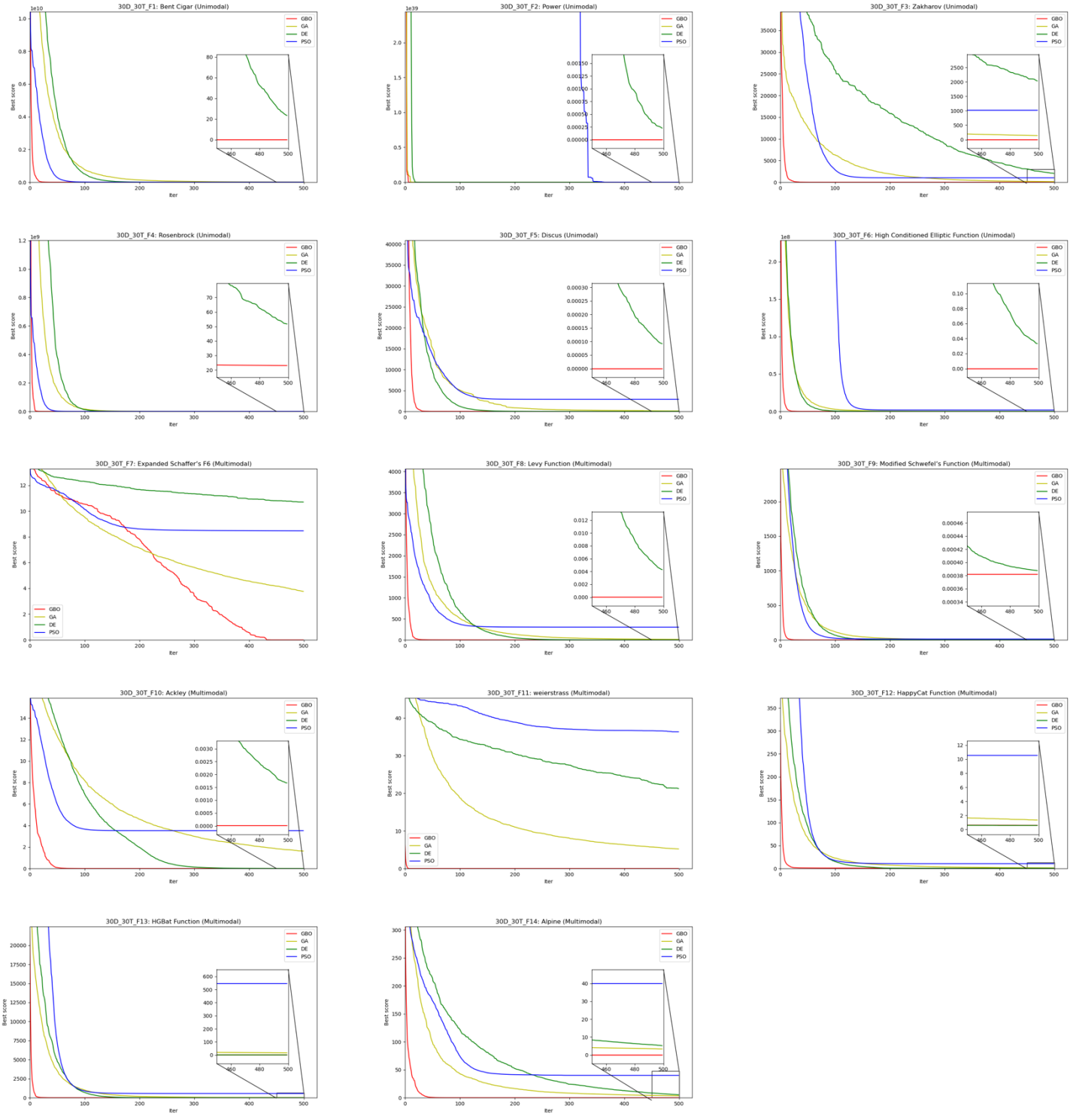

图2.3 各算法在 14 个连续优化问题上的收敛曲线

2) 勘探与逃离局部最优能力 Exploration and escaping from local optima

这里的多峰函数均是使用的多模态测试函数，这些测试函数具有大量的局部最优解，这些局部最优解的数量随着问题维数的增加呈指数增长。因此，用这里的

8 个多峰函数来评价优化算法的勘探和逃离局部最优能力。多峰函数对应表 2.3 和表 2.4 中第一列为 M 的后 8 个函数。

在平均最优解、最优解、标准差的单峰函数数据中，可以看到 GBO 算法在 f7-f11、f14 的问题上的平均最优解、最优解均达到了全局最优；在 f12、f13 上平均最优解、最优解、标准差与 DE 算法结果相似。结果表明，GBO 具有良好的勘探能力，可以提高搜索的效率，有效逃离局部最优。

3) 收敛行为 Convergence behavior

采用收敛速度来评估 GBO 算法的收敛行为。收敛曲线通常用来评价算法的收敛效率，可以明显地看到：在 f1、f3-f6、f7-f14 上，GBO 算法的收敛速度要远远超过其他算法，表明 GBO 算法在开采效率上远高于其他算法；在 f2 上，GBO 和 GA、DE 收敛速度接近；在 f7 上，GBO 算法的收敛速度在开始要慢于其他算法，而在后期，GBO 算法很快逃离了局部最优，而其他算法均陷入了局部最优，说明了 GBO 算法在多峰问题的勘探效率也要高于其他算法；在所有的问题上，GBO 算法在末期均能收敛到接近全局最优的地方。

