分 类 号_____ 学号 M201772696
学校代码 10487 密级_____

華中科技大學

硕 士 学 位 论 文

基于多任务进化算法的
连续优化研究

学位申请人： 谭显烽
学 科 专 业： 控制工程
指 导 教 师： 伍冬睿 教授
答 辩 日 期： 2019 年 5 月 21 日

A Thesis Submitted in Partial Fulfillment of the Requirements
For the Degree of Master of Engineering

Multi-Tasking Evolutionary Algorithms for Continuous Optimization

Candidate : Xianfeng Tan
Major : Control Engineering
Supervisor : Prof. Dongrui Wu

Huazhong University of Science & Technology

Wuhan 430074, P. R. China

May, 2019

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体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，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以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、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于　保密□，在 ____ 年解密后适用本授权书。

　　不保密 □。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘　要

多任务进化是进化计算和优化领域内新兴的子领域，它采用了多任务学习的思想，能够同时地解决多个优化问题。相对于一次只解决一个优化问题的经典进化算法，多任务进化算法可以有效地利用不同任务间存在的共性或互补性，从而促进种群的进化。这是容易理解的，在不同任务间迁移有用的信息，可以提升进化算法在多任务优化问题中的效率。本文主要关注单目标多任务的连续优化问题，也就是如何同时解决多个目标函数值为标量的连续优化问题。主要分为三个部分：

(1) 首先介绍基于遗传算法的多任务进化算法，我们提出了一种新的容易实现的多任务遗传算法，它能够估计并使用任务间的偏差。最后通过广泛的实验验证了我们提出的多任务遗传算法具有优异的性能。

(2) 我们进一步提出了多任务差分进化，它在多任务遗传算法的框架下，使用差分进化来搜索最优解。大量的实验验证了多任务差分进化也具有优异的性能。

(3) 我们首次将多任务优化应用于模糊控制器设计，并验证了我们提出的多任务遗传算法和多任务差分进化均能取得较好的效果。

.

关键词： 进化算法　　多任务进化　　多任务学习　　多因素优化　　模糊逻辑控制器

Abstract

Evolutionary multi-tasking, or multi-factorial optimization, is an emerging subfield of multitask optimization and evolutionary computation, which integrates evolutionary computation and multi-task learning, and can solve multiple optimization problems simultaneously. Compared with the classical evolutionary algorithm which only solves a single optimization problem at a time, it can effectively utilize the commonality or complementarity between different tasks to promote population evolution. It is easy to understand that transferring useful information between different tasks can improve the efficiency of evolutionary algorithms in multi-task optimization problems. This thesis mainly focuses on single-objective multi-tasking optimization, that is, how to solve multiple continuous optimization problems with scalar objective function simultaneously. It is mainly divided into three parts:

(1) Firstly, a multi-tasking evolutionary algorithm based on genetic algorithm is introduced. Then we propose a novel easy-to-implement multi-tasking genetic algorithm (MTGA), which copes well with significantly different optimization tasks by estimating and using the bias among them. Finally, extensive experiments demonstrated that the proposed MTGA has excellent performance.

(2) We further propose the multi-tasking differential evolution (MTDE), which uses differential evolution to search the optima in the framework of MTGA. A large number of experiments demonstrated that the proposed MTDE also has excellent performance.

(3) We apply multi-tasking optimization to the design of fuzzy logic controller for the first time, and demonstrated that the MTGA and the MTDE can find better fuzzy logic controllers than other approaches.

.

Key words: Evolutionary algorithm evolutionary multitasking multi-task learning multi-factorial optimization fuzzy logic controllers

目　录

1 绪论

1.1 进化算法

进化算法起源于19世纪50年代，friedberg首先将进化过程用于自动编程[1]，紧接着，Bremermann首先将模拟的进化用于解决数值优化问题[2]，除此之外，Bremermann还发展了一些早期的进化算法理论，表明线性可分离问题的最优变异概率应该为$1/l$，l为个体的长度[3]。在进化算法的早期，Box发展了他的进化操作的思想，包含了一个进化技术用于设计和分析工业的实验[4,5]。直到19世纪60年代中期，进化计算领域中的三个主要方向的基础得以建立，即进化编程，遗传算法和进化策略[6]：

(1) Fogel等人成功的实现了进化编程[7]，他们使用了有限状态机的模拟进化去预测关于任意准则的非平稳的时间序列，并且使用专门的操作符来保证子代语法的正确性。

(2) Holland首先提出了遗传算法[8]，用二进制的字符串表示可能的解，并且用这些单个的字符串组成一个种群。在进化的过程中维持这个种群，采用类似于染色体交叉变异的方式去调整这些个体的值。为了评价种群中每个个体的好坏，也就是谁能更好的解决某个特定的问题，算法需要给定一个适应函数，使得更好的个体拥有更高的适应度，然后通过选择机制选出最好的一部分个体，组成新种群。遗传算法被广泛的用于解决各种复杂的优化问题。

(3) 柏林工业大学的三个学生Bienert, Rechenberg和Schwefel首先提出了进化策略[9–11]，用于搜索机器人的最优形状，使得其在风洞中遇到的阻力最小。进化策略被大量地用于连续搜索空间中的黑箱优化问题。进化策略随机地产生新的候选解，最常见的是从多元正态分布中抽取新个体。它的两个突出的特点是抽样分布的参数的无偏性和自适应控制。

达尔文的进化论揭示了生物的起源与演化，“物竞天择，适者生存”，因为个体之间遗传物质的交换和变异提高了种群的多样性，使得种群能够获得新的更适应当前环境的个体，加上环境对个体的选择，使得种群中适应性更强的个体得以保留下来，较弱的个体被淘汰，环境的整体承受能力有限。进化是一个优化过程，优化了种群中的个体，从而提高了种群对当前环境的整体适应度。同理，也可以将进化的思想用于解决现实生活中的优化问题。在大部分进化算法中，都使用了种群的概念，通过种群隐式地执行了并行搜索，种群中的一个个体对应的是一个优化问题的解，不同的解在同一个优化问题中可能表现不同，在给定与优化问题相关的适应函数后，则可以计算出不同的解的好坏。进化算法的选择操作决定了种群进化的方向。

在现实世界中存在各种各样复杂的优化问题，这些问题可能是不可导，非凸或者难以解析的。为了较好的解决这些日益重要的问题，进化算法得到了广泛的应用，例如路径规划[12–14]，生产调度[15–17]，设计[18–20]，控制[21–23] 等实际问题。在进化算法中，遗传算法和差分进化算法经常被用来解决连续优化问题[22–28]。其中差分进化是由 Storn 和 Price 首先提出的[26]，是一个有竞争力和流行的进化算法，特别是在复杂的数值优化问题中表现优异。相对于遗传算法，差分进化引入了差分向量，考虑到不同个体间的差分向量可能包含了目标函数值下降时变量的变化方向，差分进化一般具有更快的收敛速度。

进化算法一般包含如下操作：

(1) 初始化种群，一般会随机生成，初始个体均匀的遍布整个搜索空间。

(2) 评估每个个体的适应度。

(3) 选择，根据个体的适应度，从父代和子代中选择出一部分个体组成新种群。

(4) 交叉，重组两个不同的个体，从而在个体间传递遗传信息。

(5) 变异，给当前的种群添加一些未知的扰动，增加种群的多样性，探索更多的未知区域。

如图 1-1所示，进化算法是一个迭代的过程，在得到被评估后的初始种群后，便进入了迭代的进化周期，每一代的个体都会先后经历评估，选择，交叉和变异，然后一部分个体顺利的进入下一个周期，剩下的被淘汰。

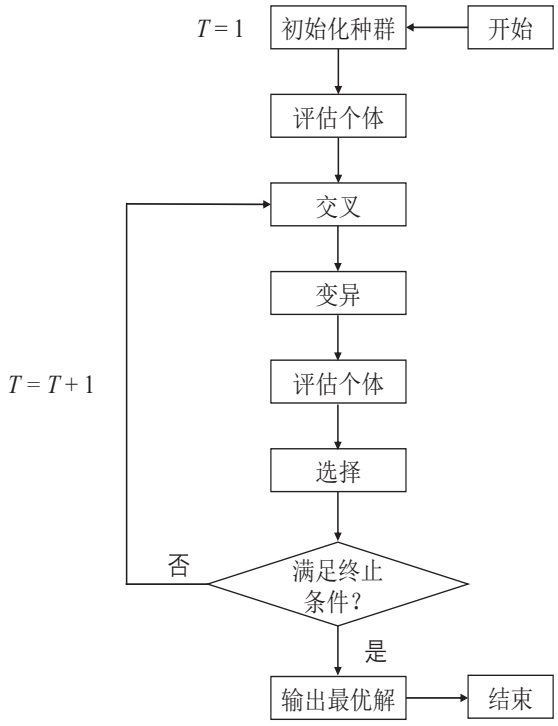


图 1-1 经典进化算法的流程图。T是进化的代数。

1.2 多任务学习

多任务学习是机器学习的子领域，它的主要思想是使用多个相关任务中有用的信息去改善这些任务的泛化性能。一般来说，机器学习算法通常需要大量的带标签的训练样本才能学到较准确的模型，但在很多情况下，我们需要花费较大的代价才能获得带标签的训练样本，例如医学图像分析，所以不得不考虑用较少的被标记的样本训练出足够准确的模型。为了解决这个问题，多任务学习不仅使用了当前任务的训练数据，还使用了相关任务中的训练数据，学习其中有用的信息。这种任务间有效的信息迁移可以提升模型的泛化性能[29]。为了准确的描述多任务学习，文献[29]中给出了如下定义：

定义 1.1 (多任务学习)： 假设有m个学习任务，这些任务或其子集是相关的。多任务学习的目的是使用这m个任务中包含的知识去改善任务T_i中模型的学习过程。

在现有的文献中，主要有如下几种方法：特征变换的方法；特征选择的方法；低秩的方法；任务聚类的方法；学习任务相关性的方法；鲁棒的多任务学习。

1.3 多任务进化算法

由于传统的进化算法往往一次只解决一个优化问题，即便是需要同时完成多个优化任务也只能分开执行。尤其是在这多个任务之间存在相关性或互补性时，这样独立地完成每个任务会导致信息和资源的浪费。为了提高进化算法在多任务场景下的效率，多任务进化算法可以同时执行多个相关的优化任务，并在优化的过程中利用任务间的相关性或互补性加速收敛。多任务进化算法的意义在于，它能够减少进化算法在解决多个优化任务时所需的计算代价，并能得到更好的最优解，这对于进化算法的应用是至关重要的。

对于单目标的多任务进化算法，目前有少量的文献对其进行了研究和应用，主要关注如下几个方面：

(1) 如何有效地利用任务间的相关信息。Gupta等人率先将多任务学习的思想引入进化算法，并且在多因素遗传模型的启发下提出了多因素进化算法(MFEA)，用于同时执行多个优化任务。在多因素进化算法中，种群中的每个个体都只属于其中一个任务，遗传因素确定了个体的值，而文化因素则确定了个体所属的任务[30]。Yuan等人提出了两个新的改进用于多因素进化算法，即一个新的统一表示方式和新的幸存者选择程序[31] 。Bali等人提出了一个线性域适应策略改进多因素进化算法，其主要思想为，将简单任务的搜索空间转变成与其对应的复杂任务相似的搜索空间，这个高阶的表示空间具备与复杂任务更高的相关性，从而给个体之前信息的迁移提供了更好的平台，减少迁移过程中信息的损失[32]。Liew等人提出了基于生物共生进化的一般性框架，可以同时处理较多任务，并且用对应的测试问题验证了该方法或

许优于多因素进化算法[33]。Ding等人提出了决策变量变换和混合策略去促进优化问题间的信息迁移，特别是这些问题的有不同的最优解或不同数量的决策变量时[34]。Feng等人采用了自动编码去显式地迁移任务间的知识，通过自动编码学习任务之间的关系，从而更有效地获取其他任务的信息，并且在单目标和多目标多任务进化问题上均取得了较好的结果[35]。Hashimoto等人将多因素进化算法看作一个特殊的岛屿模型，并且提出了一个基于标准岛屿模型的更简单的多任务进化算法[36]。

(2) 如何给不同的任务分配计算资源。Gong等人提出了使用动态资源分配策略的多任务进化算法，其主要思想通过在线地估计每个任务需要的计算资源，更加合理地分配计算资源，将较多的资源分配给更困难的任务，以取得更好的整体性能[37]。

(3) 如何动态地调整在任务间迁移的信息量。Li等人对多因素进化算法的子种群采用不同的自适应的随机交配概率，从而动态地控制了不同任务间信息迁移的数量[38]。Wen等人提出了两点改进用于多因素进化算法，分别为分离检测和资源重分配，表明了这种方法能取得更好的结果，特别是这些任务之间存在较低的相似度时[39]。

(4) 如何与其他的优化算法结合。Cheng等人基于协同多任务进化算法提出了一个粒子群优化算法，用于解决并行的全局优化任务，并且通过人造函数和现实世界中复杂的工程设计验证了此算法的性能[40]。Chen等人基于协作的协同进化模因算法提出了一个进化多任务单目标优化算法，并且通过使用拟牛顿法来加速其收敛速度[41]。Feng等人采用差分进化和粒子群优化作为多因素进化算法的基础，验证了多因素进化算法的普适性[42]。

(5) 如何在新的应用中使用。Sagarna等人将多任务进化算法用于并行的搜索软件测试中的分支，也是第一次同时优化两个以上的现实任务[43] 。Tang等人将进化多任务应用于进化模块化的拓扑极限学习机[44]。Li等人将多因素进化算法应用至稀疏重构，通过信号重建和高光谱图像分离实验验证了算法的有效性[45]。Bao等人将多因素进化算法用于云计算系统中的服务组合问题，即找出一组最优的，符合条件的原服务，然后分配给云计算系统[46]。

从上述文献中可以得知，这些方法都采用了统一表示，将不同任务的解映射至共同的统一搜索空间，然后在这个空间中完成信息的迁移。通过分析统一表示的原理，不难发现统一搜索空间中最优解的位置与变量的取值范围有关。也就是说，如果两个任务具有不同的变量范围，即使它们在原始空间中的适应景象是完全相同的，它们的最优解在统一搜索空间中的位置也会不同。在缺少先验知识的情况下，正确地设置变量的取值范围是非常有挑战性的，但是这对于现有的多任务进化算法来说却是十分重要的。而且，即便我们能正确地给定变量的取值范围，不同任务的最优解在统一搜索空间中的位置也可能是不同的。因此，考虑任务间的偏差是有意义的。

1.4 本文主要研究内容

本文主要研究多任务进化算法在单目标连续优化问题中的应用，首先总结并分析了已有算法的原理和特点，然后发现了这些算法中普遍存在且被忽视的问题，即统一表示可能会带来较大的偏差。在很多情况下，为了避免不必要的偏差，我们需要较多的先验知识来正确地设置变量的取值范围。除此之外，变量的取值范围也可能是固定的。这些问题的存在会严重影响这些算法的性能和实用性，我们需要一个更加实用且高效的多任务进化算法，为此，我们首先提出了一种新的容易实现的多任务遗传算法（MTGA），它能够估计并使用任务间的偏差，并且采用了新的多任务优化框架。在种群的进化过程中，MTGA会利用各个种群中最好的一部分个体，在线地估计任务间的偏差，然后在迁移个体时减去这个估计的偏差，从而使被迁移的个体更加适应于当前的任务。不仅如此，MTGA还能更好地利用种群进化后得到的信息，也就是，在种群进化一代后，其中最好的一部分个体会被立即迁移至其他种群，然后再依次优化其他种群。这样做的目的是，在单个周期内，一部分个体能够进化多次，因为它们会被多次迁移进新的种群，每迁移一次，则进化一次。除了MTGA，我们还提出了多任务差分进化（MTDE），MTDE在MTGA的框架下使用差分进化来优化每个种群。最后，我们首次将多任务优化应用于模糊控制器设计，并验证了我们提出的MTGA和MTDE能取得更好的效果。

全文共四章，第二章至第四章为本文的主要内容，是作者硕士阶段，在基于多任务进化算法的连续优化研究中的主要工作。论文的主要内容概括如下：

第二章介绍了我们提出的MTGA和已有的多因素进化算法，然后通过大量的实验比较了MTGA与现有的八种算法，实验结果表明MTGA具有最好的性能。另外，通过修改测试问题中变量的取值范围，我们还探究了任务间的偏差对这些算法的影响，实验结果表明，相对于其他算法，MTGA是更加鲁棒的。

第三章介绍了差分进化，多因素差分进化和我们提出的MTDE，并且通过广泛的实验验证了，MTDE具有优异的性能，在多个测试问题中均取得了最好的结果。然后进一步探究了MTGA和MTDE中估计偏差时的精度是如何变化的，通过实验发现，它们的精度一般会随着种群的进化变得越来越准确。

第四章介绍了模糊控制器设计中的应用，我们首次将多任务优化应用于模糊控制器设计，然后通过仿真验证了MTGA和MTDE能够更好地优化一型和区间二型模糊控制器。

2 基于遗传算法的多任务进化算法

经典的遗传算法[8,47]已经在连续优化领域得到了广泛应用，但是大部分文献只专注于一次解决一个优化任务。为了进一步的提高这些算法的优化效率，近几年有少部分多任务进化算法被提出。本章介绍了一些基于遗传算法的代表性工作，包括了我们最近提出的多任务遗传算法。

2.1 多因素进化算法

多因素进化算法[30] (MFEA)是最早，最流行的多任务进化算法，目前已有的大部分算法都是以此为基础做出不同的改进和应用。我们首先假设有M个待优化的任务，它们都是最小化问题。第m个任务记为T_m，T_m在搜索空间$\mathbf{X}_m$中的目标函数为$f_m : \mathbf{X}_m \to \mathbb{R}$。每个任务还可能被少数几个等式或不等式约束，同时满足所有约束的解为可行解。MFEA的目标为找出下列解集：

$$\{\mathbf{x}_1^*, ..., \mathbf{x}_M^*\} = \{\arg\min_{\mathbf{x}_1} f_1(\mathbf{x}_1), ..., \arg\min_{\mathbf{x}_M} f_M(\mathbf{x}_M)\}, \tag{式 2.1}$$

其中$\mathbf{x}_m$是$\mathbf{X}_m$, $m = 1, ..., M$中的可行解.

MFEA采用了如下重要的定义[30]：

定义 2.1： 对于给定的任务T_m，$\mathbf{x}_n$的 *factorial cost* Ψ_{nm} 为$\Psi_{nm} = \lambda \cdot \delta_{nm} + f_m(\mathbf{x}_n)$, 其中$\lambda$是一个大的惩罚乘子，$f_m(\mathbf{x}_n)$和$\delta_{nm}$分别是目标函数值和总的违反约束值。

定义 2.2： $\mathbf{x}_n$在任务T_m上的 *factorial rank* r_{nm} 是$\mathbf{x}_n$在种群中的序号，种群中的个体根据 factorial cost Ψ_{nm} 从小到大排列。

定义 2.3： $\mathbf{x}_n$的技能因子（*skill factor*）τ_n 表示了其所属的任务序号，也就是说，$\mathbf{x}_n$在所有任务中的最小 factorial rank 对应的任务为T_{τ_n}，$\tau_n = \arg\min_m r_{nm}$。

定义 2.4： $\mathbf{x}_n$在多任务场景下的适应度标量（*scalar fitness*）是$\phi_n = 1/r_{n,\tau_n}$。

MFEA对所有任务都使用了统一搜索空间（unified search space），这样可以使每一个个体都能被任意一个任务评估。假设任务T_m的搜索空间维度为d_m，则统一搜索空间的维度为$d = \max_m d_m$。一个共同的空间有利于不同任务之间信息的传递，这是至关重要的。如果我们能事先知道不同搜索空间中变量的对应关系，我们便可以根据这种关系设置特定的基因序列，以保持这种关系。否则，对于任务T_m，我们仅仅使用前d_m个变量作为被评估的个体。

在MFEA中，个体的每一个变量都被线性映射（编码）成$[0,1]$区间内的基因值[48]，这种方式被称为统一表示(*unified representation*)[30]。MFEA 还记录了每个任务中每个变量的取值范围。假设统一表示为g，它的范围是$[l,u]$。然后在个体的适应度评估阶段，首先需要将g 映射（解码）成对应的搜索空间内的值$l+g(u-l)$，然后才能计算个体对应的适应度。

MFEA的伪代码如算法 1所示。它首先在统一搜索空间中随机地生成N个个体，用来组成初始种群P，同时给每一个个体分配一个技能因子，确保每个任务中的个体数量相同，并计算每个个体的适应度标量。MFEA的其他部分类似于普通的进化算法，除了需要确定每一个子代的技能因子。

在MFEA中具有相同技能因子的个体可以被视为一个子种群，染色体交叉可以在不同的子种群间交换遗传信息。类似于生物界中的杂交现象，MFEA能够潜在地借用其他子种群中个体的优良性状，去改善当前子种群的适应度，从而得到更健壮的个体，特别是当不同的任务在统一搜索空间中具有相同的最优解或相似的适应度景象时。