2.4 本章小结

本章介绍了 GBO 算法的原理及研究了 GBO 算法在处理连续优化问题上的能力。研究发现 GBO 算法在处理高维单峰问题时效果非常好，能够很快地收敛到全局最优；在多峰问题上的效果也要优于许多传统的元启发式算法。

第三章 基于梯度的多因子进化算法

3.1 引言

本章主要介绍基于梯度的多因子优化算法(Multifactorial Gradient-Based Optimizer, MFGBO)的原理及实验验证其在多任务连续优化问题上的先进性。MFGBO 是在 GBO 算法的基础上，引入多因子策略，改造为能够有效处理多任务连续优化问题的算法。

3.2 MFGBO 原理及方法论

3.2.1 多因子进化概念

多因子进化受到多因子遗传模型的启发，该模型提出后代间复杂的发育性状受到遗传和文化因素相互作用的影响[25][26]。例如，个体的知识可能取决于它的基因型，对基因的选择又会被文化因素修改[27][28]，这些文化因素通常来源于社会学习或父辈习俗。

多因子进化为了有效地进行多任务处理，为每一个优化任务创造独立的文化环境，在这个环境中养育后代。多因子策略允许多个文化共存，每个优化任务对应一个文化环境，通过遗传因子和文化因子的相互作用，模拟了复杂环境中个体随后的进化过程。

3.2.2 种群初始化

首先为每个个体p_i定义一组属性，其中$i \in \{1,2,\dots,|P|\}$，P为种群数量。个体编码在$X_1, X_2, \dots, X_K$的统一搜索空间中，可以解码成针对每个任务的解，任务总数为K，p_i的解码形式可以写成$\{x_1^i, x_2^i, \dots, x_K^i\}$，其中$x_1^i \in X_1, x_2^i \in X_2, \dots, x_K^i \in X_K$。例如，$K = 3,\ X_1Dim = 30,\ X_2Dim = 25,\ X_3Dim = 20,\ p_iDim = \max\{X_kDim\} = 30$,其编码与解码的方式如图 3.1。在其中每一个维度上变量的范围都被定义为$[0,1]$，针对每个任务进行运算时，会将$[0,1]$映射到对应任务的变量范围内

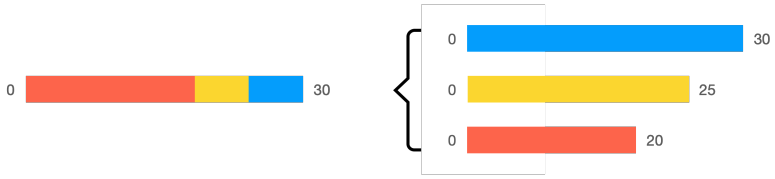

图3.1 个体编码与解码

因为每个任务的函数值不同，不同任务间不能直接用函数值来对比个体的优

劣，需要一种新的方式来进行任务间个体适应值的比较，需要定义以下几个概念：

1) 技能因子 Skill Factor

技能因子τ_i表示当前个体最适应的任务。

2) 因子函数值 Factorial Cost

当前个体p_i的在第k个任务上的函数值。

3) 因子排名 Factorial Rank

当前个体p_i的在第k个任务上的函数值的排名。

4) 标量适应值 Scalar Fitness

当前个体p_i的因子排名的倒数，是为了在不同任务间判断个体的优劣。

3.2.3 因子杂交机制

MFGBO 的一个关键特征是，个体更喜欢与那些属于相同文化背景的其他个体进行杂交，生成新的个体。在 MFGBO 中，技能因子τ被看作是个体文化偏好的表象。因此，如果$x_{best}, x_{worst}, r1, r2$与当前个体技能因子相同，则它们会共同杂交产生新个体。相反地，如果他们的技能因素不同，杂交产生新个体只能按照规定的随机交配概率rmp。根据这些规则生成子代的步骤展示在表 3.1 中。

表3.1 因子杂交机制

$if\ rand < rmp$
选取$\tau = \tau_i$的$x_{best}, x_{worst}, r1, r2$
$else$
随机选取$x_{best}, x_{worst}, r1, r2$
$endif$

文化偏好交配在自然界的出现被用于多因子策略中。随机交配概率rmp被用来平衡搜索的勘探和开采。如果rmp值接近 0 则意味着只有相同文化的个体才允许交配；而如果接近 1 则允许文化间随机交配。在前一种情况下，文化内交配有利于在当前任务上加强搜索能力，但是在当前任务容易陷入局部最优解。较大的rmp值情况下发生的跨文化交配使得算法能够探索每个任务的空间，同时可以学习到其他任务的知识，如果两个任务有相似之处就有助于逃离当前任务的局部最优，但是会导致算法收敛速度下降。因此，必须选择一个合适的rmp值，以确保在勘探和开采之间保持良好的平衡。

3.2.4 垂直文化传播机制

在每个任务上评价个体时，第一步是将个体的统一编码解码为对应该任务的变量。对于连续优化问题，直接通过解空间映射来实现。例如，一个问题的第i个变量x_i在$[L_i, U_i]$范围内，个体的第i个维度为y_i，则个体到实际问题的搜索空间的映射为$x_i = L_i + (U_i, -L_i) * y_i$。