Algorithm 1: MFEA[30,49]的伪代码。

Input: M 个任务$\{T_m\}_{m=1}^{M}$;
N，种群大小;
K，最大的进化代数;
rmp，选择交配概率。

Output: $\mathbf{x}_m^*$，每个任务$T_m, m = 1, ..., M$的最优解。

随机地产生N 个个体$\{\mathbf{x}_n\}_{n=1}^{N}$组成初始化种群$P$;
for $n = 1, ..., N$ **do**
    分配技能因子$\tau_n =$ mod $(n, M) + 1$给$\mathbf{x}_n$;
    仅仅评估$\mathbf{x}_n$在任务T_{τ_n}中的目标函数值;
    计算$\mathbf{x}_n$的适应度标量;
end
设置$k = 1$;
while $k \leq K$ **do**
    初始化子代种群$O = \emptyset$;
    for $n = 1, ..., N/2$ **do**
        从P中随机地选出两个个体$\mathbf{x}_a$和$\mathbf{x}_b$作为父代;
        在$[0, 1]$中产生一个随机值r;
        if $\tau_a = \tau_b$或$r < rmp$ **then**
            对$\mathbf{x}_a$和$\mathbf{x}_b$执行交叉操作，得到两个子代$\mathbf{x}_e$和$\mathbf{x}_f$;
            $\mathbf{x}_e$变异，$\mathbf{x}_e$随机地继承$\mathbf{x}_a$或$\mathbf{x}_b$中的技能因子;
            $\mathbf{x}_f$变异，$\mathbf{x}_f$随机地继承$\mathbf{x}_a$或$\mathbf{x}_b$中的技能因子;
        else
            $\mathbf{x}_a$变异后得到一个子代$\mathbf{x}_e$;
            $\mathbf{x}_e$继承$\mathbf{x}_a$的技能因子;
            $\mathbf{x}_b$变异后得到一个子代$\mathbf{x}_f$;
            $\mathbf{x}_f$继承$\mathbf{x}_b$的技能因子;
        end
        根据它们的技能因子评估$\mathbf{x}_e$和$\mathbf{x}_f$;
        计算$\mathbf{x}_e$和$\mathbf{x}_f$的适应度，将$\mathbf{x}_e$和$\mathbf{x}_f$添加至子代种群O;
    end
    从$P \cup O$选出最好的N个个体组成新种群P;
    $k = k + 1$;
end
Return 每个任务$T_m, m = 1, ..., M$的最优个体。

2.2 多任务遗传算法

多任务遗传算法（MTGA）是我们最近提出的一种新的多任务进化算法，为了方便阐述，我们这里仅考虑两个任务的情况，也就是$M = 2$。当任务数超过两个时，也能够直接应用。MTGA的伪代码如算法 2 所示。

2.2.1 MTGA的起源

当每个任务的最优解不同时，MFEA可能不能提供有效的帮助，特别是如果统一搜索空间中不同任务的最优解或适应度景象显著不同。不幸的是，在实际应用中，我们一般不能提前知道这些任务的相似度，我们需要一个稳定的多任务优化算法，它在最困难的场景下也能取得较好的性能，即这些任务是显著不同的情况下。

MTGA可以用来解决上述问题，它尝试解决两个重要的问题: 1) 怎样去估计两个任务的最优解的不同；2) 怎样在两个种群（任务）之间有效地迁移最好的个体。MTGA的主要思想是估计两个任务的最优解之间的偏差，并在迁移个体时消除这个偏差，从而使这两个任务的最优解在迁移后变得更近。按照这种方式，在一个任务中表现很好的个体也能被转变成另一个任务中表现很好的个体，加速算法的收敛过程。

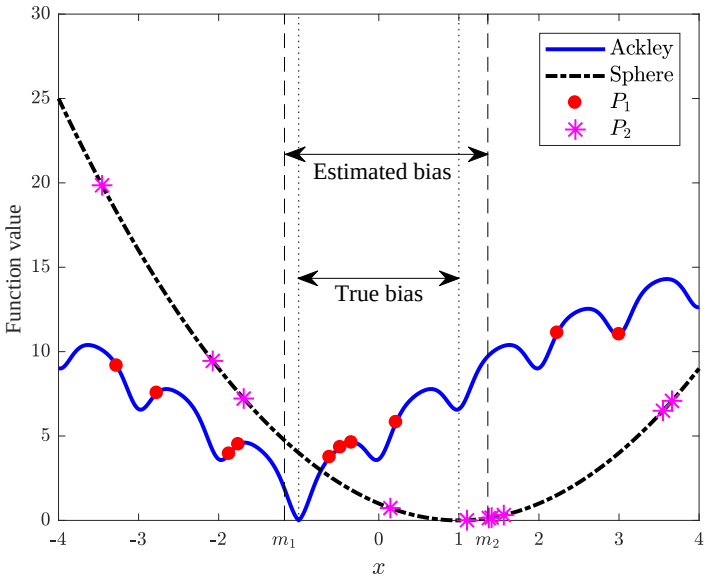


图 2-1 阐述了MTGA如何估计两个任务之间的偏差。P_1 (P_2)是一维Ackley (Sphere)的种群。$\mathbf{m}_1$ ($\mathbf{m}_2$)是P_1 (P_2)中最好的四个个体的均值。$|\mathbf{m}_2 - \mathbf{m}_1|$是估计的偏差。

MTGA估计偏差的方法可以用图 2-1来解释，假设待优化的两个任务的目标函数分别为一维的Ackley和Sphere函数。种群P_1的优化目标为Ackley，种群P_2的优化目标为Sphere。P_1 和P_2分别有10个个体。从图 2-1中可以清楚地看出这两个任务的最优解存在较大的偏差，如果盲目的将P_1中好的个体迁移至P_2（或者从P_2至P_1），很可

能不能取得好的结果。MTGA首先分别计算每个种群中少部分最好的个体的平均值（本例中的数量为4），然后将这些均值的差值作为最优解偏差的估计值。当从一个种群中迁移一个有希望的个体至其他种群中时，MTGA会给这个个体加上（或者减去，取决于迁移的方向）对应的偏差，使其更加适应于新任务。在进化的前几代中，估计的偏差或许不是非常准确，因为在每个种群中最优的一部分个体或许远离全局最优解。然而，随着种群的不断进化，最好的个体将会逐渐移向对应的全局最优解，因此，估计的偏差将会更加准确，也会更加有利于知识的迁移。

为了有效地在两个种群间迁移最好的个体，MTGA在每个进化周期内采用了有序的迁移方式，也就是，P_1首先将它的最好的一部分个体迁移至P_2（在消除偏差之后），然后P_2再通过交叉，变异，适应值评估和选择操作得到新的P_2。新的P_2中最好的一部分个体再迁移至P_1（在消除偏差之后），然后P_1再通过交叉，变异，适应值评估和选择操作得到新的P_1。通过这种方式，在同一进化周期内，单个种群中被更新的最好的一部分个体，能够立即地被同一代中的其他种群使用，从而加速收敛。相反地，MFEA在每一代中同时地迁移个体，即两个任务同时地迁移它们最好的一部分个体给对方（不考虑偏差），然后分别执行交叉，变异，适应值评估和选择操作。这会导致单个任务被更新后的最好的一部分个体，在进入下一代前无法被所有任务共享，从而浪费有用的信息。MTGA与MFEA的不同如图 2-2所示。

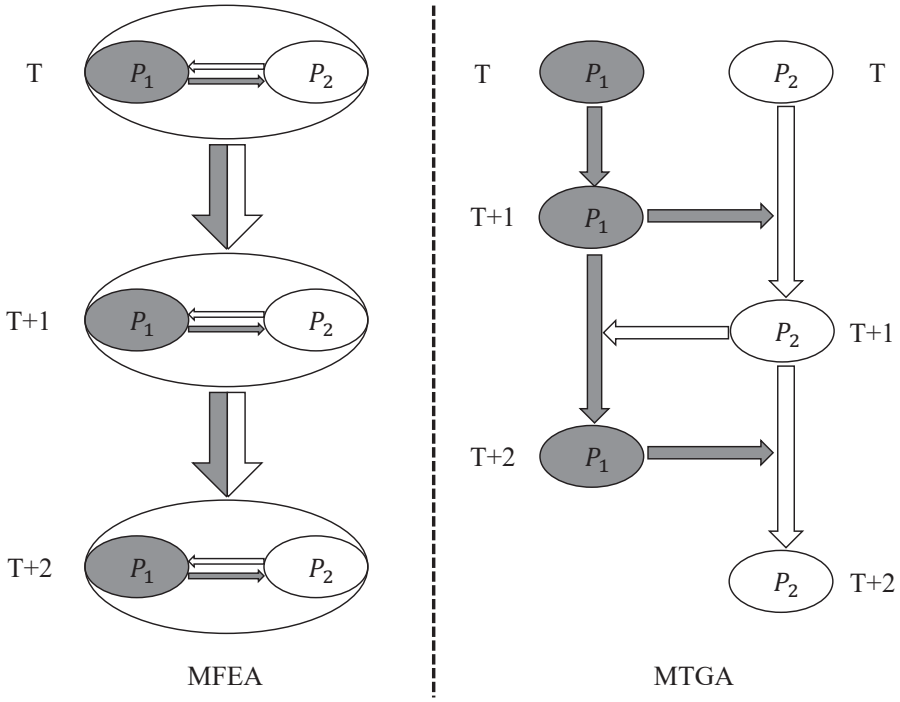


图 2-2 MTGA和MFEA的流程示意图。T 是进化的代数。

Algorithm 2: MTGA的伪代码。

Input: 两个任务T_1和T_2;
N，种群大小;
K，最大的进化代数;
n_t，迁移个体的数量.
Output: $\mathbf{x}_m^*$，每个任务$T_m, m = 1, 2$的最优解。
for $m = 1, 2$ **do**
随机地产生N个个体$\{\mathbf{x}_{m,n}\}_{n=1}^{N}$组成任务$T_m$的初始化种群$P_m$;
计算P_m中每个$\mathbf{x}_{m,n}$的适应度;
根据个体的适应度将P_m中的所有个体按降序排序;
end
设置$k = 1$;
while $k \leq K$ **do**
根据式 2.2计算$\mathbf{m}_1$和$\mathbf{m}_2$;
for $m = 1, 2$ **do**
初始化一个临时种群$P_t = \emptyset$;
根据式 2.4，用P_{3-m}中最好的n_t个个体组成P_t的前n_t个个体;
用P_m中最好的$N - n_t$个个体组成P_t中剩余的$N - n_t$个个体;
构建一个指针向量S，作为$[1, ..., N]$的随机序列;
初始化子代种群$O = \emptyset$;
for $n = 1, ..., N/2$ **do**
挑选两个父代个体$\mathbf{x}_{m,S(n)}$和$\mathbf{x}_{m,S(N/2+n)}$;
根据式 2.6和 2.7交叉产生两个子代个体$\mathbf{x}_e$和$\mathbf{x}_f$;
对$\mathbf{x}_e$和$\mathbf{x}_f$执行变异;
将$\mathbf{x}_e$和$\mathbf{x}_f$添加至O;
end
评估O中每个个体的适应度;
设置$P = P_m \cup O$;
根据个体的适应度将P中的个体按降序排序;
用P中最好的N个个体组成新的种群P_m;
end
$k = k + 1$;
end
Return 每个任务$T_m, m = 1, 2$的最优个体。

2.2.2 种群初始化

MFEA在进化的过程中始终保持单个种群，通过个体的技能因子确定个体所属的任务。而MTGA给每个任务都分配一个种群。

假设N是每个任务中种群的个体数量，T_m $(m = 1, 2)$ 的种群记为P_m。MTGA随机初始化第一代的P_m。

2.2.3 个体迁移

在后续的每一轮迭代中都会考虑个体迁移。我们首先分别计算P_1和P_2中最好的n_t个个体的均值，并标记为$\mathbf{m}_1$和$\mathbf{m}_2$。$\mathbf{m}_1$和$\mathbf{m}_2$的差值代表了两个任务间的偏差，使用偏差能够使被迁移的个体与新种群更一致。

以T_1为例. 我们从P_2中迁移n_t个最好的个体至P_1中, 替换掉P_1中最差的n_t个个体。假设分别将P_1和P_2中的个体根据适应值从好到坏排列。分别定义这些被排序的个体为$\{\mathbf{x}_{1,m}\}_{m=1}^{M}$和$\{\mathbf{x}_{2,m}\}_{m=1}^{M}$。然后计算$\mathbf{m}_1$和$\mathbf{m}_2$。

$$\mathbf{m}_1 = \frac{1}{n_t}\sum_{m=1}^{n_t}\mathbf{x}_{1,m}, \quad \mathbf{m}_2 = \frac{1}{n_t}\sum_{m=1}^{n_t}\mathbf{x}_{2,m} \tag{式 2.2}$$

紧接着，我们构建一个临时种群P_t为:

$$P_t = [\mathbf{x}'_{1,1}, ..., \mathbf{x}'_{1,n_t}, \mathbf{x}_{1,1}, ..., \mathbf{x}_{1,N-n_t}], \tag{式 2.3}$$

其中 $\mathbf{x}'_{1,m}$ $(m = 1, ..., n_t)$ 是一个从P_2中迁移出来的个体, 按如下的方式计算.

假设$d_m = |\mathbf{X}_m|$是任务T_m的搜索空间的维度，或者等于P_m, $m = 1, 2$中单个个体的基因数量。如果$d_1 \geq d_2$, 我们则从d_1个位置中随机地选取d_2个位置来构建一个指针集合I。如果$d_1 < d_2$, 我们则从d_2个位置中随机地选取d_1个位置来构建一个指针集合I。然后，

$$\mathbf{x}'_{1,m}(i) = \mathbf{x}_{2,m}(I(i)) - \mathbf{m}_2(I(i)) + \mathbf{m}_1(i), \quad m = 1, ..., n_t \tag{式 2.4}$$

其中$\mathbf{x}'_{1,m}(i)$是$\mathbf{x}'_{1,m}$的第i个基因，$\mathbf{x}_{2,m}(I(i))$是$\mathbf{x}_{2,m}$的第$I(i)$个基因，$\mathbf{m}_2(I(i))$是 $\mathbf{m}_2$的第$I(i)$个基因，$\mathbf{m}_1(i)$ 是$\mathbf{m}_1$的第i个基因。

注意I是分别对每个被迁移的个体随机构建的，即$\mathbf{x}'_{1,m}$的第i个基因是随机地对应于$\mathbf{x}_{2,m}$的一个基因（不一定是第i个基因），并且不同的m的对应关系也是不同的。由于在实际应用中，我们一般不知道T_2的哪个基因会最有利于T_1的基因，所以我们采用了随机的匹配方式，而不是固定的。随机地对不同的m采用不同的匹配关系能够增加种群的多样性，并且更有可能找到一个更好的匹配关系，相对于盲目的固定匹配关系。

一旦$\mathbf{x}'_{1,m}(i)$和$\mathbf{x}_{2,m}(I(i))$的匹配关系被建立起来，$\mathbf{m}_1(i)-\mathbf{m}_2(I(i))$便代表了两个任务中两个基因间的偏差，从$\mathbf{x}_{2,m}(I(i))$中减去这个偏差会使两个任务中的基因更加一致，从而促进了知识的迁移。

2.2.4 交叉和变异

接下来执行交叉操作，并且确保n_t个从P_2中迁移出的个体都被使用在交叉操作中。

我们定义了一个指针向量，作为$[1,...,N]$的随机排列。我们每次挑选两个父代个体$\mathbf{x}_{S(i)}$和$\mathbf{x}_{S(i+N/2)}$ $(i=1,...,N/2)$，然后使用模拟二进制交叉操作（SBX）[50]和多项式变异[51]，和文献[49]中使用的相同。

假设两个父代个体分别为$\mathbf{x}_{S(i)}=[x_a^1,...,x_a^d]$ 和$\mathbf{x}_{S(i+N/2)}=[x_b^1,...,x_b^d]$，$d$是搜索空间的维度。然后在SBX中，我们首先计算：

$$c^i=\begin{cases}(2r)^{1/(\beta+1)}, & r\le 0.5\\ [2(1-r)]^{-1/(\beta+1)}, & r>0.5\end{cases},\quad i=1,...,d \tag{式 2.5}$$

其中β是用户自定义的参数，r是$[0,1]$中的随机数，对每个i都随机生成。通过SBX获得的两个子代个体，$\mathbf{x}_e=[x_e^1,...,x_e^d]$和$\mathbf{x}_f=[x_f^1,...,x_f^d]$是:

$$x_e^i=[(1+c^i)x_a^i+(1-c^i)x_b^i]/2,\ i=1,...,d \tag{式 2.6}$$

$$x_f^i=[(1+c^i)x_b^i+(1-c^i)x_a^i]/2,\ i=1,...,d \tag{式 2.7}$$

不难发现，$\mathbf{x}_e+\mathbf{x}_f=\mathbf{x}_{S(i)}+\mathbf{x}_{S(i+N/2)}$。同时也容易观察到$\mathbf{x}_e$更靠近$\mathbf{x}_{S(i)}$，$\mathbf{x}_f$更靠近$\mathbf{x}_{S(i+N/2)}$，因为$c^i>0$。

η是用户自定义的参数，r是$[0,1]$中的随机数。然后对范围为$[l,u]$的基因g执行多项式变异，按如下公式计算[51]：

$$g'=\begin{cases}g+[(2r)^{1/(1+\eta)}-1](g-l), & r\le 0.5\\ g+[1-(2(1-r))^{1/(1+\eta)}](u-g), & r>0.5\end{cases} \tag{式 2.8}$$

在变异后，$\mathbf{x}_e$和$\mathbf{x}_f$会被添加至子代种群O中。上述交叉和变异操作被重复执行$N/2$次，直至O包含了N个个体。

2.2.5 选择

我们然后评估O中每个个体的适应值，混合O和P_m，根据个体的适应值从好到坏排列这$2N$个个体。对于任务T_m，我们通过精英选择机制选出前N个最好的个体进入下一代种群。

2.3 实验部分

实验部分比较了MTGA与8种当前的单任务或多任务进化算法，通过文献[49]中介绍的9个测试问题。表 2.1总结了这些测试问题，其具体描述被放置在附录 2中。

表 2.1 9 个单目标多任务测试问题[49]。

No.	Category	d_1	$[l,u]$	d_2	$[l,u]$	Intersection[1]	Similarity[2]
1	CI+HS	50	$[-100,100]$	50	$[-50,50]$	CI	1.0000
2	CI+MS	50	$[-50,50]$	50	$[-50,50]$	CI	0.2261
3	CI+LS	50	$[-50,50]$	50	$[-500,500]$	CI	0.0002
4	PI+HS	50	$[-50,50]$	50	$[-100,100]$	PI	0.8670
5	PI+MS	50	$[-50,50]$	50	$[-50,50]$	PI	0.2154
6	PI+LS	50	$[-50,50]$	25	$[-0.5,0.5]$	PI	0.0725
7	NI+HS	50	$[-50,50]$	50	$[-50,50]$	NI	0.9434
8	NI+MS	50	$[-100,100]$	50	$[-0.5,0.5]$	NI	0.3669
9	NI+LS	50	$[-50,50]$	50	$[-500,500]$	NI	0.0016

[1] CI (complete intersection): 两个任务的最优解的所有变量的统一表示都是相等的。

PI (partial intersection): 两个任务的最优解的一部分变量的统一表示是相等的。

NI (no intersection): 两个任务的最优解的所有变量的统一表示都是不相等的。

HS: 高度相似（High similarity）; MS: 中度相似（Moderate similarity）; LS: 低度相似（Low similarity）.

[2] 这里的相似性等于两个任务的解之间的Spearman次序相关系数[49]。

2.3.1 性能度量

除了每个任务的适应度，文献[49]中提出的简单的性能度量也被用来评估不同算法的性能。

假设有K个算法，$A_1,...,A_K$，被用来解决M个最小化任务$T_1,...,T_M$，每一个算法都被重复运行L次。算法A_k在第l次运行时，获得任务T_m中的最好结果表示为$I(k,m)_l$。μ_m和σ_m分别是$I(k,m)_l$, $k=1,...,K$, $l=1,...,L$的均值和标准差。按如下公式计算正则化后的性能：

$$I'(k,m)_l = \frac{I(k,m)_l - \mu_m}{\sigma_m} \tag{式 2.9}$$

算法A_m的性能分数为:

$$s_m = \sum_{k=1}^{K} \sum_{l=1}^{L} I'(k,m)_l \quad \text{(式 2.10)}$$

性能分数越小，算法的整体性能越好。

2.3.2 实验设置

我们比较了我们提出的MTGA与如下八种现有的方法:

(1) 单目标的进化算法(SOEA)，独立地考虑每个任务。本质上是使用模拟二进制交叉和多项式变异的遗传算法。

(2) MFEA。

(3) 共生的生物进化(EBS)[33]，能够处理任务数较多的问题，EBS使用的基础进化算法与SOEA和MFEA采用的遗传算法相同。

(4) MFEA-LBS[31], 采用了基于序列的统一表示和基于水平的选择去增强原始的MFEA。

(5) 资源再分配的多因素进化算法(MFEARR)[39]，使用资源分配机制促进任务间协同关系的发现和利用。

(6) 线性域适应多因素进化算法(LDA-MFEA)[32]，使用线性域适应将简单任务的搜索空间转换成一个新的与复杂任务相似的空间，有利于信息的传递。

(7) 泛化的多因素进化算法(G-MFEA)[34]，使用了决策变量转换和混合策略去促进优化问题间知识的迁移。

(8) 基于显式自编码的进化多任务(EMEA)[35]，在进化多任务的场景下使用自动编码器显式地迁移任务间的知识。

MFEA及其变化后的算法的种群大小均为200。对于SOEA, G-MTEA, EMEA和MTGA，每个任务的种群大小均为100。所有算法的最大函数计算次数都为100,000，即所有算法在进化500代后终止。在模拟二进制交叉的式 2.5中$rmp = 0.3$，$\beta = 2$。多项式变异的式 2.8中$\eta = 5$。所有算法都使用模拟二进制交叉和多项式变异。另外，根据文献[35]，EMEA中的$n_t = 10$。在MTGA中$n_t = 40$。

为了减少随机性的影响，每个算法都被运行20次，每一次都随机地产生初始种群。然后计算少数几个统计学数值，例如两个任务中目标函数的均值和标准差。

2.3.3 实验结果

9个算法在9个测试问题中的平均性能分数如图 2-3所示，单个的实验结果如图 2-4所示，从中可以观察到:

(1) 所有算法的整体性能都优于SOEA，这表明了任务之间的信息迁移确实能改善整体的优化性能。

(2) EBS和MFEARR的平均性能比MFEA差，MFEA-LBS和MFEA的大致相同。

(3) 在函数计算次数较小时LDA-MFEA具有最好的性能，但是会随着计算次数的增加而逐渐变差。

(4) 在函数计算次数较大时G-MFEA略好于MFEA。

(5) 在前面的8种算法中，EMEA在函数计算次数较小时具有第二好的平均性能（仅比LDA-MFEA差），在计算次数较大时具有最好的平均性能。

(6) 在所有的9种算法中，我们提出的MTGA在函数计算次数较小时具有第二好的平均性能（仅比LDA-MFEA差），在计算次数较大时具有最好的平均性能。

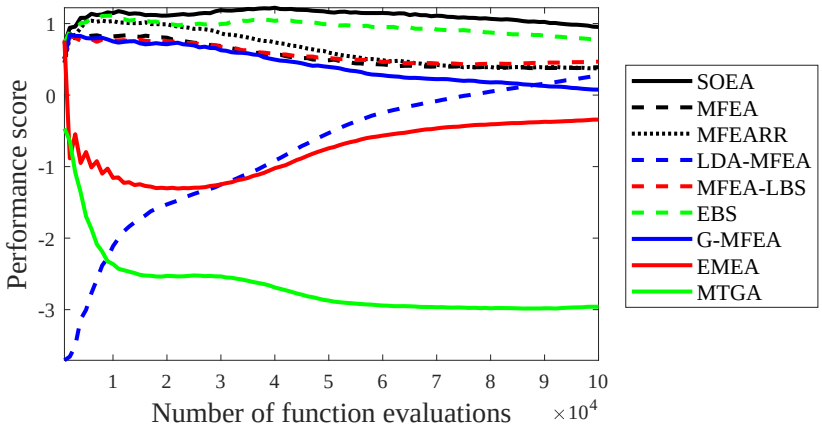


图 2-3 在9个原始的测试问题中的平均性能分数。

经过100,000次函数计算后，不同算法在不同任务中获得的平均结果（均值和标准差）如表 2.2所示。我们提出的MTGA在8个测试问题中都获得了最好的性能分数，整体上它支配了其他8种算法。根据SOEA的结果归一化不同的算法在计算100,000次函数后的均值和标准差，如图 2-5所示。平均来看，MTGA具有最小的均值和标准差，表明MTGA确实优于其他方法。

我们还进一步比较了不同算法的计算代价。对于每一个测试问题，我们用SOEA去归一化其他8个算法的计算时间，并将其结果表示在图 2-6中。SOEA，EMEA和MTGA有差不多的计算代价，它们在这9个算法中是最快的。MFEA-LBS和EBS有差不多的计算代价，并且都少于MFEA，G-MFEA和MFEARR的计算代价。LDA-MFEA的计算代价差不多是MTGA的三倍。

总之，我们可以推断MTGA是高效且稳定的。

2.3.4 MTGA的参数敏感性

除了在进化算法中被考虑的标准参数，例如种群大小，最大迭代次数，随机交配概率，我们提出的MTGA还有一个参数n_t，代表两个任务之间迁移个体的数量。我们研究了MTGA关于n_t的敏感性。

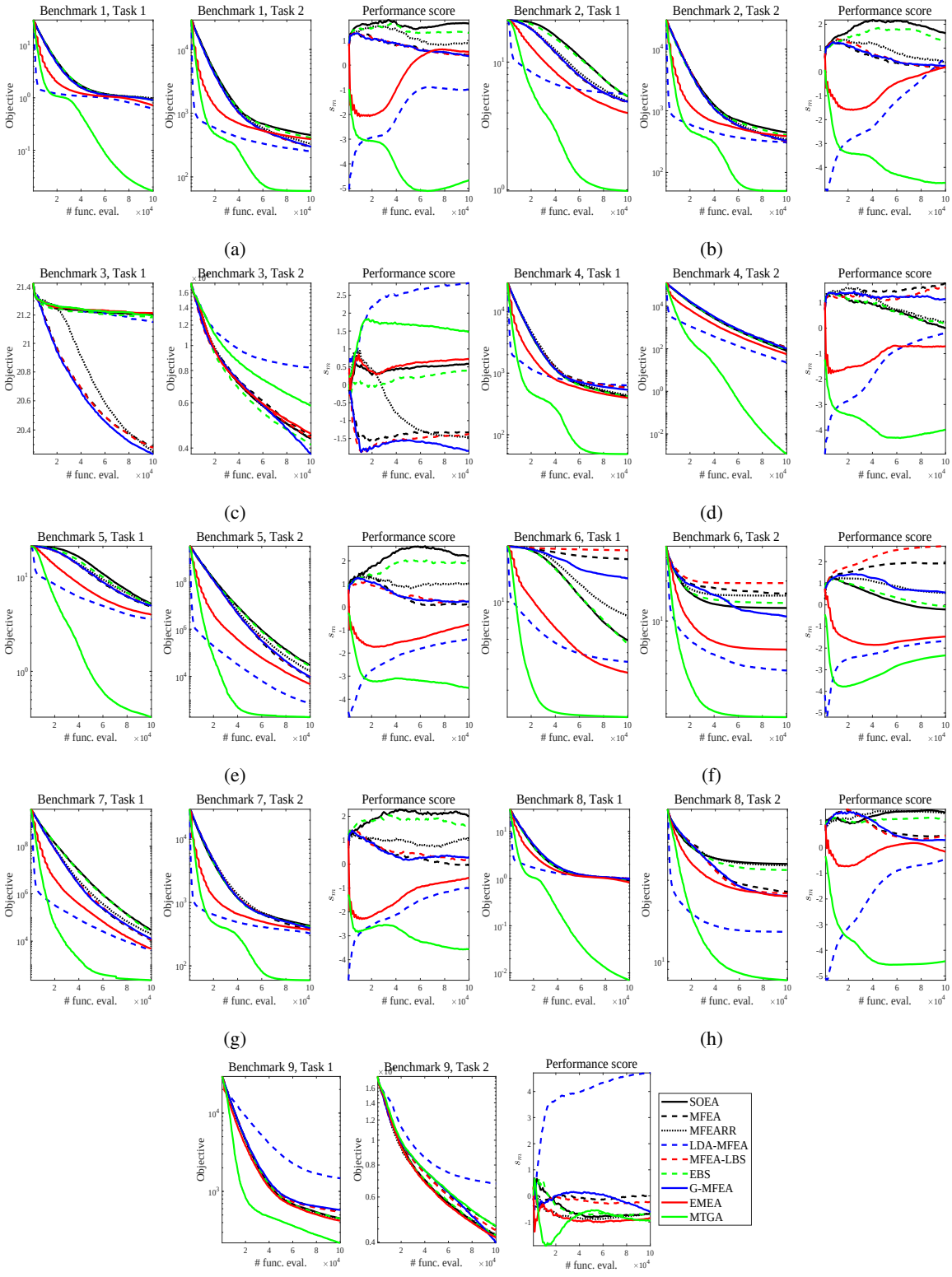


图 2-4 在9个原始的测试问题中的实验结果。(a) 测试问题 1；(b) 测试问题 2；(c) 测试问题 3；(d) 测试问题 4；(e) 测试问题 5；(f) 测试问题 6；(g) 测试问题 7；(h) 测试问题 8；(i) 测试问题 9。

表 2.2 不同的算法在**原始**的测试问题中的平均性能（均值和括号内的标准差）。最好的性能已被加粗表示。

	Benchmark		1	2	3	4	5	6	7	8	9
SOEA	T1	mean	0.90	5.37	21.21	417.82	5.25	5.91	29341.51	0.91	452.87
		std	(0.07)	(1.05)	(0.04)	(50.36)	(0.64)	(3.59)	(13428.47)	(0.05)	(63.04)
	T2	mean	453.96	446.07	4350.45	81.96	30552.12	12.53	431.07	37.41	4261.66
		std	(54.26)	(50.76)	(567.06)	(23.41)	(12233.19)	(2.51)	(51.22)	(4.28)	(438.00)
	Score		1.68	1.62	0.59	0.00	2.20	−0.21	2.01	1.37	−0.67
MFEA	T1	mean	0.93	4.88	20.28	587.56	4.87	17.87	10833.53	0.98	568.35
		std	(0.05)	(0.57)	(0.05)	(72.48)	(0.39)	(5.53)	(3463.65)	(0.04)	(85.78)
	T2	mean	294.61	316.08	4450.49	116.85	8843.01	16.11	371.84	25.67	4609.52
		std	(40.86)	(29.19)	(646.83)	(22.86)	(2198.81)	(5.17)	(49.95)	(3.48)	(495.99)
	Score		0.44	0.17	−1.33	1.77	0.12	1.92	−0.05	0.47	0.00
MFEARR	T1	mean	0.98	4.88	20.25	439.15	5.22	8.23	19331.32	0.92	448.33
		std	(0.04)	(0.37)	(0.08)	(74.14)	(0.48)	(6.15)	(8251.89)	(0.09)	(49.12)
	T2	mean	328.06	341.42	4343.43	86.20	16927.11	15.54	419.07	36.70	4286.25
		std	(30.52)	(44.22)	(586.37)	(27.31)	(6844.10)	(2.91)	(33.86)	(4.82)	(517.42)
	Score		0.88	0.39	−1.46	0.22	1.03	0.55	1.06	1.34	−0.66
LDA-MFEA	T1	mean	0.62	5.63	21.15	633.42	3.57	4.41	4250.29	1.01	1472.91
		std	(0.13)	(0.89)	(0.10)	(126.17)	(0.40)	(0.83)	(1591.87)	(0.04)	(377.93)
	T2	mean	252.72	307.06	8187.26	22.49	723.60	4.24	326.04	14.95	6773.09
		std	(67.86)	(78.78)	(2356.72)	(9.10)	(211.25)	(1.26)	(69.91)	(1.85)	(882.55)
	Score		−0.97	0.58	2.84	−0.18	−1.39	−1.67	−0.99	−0.44	4.71
MFEA-LBS	T1	mean	0.90	4.85	20.28	577.36	5.02	20.18	11263.10	0.99	545.56
		std	(0.06)	(0.43)	(0.07)	(86.99)	(0.41)	(0.08)	(3922.94)	(0.03)	(75.98)
	T2	mean	304.32	319.22	4404.16	111.66	9756.55	19.31	393.86	25.12	4424.13
		std	(42.99)	(39.12)	(620.69)	(22.30)	(3668.28)	(2.13)	(68.95)	(3.85)	(550.68)
	Score		0.43	0.18	−1.37	1.59	0.29	2.71	0.17	0.46	−0.25
EBS	T1	mean	0.89	5.40	21.18	415.14	5.17	5.68	26599.15	0.91	450.39
		std	(0.07)	(1.42)	(0.05)	(43.37)	(0.51)	(2.16)	(12441.12)	(0.05)	(51.02)
	T2	mean	412.79	401.54	4121.20	89.36	26750.32	13.68	416.40	34.45	4245.47
		std	(48.40)	(49.43)	(773.88)	(20.23)	(10428.62)	(2.95)	(43.35)	(7.94)	(569.82)
	Score		1.31	1.25	0.39	0.16	1.82	−0.07	1.65	1.09	−0.70
G-MFEA	T1	mean	0.90	4.90	**20.24**	538.32	4.98	13.77	12605.29	0.96	579.11
		std	(0.07)	(0.43)	(0.06)	(63.80)	(0.43)	(9.01)	(5139.68)	(0.05)	(106.98)
	T2	mean	298.48	325.50	**3790.29**	101.28	9254.20	10.82	395.50	24.25	**3972.57**
		std	(32.05)	(37.51)	(565.23)	(24.64)	(3145.37)	(7.18)	(45.47)	(4.25)	(503.32)
	Score		0.37	0.26	**−1.84**	1.13	0.22	0.57	0.29	0.30	−0.62
EMEA	T1	mean	0.72	4.00	21.21	398.53	4.01	3.81	4816.00	0.83	422.83
		std	(0.07)	(0.45)	(0.03)	(49.32)	(0.29)	(0.36)	(2148.75)	(0.09)	(47.21)
	T2	mean	391.74	386.59	4530.94	56.21	4667.83	6.09	370.28	24.17	4164.94
		std	(68.70)	(47.18)	(760.37)	(11.75)	(1854.26)	(2.58)	(54.69)	(4.85)	(608.49)
	Score		0.54	0.21	0.71	−0.71	−0.78	−1.47	−0.57	−0.16	−0.85
MTGA	T1	mean	**0.02**	**0.98**	21.19	**48.52**	**0.33**	**2.08**	**226.92**	**0.01**	**220.23**
		std	(0.02)	(0.67)	(0.04)	(13.06)	(0.41)	(0.49)	(404.06)	(0.01)	(63.50)
	T2	mean	**60.17**	**50.15**	5844.60	**0.00**	**187.55**	**1.89**	**59.30**	**7.77**	4602.33
		std	(19.15)	(15.13)	(719.33)	(0.00)	(339.17)	(1.81)	(19.65)	(7.07)	(492.29)
	Score		**−4.67**	**−4.66**	1.48	**−3.98**	**−3.51**	**−2.34**	**−3.56**	**−4.42**	**−0.96**

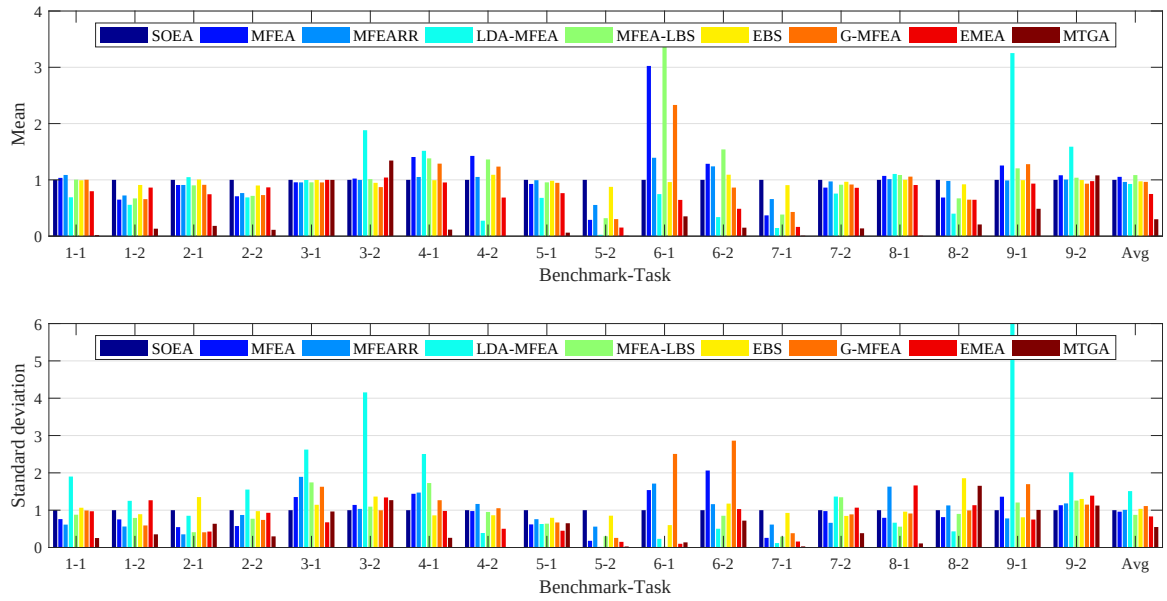


图 2-5 在原始的测试问题中不同算法归一化后的均值和标准差。

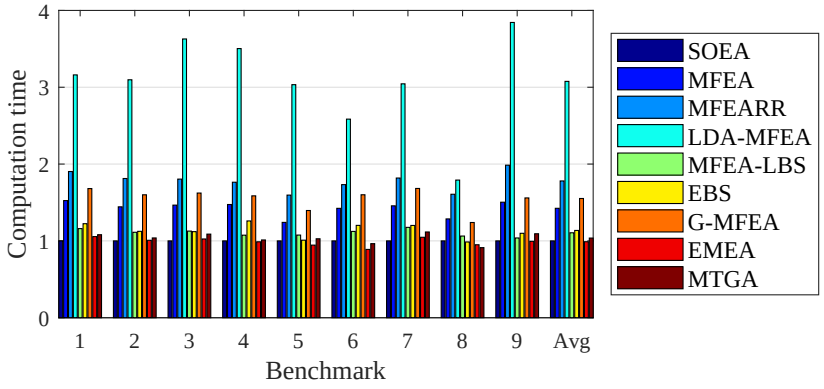


图 2-6 9个算法的计算时间。

因此，我们比较了SOEA，六个版本的MTGA（$n_t = \{10, 20, 30, 40, 50, 60\}$）和不考虑偏差的MTGA (MTGA-NB)。MTGA-NB的$n_t = 40$，它直接在两个任务之间迁移40个最好的个体。这8个算法在9个测试问题中的平均性能如图 2-7所示。可以看出，具有不同n_t的MTGA均优于SOEA和MTGA-NB，表明考虑偏差是很重要的。具有不同n_t的MTGA拥有相似的性能，表明MTGA对n_t的变化是鲁棒的。

2.3.5 更实际的测试问题

上述的大部分测试问题都是很特殊的，因为$\mathbf{x}_1$和$\mathbf{x}_2$的搜索空间都是对称的，并且$\mathbf{x}_1^*$和$\mathbf{x}_2^*$的统一表示也是完全或部分重叠的，因此我们很容易理解它们之间共享信息能够加速优化。然而在实际的搜索空间中，$\mathbf{x}_1$和$\mathbf{x}_2$的搜索空间不一定是对称的，我们或许不能提前知道$\mathbf{x}_1^*$和$\mathbf{x}_2^*$经过统一表示后是否会重叠。这里研究了一个更一般和更有挑战性的问题，并且试图去回答如下两个问题：

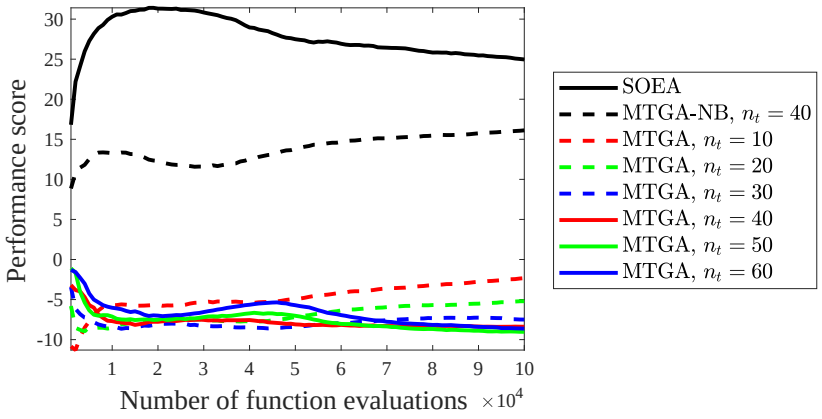


图 2-7 使用不同的n_t时，8个算法在9个原始的测试问题中的平均性能分数。

Q1: 当$\mathbf{x}_1^*$和$\mathbf{x}_2^*$的统一表示完全不重叠的时候，多任务还能有助于优化吗?

Q2: 当搜索空间变小时，多任务搜索会变得更简单吗?