表3.2 垂直文化传播机制

$if\ rand < pm$
随机变异技能因子τ
$else$
遗传父个体的技能因子τ
$endif$

如果对每个个体在每个任务上都进行评价，那这样会极大增加计算量，显然这样做是不可取的，为了使 MFGBO 计算更有效率，算法必须减少评价的总数量。一个重要特征是，在 MFGBO 中生成的个体一般不会在所有任务中都表现非常好，因此，只对选定的最有可能表现良好的任务对个体进行评价会大大增强算法的效率。利用垂直文化传播的概念，根据表 3.2 让子代遗传父本的技能因子(文化特性)以及进行一定概率的变异，仅对子代在技能因子对应的任务上进行评价。

3.2.5 选择操作

使用 1 对 1 锦标赛策略进行选择，让子代个体只和自己的父代个体评比，选取其中较好的留存到下一代。因为子代和父代的技能因子相同，所以不需要再进行一次大规模排序选择，将基础 MFEA 算法[17]中选择操作的时间复杂度$O(n * \log(n))$降低到了$O(n)$。

3.2.6 MFGBO 算法框架

表3.3 MFGBO 算法伪代码

步骤 1. 初始化
定义迭代次数M、种群数量N、逃逸概率pr、杂交概率rmp、变异概率pm
生成初始种群，评价每个个体在每个任务上的因子函数值、因子排名、标量适应值
计算每个个体的技能因子，选出每个任务上的x_{best}, x_{worst}
步骤 2. 主循环
$while\ (m < M\ \&\&\ eva < MaxEva)$
$for\ n = 1{:}N$
根据表 3.1 选取 $x_{best}, x_{worst}, r1, r2, r3, r4$
根据式(2.24)计算 x_n^{m+1}
$if\ rand < pr$
根据表 2.1 计算 x_{LEO}^m
$x_n^{m+1} = x_{LEO}^m$
$endif$
根据表 3.2 遗传或变异x_n^m的技能因子τ_m
只评价x_n^{m+1}在τ_m任务上的因子函数值
在τ_m对应对任务上根据因子函数值进行 1 对 1 锦标赛选择
更新x_{best}, x_{worst}
$endfor$
$endwhile$

3.3 实验结果与分析

3.3.1 实验设计

1) 测试集 Benchmark

第一个，B. Da 在 2016 年提出的单目标多任务测试集 Single-Objective MFO benchmark problems(SOCO2016-2T)[24]，是多任务处理中非常经典的一个测试集，其中问题的分类如表 3.4 所示。

表3.4 SOCO2016-2T 测试集问题类型

	任务间高相似度	任务间中相似度	任务间低相似度
全局最优完全相交	CI_H	CI_M	CI_L
全局最优部分相交	PI_H	PI_M	PI_L
全局最优无相交	NI_H	NI_M	NI_L

第二个，WCCI 2020 Competition on Evolutionary Multi-task Optimization 中的复杂 2 任务测试集。

第三个，WCCI 2020 Competition on Evolutionary Multi-task Optimization 中的 10 任务(超多任务优化 Many-Task Optimization)测试集。

2) SOGBO、MFGBO 对比

SOGBO(Single Objective Gradient-Based Optimizer)为单目标 GBO 算法，即第二章提到的基于梯度度进化算法 GBO；MFGBO 为多任务 GBO 算法，即第三章构建的基于梯度的多因子进化算法 MFGBO。使用多因子进化算法同时对多个任务优化，对比使用 SOGBO 算法分别对每个任务进行优化。

3) 对比多因子算法

用 MFGBO 和其他 3 个经典多因子算法进行对比：

多因子遗传算法(Multifactorial Genetic Algorithm, MFGA)[17]；

多因子差分演化算法(Multifactorial Differential Evolution, MFDE)[18]；

多因子粒子群算法(Multifactorial Particle Swarm Optimization, MFPSO)[18]。

4) 参数设置

种群数量 MF/SO (N)：*100*；

问题维度(D)：*50*；

最大迭代次数(M)：*500*；

最大评价次数(MaxEva)：*100 * 500*；

独立运行次数：*30*；

MFGBO：逃逸概率 *pr=0.5,* 文化交叉概率 *rmp=0.3,* 文化变异概率 *pm=0.1*；

MFGA：基因长度 *lenChrome=10,* 文化交叉概率 *rmp=0.3*；

MFDE：交叉概率 *pc=0.9,* 缩放因子 *F=0.5,* 文化交叉概率 *rmp=0.3*；

MFPSO：*w* 初始值 *wInital=0.9, w* 结束值 *wEnd=0.4, c1=0.2, c2=0.2, c3=0.2,* 文化交叉概率 *rmp=0.3*。

5) 评价指标

平均最优解 Average Best Cost，每次独立运行后算法取得最优解的平均值；

标准差 Standard Deviation，每次独立运行后算法取得最优解的标准差；

运行速度 Running Speed，每次独立运行后算法运行时间的平均值。

3.3.2 SOGBO、MFGBO 对比实验结果与分析

表 3.5 表 3.6 展示了 SOCO2016-2T、WCCI2020-Complex、WCCI2020-ManyTask 测试集上 SOGBO、MFGBO 的平均最优解、标准差、运行时间，第一列表示每个任务组合，第二列表示每个任务组合中的任务(WCCI2020-ManyTask 测试集为所有任务平均值)。其中每个任务对应的平均最优解、标准差、运行时间的最优值标绿突出显示。

表3.5 各测试集 SO/MF 平均最优解、标准差、运行时间

		平均最优解		标准差		运行时间	
Benchmark		MFGBO	SOGBO	MFGBO	SOGBO	MFGBO	SOGBO
SOCO2016-2T							
CI_H	1	0.00E+00	1.09E-01	0.00E+00	4.63E-02	1.07E+02	9.65E+01
	2	1.62E-04	4.83E+02	2.50E-04	9.75E+01		
CI_M	1	3.43E-04	5.81E+00	2.80E-04	1.19E+00	1.24E+02	1.07E+02
	2	1.32E-04	5.15E+02	3.04E-04	1.11E+02		
CI_L	1	3.67E-04	1.71E+01	2.45E-04	7.32E+00	1.13E+02	1.10E+02
	2	6.57E-04	1.12E+04	3.90E-05	7.27E+02		
PI_H	1	3.55E-01	4.92E+02	4.49E-01	1.23E+02	1.10E+02	9.10E+01
	2	3.79E+03	1.22E+00	1.42E+02	9.37E-01		
PI_M	1	3.27E+00	5.77E+00	6.08E-02	1.02E+00	1.15E+02	1.29E+02
	2	2.45E+00	5.21E+02	1.08E+01	3.61E+02		
PI_L	1	2.82E-04	5.80E+00	2.05E-04	1.25E+00	2.27E+02	1.76E+02
	2	1.37E-02	1.23E+01	6.52E-03	4.44E+00		
NI_H	1	1.04E-01	5.05E+02	1.32E-01	2.95E+02	1.14E+02	1.03E+02
	2	1.62E-02	5.07E+02	2.31E-02	1.19E+02		

表3.6 各测试集 SO/MF 平均最优解、标准差、运行时间

		平均最优解		标准差		运行时间	
Benchmark		MFGBO	SOGBO	MFGBO	SOGBO	MFGBO	SOGBO
SOCO2016-2T							
NI_M	1	3.32E-04	1.00E-01	3.63E-04	3.86E-02	2.28E+02	2.16E+02
	2	5.15E-01	2.99E+01	2.29E-01	3.51E+00		
NI_L	1	7.02E+00	4.47E+02	7.74E+00	8.75E+01	1.06E+02	9.39E+01
	2	8.06E-01	1.12E+04	6.85E-01	5.42E+02		
WCCI2020-Complex							
f1	1	6.88E+02	6.87E+02	1.35E-01	3.00E-01	1.95E+02	1.96E+02
	2	6.88E+02	6.87E+02	1.46E-01	2.73E-01		
f2	1	2.54E+03	2.54E+03	1.19E+00	4.03E-01	1.08E+02	9.19E+01
	2	2.54E+03	2.54E+03	7.65E-01	3.29E-01		
f3	1	3.76E+09	3.74E+09	6.05E+06	8.75E+04	1.00E+02	9.53E+01
	2	3.76E+09	3.74E+09	5.11E+06	5.34E+04		
f4	1	1.31E+03	1.31E+03	4.83E-03	1.48E-03	1.10E+02	9.32E+01
	2	1.31E+03	1.31E+03	5.19E-03	1.73E-03		
f5	1	2.38E+07	2.35E+07	7.26E+04	1.15E+04	1.18E+02	1.26E+02
	2	2.38E+07	2.35E+07	7.37E+04	1.45E+04		
f6	1	1.72E+09	1.71E+09	3.21E+06	7.48E+01	1.07E+02	9.63E+01
	2	1.72E+09	1.71E+09	3.09E+06	8.63E+01		
f7	1	5.89E+06	5.88E+06	6.17E+03	2.48E+02	1.10E+02	9.63E+01
	2	5.88E+06	5.88E+06	6.20E+03	4.06E+01		
f8	1	5.21E+02	5.20E+02	4.74E-02	1.17E-01	1.04E+02	9.45E+01
	2	5.21E+02	5.20E+02	3.41E-02	1.27E-01		
f9	1	1.85E+04	1.79E+04	6.14E+01	3.89E+01	4.87E+02	8.61E+01
	2	1.62E+03	1.62E+03	6.67E-02	4.31E-02		
f10	1	3.14E+09	3.13E+09	2.96E+06	2.70E+03	4.40E+02	7.55E+01
	2	1.72E+09	1.71E+09	4.90E+06	9.05E+01		
WCCI2020-ManyTask							
f1		1.55E+03	1.42E+03	1.55E+03	1.42E+03	1.54E+02	1.30E+02
f2		6.74E+05	4.74E+05	6.74E+05	4.74E+05	1.41E+02	9.75E+01
f3		7.28E+02	4.95E+02	7.28E+02	4.95E+02	1.16E+02	1.01E+02
f4		3.05E+07	2.75E+07	3.05E+07	2.75E+07	1.15E+02	9.77E+01
f5		1.68E+03	1.57E+03	1.68E+03	1.57E+03	1.81E+02	1.70E+02
f6		3.32E+07	2.90E+07	3.32E+07	2.90E+07	1.13E+02	9.58E+01
f7		1.21E+03	1.12E+03	1.21E+03	1.12E+03	1.78E+02	1.62E+02
f8		1.55E+08	1.45E+08	1.55E+08	1.45E+08	1.59E+02	1.41E+02
f9		1.88E+08	1.76E+08	1.88E+08	1.76E+08	1.37E+02	1.21E+02
f10		6.79E+03	6.70E+03	6.79E+03	6.70E+03	1.62E+02	1.48E+02