出于这个目的，在每个测试问题中，我们轻微的修改了至少一个搜索空间，使得它变得不再对称，同时确保原始的全局最优解仍然位于当前的空间内，如表 2.3 所示。这个简单的修改使得我们能够回答如下问题：

表 2.3 9 个被修改后的单目标多任务测试问题。

No.	Category	d_1	$[l, u]$	d_2	$[l, u]$
1	CI+HS	50	$[-50, 100]$	50	$[-50, 50]$
2	CI+MS	50	$[-20, 50]$	50	$[-50, 50]$
3	CI+LS	50	$[30, 50]$	50	$[-500, 500]$
4	PI+HS	50	$[-20, 50]$	50	$[-100, 50]$
5	PI+MS	50	$[-10, 50]$	50	$[-50, 10]$
6	PI+LS	50	$[-10, 50]$	25	$[-0.5, 0.5]$
7	NI+HS	50	$[-20, 50]$	50	$[-50, 20]$
8	NI+MS	50	$[-20, 100]$	50	$[-0.5, 0.5]$
9	NI+LS	50	$[-20, 50]$	50	$[-500, 500]$

(1) 这会使得所有测试问题中$\mathbf{x}_1^*$和$\mathbf{x}_2^*$的统一表示完全不同，因此我们能够研究问题*Q1*。例如，对于表 2.1中的原始测试问题 1 ，所有任务的全局最优解的统一表示都是$[0.5, ..., 0.5] \in \mathbb{R}^{50}$，因此，得到更好的多任务学习性能是意料之中的。对于表 2.3中的修改后的测试问题 1，任务 1 的全局最优解仍然是$[0, ..., 0] \in \mathbb{R}^{50}$，但是它的统一表示变成了$[1/3, ..., 1/3] \in \mathbb{R}^{50}$，同时任务 2 的全局最优解的统一表示仍然是$[0.5, ..., 0.5] \in \mathbb{R}^{50}$。这两者的统一表示是不同的，表明它们的相关性更低。

(2) 修改后的搜索空间总是比原始的搜索空间更小，这使得我们能够研究问题*Q2*。直观地说，我们期望一个更小的搜索空间能使算法更容易找到全局最优解。

值得注意的是，在这9个原始的测试问题中也包含了少数几个任务之间具有低相似性的例子。因此，通过关注这些测试问题，我们也能够研究问题*Q1*，但是不能研究问题*Q2*。

9个算法在修改后的测试问题中的平均性能分数如图 2-8所示，单个实验结果如图 2-9所示，从中可以观察到：

(1) EMEA和MTEA的性能仍然优于SOEA，但是EMEA的改进更小。

(2) 当两个任务完全不同时，MFEA，G-MFEA和MFEA-LBS有差不多的性能，它们都比SOEA更差。

(3) MFEARR和EBS在原始的测试问题中比MFEA更差，在修改后的问题中优于MFEA，主要是因为MFEA变差了，而不是MFEARR和EBS的性能变好了。SOEA，MFEARR和EBS有差不多的性能。

(4) 在函数计算次数非常小时LDA-MFEA仍然具有最好的性能，但是会随着计算次数的增加而逐渐变差。

(5) 在现有的8种算法中，EMEA在函数计算次数较小时具有第二好的平均性能（仅比LDA-MFEA差），在计算次数较大时具有最好的平均性能，和在原始测试问题中的排序相同。

(6) 在所有的9种算法中，我们提出的MTGA再一次取得了最好的整体性能，并且大幅度的优于其他8种算法，表明它能有效地处理多任务连续优化问题，不管任务之间是否相似。

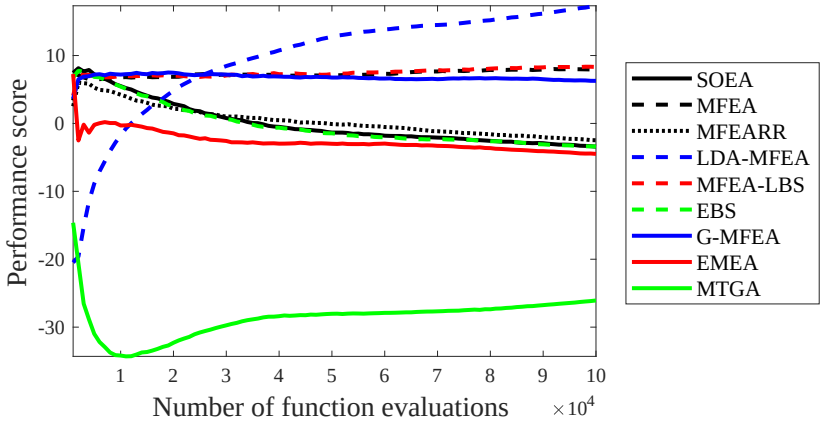


图 2-8 在9个修改后的测试问题中的平均性能分数。

表 2.4 给出了在所有被修改后的测试问题中，所有算法分别计算100,000次函数值后得到的最终的优化结果。我们提出的MTGA仍然支配其他8种算法：它在所有被修改的测试问题中都获得了最好的性能分数。

为了更好地可视化表 2.2和表 2.4的结果，我们画出了完成100,000次函数计算后最终的性能分数，如图 2-10所示，其中彩色的直条代表原始问题中的性能分数，彩色直条中间的黑色直条代表修改后的问题中对应的性能分数。因为原始的问题和修

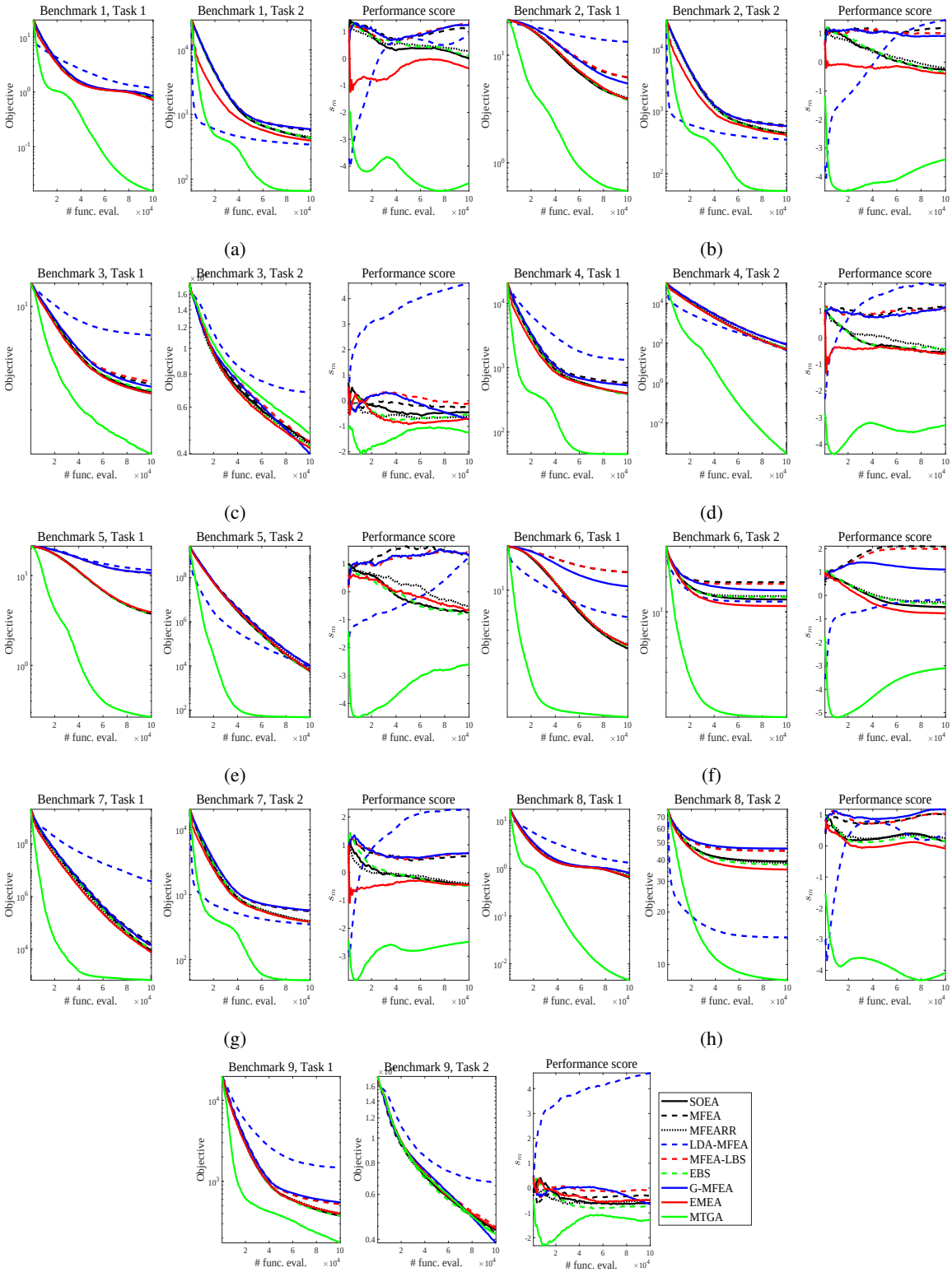


图 2-9 在9个修改后的测试问题中的实验结果。(a) 测试问题 1；(b) 测试问题 2；(c) 测试问题 3；(d) 测试问题 4；(e) 测试问题 5；(f) 测试问题 6；(g) 测试问题 7；(h) 测试问题 8；(i) 测试问题 9。

改后的问题具有相同的全局最优解，所以可以直接比较它们的性能分数。

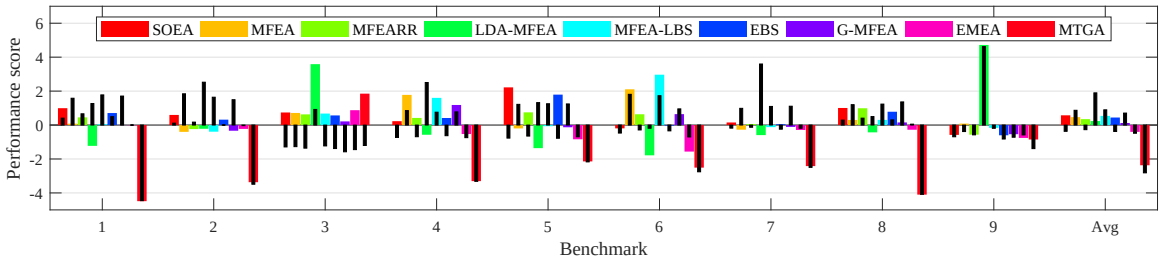


图 2-10 在原始和修改的测试问题中不同算法的性能分数（越小越好）。

从图 2-10中可以观察到：

(1) 在修改后的测试问题中的SOEA始终优于原始测试问题中的SOEA，这是合理的，因为修改后的测试问题中搜索空间更小，所以SOEA可以更广泛地搜索最优解。

(2) 在修改后的测试问题中的MTGA始终优于原始测试问题中的MTGA，同样是因为修改后的测试问题中搜索空间更小，所以MTGA可以更广泛的搜索最优解。

(3) 即使所有算法在原始的测试问题中都优于SOEA，但是它们中的大部分算法（除了EBS, EMEA和MTGA）都在修改后的测试问题中比SOEA更差。

(4) 在原始和修改后的测试问题中，我们提出的MTGA都显著的优于其他八种算法，这表明MTGA是非常鲁棒的，在实际应用中使用它会更加安全，因为我们一般不知道任务间的相似性。

总之，对于多任务连续优化任务，即使修改后的测试问题的搜索空间是更小的，但是任务之间差异的增大还是会使这些被修改后的问题更加困难。尽管如此，一种精心设计的多任务算法（如MTGA）仍然可以通过同时学习多个任务获益。无论任务之间是相似的还是显著不同的，它都是非常鲁棒的。因此，在实际应用中，应该更倾向于使用MTGA。

表 2.4 不同的算法在**修改后**的测试问题中的平均性能（均值和括号内的标准差）。最好的性能已被加粗表示。

	Benchmark		1	2	3	4	5	6	7	8	9
SOEA	T1	mean	0.77	3.93	2.64	399.16	3.75	3.66	9263.79	0.64	376.82
		std	(0.07)	(0.34)	(0.36)	(66.83)	(0.33)	(0.23)	(3912.13)	(0.09)	(58.93)
	T2	mean	430.55	442.82	4403.29	48.60	5859.80	12.79	385.53	39.05	4350.45
		std	(71.79)	(57.83)	(502.57)	(11.03)	(2932.45)	(2.76)	(65.94)	(4.09)	(567.06)
	Score		0.03	−0.26	−0.46	−0.57	−0.76	−0.50	−0.44	0.14	−0.60
MFEA	T1	mean	0.86	6.22	3.00	581.11	10.62	13.58	15293.98	0.78	538.58
		std	(0.07)	(4.62)	(0.45)	(137.72)	(7.73)	(7.86)	(6746.19)	(0.07)	(65.93)
	T2	mean	565.67	596.84	4373.45	83.01	10119.29	18.02	555.44	45.08	4230.30
		std	(76.45)	(96.40)	(405.58)	(15.62)	(3837.31)	(2.29)	(53.03)	(4.77)	(589.46)
	Score		1.13	1.20	−0.24	1.15	1.31	2.08	0.59	1.01	−0.30
MFEARR	T1	mean	0.81	3.99	2.58	393.91	3.75	3.91	7990.25	0.70	406.23
		std	(0.06)	(0.31)	(0.33)	(48.62)	(0.44)	(0.30)	(3095.45)	(0.09)	(55.67)
	T2	mean	448.07	449.02	4318.28	50.86	6970.91	13.68	393.16	38.28	4384.89
		std	(66.51)	(59.61)	(640.28)	(14.84)	(2521.57)	(2.78)	(56.09)	(4.40)	(709.05)
	Score		0.27	−0.21	−0.59	−0.51	−0.52	−0.29	−0.39	0.25	−0.49
LDA-MFEA	T1	mean	1.19	13.30	6.40	1314.05	11.55	6.28	3719548.65	1.32	1473.19
		std	(0.09)	(1.97)	(0.88)	(296.26)	(1.59)	(1.46)	(1571476.27)	(0.15)	(455.31)
	T2	mean	343.13	352.03	6810.70	44.87	9135.99	12.26	351.07	14.30	6679.11
		std	(56.63)	(66.82)	(523.28)	(13.80)	(6884.91)	(1.66)	(66.55)	(2.03)	(823.08)
	Score		0.90	1.48	4.61	1.97	1.26	−0.18	2.26	0.40	4.63
MFEA-LBS	T1	mean	0.87	6.22	3.12	541.97	10.62	13.66	12643.90	0.80	521.75
		std	(0.08)	(4.60)	(0.67)	(69.97)	(7.64)	(7.78)	(4467.27)	(0.07)	(100.43)
	T2	mean	594.45	565.74	4412.97	84.87	10451.12	17.45	572.25	44.96	4460.62
		std	(84.75)	(72.83)	(502.55)	(16.17)	(3447.07)	(2.07)	(78.36)	(5.67)	(524.24)
	Score		1.32	1.02	−0.12	1.10	1.38	1.98	0.69	1.04	−0.11
EBS	T1	mean	0.79	3.84	2.71	380.83	3.78	3.80	9612.78	0.69	374.77
		std	(0.11)	(0.35)	(0.38)	(61.55)	(0.40)	(0.43)	(4514.29)	(0.07)	(54.76)
	T2	mean	435.82	430.62	4264.97	54.93	5723.13	13.45	376.40	37.42	4227.34
		std	(53.21)	(56.74)	(590.30)	(15.85)	(2708.37)	(2.39)	(52.51)	(3.67)	(466.16)
	Score		0.11	−0.36	−0.55	−0.41	−0.78	−0.35	−0.49	0.15	−0.73
G-MFEA	T1	mean	0.85	5.45	2.87	531.12	10.72	10.68	13996.14	0.81	550.45
		std	(0.06)	(3.37)	(0.34)	(69.63)	(7.60)	(7.73)	(5540.70)	(0.05)	(89.98)
	T2	mean	589.90	580.61	**3983.81**	87.01	10134.47	15.34	573.89	46.35	**3873.61**
		std	(86.07)	(81.14)	(620.70)	(14.20)	(2998.57)	(4.06)	(98.15)	(5.42)	(556.78)
	Score		1.25	0.92	−0.72	1.14	1.33	1.11	0.70	1.17	−0.64
EMEA	T1	mean	0.72	3.89	2.60	401.67	3.90	3.87	7580.20	0.66	399.54
		std	(0.06)	(0.30)	(0.32)	(56.81)	(0.38)	(0.32)	(2041.63)	(0.06)	(81.39)
	T2	mean	394.25	421.56	4197.27	47.74	6247.03	11.22	384.50	35.10	4397.57
		std	(61.00)	(54.68)	(500.80)	(9.33)	(2936.33)	(2.33)	(49.69)	(4.16)	(549.19)
	Score		−0.37	−0.40	−0.70	−0.59	−0.65	−0.76	−0.45	−0.09	−0.49
MTGA	T1	mean	**0.02**	**0.55**	**1.01**	**45.38**	**0.27**	**1.12**	**663.25**	**0.00**	**174.32**
		std	(0.01)	(0.62)	(0.57)	(12.35)	(0.42)	(0.62)	(713.91)	(0.00)	(56.40)
	T2	mean	**65.58**	**53.00**	4763.65	**0.00**	**47.87**	**1.23**	**48.42**	**8.12**	4198.22
		std	(22.90)	(16.99)	(816.25)	(0.00)	(0.34)	(1.07)	(15.68)	(3.89)	(504.69)
	Score		**−4.64**	**−3.39**	**−1.23**	**−3.29**	**−2.58**	**−3.10**	**−2.48**	**−4.08**	**−1.28**

2.4 本章小结

本章首先介绍了两种基于遗传算法的多任务进化算法，MFEA和MTGA，其中MTGA是作者硕士阶段提出的算法。然后通过大量的实验证明了，在任务之间存在明显的相关性时，现有的多任务进化算法均能较好地提升遗传算法的整体性能，其中MTGA的效果最好；在任务之间的相关性较低，或者任务的最优解之间存在较大偏差时，大部分多任务进化算法的性能会显著降低，甚至弱于单任务的遗传算法，在这种情况下MTGA依然取得了最好的结果。除了比较算法的最终结果和计算代价，我们还测试了MTGA对迁移个体数量的敏感性，实验结果证明了MTGA对迁移个体数量的变化是鲁棒的。总之，相对于其他算法，我们提出的MTGA是更加有效和稳定的。为了进一步探究MTGA的实用性和可扩展性，我们将在下一章介绍我们提出的多任务差分进化（MTDE），它使用了MTGA的框架和思想，用差分进化替代了遗传算法，将其作为多任务进化的基础。

3 基于差分进化的多任务进化算法

本章主要介绍基于差分进化（DE）的多任务进化算法，包括我们提出的多任务差分进化。相对于我们提出的多任务遗传算法，多任务差分进化采用差分进化去优化每个种群。

3.1 差分进化算法

DE是一个非常流行且性能很好的进化算法，已被广泛的用于连续优化问题。它的主要思想是用多个参数向量组成的种群并行地搜索最优解，这里的参数向量本质上类似于遗传算法（GA）中的个体，向量的个数始终保持不变。一般情况下，DE也采用随机的方法生成初始种群。为了更新种群，DE将乘以权重的两个种群向量的差值添加至第三个向量，从而得到一个新的参数向量，这一步称为变异。然后将这个新的参数向量与旧的交叉，得到的向量称之为实验向量。如果实验向量的代价函数值小于（假设为最小化问题）目标向量的，则用实验向量替换目标向量，并加入下一代种群，这一步称为选择[26]。

为了更具体地描述DE的优化过程，这里给出文献中常见的5种变异策略[28]：

(1) DE/rand/1 :

$$\mathbf{V}_i^g = \mathbf{X}_{r1}^g + F \cdot (\mathbf{X}_{r2}^g - \mathbf{X}_{r3}^g) \tag{式 3.1}$$

(2) DE/best/1 :

$$\mathbf{V}_i^g = \mathbf{X}_{best}^g + F \cdot (\mathbf{X}_{r1}^g - \mathbf{X}_{r2}^g) \tag{式 3.2}$$

(3) DE/current-best/2 :

$$\mathbf{V}_i^g = \mathbf{X}_i^g + F \cdot (\mathbf{X}_{best}^g - \mathbf{X}_i^g + \mathbf{X}_{r2}^g - \mathbf{X}_{r3}^g) \tag{式 3.3}$$

(4) DE/best/2 :

$$\mathbf{V}_i^g = \mathbf{X}_{best}^g + F \cdot (\mathbf{X}_{r1}^g - \mathbf{X}_{r2}^g + \mathbf{X}_{r3}^g - \mathbf{X}_{r4}^g) \tag{式 3.4}$$

(5) DE/rand/2 :

$$\mathbf{V}_i^g = \mathbf{X}_{r1}^g + F \cdot (\mathbf{X}_{r2}^g - \mathbf{X}_{r3}^g + \mathbf{X}_{r4}^g - \mathbf{X}_{r5}^g) \tag{式 3.5}$$

其中$\mathbf{V}_i^g$是$\mathbf{X}_i^g$在第g代产生的变异向量，$\mathbf{X}_{r1}^g$，$\mathbf{X}_{r2}^g$，$\mathbf{X}_{r3}^g$，$\mathbf{X}_{r4}^g$和$\mathbf{X}_{r5}^g$是五个从第g代种群中随机抽取的向量，$\mathbf{X}_{best}^g$是第g代已知的最优解。$r1$，$r2$，$r3$，$r4$和$r5$互不相等，F是控制差分向量权重的参数，$F > 0$。

我们一般采用如下交叉操作得到实验向量$\mathbf{U}_i^g$：

$$\mathbf{U}_i^g(j) = \begin{cases} \mathbf{V}_i^g(j), & if(rand < CR) or (j = rand_j) \\ \mathbf{X}_i^g(j), & otherwise \end{cases} \quad \text{(式 3.6)}$$

其中，$j = 1, ..., D$，D是参数向量的长度，$\mathbf{X}_i^g(j)$是第g代种群中的第i个向量的第j个参数，以此类推。*rand*是一个$[0, 1]$中均匀的随机数，CR是交叉概率，它的大小位于$[0, 1]$ 中。$rand_j$是$\{1, ..., D\}$中的一个随机整数，可以防止$\mathbf{U}_i^g$和$\mathbf{X}_i^g$相同。

然后评估$\mathbf{U}_i^g$的代价函数值，比较$\mathbf{U}_i^g$和目标向量$\mathbf{X}_i^g$，选择其中更好的向量进入下一代种群。

达到预先设定的终止条件后，种群会停止进化，并输出最后得到的最优解。这里讨论的是最简单的，也是最早的DE。虽然目前已有大量的文献提出了各种各样改进后的DE，但是由于本章的重点是如何用多任务进化的思想改进DE的优化效率，所以这里没有一一赘述，仅以此方法作为DE的代表，其他的DE也很容易与当前的多任务进化算法结合。

3.2 多因素差分进化

Feng等人[42]给出了多因素差分进化(MFDE)的基本流程与实验结果，同时验证了MFEA的广泛适用性，表明MFEA不仅可以提升GA的效率，还能够提升DE和粒子群优化算法的效率。相对于MFEA，MFDE舍弃了广泛使用的模拟二进制交叉操作（SBX）[50]和多项式变异[51]，改用经典的DE操作更新当前种群。其主要特点体现在，MFDE在选择交配（*assortative mating*）中采用了“DE/rand/1”，如算法 3 所示。MFDE的其他部分与MFEA保持一致。

Algorithm 3: MFDE[42]中选择交配的伪代码。

在$[0, 1]$中产生一个随机值r;
if $\tau_a = \tau_b$ *or* $r < rmp$ **then**
  $\mathbf{V}_i^g = \mathbf{X}_{r1}^g + F \cdot (\mathbf{X}_{r2}^{'g} - \mathbf{X}_{r3}^{'g})$;
else
  $\mathbf{V}_i^g = \mathbf{X}_{r1}^g + F \cdot (\mathbf{X}_{r2}^g - \mathbf{X}_{r3}^g)$;
end
备注：$\mathbf{X}_{r2}^{'g}, \mathbf{X}_{r3}^{'g}$与$\mathbf{X}_i^g$拥有不同的技能因子，$\mathbf{X}_{r1}^g, \mathbf{X}_{r2}^g, \mathbf{X}_{r3}^g$与$\mathbf{X}_i^g$拥有相同的技能因子，$rmp$是选择交配概率。