SOGBO、MFGBO 在运行时间上相差无几，因为是根据评价次数来终止算法的运行，两个算法时间复杂度相同，在多任务问题上相同评价次数下 SOGBO、MFGBO 运行时间也相似。

在 SOCO2016-2T 测试集上，MFGBO 在绝大多数问题上的平均最优解要远远好于用 SOGBO 算法单独计算每个任务。在 PI_H-Task2 上 MFGBO 结果要差于 SOGBO，原因是这个任务是一个单峰函数，SOGBO 单独运行会快速收敛到全局最优，而 MFGBO 中受另一个多峰任务影响，没有收敛到全局最优。在大多数情况下 MFGBO 都能利用两个任务间的相似性来互相学习，迁移有用的知识，即使任务相似度不高，通过垂直文化传播机制也能让个体更少地受另一个任务影响。

在 WCCI2020-Complex 测试集上，由于任务较多，故使用 10 个任务的平均值作为评价指标。MFGBO 平均最优解均稍差于 SOGBO，但是结果相差无几。出现这种情况的原因主要是 MFGBO 中通过杂交的知识迁移机制不能有效地解决复杂且差异较大的任务。

在 WCCI2020-ManyTask 测试集上，MFGBO 平均最优解均要差于 SOGBO，在第 2、3 个问题上 MFGBO 的解离 SOGBO 还有一定距离，说明 MFGBO 在处理超多任务问题上的效果并不是很明显，还要稍差于 SOGBO 分别运行每个任务。出现这种情况的原因是 MFGBO 中通过文化间杂交来进行知识迁移的方法对超多任务问题效果不好，因为随着任务的增加，任务间相似性指数级降低，需要一种新的方法来处理超多任务问题。

3.3.3 多因子算法横向对比实验结果与分析

表 3.7 表 3.8 展示了 SOCO2016-2T、WCCI2020-Complex 测试集上 MFGBO、MFGA、MFDE、MFPSO 的平均最优解、标准差。表 3.9 展示了各测试集上各算法的运行时间。每个表格的第一列表示每个任务组合，第二列表示每个任务组合中的若干任务。其中每个任务对应的各项最优值已标绿进行突出显示。

表3.7 各测试集各算法平均最优解、标准差

		平均最优解				标准差			
Benchmark		MFGBO	MFGA	MFDE	MFPSO	MFGBO	MFGA	MFDE	MFPSO
SOCO2020-2T									
CI_H	1	0.00E+00	1.15E+00	7.18E-02	1.08E+00	0.00E+00	2.37E-02	3.36E-02	2.88E-02
	2	1.62E-04	4.58E+02	1.41E+02	4.55E+02	2.50E-04	3.76E+01	8.32E+01	4.12E+01
CI_M	1	3.43E-04	8.71E+00	7.39E-01	8.11E+00	2.80E-04	5.05E-01	3.54E-01	1.01E+00
	2	1.32E-04	5.49E+02	8.44E+01	6.12E+02	3.04E-04	6.23E+01	6.23E+01	9.55E+01
CI_L	1	3.67E-04	2.11E+01	2.12E+01	2.13E+01	2.45E-04	9.50E-02	2.37E-02	4.45E-02
	2	6.57E-04	1.01E+04	1.37E+04	1.58E+04	3.90E-05	7.95E+02	1.10E+03	9.70E+02

表3.8 各测试集各算法平均最优解、标准差

		平均最优解				标准差			
Benchmark		MFGBO	MFGA	MFDE	MFPSO	MFGBO	MFGA	MFDE	MFPSO
SOCO2020-2T									
PI_H	1	3.55E-01	8.19E+02	2.93E+02	1.05E+03	4.49E-01	6.93E+01	7.86E+01	1.66E+02
	2	3.79E+03	4.45E+02	1.33E+00	5.74E+03	1.42E+02	8.09E+01	1.18E+00	1.27E+03
PI_M	1	3.27E+00	8.14E+00	6.44E-01	7.39E+00	6.08E-02	4.43E-01	1.74E-01	6.62E-01
	2	2.45E+00	1.92E+05	1.75E+02	1.89E+05	1.08E+01	6.15E+04	3.15E+01	1.07E+05
PI_L	1	2.82E-04	2.09E+01	3.11E+00	1.38E+01	2.05E-04	1.30E-01	8.54E-01	2.65E+00
	2	1.37E-02	2.17E+01	1.72E+00	1.14E+01	6.52E-03	3.31E+00	8.90E-01	4.03E+00
NI_H	1	1.04E-01	1.98E+05	3.64E+02	6.84E+05	1.32E-01	5.10E+04	1.35E+02	2.90E+05
	2	1.62E-02	5.81E+02	3.34E+02	6.04E+02	2.31E-02	8.44E+01	4.02E+01	8.26E+01
NI_M	1	3.32E-04	1.11E+00	9.08E-02	1.22E+00	3.63E-04	2.02E-02	3.65E-02	1.09E-01
	2	5.15E-01	3.00E+01	6.04E+00	3.34E+01	2.29E-01	2.09E+00	1.00E+00	2.19E+00
		平均最优解				标准差			
Benchmark		MFGBO	MFGA	MFDE	MFPSO	MFGBO	MFGA	MFDE	MFPSO
SOCO2020-2T									
NI_L	1	7.02E+00	8.22E+02	4.65E+02	2.79E+03	7.74E+00	8.11E+01	3.29E+01	7.43E+02
	2	8.06E-01	1.00E+04	4.40E+03	1.58E+04	6.85E-01	6.60E+02	6.30E+02	7.72E+02
WCCI2020-Complex									
f1	1	6.88E+02	6.88E+02	6.85E+02	6.87E+02	1.35E-01	3.40E-01	1.45E-01	2.90E-01
	2	6.88E+02	6.88E+02	6.85E+02	6.87E+02	1.46E-01	3.40E-01	1.51E-01	2.90E-01
f2	1	2.54E+03	2.54E+03	2.54E+03	2.54E+03	1.19E+00	6.83E-01	1.06E-03	2.47E+00
	2	2.54E+03	2.54E+03	2.54E+03	2.54E+03	7.65E-01	6.89E-01	1.06E-03	2.47E+00
f3	1	3.76E+09	3.74E+09	3.74E+09	3.75E+09	6.05E+06	1.66E+06	1.33E+00	1.02E+07
	2	3.76E+09	3.74E+09	3.74E+09	3.75E+09	5.11E+06	1.69E+06	1.39E+00	1.02E+07
f4	1	1.31E+03	1.31E+03	1.31E+03	1.31E+03	4.83E-03	1.77E-03	4.00E-06	8.29E-03
	2	1.31E+03	1.31E+03	1.31E+03	1.31E+03	5.19E-03	1.76E-03	4.00E-06	8.29E-03
f5	1	2.38E+07	2.36E+07	2.35E+07	2.36E+07	7.26E+04	2.16E+04	2.09E+01	9.19E+04
	2	2.38E+07	2.36E+07	2.35E+07	2.36E+07	7.37E+04	2.19E+04	2.00E+01	9.19E+04
f6	1	1.72E+09	1.72E+09	1.71E+09	1.71E+09	3.21E+06	5.60E+05	2.29E-01	1.98E+06
	2	1.72E+09	1.72E+09	1.71E+09	1.71E+09	3.09E+06	5.62E+05	2.24E-01	1.98E+06
f7	1	5.89E+06	5.88E+06	5.88E+06	5.88E+06	6.17E+03	5.22E+02	1.03E-01	3.44E+03
	2	5.88E+06	5.88E+06	5.88E+06	5.88E+06	6.20E+03	5.25E+02	1.03E-01	3.44E+03
f8	1	5.21E+02	5.20E+02	5.20E+02	5.20E+02	4.74E-02	8.96E-02	6.32E-02	1.31E-01
	2	5.21E+02	5.20E+02	5.20E+02	5.20E+02	3.41E-02	8.95E-02	6.31E-02	1.31E-01
f9	1	1.85E+04	1.80E+04	1.78E+04	1.81E+04	6.14E+01	3.04E+01	1.19E+00	8.50E+01
	2	1.62E+03	1.62E+03	1.62E+03	1.62E+03	6.67E-02	4.44E-02	1.52E-03	7.23E-02
f10	1	3.14E+09	3.14E+09	3.13E+09	3.14E+09	2.96E+06	6.68E+05	1.49E+00	2.29E+06
	2	1.72E+09	1.72E+09	1.71E+09	1.71E+09	4.90E+06	1.08E+06	2.55E+00	7.79E+01

表3.9 各测试集各算法运行时间

Benchmark	MFGBO	MFGA	MFDE	MFPSO
SOCO2016-2T				
CI_H	1.07E+02	1.53E+02	1.58E+02	1.49E+02
CI_M	1.24E+02	1.67E+02	1.69E+02	1.13E+02
CI_L	1.13E+02	1.51E+02	1.42E+02	1.08E+02
PI_H	1.10E+02	1.47E+02	1.64E+02	1.17E+02
PI_M	1.15E+02	1.69E+02	1.68E+02	1.51E+02
PI_L	2.27E+02	2.19E+02	2.18E+02	1.73E+02
NI_H	1.14E+02	1.53E+02	1.52E+02	1.10E+02
NI_M	2.28E+02	2.74E+02	2.79E+02	2.10E+02
NI_L	1.06E+02	1.42E+02	1.43E+02	1.05E+02
Benchmark	MFGBO	MFGA	MFDE	MFPSO
WCCI2020-Complex				
f1	1.95E+02	2.94E+02	2.70E+02	2.21E+02
f2	1.08E+02	1.66E+02	1.47E+02	1.00E+02
f3	1.00E+02	1.42E+02	1.42E+02	1.13E+02
f4	1.10E+02	1.67E+02	1.73E+02	1.15E+02
f5	1.18E+02	1.94E+02	1.83E+02	1.09E+02
f6	1.07E+02	1.51E+02	1.49E+02	1.08E+02
f7	1.10E+02	1.49E+02	1.48E+02	1.07E+02
f8	1.04E+02	1.48E+02	1.49E+02	1.08E+02
f9	4.87E+02	4.73E+02	6.00E+02	2.56E+02
f10	4.40E+02	4.56E+02	6.51E+02	2.78E+02