3.3 多任务差分进化

我们提出的多任务差分进化（MTDE）是整合了DE与MTGA后得到的一种新的多任务进化算法，为了方便阐述，我们这里同样只考虑两个任务的情况，也就是$M = 2$。当任务数超过两个时，也能够直接应用。MTDE的伪代码如算法 4 所示。

MTDE采用MTGA中的种群初始化和个体迁移方法，不同的是MTDE采用DE去优化单个种群，为了便于理解，这里采用的变异策略是“DE/rand/1”：

$$DE/rand/1 : \mathbf{V}_i^g = \mathbf{X}_{r1}^g + F \cdot (\mathbf{X}_{r2}^g - \mathbf{X}_{r3}^g) \tag{式 3.7}$$

其中$\mathbf{X}_{r1}^g$，$\mathbf{X}_{r2}^g$和$\mathbf{X}_{r3}^g$是三个从第g代种群中随机抽取的个体。$r1$，$r2$ 和$r3$互不相等，$F > 0$。

在得到变异个体$\mathbf{V}_i^g$后，再执行如下交叉操作：

$$\mathbf{U}_i^g(j) = \begin{cases} \mathbf{V}_i^g(j), & if(rand < CR) or (j = rand_j) \\ \mathbf{X}_i^g(j), & otherwise \end{cases} \tag{式 3.8}$$

其中，$j = 1, ..., D$，D是个体的长度，$\mathbf{X}_i^g(j)$是第g代种群中的第i个个体的第j个参数，以此类推。*rand*是一个$[0, 1]$中均匀的随机数，CR是交叉概率，它的大小位于$[0, 1]$中。$rand_j$是$\{1, ..., D\}$中的一个随机整数，可以防止$\mathbf{U}_i^g$和$\mathbf{X}_i^g$相同。

然后评估$\mathbf{U}_i^g$的适应度，比较$\mathbf{U}_i^g$和目标个体$\mathbf{X}_i^g$，它们中更好的个体被选择进入下一代种群。

Algorithm 4: MTDE的伪代码。

Input: 两个任务T_1和T_2;
N，种群大小;
K，最大的进化代数;
n_t，迁移个体的数量.
Output: $\mathbf{x}_m^*$，每个任务$T_m, m = 1, 2$的最优解。
for $m = 1, 2$ **do**
随机地产生N 个个体$\{\mathbf{x}_{m,n}\}_{n=1}^{N}$组成任务$T_m$的初始化种群$P_m$;
计算P_m 中每个$\mathbf{x}_{m,n}$ 的适应度;
根据个体的适应度将P_m 中的所有个体按降序排序;
end
设置$k = 1$;
while $k \leq K$ **do**
根据式 2.2 计算$\mathbf{m}_1$和$\mathbf{m}_2$;
for $m = 1, 2$ **do**
初始化一个临时种群$P_t = \emptyset$;
根据式 2.4，用P_{3-m}中最好的n_t个个体组成P_t的前n_t个个体;
用P_m中最好的$N - n_t$个个体组成P_t中剩余的$N - n_t$个个体;
初始化子代种群$O = \emptyset$;
P_t 为父代种群;
for $n = 1, ..., N$ **do**
根据式 3.7产生一个变异个体$\mathbf{V}_i^g$;
根据式 3.6交叉$\mathbf{V}_i^g$和$\mathbf{X}_i^g$;
评估$\mathbf{U}_i^g$的适应度;
$\mathbf{U}_i^g$ 和$\mathbf{X}_i^g$中更好的个体进入新的种群P_m;
end
根据个体的适应度将P_m中的所有个体按降序排序;
end
$k = k + 1$;
end
Return 每个任务$T_m, m = 1, 2$的最优个体。

3.4 实验部分

3.4.1 实验设置

这里依然分别用表 2.1和表 2.3中的测试问题来测试基于DE的多任务进化算法，同时采用完全相同的实验设置。我们比较了上述单任务的差分进化算法（SODE），MFDE，MTDE和上一章中测试的9种基于GA的多任务进化算法。

9种基于GA的多任务进化算法采用上一章中的实验设置。MFDE的种群大小为200，$F = 0.1$，$CR = 0.5$。MTDE中单个任务的种群大小为100，$F = 0.1$，$CR = 0.5$，$n_t = 40$。所有算法的最大函数计算次数都为100,000，所有基于DE的多任务进化算法均采用“DE/rand/1”变异策略。为了减少随机的影响，这里依然将每个算法都重复运行20次。

3.4.2 实验结果

12个算法在9个测试问题中的平均性能分数如图 3-1所示，单个实验结果如图 3-2所示，从中可以观察到：

(1) SODE, MFDE, MTDE整体性能分别优于SOEA, MFEA, MTGA，这表明我们采用的DE比GA更适合于这些测试问题。

(2) MFDE, MTDE都优于SODE，表明多任务优化也能提升DE的性能，再次验证了MFEA和我们提出的MTGA的核心思想是有效的。

(3) 在大部分情况下，我们提出的MTDE的整体性能优于MFDE，尤其在进化后期，MTDE的整体性能显著优于MFDE。

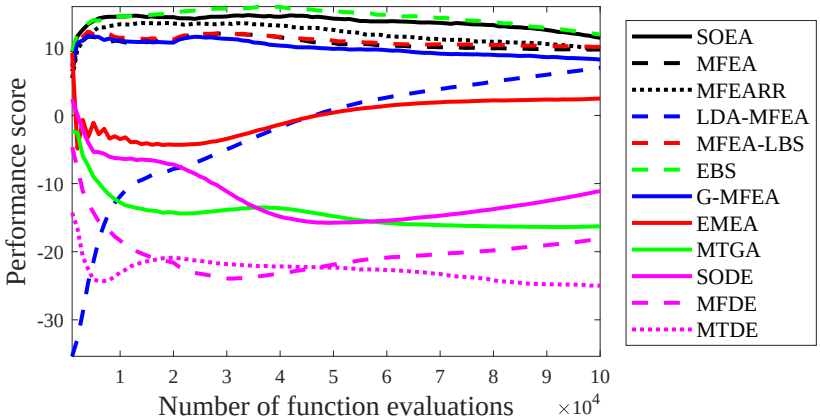


图 3-1 在9个原始的测试问题中的平均性能分数。

同样，我们也在表 3.1中给出了SODE, MFDE和MTDE经过100,000次函数计算后，在不同任务中获得的平均结果（均值和标准差）。我们这里主要关注于基于 DE的多任务进化算法，也就是MFDE和MTDE的优化效果。在SODE, MFDE

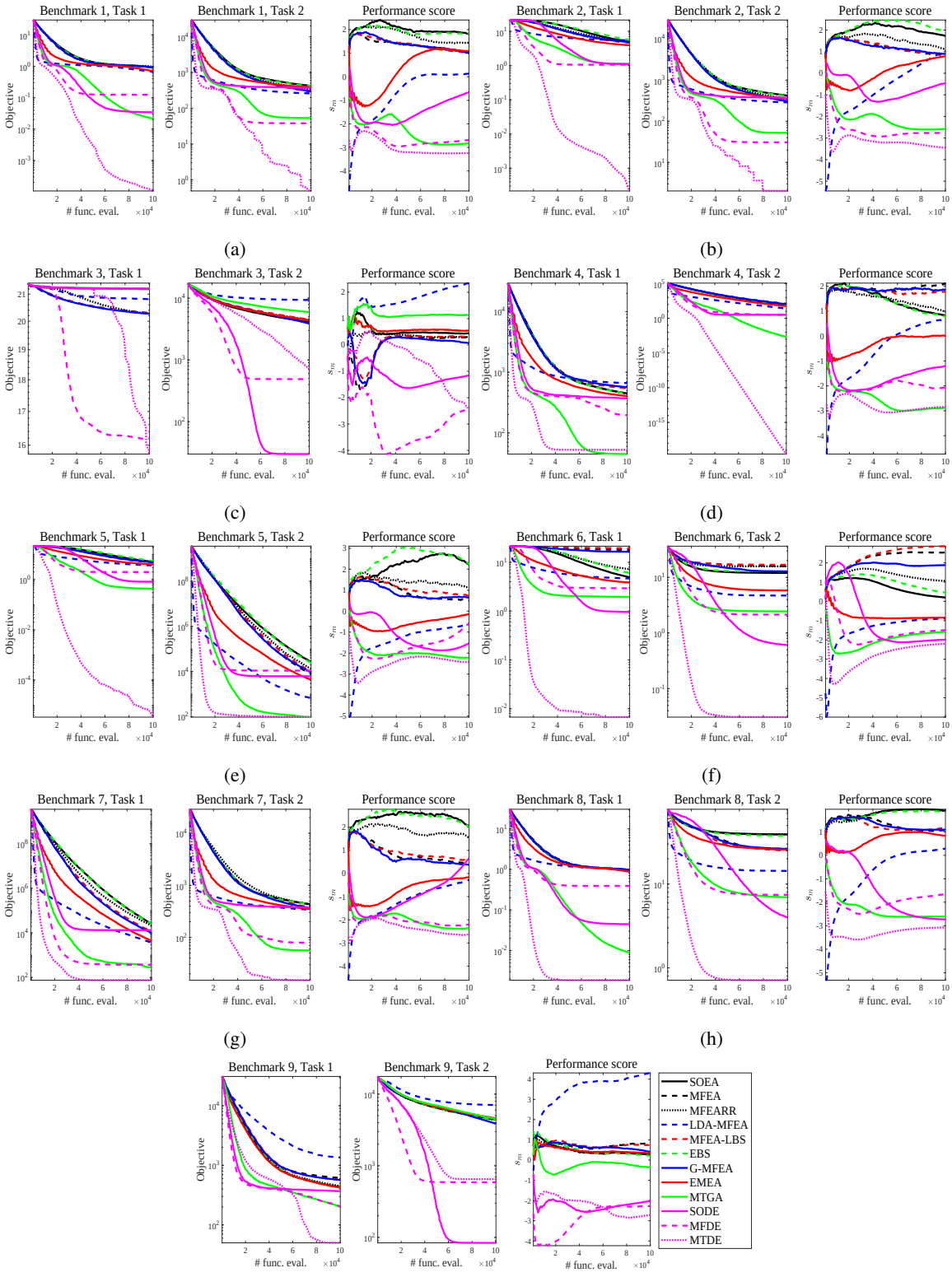


图 3-2 在9个原始的测试问题中的实验结果。(a) 测试问题 1；(b) 测试问题 2；(c) 测试问题 3；(d) 测试问题 4；(e) 测试问题 5；(f) 测试问题 6；(g) 测试问题 7；(h) 测试问题 8；(i) 测试问题 9。

表 3.1 3种算法在**原始**的测试问题中的平均性能（均值和括号内的标准差）。最好的性能已被加粗表示。

	Benchmark		1	2	3	4	5	6	7	8	9
SODE	T1	mean	0.03	1.11	21.20	371.88	0.82	0.95	13198.02	0.04	368.90
		std	(0.04)	(0.91)	(0.05)	(20.74)	(0.79)	(1.07)	(27294.73)	(0.08)	(14.16)
	T2	mean	372.28	371.38	**29.92**	3.21	6182.57	0.61	373.32	4.00	**83.93**
		std	(15.98)	(20.53)	(52.52)	(6.42)	(8818.72)	(0.63)	(12.34)	(2.13)	(95.55)
	Score		−0.66	−0.46	−1.16	−1.20	−1.53	−1.98	0.65	−2.73	−2.03
MFDE	T1	mean	0.12	1.05	16.18	195.98	1.99	2.93	368.07	0.39	205.52
		std	(0.09)	(0.76)	(6.17)	(93.24)	(1.04)	(0.61)	(283.47)	(0.23)	(101.98)
	T2	mean	37.98	30.81	488.77	3.19	10964.62	2.14	79.23	7.44	588.53
		std	(18.58)	(28.47)	(224.07)	(9.67)	(21811.14)	(0.91)	(57.87)	(1.48)	(242.22)
	Score		−2.68	−2.76	−2.40	−2.07	−0.57	−1.49	−2.19	−1.65	−2.27
MTDE	T1	mean	**0.00**	**0.00**	**15.75**	**52.57**	**0.00**	**0.01**	**74.81**	**0.00**	**50.00**
		std	(0.00)	(0.00)	(8.39)	(23.61)	(0.00)	(0.02)	(33.75)	(0.00)	(30.36)
	T2	mean	**0.49**	**2.02**	720.86	**0.00**	**96.89**	**0.03**	**15.04**	**0.71**	650.43
		std	(0.61)	(8.81)	(494.08)	(0.00)	(59.13)	(0.08)	(12.87)	(0.72)	(193.66)
	Score		**−3.23**	**−3.45**	**−2.43**	**−2.85**	**−2.43**	**−2.20**	**−2.62**	**−3.07**	**−2.71**

和MTDE中，我们可以明显地看出MTDE在所有测试问题中均取得了最好的性能分数。不仅如此，在图 3-3中，我们可以看到所有算法归一化后的计算时间，从中不难发现MTDE的计算代价仅仅略高于SODE，而MFDE的计算代价显著的高于SODE和MTDE，MFDE的平均计算时间是MTDE的两倍多一点。

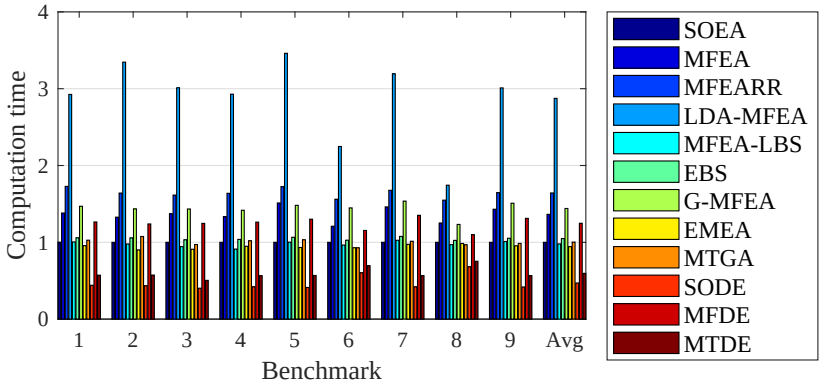


图 3-3 12个算法的计算时间。

同时，我们依然研究了MTDE对于迁移个体数量n_t的敏感性，以及考虑偏差带来的影响。如图 3-4所示，我们比较了SODE，六个版本的MTDE（$n_t = \{10, 20, 30, 40, 50, 60\}$）和不考虑偏差的MTDE (MTDE-NB)，MTDE-NB的$n_t = 40$。从中可以看出，具有不同n_t的MTDE均优于SODE和MTDE-NB，再一次表明考虑偏差确实是很重要的。具有不同n_t的MTDE拥有相近的性能，表明MTDE对于n_t的变化是稳定的。值得注意的是，相对于图 2-7中的结果，MTDE与MTDE-NB的差距更小，

MTGA与MTGA-NB的差距更明显。这是合理的，因为在MTGA当中，我们不仅迁移了最优的个体，还隐式地迁移了相关任务的进化方向，所以考虑偏差对MTGA-NB的提升较大。但是对于DE来说，由于基础的DE本身就在变异的过程中使用了种群的进化方向，即便我们在迁移个体过程中不考虑任务之间存在的偏差，MTDE-NB依然能够获取相关任务的进化方向，只是迁移过来的最优解不一定能在当前任务中表现很好。而且相对于GA，我们采用的DE是一个更加贪婪的进化算法。一般来说，进化后期的种群中的个体都非常相似，那么这些个体之间的差分向量往往会非常接近于零向量，再加上较为保守的交叉操作，也就导致了DE难以跳出当前的局部最优区域，所以此时的DE更加需要增加种群的多样性，在迁移的过程中不考虑偏差反而会大幅度的提升种群的多样性，进而会给种群的进化提供更多可能，这会抵消掉一部分因为考虑偏差带来的积极作用，尤其是在两个任务的偏差为零的情况下，因为此时对偏差的估计或许是不必要的。为了验证我们的想法，我们又在修改后的测试问题（表 2.3）中测试了这8种算法，如图 3-5所示，此时MTDE-NB的性能甚至显著的弱于SODE，而6种具有不同n_t的MTDE依然都优于SODE，这符合我们的期望。从更加实际的角度来说，我们要解决的大部分相关任务，都不是完全重叠的，即便在原始的搜索空间中具有相同的最优解，因为经过统一表示后，由于变量范围的不同，也会使不同任务的最优解分离，所以在我们没有关于任务的先验知识的情况下，考虑偏差是必要的。

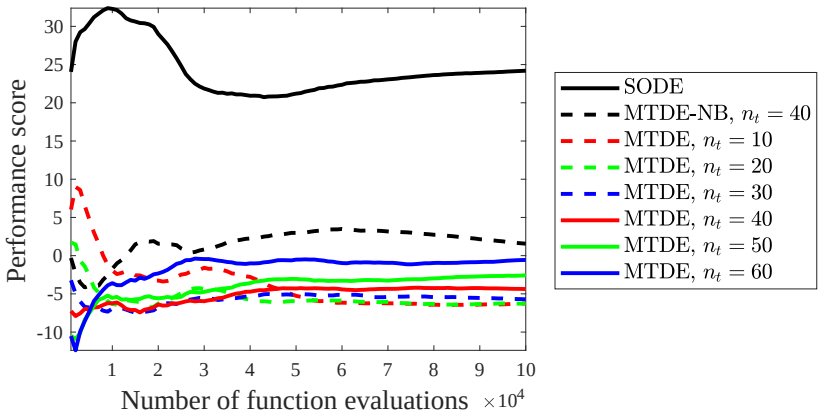


图 3-4 使用不同的n_t时，8个算法在9个原始的测试问题中的平均性能分数。

在修改后的9个测试问题中，平均性能分数如图 3-6所示，单个问题的实验结果图 3-7所示，从中可以看出，MTDE仍然整体上优于SODE，而MFDE则在进化后期弱于SODE。表 2.4 给出了在所有被修改后的测试问题中，3种基于DE的算法分别计算100,000次函数值后得到的最终的优化结果。在SODE, MFDE和MTDE中，MTDE在5个测试问题中都显著地优于其他两种算法，并且在剩下的4个测试问题中，MTDE也具有和SODE差不多的性能分数，SODE在这4个测试问题中的整体性能最好。MTDE在所有修改后的测试问题中都优于MFDE。

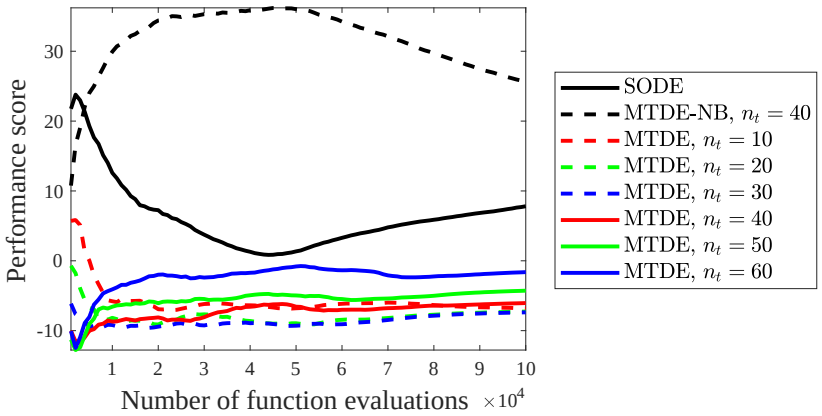


图 3-5 使用不同的n_t时，8个算法在9个修改后的测试问题中的平均性能分数。

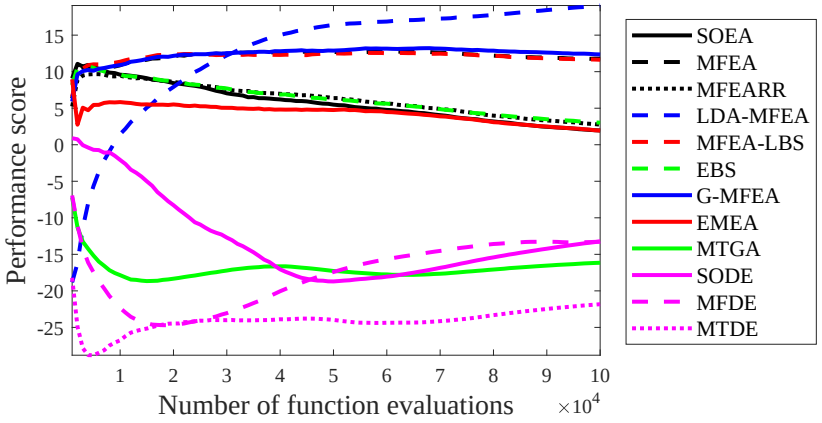


图 3-6 在9个修改后的测试问题中的平均性能分数。

表 3.2 3种算法在**修改后**的测试问题中的平均性能（均值和括号内的标准差）。最好的性能已被加粗表示。

	Benchmark		1	2	3	4	5	6	7	8	9
SODE	T1	mean	0.06	0.52	0.10	379.04	0.55	**0.88**	11864.13	0.05	375.38
		std	(0.07)	(0.49)	(0.10)	(14.40)	(0.58)	(0.77)	(34353.95)	(0.05)	(16.01)
	T2	mean	380.11	375.66	**54.82**	1.67	47002.68	0.75	370.17	4.40	**53.48**
		std	(12.64)	(14.78)	(70.96)	(2.46)	(164036.37)	(0.65)	(14.80)	(1.54)	(81.25)
	Score		−0.17	−0.90	**−2.88**	**−2.27**	−0.27	**−2.20**	−0.16	−1.43	**−2.95**
MFDE	T1	mean	0.10	2.81	0.68	226.59	2.67	3.07	6338.84	0.19	251.20
		std	(0.12)	(0.68)	(0.28)	(104.12)	(0.73)	(0.87)	(19409.56)	(0.17)	(110.65)
	T2	mean	204.33	213.83	701.27	2.41	77658.96	3.10	198.91	12.24	519.64
		std	(90.18)	(87.41)	(255.90)	(5.38)	(69236.73)	(1.48)	(108.22)	(2.17)	(211.06)
	Score		−2.18	−1.21	−2.48	−1.65	0.63	−1.38	−1.11	−1.89	−2.00
MTDE	T1	mean	**0.00**	**0.00**	**0.00**	**50.73**	**0.00**	1.79	**747.28**	**0.00**	**43.0433**
		std	(0.00)	(0.00)	(0.00)	(28.50)	(0.00)	(0.79)	(850.60)	(0.00)	(17.09)
	T2	mean	**24.81**	**39.44**	815.25	**0.00**	**14701.35**	**0.08**	**21.81**	**0.53**	719.52
		std	(4.62)	(13.90)	(250.79)	(0.00)	(14261.51)	(0.23)	(4.54)	(0.75)	(300.05)
	Score		**−3.37**	**−2.77**	−2.84	−2.19	**−0.94**	−2.07	**−2.11**	**−3.05**	−2.50