在运行时间上，MFGBO、MFPSO 要优于 MFGA、MFDE。原因是 MFGBO 在选取下一代时用了 1 对 1 锦标赛策略，MFPSO 也只是对每个个体进行更新，选择操作的时间复杂度为$O(n)$，而 MFGA、MFDE 对父、子种群进行了合并和排序，选取最好的一半作为下一代，时间复杂度为$O(n * \log(n))$。

在 SOCO2016-2T 测试集上，MFGBO 在绝大多数问题上的平均最优解要远远好于其他 3 个算法。在 PI_H-Task2、PI_M-Task1 上 MFGBO 结果稍差于并接近 MFDE。

在 WCCI2020-Complex 测试集上，4 个算法结果整体结果都不好，其中 MFDE 要稍微好一点。说明通过文化间杂交进行知识迁移的多因子算法不能很好地处理复杂的多任务问题。

3.4 本章小结

本章介绍了 MFGBO 的原理及研究了 MFGBO 在处理多任务连续优化问题上的能力。研究发现相比于使用单任务算法对多个任务分别计算，通过杂交操作进行知识迁移的多因子算法能够有效地解决任务间相似度高的连续优化问题，但是不能有效解决超多任务问题和较难进行知识迁移的复杂多任务问题。

MFGBO 相较于[17][18]中提出的 MFGA、MFDE、MFPSO 算法，在处理相似度高的多任务连续优化问题上要更有优势，得益于 GBO 算法在处理连续优化问题上的优势。

第四章 总结与展望

4.1 创新点和优势

本文的创新点主要是在求解连续优化问题效果优异的基于梯度的进化算法 GBO 中嵌入多因子策略 MF，创造出一种基于梯度的多因子进化算法 MFGBO 来解决多任务连续优化问题，利用了迁移学习的思想，让种群学习任务间潜在的相似性知识，以加强云计算中多任务优化的效率，满足用户日益增长的需求。

基于梯度的多因子进化算法 MFGBO 在解决相似的多任务优化问题上有良好的效果，并且相比于未加入多因子策略的 GBO 算法在多任务优化问题上有着显著的提升，和其他多因子算法的横向对比也显示出 MFGBO 的优势。

MFGBO 的主要优势在于：

1) 开采能力出众，收敛速度快；

2) 勘探能力较优，容易脱离局部最优；

3) 很好地平衡了勘探和开采阶段，算法效率高；

4) 多因子策略能够在相似的多任务问题间进行知识迁移，促进任务的收敛；

5) 在问题相似度较低的时候也能够避免任务间的负知识迁移，最差也能达到单任务算法独立解决任务的效率。

4.2 不足之处

在处理相似度不高的复杂多任务问题时，MFGBO 不能很好地迁移任务间知识，原因是利用杂交进行知识迁移的方法对多任务优化的效果取决于任务间的相似度，如果任务间最优解位置差异较大，MFGBO 就不能够有效的进行知识迁移。

在超多任务优化问题上，MFGBO 不能取得很好的效果，一个原因是 MFGBO 在初始化的过程中不能保证每个任务都有均匀的个体。比如种群大小为 100，任务为 10，初始化后可能有的任务有 20 个个体，有的只有不到 5 个，对于个体少的任务就不能高效率地优化，仅靠因子的变异不能很好地解决这个问题。另外的一个原因就是如果同时处理非常多个任务，任务间的差异就会指数上涨，目前采用的多因子策略对其中某个任务来说不能高效地从其他较相似的任务中学习到知识，这样就会导致收敛性不足。

4.3 展望

目前个体的编码方式是固定每个任务的每个维度映射到基因上，未来可以使用自适应任务基因编码映射的方法来解决这个问题。这样如果两个问题的最优解位置差异较大，种群通过迭代去转换每个任务解的映射，使两个不相似的任务最优解位置通过改变解的映射达到同样的位置，就能够很好地从任务间迁移知识。

目前进行任务间知识迁移的方式是通过杂交的操作，而杂交操作有多种不同的算子，每种算子对于不同任务知识迁移对效果是不同的。可以考虑将多种算子加入到算法中，利用强化学习的思想，对每个个体使用的杂交算子进行自适应选择，每经过一定迭代次数就根据算子产生的效果重新选择。

超多任务优化可以改进的方向是将超多任务划分为若干子任务组合，在每个组合内进行杂交，每迭代一定次数后对组合间任务的相似性进行判断，对相似度低的组合打乱重组，这样就能避免超多任务带来的任务间相似度指数级下降问题。

参考文献

[1] Yang X . Nature-Inspired Metaheuristic Algorithms: Second Edition[J]. 2010.

[2] Ahmadianfar I , Khajeh Z , Pari S , et al. Developing optimal policies for reservoir systems using a multi-strategy optimization algorithm[J]. Applied Soft Computing, 2019.

[3] A U M , B S B . Genetic algorithm-based clustering technique[J]. Pattern Recognition, 2000, 33(9):1455-1465.

[4] Kasbawati, Gunawan A Y , Sidarto K A , et al. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces[C] Whfreeman. Whfreeman, 2015.

[5] 贺毅朝, 王熙照, 刘坤起, 等. 差分演化的收敛性分析与算法改进[J]. 软件学报, 2010, 21(5): 875-885.