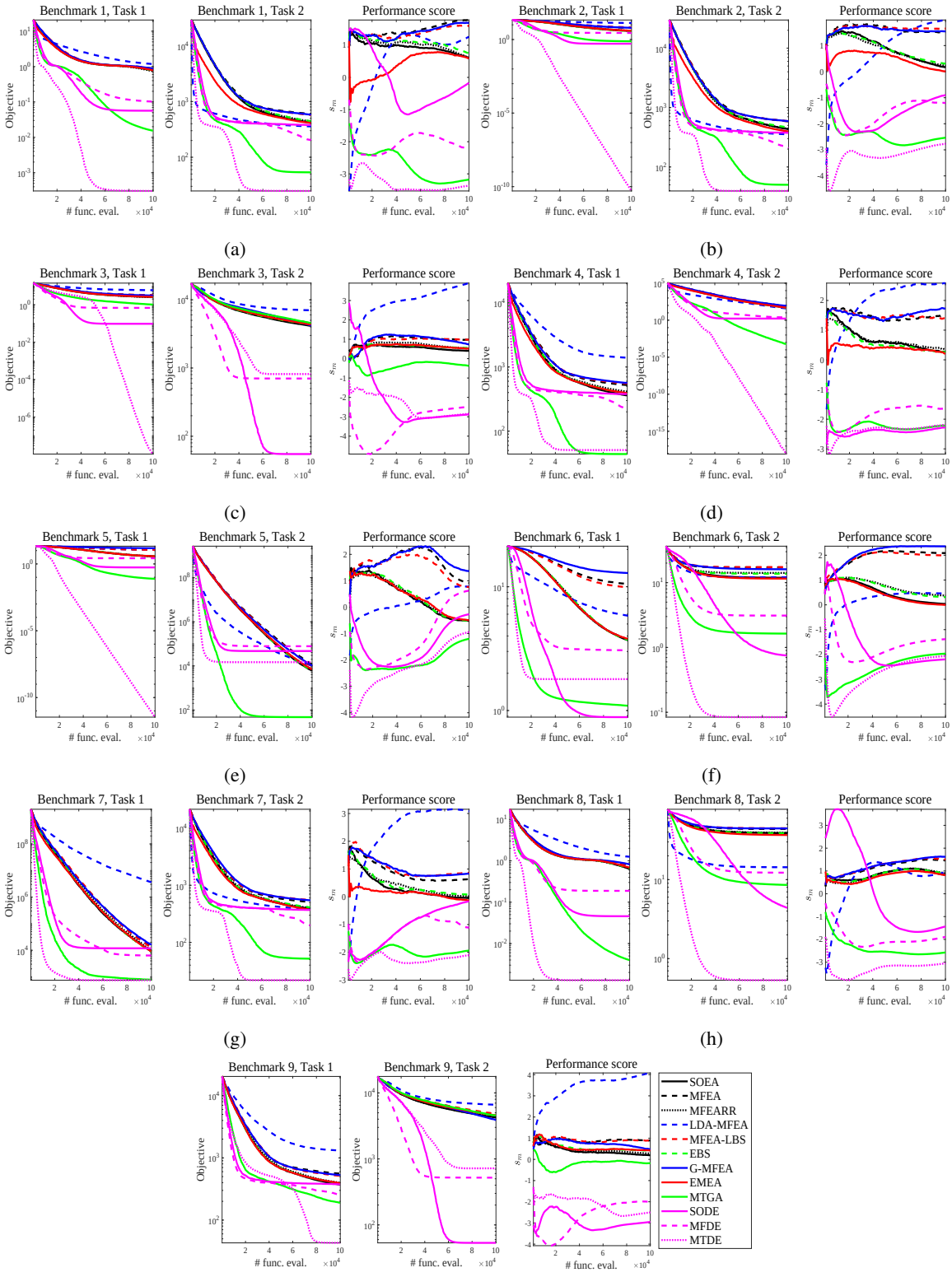


图 3-7 在9个修改后的测试问题中的实验结果。(a) 测试问题 1；(b) 测试问题 2；(c) 测试问题 3；(d) 测试问题 4；(e) 测试问题 5；(f) 测试问题 6；(g) 测试问题 7；(h) 测试问题 8；(i) 测试问题 9。

为了更深入的研究估计和使用偏差的意义，图 3-8中给出了在统一搜索空间中，MTGA和MTDE估计偏差时单个变量的平均绝对误差(MAE, 式3.9)，随函数计算次数变化的过程。

$$\mathbf{MAE} = \frac{||\mathbf{m}_1 - \mathbf{m}_2 - (\mathbf{x}_1^* - \mathbf{x}_2^*)||_1}{d} \tag{式 3.9}$$

其中，$\mathbf{x}_1^*$，$\mathbf{x}_2^*$分别是任务 1和任务 2的全局最优解。这里假设任务 1和任务 2的变量个数同等，d为变量个数，如果不相等，则$d = \min(d_1, d_2)$，只考虑前d个变量的平均绝对误差。

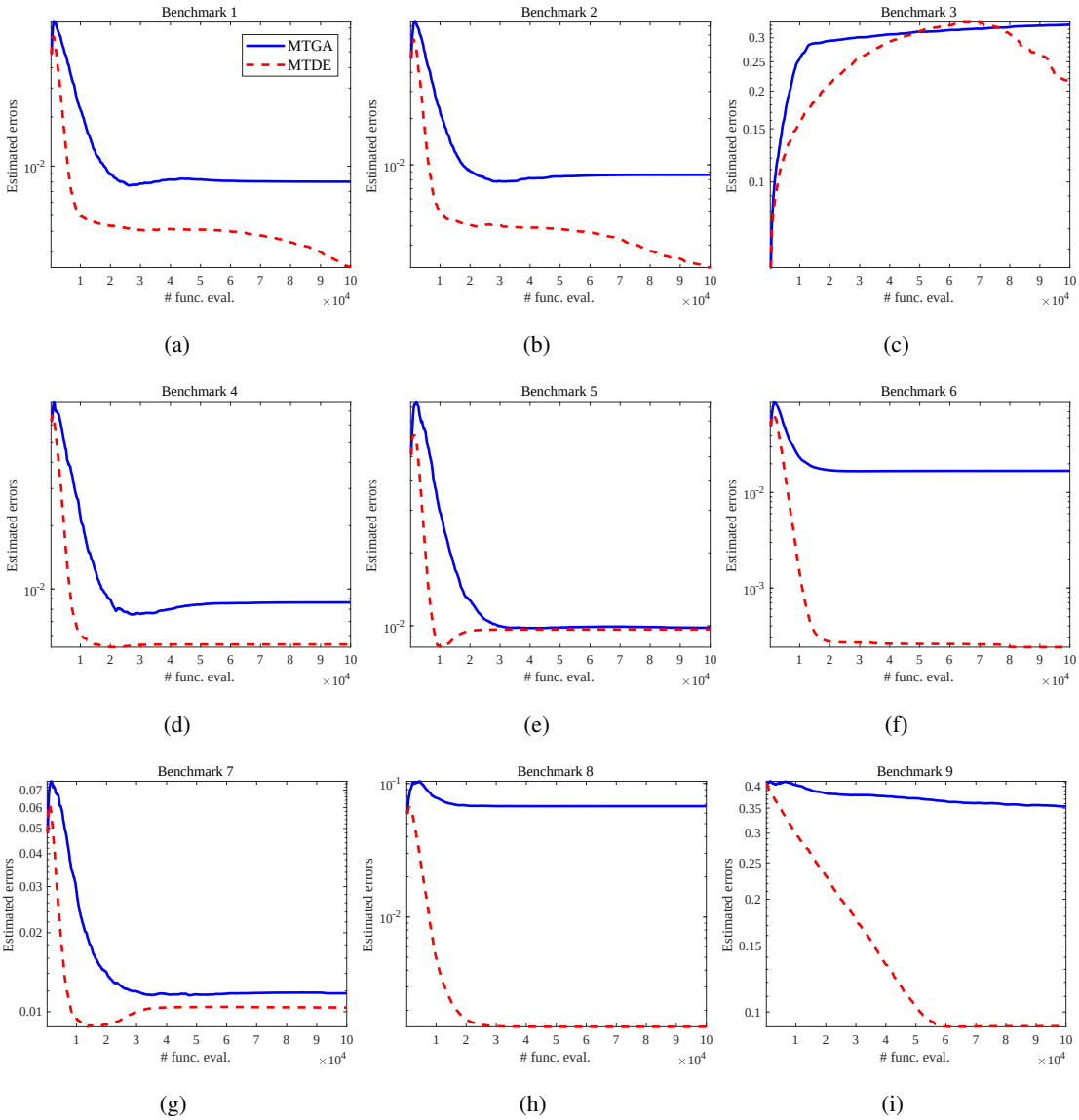


图 3-8 在9个原始的测试问题中偏差的估计误差。(a) 测试问题 1；(b) 测试问题 2；(c) 测试问题 3；(d) 测试问题 4；(e) 测试问题 5；(f) 测试问题 6；(g) 测试问题 7；(h) 测试问题 8；(i) 测试问题 9。

从中可以看出，大多数情况下，估计的偏差会随着种群的进化变得更加准确。在少数几个测试问题中，估计偏差的误差较大，同时MTGA和MTDE的结果也不够理想。这是由于在这些测试问题中，有些任务的最优解过于靠近决策边界，使得种群中最好的一部分个体不再对称地分布于最优解两侧。一部分个体经过交叉，变异等操作后超出了变量的范围，所以需要被“截断”后，才能进入子代种群。这是MTGA和MTDE的局限所在，也是可以进一步改进的地方。

3.5 本章小结

本章介绍了经典的差分进化算法，以及两种基于差分进化的多任务进化算法，MFDE和MTDE，其中的MTDE也是作者硕士阶段的工作。MTDE借用了MTGA的框架和思想，在迁移个体的过程中估计并使用任务间的偏差，然后将其与DE结合。相对于其他多任务进化算法，我们提出的MTGA和MTDE都具有较小的计算代价。通过广泛的实验和深入的理论分析，我们进一步地理解了多任务进化的思想。实验结果表明MTDE也具有优异的性能，在多个测试问题中均取得了最好的结果，再一次验证了我们提出的MTGA的框架和思想是十分有效的，能够同时提升GA和DE的性能，具有较好的普适性。同时，我们也分析了GA与DE的区别，对于这些测试问题，DE是更好的选择。最后，我们还通过实验测试了MTGA和MTDE中估计偏差的精度，证明了一般情况下估计偏差的精度会随着种群的进化变得越来越准确，而且在大多数情况下，MTDE中估计偏差的精度会更准确。

4 模糊控制器设计中的应用

4.1 模糊逻辑系统

作为智能计算领域内另一个重要的分支，模糊逻辑系统[52]在近几十年得到了广泛且深入的研究，在实际生活中得到了较多的应用，特别是用于智能控制。模糊逻辑系统用于控制时被称为模糊逻辑控制器(FLCs)，FLCs已经被成功地用于控制柴油机[53]，移动机器人[54]，倒立摆[55]和双水箱[22,23]等系统。一般来说，模糊逻辑系统可以分为一型模糊逻辑系统和二型模糊逻辑系统，其结构框图分别如图 4-1和图 4-2所示。它们的区别在于，一型模糊逻辑系统论域中每个输入值的隶属度函数都是确定的，而二型模糊逻辑系统中至少有一个隶属度函数是不确定的，也就是它的规则库中至少有一个模糊集是二型模糊集。为了能通过去模糊化得到一个精确的输出值，二型模糊逻辑系统不得不增加一个降阶器把二型模糊集转换成一型模糊集，常用的降阶器为Karnik-Mendel(KM) 算法[56]。在二型模糊逻辑系统中，最常见的是区间二型模糊系统[57]，其对应的区间二型模糊集将模糊集的隶属度限定为$[0,1]$中的一个区间（如图 4-3(b)所示），而不是一个单个值或一个一型模糊集，可以减少二型模糊逻辑系统的计算代价。目前，区间二型模糊逻辑系统的计算代价仍显著地高于一型模糊逻辑系统。

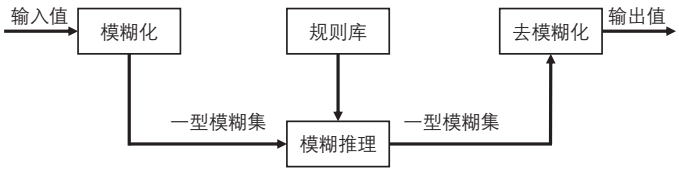


图 4-1 一型模糊逻辑系统。

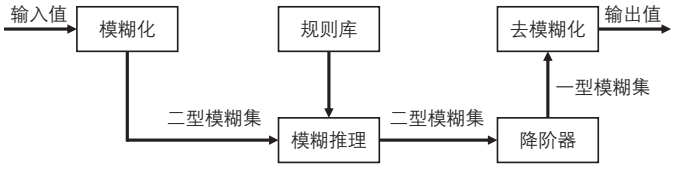
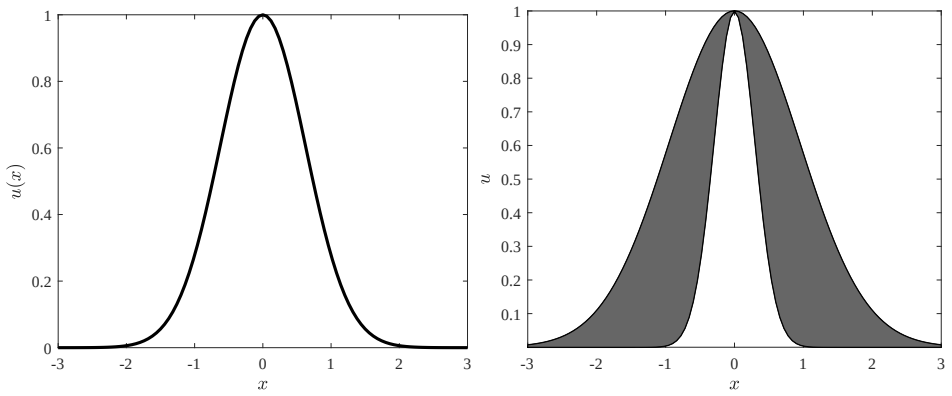


图 4-2 二型模糊逻辑系统。

由于模糊系统的设计通常需要专家花费较多的精力，而且即便如此，也依然难以精确地确定隶属度函数的参数，因此迫切地需要引入自动优化的方法来调整系统



(a) (b)

图 4-3 (a) 一型模糊集；(b) 区间二型模糊集。

的参数。由于这些参数的优化过程往往是很复杂的，难以解析的，所以进化算法是一个不错的选择。然而大部分工作都只是单独地优化了一型或二型的模糊逻辑系统，少部分工作在优化二型模糊逻辑系统之前，首先尽可能地优化一型模糊逻辑系统的参数，然后用它来初始化二型模糊逻辑系统的参数，从而减少计算代价。为了充分利用一型和二型模糊逻辑系统的相关性，我们首次将多任务进化算法用于同时优化一型和二型模糊控制器（FLC_1和FLC_2）的参数。

4.2 水箱系统

我们采用如图 4-4所示的双水箱液位控制系统作为现实生活中的优化问题，分别用优化后的FLC_1和FLC_2控制水箱液位，然后根据控制系统在4个不同的模型中的整体表现来评估优化算法的性能。该系统主要由两个水箱组成，每个水箱都有一个独立的水泵，可以控制液体输入的流量，并且都有一个出水口，水箱内安装了可以监测水位的传感器，并且可以通过上下移动两个水箱之间的隔板控制水的流动。由于返回水箱的水量近似地正比于水箱内水位高度的平方根，所以此系统是一个非线性系统[22,23]。

水箱系统的数学模型如下所示：

$$A_1\frac{dH_1}{dt} = Q_1 - \alpha_1\sqrt{H_1} - \alpha_3\sqrt{H_1 - H_2} \quad \text{(式 4.1)}$$

$$A_2\frac{dH_2}{dt} = Q_2 - \alpha_2\sqrt{H_2} + \alpha_3\sqrt{H_1 - H_2} \quad \text{(式 4.2)}$$

其中，A_1，A_2 分别是水箱1，2的横截面积，H_1，H_2分别是水箱1，2的水位高度，Q_1，Q_2 分别是水泵1，2的流量($\mathbf{cm^3/s}$)，α_1，α_2，α_3分别是对应于$\sqrt{H_1}$，$\sqrt{H_2}$，$\sqrt{H_1 - H_2}$ 的比例常数。

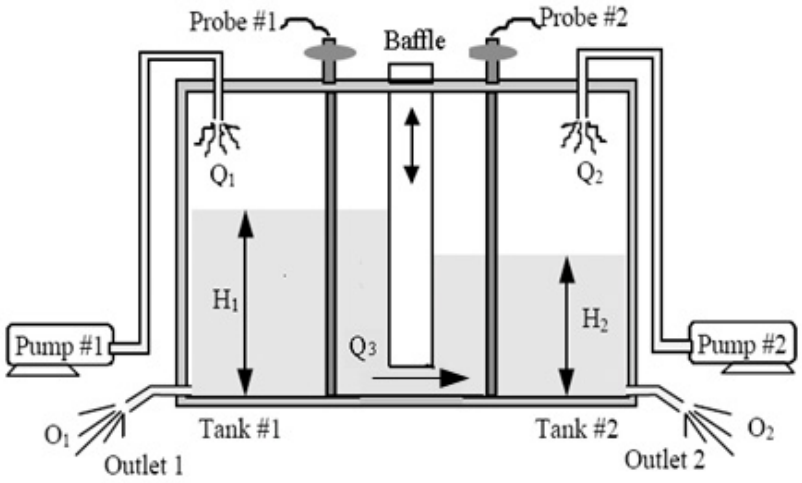


图 4-4 双水箱水位控制模型

4.3 适应函数

式 4.2描述的是通过物理公式推导得到的理想模型，与实际模型可能存在差异。为了训练出更鲁棒的FLCs，我们采用表 4.1 中4 种不同的模型去测试每个个体，同时使用时间权重的绝对误差和（ITAE）作为个体的适应函数，ITAE越小，算法控制性能越好，其计算方式如下所示：

$$F = \sum_{i=1}^{4} \alpha_i \left[\sum_{j=1}^{N_i} j \cdot |e_i(j)| \right] \tag{式 4.3}$$

其中，$e_i(j)$代表第i个模型在第j个采样点时，实际液位与设定点之间的绝对差值，α_i为第i个模型的ITAE的权重。采样的次数统一设置为$N_i = 200$，步长为1s ，因为第三个模型的ITAE一般会是其他模型的几倍，所以为了同等地考虑每个模型，这里使$\alpha_3 = 1/3$，剩下的3个权重都为1。

表 4.1 被控制的4种模型

	I	II	III	IV
$A_1 = A_2$ (cm^2)	36.52	36.52	36.52	36.52
$\alpha_1 = \alpha_2$	5.6186	5.6186	5.6186	5.6186
α_3	10	10	10	8
设定点(cm)	$0 \rightarrow 15$	$0 \rightarrow 15$	$0 \rightarrow 22.5 \rightarrow 7.5$	$0 \rightarrow 15$
传输延迟(s)	0	2	0	0

4.4 模糊控制器的结构和参数

FLC_1和FLC_2的输入均为反馈误差e和误差变化率$\dot{e}$，输出为控制信号的变化率$\dot{u}$。FLC_1和FLC_2把每个输入的论域分成3个模糊集（高斯型隶属度函数），分别记为N (negative)，Z (zero)和P (positive)，并且使用了5个不同的精确值$\dot{u}_i$ $(i = 1, 2, ..., 5)$作为输出。表 4.4给出了FLC_1和FLC_2采用的规则库。每个一型的高斯隶属度函数包含2个参数：均值(m)和标准差(δ)。每个二型的高斯隶属度函数包含3个参数：均值(m)，标准差的范围$([\delta_1, \delta_2])$。因此我们可以知道FLC_1和FLC_2中待优化的参数分别有17和23个，将这些参数编码成染色体的形式后如图 4-5所示，值得注意的是，这些参数的排列顺序需要满足一些约束条件，例如$m_{e_1} < m_{e_2} < m_{e_3}$，$m_{e_1}, m_{e_2}, m_{e_3}$分别是$N_e$, Z_e, P_e的高斯隶属度函数的均值，所以在评估个体之前需要调整基因的顺序。同时图 4-6给出了迁移个体的过程中，FLC_1和FLC_2的相关参数的对应关系，也就是不同种群的个体间的基因匹配关系，同一列的参数具有相关性。

表 4.2 FLC_1和FLC_2的规则库

$e\backslash\dot{e}$	$N_{\dot{e}}$	$Z_{\dot{e}}$	$P_{\dot{e}}$
N_e	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$
Z_e	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$
P_e	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
FLC_1	m_{e_1}	m_{e_2}	m_{e_3}	$m_{\dot{e}_1}$	$m_{\dot{e}_2}$	$m_{\dot{e}_3}$	δ_{e_1}	δ_{e_2}	δ_{e_3}	$\delta_{\dot{e}_1}$	$\delta_{\dot{e}_2}$	$\delta_{\dot{e}_3}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$						
FLC_2	m_{e_1}	m_{e_2}	m_{e_3}	$m_{\dot{e}_1}$	$m_{\dot{e}_2}$	$m_{\dot{e}_3}$	δ_{1e_1}	δ_{2e_1}	δ_{1e_2}	δ_{2e_2}	δ_{1e_3}	δ_{2e_3}	$\delta_{1\dot{e}_1}$	$\delta_{2\dot{e}_1}$	$\delta_{1\dot{e}_2}$	$\delta_{2\dot{e}_2}$	$\delta_{1\dot{e}_3}$	$\delta_{2\dot{e}_3}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$

$m_{e_1} < m_{e_2} < m_{e_3}$, $m_{\dot{e}_1} < m_{\dot{e}_2} < m_{\dot{e}_3}$, $\delta_{1e_i} < \delta_{2e_i}$, $\delta_{1\dot{e}_i} < \delta_{2\dot{e}_i}$, $i = 1, 2, 3$, $\dot{u}_1 < \dot{u}_2 < \dot{u}_3 < \dot{u}_4 < \dot{u}_5$

$m_{\boldsymbol{e}} \in [-20, 20]$, $m_{\dot{\boldsymbol{e}}} \in [-3, 3]$, $\delta_{\boldsymbol{e}} \in [1, 6]$, $\delta_{\dot{e}} \in [0.1, 0.8]$, $\dot{u} \in [-1, 1]$

图 4-5 FLC_1和FLC_2的编码格式。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
FLC_1	m_{e_1}	m_{e_2}	m_{e_3}	$m_{\dot{e}_1}$	$m_{\dot{e}_2}$	$m_{\dot{e}_3}$	δ_{e_1}	δ_{e_2}	δ_{e_3}	$\delta_{\dot{e}_1}$	$\delta_{\dot{e}_2}$	$\delta_{\dot{e}_3}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$
	1	2	3	4	5	6	7	9	11	13	15	17	19	20	21	22	23
FLC_2	m_{e_1}	m_{e_2}	m_{e_3}	$m_{\dot{e}_1}$	$m_{\dot{e}_2}$	$m_{\dot{e}_3}$	δ_{1e_1}	δ_{1e_2}	δ_{1e_3}	$\delta_{1\dot{e}_1}$	$\delta_{1\dot{e}_2}$	$\delta_{1\dot{e}_3}$	$\dot{u}_1$	$\dot{u}_2$	$\dot{u}_3$	$\dot{u}_4$	$\dot{u}_5$

图 4-6 FLC_1和FLC_2的参数对应关系。

4.5 实验部分

4.5.1 实验设置

我们的目的是探究多任务进化算法同时优化FLC_1和FLC_2时的效率，相对于用单任务进化算法分别优化FLC_1和FLC_2时是否更有效。为此，我们这里比较了下面6种算法：1）SOEA；2）MFEA；3）MTGA；4）SODE；5）MFDE；6）MTDE。依然采用前面两章给定的相关参数，MFEA和MFDE的种群大小为200，其他算法中每个任务的种群大小均为100。其他参数分别为$rmp = 0.3$，$\beta = 2$，$\eta = 5$，$F = 0.1$，$CR = 0.5$，$n_t = 40$。所有算法的最大函数计算次数都为100,000，也就是只进化500代。FLC_2采用的降阶器均为EIASC算法[58]。为了减少随机性的影响，我们依然重复了20次实验。

4.5.2 实验结果

图 4-7(a)给出了六个算法在这两个任务中平均的优化效果，不难发现：

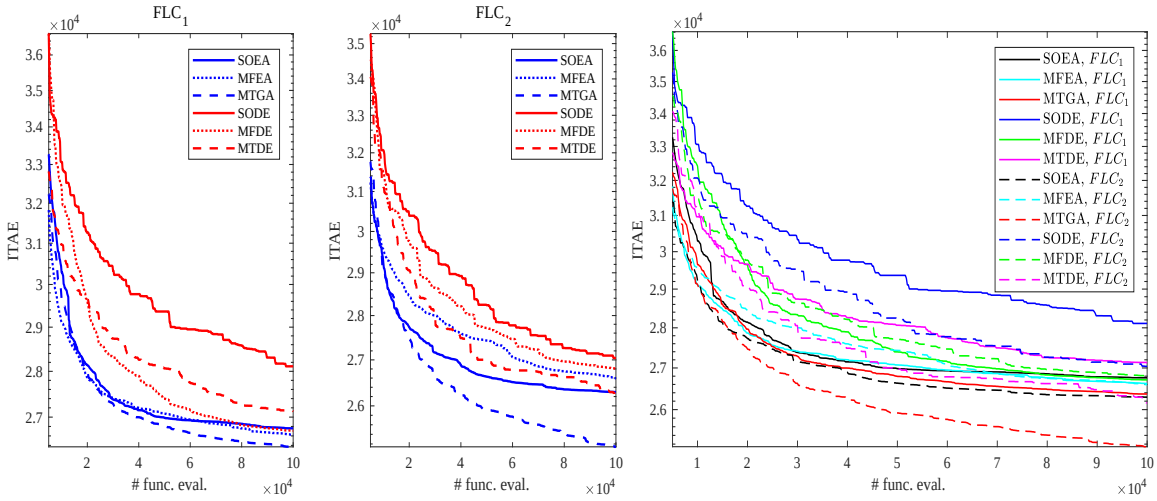
(1) 根据最终取得的IATE，除了MFEA优化FLC_2时的效率低于SOEA之外，四种多任务进化算法均优于其对应的单任务进化算法（SOEA或SODE），这表明任务之间的信息迁移确实能改善整体的优化性能。

(2) 在这两个任务中，SOEA均优于SODE。

(3) 在这两个任务中，我们提出的MTGA均取得了最好的效果，同时我们提出的MTDE也显著的优于SODE，这表明MTGA和MTDE均具有较好的实用性，有助于设计模糊控制器。

图 4-7(b)将六个算法在这两个任务中的结果混合在一张子图中，因为FLC_1和FLC_2控制的系统是相同的，所以我们还可以直接比较FLC_1和FLC_2的性能，从中可以看出，大部分FLC_2的结果要优于同一算法优化的FLC_1的，也就是同一颜色的虚线在实线的下方。这表明FLC_2的性能整体上优于FLC_1的。

图 4-8给出了六个算法分别优化20次后，FLC_1 (F1) 和 FLC_2 (F2) 最终取得的ITAE的箱型图。除了前面已得到的信息外，还可以发现FLC_2的方差普遍比FLC_1的更大，这是因为FLC_2的参数更多，参数变化带来的影响更大。为了更直接的观察六个算法在不同模型中的表现，如图 4-9和 4-10所示，我们分别画出了所有FLC_1和FLC_2在4 个模型中仿真得到的阶跃响应以及对应的控制信号变化曲线（均为20次实验中获得的最好结果，与平均结果的排序不一定相同，见图 4-8）。从中可以看出，六个算法对应的曲线具有明显的差异，整体上，ITAE越小，阶跃响应的超调量也越小，也更容易稳定。



(a) (b)

图 4-7 六个算法的ITAE的变化过程。(a) 分别比较FLC_1和FLC_2；(b) 混合比较FLC_1和FLC_2。

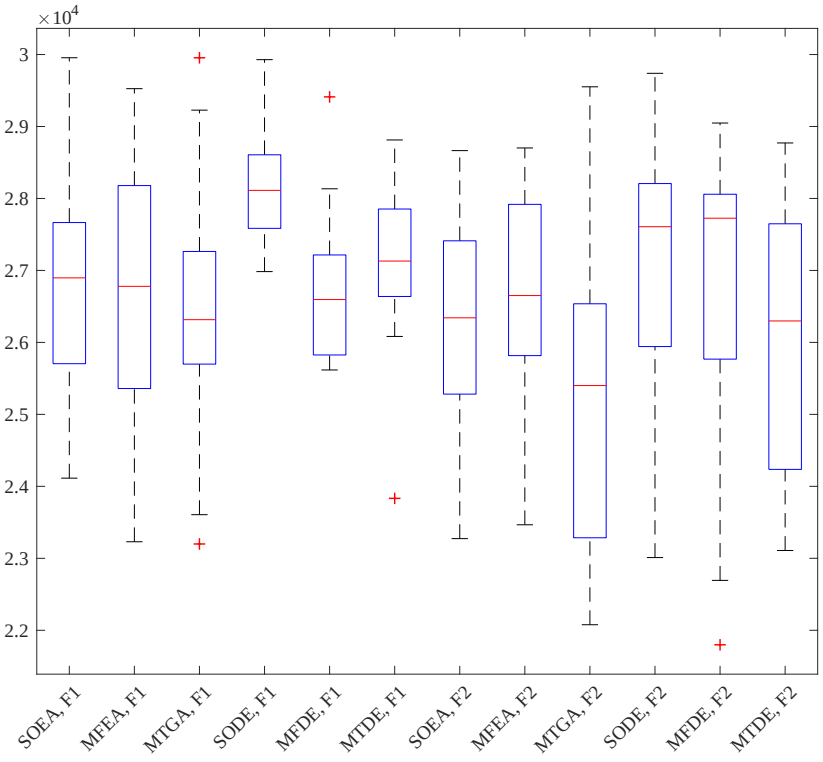


图 4-8 所有FLC_1 (F1) 和 FLC_2 (F2) 最终取得的ITAE。

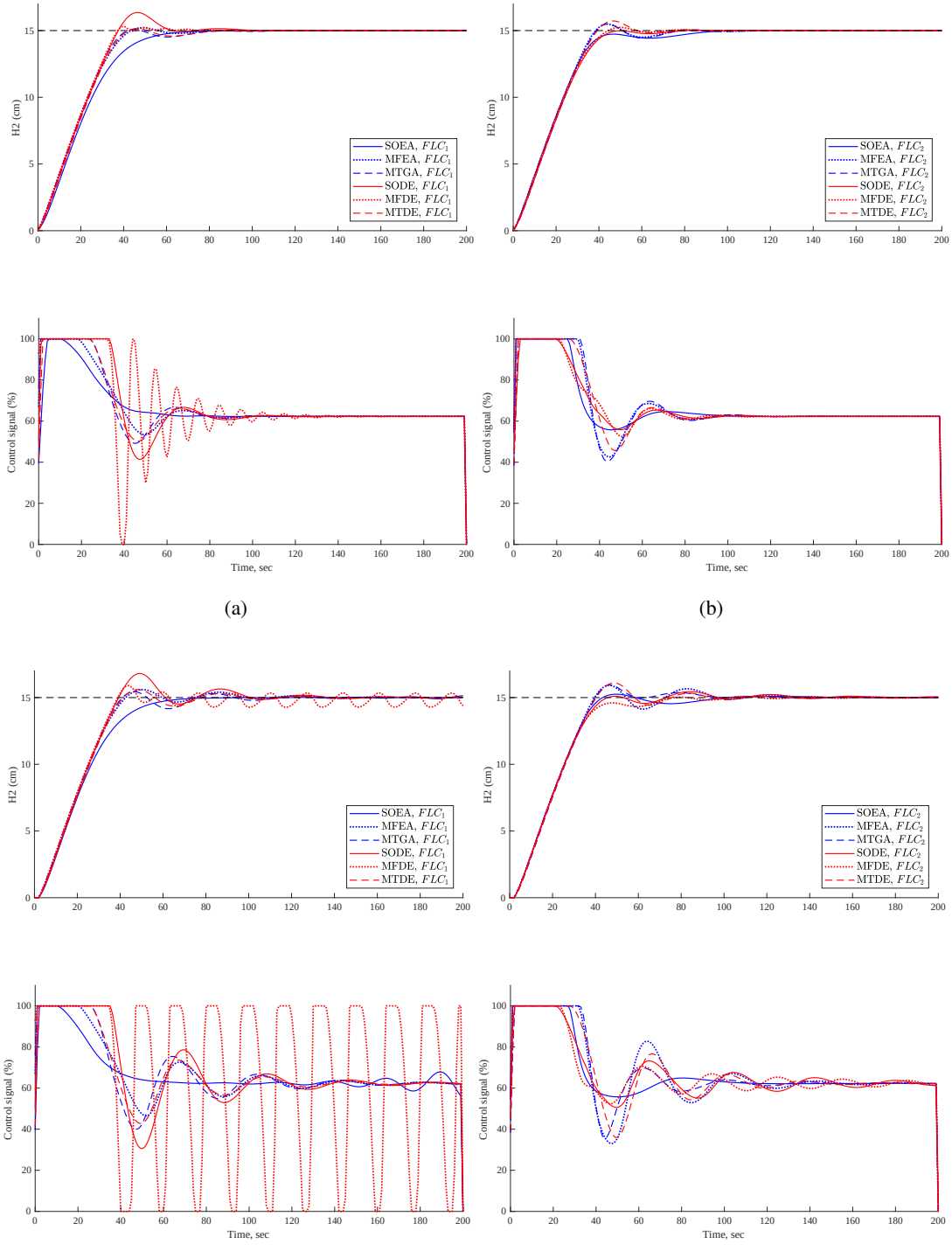


图 4-9 FLC_1和FLC_2在模型 1和 2中的阶跃响应（上）以及对应的控制信号变化曲线（下）。(a) 所有FLC_1在模型 1中；(b) 所有FLC_2在模型 1中；(c) 所有FLC_1在模型 2 中；(d) 所有FLC_2在模型 2 中。

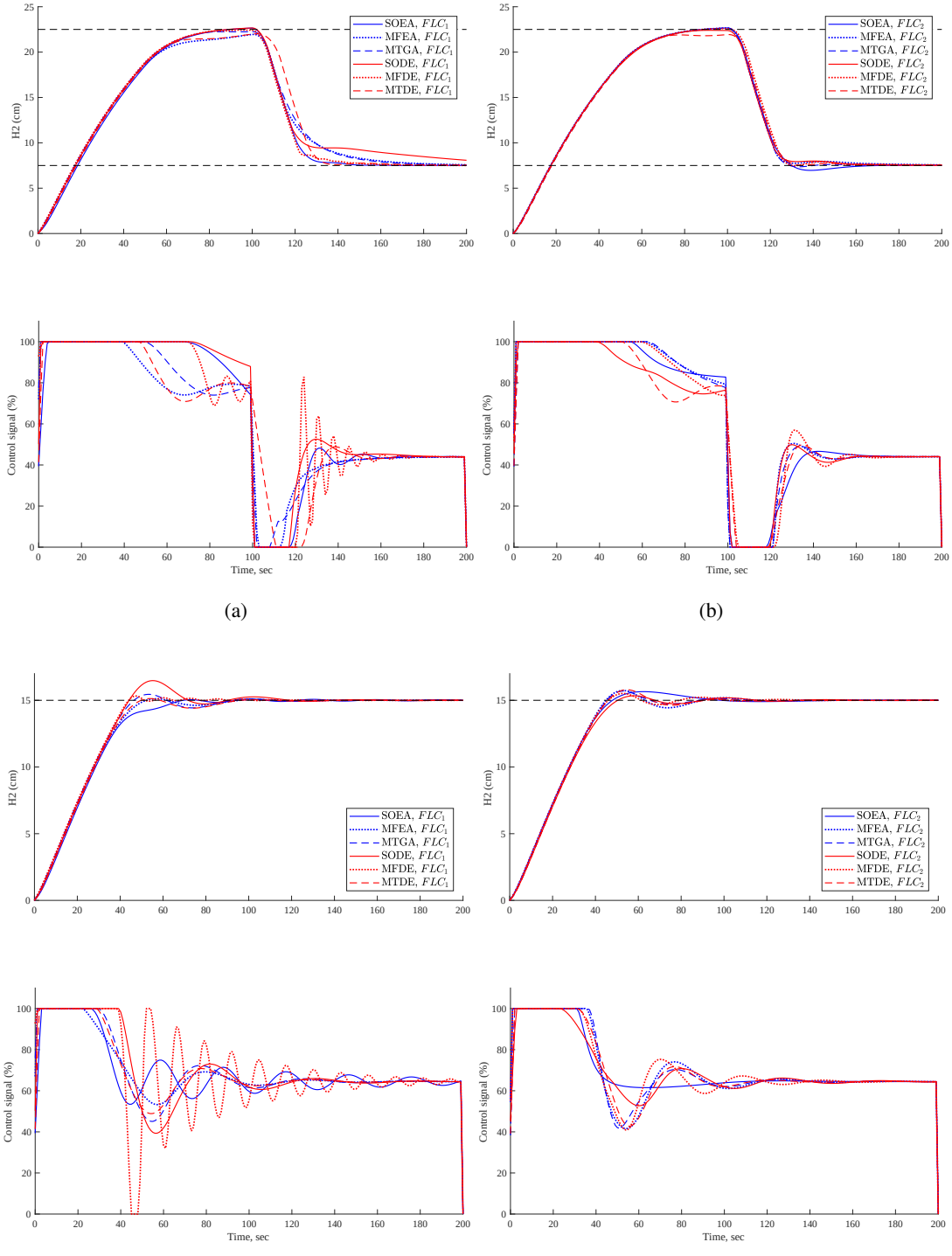


图 4-10 FLC_1和FLC_2在模型 3和 4中的阶跃响应（上）以及对应的控制信号变化曲线（下）。(a) 所有FLC_1在模型 3中；(b) 所有FLC_2在模型 3中；(c) 所有FLC_1在模型 4中；(d) 所有FLC_2在模型 4中。

4.6 本章小结

本章首先概述了模糊逻辑系统的应用和分类，然后我们首次将多任务优化应用于模糊控制器设计。因为同一系统的一型和二型模糊控制器的参数之间，存在明显的相关关系，所以我们可以用多任务进化算法来同时优化一型和二型模糊控制器的参数。除此之外，本章还详细描述了具体的控制对象，水箱系统。在水箱系统中，我们构造了四个不同的模型，来测试模糊控制器的性能，进而比较不同进化算法的性能。实验结果验证了我们提出的MTGA和MTDE均能找到较好的模糊控制器。

5 结论

5.1 全文总结

多任务进化是进化算法领域内一个新兴的子领域，它整合了进化算法和多任务学习，可以用来同时的执行多个优化任务。本文主要关注单目标多任务的连续优化问题。为了解决这类问题，我们提出了多任务遗传算法（MTGA）和多任务差分进化（MTDE），它们的共同特点是在线地估计并使用任务间的偏差，同时采用新的多任务进化框架，使得单个种群中被更新的最好的一部分个体，能够立即地被同一代中的其他种群使用。大量的实验结果证明了我们提出的MTGA和MTDE均具有优异的性能。不仅如此，我们还首次将多任务优化应用于模糊控制器设计。我们用多任务进化算法去同时优化一型和二型模糊控制器的参数，实验结果表明，我们提出的MTGA和MTDE在模糊控制器设计中表现优异，具有较强的实用性。本文的主要内容包括以下几个部分：

(1) 介绍了我们提出的MTGA和已有的多因素进化算法。不仅在理论上解释了MTGA为什么具有较好的性能，而且通过广泛的实验验证了MTGA是更加有效和鲁棒的。

(2) 介绍了我们提出的MTDE，并解释了考虑任务间的偏差会对MTGA和MTDE造成的影响，同时指出了不同之处。大量的实验验证了MTDE也具有优异的性能。

(3) 首次将多任务优化应用于模糊控制器设计，并验证了我们提出的MTGA和MTDE均能取得较好的效果。

通过上述内容，我们还可以了解到当前多任务进化的不足，以及需要进一步解决的问题。比如：

(1) 很多情况下，我们不能提前知道待优化任务间的相关性，统一表示[30]也许会降低任务间的相关性，变量取值范围的波动也会造成统一搜索空间中最优解的偏离。所以我们需要一种自适应的方法将不同任务的解变换至统一搜索空间。

(2) 如何有效地表示任务间的相关关系？也就是通过怎样的方式可以从其他任务中获取对当前任务有用的知识。线性域适应[32]和自动编码器[35]已被用来解决这个问题，但是对于前者来说，一方面，计算代价太大，以至于违背了多任务进化的初衷；另一方面，线性变换的精度或许难以满足进化后期的要求，因为此时的种群已经逐渐靠近最优解区域，种群中的个体比较相似，变换后的个体很可能会远离这个区域，阻碍种群的收敛。对于后者，更多的是精度的问题。考虑偏差的方法相对来说是一个更稳定的方法，因为大多数情况下估计偏差的精度会随着种群的进化越来越准确，偏差的波动范围较小，变换后的个体也不会远离最优区域。但是对于存在多个分离的最优解，或者最优解非常靠近决策边界的优化任务，准确地估计偏差不

再是一件容易的事情，在这种情况下，我们需要更有效的方法来解决这个问题。

致　　谢

毕业论文即将完成，两年的研究生时光也即将过去，回想起过去的点点滴滴，我最想说的是感谢。感谢伍冬睿老师的指导与帮助，记得最开始进实验室时，我拿着电脑让老师给我讲他写的MATLAB程序，他很耐心地给我讲了每一个细节，我那时还什么都不懂。也记得凌晨三点发的问题，却能立马收到回复的意外。也记得自己写的惨不忍睹的论文，被老师一遍又一遍的修改，... 。有太多太多的事值得怀念，他严谨的科研思维，细心的做事风格，勤恳的工作作风深深地感染了我，是我人生路上的榜样。每一行代码，每一段文字，都是一种收获。

感谢实验室的同学们，感谢你们的支持和帮助，是你们陪伴我度过这难忘的两年，在我无助，焦虑，甚至迷茫的时候，是你们的鼓励和帮助让我走出了阴影。

感谢关心我的朋友们，在我最困难的时候，是你们伸出了援手，让我在这平凡的岁月里感受到了温暖。

特别感谢父母和外婆，感谢你们默默的付出，感谢你们理解并支持我追寻理想，没有你们，就没有这一切。

最后，再次衷心地感谢所有帮助和关心过我的老师，朋友和同学们，谢谢！

参考文献

[1] Friedberg R M. A Learning Machine: Part I. IBM Journal of Research and Development, 1958, 2(1):2–13.