[6] Shi Y H , Eberhart R C . Empirical study of particle swarm optimization[C] Congress on Evolutionary Computation. IEEE, 2002.

[7] Ypma T J . Historical Development of the Newton–Raphson Method[J]. Siam Review, 1995, 37(4):531-551.

[8] Bazaraa M S , Sherali H D , Shetty C M . Nonlinear programming. Theory and algorithms. 2. ed[M]. Wiley, 1993.

[9] Hestenes M R , Steifel E . Method of Conjugate Gradients for Solving Linear Systems. J.res.nat.standards, 1952.

[10] Salajegheh F , Salajegheh E . PSOG: Enhanced particle swarm optimization by a unit vector of first and second order gradient directions[J]. Swarm and Evolutionary Computation, 2019.

[11] Liouane N . A hybrid method based on conjugate gradient trained neural network and differential evolution for non linear systems identification[C] International Conference on Electrical Engineering & Software Applications. IEEE, 2013.

[12] 席红雷, 行小帅, 张清泉. 基于梯度优化的自适应小生境遗传算法[J]. 计算机工程, 2008(11):186-188.

[13] 韩飞, 杨春生, 刘清. 一种改进的基于梯度搜索的粒子群优化算法[J]. 南京大学学报(自然科学), 2013(02):196-201.

[14] Ahmadianfar I, Bozorg-Haddad O, Chu X. Gradient-based optimizer: A new Metaheuristic optimization algorithm[J]. Information Sciences, 2020, 540: 131-159.

[15] Guo S M , Yang C C . Enhancing Differential Evolution Utilizing Eigenvector-Based Crossover Operator[J]. IEEE Transactions on Evolutionary Computation, 2015, 19(1):31-49.

[16] Fonseca C M , Fleming P J . An Overview of Evolutionary Algorithms in Multiobjective Optimization[J]. Evolutionary Computation, 1999, 3(1):1-16.

[17] Gupta A, Ong Y S, Feng L. Multifactorial evolution: toward evolutionary multitasking[J]. IEEE Transactions on Evolutionary Computation, 2015, 20(3): 343-357.

[18] Feng L , Zhou W , Zhou L , et al. An empirical study of multifactorial PSO and multifactorial DE[C] 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2017.

[19] Feng L , Zhou L , Zhong J , et al. Evolutionary Multitasking via Explicit Autoencoding[J]. IEEE Transactions on Cybernetics, 2018:1-14.

[20] Tang Z , Gong M , Wu Y , et al. A Multifactorial Optimization Framework Based on Adaptive Intertask Coordinate System[J]. IEEE Transactions on Cybernetics, 2021, PP(99):1-14.

[21] 么双双, 董志明, 王显鹏. 基于分解的多目标多因子进化算法[J]. 控制与决策, 2021, 36(3): 637-644.

[22] 尚青霞, 周磊, 冯亮. 基于降噪自动编码器的多任务优化算法[J]. 大连理工大学学报, 2019 (2019 年 04): 417-426.

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

[24] Da B, Ong Y S, Feng L, et al. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results[J]. arXiv preprint arXiv:1706.03470, 2017.

[25] Rice J, Cloninger C R, Reich T. Multifactorial inheritance with cultural transmission and assortative mating. I. Description and basic properties of the unitary models[J]. American journal of human genetics, 1978, 30(6): 618.

[26] Cloninger C R, Rice J, Reich T. Multifactorial inheritance with cultural transmission and assortative mating. II. a general model of combined polygenic and cultural inheritance[J]. American journal of human genetics, 1979, 31(2): 176.

[27] Cavalli-Sforza L L, Feldman M W. Cultural versus biological inheritance: phenotypic transmission from parents to children.(A theory of the effect of parental phenotypes on children's phenotypes)[J]. American journal of human genetics, 1973, 25(6): 618.

[28] Feldman M W, Laland K N. Gene-culture coevolutionary theory[J]. Trends in ecology & evolution, 1996, 11(11): 453-457.

致　　谢

作为即将成为研究生的我来说，本文是龚文引老师指导带我入门研究生道路的第一篇文章，首先特别感谢龚老师的指导，龚老师在我开始做本文相关的研究之前给我推荐了许多篇文献和相关网站，我在学习了不久后就入门了优化算法和多任务优化问题，对本文的完成提供了非常大的帮助。

同时非常感谢实验室的各位师兄师姐，感谢明飞师兄、李水佳师兄、李瑞师兄、余方舟师兄的帮助。我在完成本文的过程中遇到过非常多的问题，解决不了的都是直接向各位师兄请教。明飞师兄在论文、优化算法方面给我提出了许多建议，带来了很大的帮助；李水佳师兄在多任务优化方向给我提供了很多指导；李瑞师兄给我指点了本科毕业论文相关的内容；余方舟师兄和我探讨了基于梯度的进化算法代码上的问题。有幸在本院读继续攻读研究生，本学期就提前进入了龚老师的实验室，实验室师兄师姐都给了我莫大的帮助。

感谢本科四年以来帮助我的辅导员彭堂树老师，感谢在学生会期间教导我的傅苑老师、李欢欢老师，感谢一路以来帮助过我的同学们。最后感谢我的父母、长辈们对我的抚养，让我在求学路上没有后顾之忧。

附　　录

附录一　连续优化问题的 14 个测试函数在 2 维空间的图像