[2] Bremermann H J. Optimization through evolution and recombination. Self-organizing systems, 1962, 93:106.

[3] Bremermann H J, Rogson M, Salaff S. Search by evolution. Biophysics and Cybernetic Systems, 1965, pages 157–167.

[4] Box G E P. Evolutionary Operation: a Method for Increasing Industrial Productivity. Journal of the Royal Statistical Society: Series C (Applied Statistics), 1957, 6(2):81–101.

[5] Box G E P, Draper N R. Evolutionary operation: A statistical method for process improvement, volume 25. Wiley New York, 1969.

[6] Bäck T, Fogel D B, Michalewicz Z. Handbook of evolutionary computation. CRC Press, 1997.

[7] Fogel L J, Owens A J, Walsh M J. Artificial Intelligence through Simulated Evolution. New York, USA: John Wiley, 1966.

[8] Holland J H. Adaptation in Natural and Artificial Systems. MI: University of Michigan Press, 1975.

[9] Bienert P. Aufbau einer Optimierungsautomatik für drei Parameter. Engineering Optimization, 1967.

[10] Rechenberg I. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment Library Translation 1122, 1965.

[11] Schwefel H P. Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. Diploma thesis, Technical Univ. of Berlin, 1965.

[12] Goldberg D E, Lingle R. Alleles, loci, and the traveling salesman problem. in: Proceedings of Proceedings of an international conference on genetic algorithms and their applications, 1985, 154–159.

[13] Fogel D B. Applying evolutionary programming to selected traveling salesman problems. Cybernetics and Systems, 1993, 24(1):27–36.

[14] Tan X, Lei D, Wu D, et al. Robot Path Planning Using an Improved Genetic Algorithm with Ordered Feasible Subpaths. in: Proceedings of 2018 Chinese Automation Congress (CAC), Xi'an, Shaanxi, China, Nov., 2018, 4293-4297.

[15] Davis L. Job shop scheduling with genetic algorithms. in: Proceedings of Proceedings of an international conference on genetic algorithms and their applications, 1985.

[16] Lei D, Tan X. Shuffled frog-leaping algorithm for order acceptance and scheduling in flow shop. in: Proceedings of 2016 35th Chinese Control Conference (CCC), July, 2016, 9445-9450.

[17] Lei D, Tan X. Local search with controlled deterioration for multi-objective scheduling in dual-resource constrained flexible job shop. in: Proceedings of 2016 Chinese Control and Decision Conference (CCDC), May, 2016, 4921-4926.

[18] Etter D, Hicks M, Cho K. Recursive adaptive filter design using an adaptive genetic algorithm. in: Proceedings of ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing, May, 1982, 635-638.

[19] Miller G F, Todd P M, Hegde S U. Designing Neural Networks using Genetic Algorithms. in: Proceedings of ICGA, 1989, 379–384.

[20] Davis L, Orvosh D, Cox A, et al. A genetic algorithm for survivable network design. in: Proceedings of Proceedings of the 5th International Conference on Genetic Algorithms, 1993, 408–415.

[21] De Jong K. Adaptive System Design: A Genetic Approach. IEEE Transactions on Systems, Man, and Cybernetics, 1980, 10(9):566–574.

[22] Wu D, Tan W W. Genetic Learning and Performance Evaluation of Type-2 Fuzzy Logic Controllers. Engineering Applications of Artificial Intelligence, 2006, 19(8):829–841.

[23] Wu D, Tan W W. A Simplified Type-2 Fuzzy Controller for Real-Time Control. ISA Trans., 2006, 15(4):503–516.

[24] Wu D, Mendel J M. Designing Practical Interval Type-2 Fuzzy Logic Systems Made Simple. in: Proceedings of Proc. IEEE World Congress on Computational Intelligence, Beijing, China, July, 2014, 800-807.

[25] Kim J, Moon Y, Zeigler B P. Designing Fuzzy Net Controllers Using Genetic Algorithms. IEEE Control Systems Magazine, 1995, 15(3):66–72.

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

[27] Elsayed S M, Sarker R A, Essam D L. An Improved Self-Adaptive Differential Evolution Algorithm for Optimization Problems. IEEE Transactions on Industrial Informatics, 2013, 9(1):89–99.

[28] Qin A K, Suganthan P N. Self-adaptive differential evolution algorithm for numerical optimization. in: Proceedings of 2005 IEEE Congress on Evolutionary Computation, Sep., 2005, 1785-1791 Vol. 2.

[29] Zhang Y, Yang Q. A survey on multi-task learning. arXiv preprint arXiv:1707.08114, 2017.

[30] Gupta A, Ong Y S, Feng L. Multifactorial evolution: Toward evolutionary multitasking. IEEE Trans. on Evolutionary Computation, 2016, 20(3):343–357.

[31] Yuan Y, Ong Y S, Gupta A, et al. Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP QAP LOP and JSP. in: Proceedings of Proc. IEEE Region 10 Conf., 2016, 3157-3164.

[32] Bali K K, Gupta A, Feng L, et al. Linearized domain adaptation in evolutionary multitasking. in: Proceedings of Proc. IEEE Int'l Conf. on Evolutionary Computation, San Sebastian, Spain, June, 2017, 1295-1302.

[33] Liaw R T, Ting C K. Evolutionary many-tasking based on biocoenosis through symbiosis: A framework and benchmark problems. in: Proceedings of Proc. IEEE Int'l Conf. on Evolutionary Computation, San Sebastian, Spain, June, 2017, 2266-2273.

[34] Ding J, Yang C, Jin Y, et al. Generalized Multi-tasking for Evolutionary Optimization of Expensive Problems. IEEE Trans. on Evolutionary Computation, 2018. In press.

[35] Feng L, Zhou L, Zhong J, et al. Evolutionary Multitasking via Explicit Autoencoding. IEEE Trans. on Cybernetics, 2018. In press.

[36] Hashimoto R, Ishibuchi H, Masuyama N, et al. Analysis of Evolutionary Multi-tasking As an Island Model. in: Proceedings of Proc. of the Genetic and Evolutionary Computation Conf. Companion, Kyoto, Japan, Jul., 2018, 1894–1897.

[37] Gong M, Tang Z, Li H, et al. Evolutionary Multitasking with Dynamic Resource Allocating Strategy. IEEE Trans. on Evolutionary Computation, 2019. In press.

[38] Li G, Zhang Q, Gao W. Multipopulation Evolution Framework for Multifactorial Optimization. in: Proceedings of Proceedings of the Genetic and Evolutionary Computation Conference Companion, New York, NY, USA: ACM, 2018, 215–216.

[39] Wen Y W, Ting C K. Parting ways and reallocating resources in evolutionary multitasking. in: Proceedings of Proc. IEEE Int'l Conf. on Evolutionary Computation, San Sebastian, Spain, June, 2017, 2404-2411.

[40] Cheng M Y, Gupta A, Ong Y S, et al. Coevolutionary multitasking for concurrent global optimization: With case studies in complex engineering design. Engineering Applications of Artificial Intelligence, 2017, 64:13–24.

[41] Chen Q, Ma X, Zhu Z, et al. Evolutionary Multi-tasking Single-Objective Optimization Based on Cooperative Co-evolutionary Memetic Algorithm. in: Proceedings of Proc. 13th Int'l Conf. on Computational Intelligence and Security, 2017, 197-201.

[42] Feng L, Zhou W, Zhou L, et al. An empirical study of multifactorial PSO and multifactorial DE. in: Proceedings of Proc. IEEE Int'l Conf. on Evolutionary Computation, San Sebastian, Spain, June, 2017, 921-928.

[43] Sagarna R, Ong Y S. Concurrently searching branches in software tests generation through multitask evolution. in: Proceedings of Proc. IEEE Symposium Series on Computational Intelligence, Athens, Greece, Dec., 2016, 1-8.

[44] Tang Z, Gong M, Zhang M. Evolutionary multi-task learning for modular extremal learning machine. in: Proceedings of Proc. IEEE Int'l Conf. on Evolutionary Computation, San Sebastian, Spain, June, 2017, 474-479.

[45] Li H, Ong Y, Gong M, et al. Evolutionary Multitasking Sparse Reconstruction: Framework and Case Study. IEEE Trans. on Evolutionary Computation, 2018. In press.

[46] Bao L, Qi Y, Shen M, et al. An Evolutionary Multitasking Algorithm for Cloud Computing Service Composition. in: Proceedings of World Congress on Services, Seattle, WA, USA, June, 2018, Springer, 130–144.

[47] Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.

[48] Bean J C. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on computing, 1994, 6(2):154–160.

[49] Da B S, Ong Y S, Feng L, et al. Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metrics and baseline results. Technical report, Nanyang Technological University, 2016.

[50] Deb K, Agrawal R B. Simulated binary crossover for continuous search space. Complex Systems, 1994, 9(2):1–15.

[51] Deb K, Agrawal S. A niched-penalty approach for constraint handling in genetic algorithms. in: Proceedings of Proc. Int'l Conf. on Artificial Neural Networks and Genetic Algorithms, 1999, 235-243.

[52] Zadeh L A. Fuzzy sets. Information and Control, 1965, 8:338–353.

[53] Lynch C, Hagras H, Callaghan V. Using Uncertainty Bounds in the Design of an Embedded Real-Time Type-2 Neuro-Fuzzy Speed Controller for Marine Diesel Engines. in: Proceedings of Proc. IEEE Int'l Conf. on Fuzzy Systems, Vancouver, Canada, July, 2006, 7217-7224.

[54] Hagras H. A Hierarchical Type-2 Fuzzy Logic Control Architecture for Autonomous Mobile Robots. IEEE Trans. on Fuzzy Systems, 2004, 12:524–539.

[55] Huang J, Ri M, Wu D, et al. Interval Type-2 Fuzzy Logic Modeling and Control of a Mobile Two-Wheeled Inverted Pendulum. IEEE Transactions on Fuzzy Systems, 2018, 26(4):2030–2038.

[56] Mendel J M. Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Upper Saddle River, NJ: Prentice-Hall, 2001.

[57] Mendel J. Uncertain rule-based fuzzy systems: introduction and new directions, 2nd ed. Springer, 2017.

[58] Wu D, Nie M. Comparison and Practical Implementation of Type-Reduction Algorithms for Type-2 Fuzzy Sets and Systems. in: Proceedings of Proc. IEEE Int'l Conf. on Fuzzy Systems, Taipei, Taiwan, June, 2011, 2131-2138.

附录 1　攻读学位期间发表论文目录

[1] Dongrui Wu and Xianfeng Tan. “Multi-Tasking Genetic Algorithm (MTGA) for Fuzzy System Optimization”, IEEE Trans. on Fuzzy Systems, 2019, submitted.

[2] Xianfeng Tan, Deming Lei, Dongrui Wu, Zheng Li. “Robot Path Planning Using an Improved Genetic Algorithm with Ordered Feasible Subpaths”, in 2018 Chinese Automation Congress (CAC), Xi’an, Shaanxi, China, Nov. 2018, pp. 4293–4297.

[3] 伍冬睿, 谭显烽. 一种适用于云计算系统的多任务处理方法. 中国, CN201811434588.6[P]. 2018-11-28.

附录 2　测试问题

文献[49]中的9个测试问题：

1. 测试问题1:

$$Task\ 1 : \min f_1(\mathbf{x}_1) = 1 + \frac{1}{4000}\sum_{i=1}^{d} z_i^2 - \prod_{i=1}^{d} \cos\left(\frac{z_i}{\sqrt{i}}\right),$$
$$\mathbf{z} = \mathbf{M}_{ch1}\mathbf{x}_1, d = 50, \mathbf{x}_1 \in [-100, 100]^d$$
$$Task\ 2 : \min f_2(\mathbf{x}_2) = \sum_{i=1}^{d} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right),$$
$$\mathbf{z} = \mathbf{M}_{ch2}\mathbf{x}_2, d = 50, \mathbf{x}_2 \in [-50, 50]^d$$

任务 1的全局最优解是$\mathbf{x}_1^* = [0, ..., 0] \in \mathbb{R}^{50}$，任务 2的全局最优解是$\mathbf{x}_2^* = [0, ..., 0] \in \mathbb{R}^{50}$。它们的统一表示都是$[0.5, ..., 0.5] \in \mathbb{R}^{50}$，因为$-100 + 0.5[100 - (-100)] = -50 + 0.5[50 - (-50)] = 0$。

2. 测试问题2:

$$Task\ 1 : \min f_1(\mathbf{x}_1) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} z_i^2}\right)$$
$$- \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)\right) + 20 + e,$$
$$\mathbf{z} = \mathbf{M}_{cm1}\mathbf{x}_1, d = 50, \mathbf{x}_1 \in [-50, 50]^d$$
$$Task\ 2 : \min f_2(\mathbf{x}_2) = \sum_{i=1}^{d} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right),$$
$$\mathbf{z} = \mathbf{M}_{cm2}\mathbf{x}_2, d = 50, \mathbf{x}_2 \in [-50, 50]^d$$

任务 1的全局最优解是$\mathbf{x}_1^* = [0, ..., 0] \in \mathbb{R}^{50}$，任务 2的全局最优解是$\mathbf{x}_2^* = [0, ..., 0] \in \mathbb{R}^{50}$。它们的统一表示都是$[0.5, ..., 0.5] \in \mathbb{R}^{50}$。

3. 测试问题3:

$$Task\ 1 : \min f_1(\mathbf{x}_1) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} z_i^2}\right)$$
$$- \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)\right) + 20 + e,$$

$$\mathbf{z} = \mathbf{M}_{cl1}\mathbf{x}_1 - \mathbf{x}_1^*, d = 50, \mathbf{x}_1 \in [-50, 50]^d$$

$$Task\ 2 : \min f_2(\mathbf{x}_2) = 418.9828d - \sum_{i=1}^{d} x_i \sin\left(|x_i|^{\frac{1}{2}}\right),$$

$$d = 50, \mathbf{x}_2 \in [-500, 500]^d$$

任务 1的全局最优解是$\mathbf{x}_1^* = [42.0969, ..., 42.0969] \in \mathbb{R}^{50}$，任务 2 的全局最优解是$\mathbf{x}_2^* = [420.9687, ..., 420.9687] \in \mathbb{R}^{50}$。即使$\mathbf{x}_1^* \neq \mathbf{x}_2^*$，但它们的统一表示都是$[0.920969, ..., 0.920969] \in \mathbb{R}^{50}$。

4. 测试问题4:

$$Task\ 1 : \min f_1(\mathbf{x}_1) = \sum_{i=1}^{d} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right),$$

$$\mathbf{z} = \mathbf{M}_{ph1}\mathbf{x}_1, d = 50, \mathbf{x}_1 \in [-50, 50]^d$$

$$Task\ 2 : \min f_2(\mathbf{x}_2) = \sum_{i=1}^{d} z_i^2,$$

$$\mathbf{z} = \mathbf{x}_2 - \mathbf{x}_2^*, d = 50, \mathbf{x}_2 \in [-100, 100]^d$$

任务 1的全局最优解是$\mathbf{x}_1^* = [0, ..., 0] \in \mathbb{R}^{50}$，任务 2的全局最优解是$\mathbf{x}_2^* = [0, ..., 0, 20, ..., 20] \in \mathbb{R}^{50}$ (前25个变量为0，剩下的为20). $\mathbf{x}_1^*$ 的统一表示为$[0.5, ..., 0.5] \in \mathbb{R}^{50}$，$\mathbf{x}_2^*$ 的统一表示为$[0.5, ..., 0.5, 0.6, ..., 0.6] \in \mathbb{R}^{50}$。这两个统一表示的前25个元素是相等的。

5. 测试问题5:

$$Task\ 1 : \min f_1(\mathbf{x}_1) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} z_i^2}\right)$$

$$- \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)\right) + 20 + e,$$

$$\mathbf{z} = \mathbf{M}_{pm1}\mathbf{x}_1, d = 50, \mathbf{x}_1 \in [-50, 50]^d$$

$$Task\ 2 : \min f_2(\mathbf{x}_2) = \sum_{i=1}^{d-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right),$$

$$d = 50, \mathbf{x}_2 \in [-50, 50]^d$$

任务 1的全局最优解是$\mathbf{x}_1^* = [0, ..., 0, 1, ..., 1] \in \mathbb{R}^{50}$ (前25个变量为0，剩下的为1)，任务 2的全局最优解是$\mathbf{x}_2^* = [1, ..., 1] \in \mathbb{R}^{50}$。$\mathbf{x}_1^*$ 的统一表示为$[0.5, ..., 0.5, 0.51, ..., 0.51] \in \mathbb{R}^{50}$，$\mathbf{x}_2^*$ 的统一表示为$[0.51, ..., 0.51] \in \mathbb{R}^{50}$。这两个统一表示的后25个元素是相等的。

6. 测试问题6:

$$
\begin{aligned}
Task\ 1: \min f_1(\mathbf{x}_1) = &-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} z_i^2}\right) \\
&- \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi z_i)\right) + 20 + e, \\
&\mathbf{z} = \mathbf{M}_{pl1}\mathbf{x}_1, d = 50, \mathbf{x}_1 \in [-50, 50]^d \\
Task\ 2: \min f_2(\mathbf{x}_2) = &\sum_{i=1}^{d}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos(2\pi b^k(z_i+0.5))\right]\right) \\
&- d\sum_{i=1}^{k_{\max}}\left[a^k\cos(\pi b^k)\right] \\
&a = 0.5, b = 3, k_{\max} = 20, \\
&\mathbf{z} = \mathbf{M}_{pl2}\mathbf{x}_2, d = 25, \mathbf{x}_2 \in (-0.5, 0.5)^d
\end{aligned}
$$

任务1的全局最优解是$\mathbf{x}_1^* = [0, ..., 0] \in \mathbb{R}^{50}$，任务2的全局最优解是$\mathbf{x}_2^* = [0, ..., 0] \in \mathbb{R}^{25}$。$\mathbf{x}_1^*$的统一表示为$[0.5, ..., 0.5] \in \mathbb{R}^{50}$，$\mathbf{x}_2^*$的统一表示为$[0.5, ..., 0.5] \in \mathbb{R}^{25}$。这两个统一表示的前25个元素是相等的。

7. 测试问题7:

$$
\begin{aligned}
Task\ 1: \min f_2(\mathbf{x}_1) = &\sum_{i=1}^{d-1}\left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right), \\
&d = 50, \mathbf{x}_1 \in [-50, 50]^d \\
Task\ 2: \min f_2(\mathbf{x}_2) = &\sum_{i=1}^{d}\left(z_i^2 - 10\cos(2\pi z_i) + 10\right), \\
&\mathbf{z} = \mathbf{M}_{nh2}\mathbf{x}_2, d = 50, \mathbf{x}_2 \in [-50, 50]^d
\end{aligned}
$$

任务1的全局最优解是$\mathbf{x}_1^* = [1, ..., 1] \in \mathbb{R}^{50}$，任务2的全局最优解是$\mathbf{x}_2^* = [0, ..., 0] \in \mathbb{R}^{50}$。$\mathbf{x}_1^*$的统一表示为$[0.51, ..., 0.51] \in \mathbb{R}^{50}$，$\mathbf{x}_2^*$的统一表示为$[0.5, ..., 0.5] \in \mathbb{R}^{50}$。这两个统一表示是略微不同的。

8. 测试问题8:

$$
\begin{aligned}
Task\ 1: \min f_1(\mathbf{x}_1) = &1 + \frac{1}{4000}\sum_{i=1}^{d} z_i^2 - \prod_{i=1}^{d}\cos\left(\frac{z_i}{\sqrt{i}}\right), \\
&\mathbf{z} = \mathbf{M}_{nm1}(\mathbf{x}_1 - \mathbf{x}_1^*), d = 50, \mathbf{x} \in [-100, 100]^d \\
Task\ 2: \min f_2(\mathbf{x}_2) = &\sum_{i=1}^{d}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos(2\pi b^k(z_i+0.5))\right]\right)
\end{aligned}
$$

$$
\begin{aligned}
&- d\sum_{i=1}^{k_{\max}}\left(a^k\cos(\pi b^k)\right)\\
&a=0.5, b=3, k_{\max}=20,\\
&\mathbf{z}=\mathbf{M}_{nm2}\mathbf{x}_2, d=25, \mathbf{x}_2\in[-0.5,0.5]^d
\end{aligned}
$$

任务 1的全局最优解是$\mathbf{x}_1^*=[10,...,10]\in\mathbb{R}^{50}$，任务 2的全局最优解是$\mathbf{x}_2^*=[0,...,0]\in\mathbb{R}^{50}$。$\mathbf{x}_1^*$ 的统一表示为$[0.55,...,0.55]\in\mathbb{R}^{50}$，$\mathbf{x}_2^*$ 的统一表示为$[0.5,...,0.5]\in\mathbb{R}^{50}$。这两个统一表示是略微不同的。

9. 测试问题9:

$$
\begin{aligned}
Task\ 1: \min f_1(\mathbf{x}_1)&=\sum_{i=1}^{d}\left(z_i^2-10\cos(2\pi z_i)+10\right),\\
\mathbf{z}&=\mathbf{M}_{ph1}\mathbf{x}_1, d=50, \mathbf{x}_1\in[-50,50]^d\\
Task\ 2: \min f_2(\mathbf{x}_2)&=418.9829d-\sum_{i=1}^{d}x_i\sin\left(|x_i|^{\frac{1}{2}}\right),\\
d&=50, \mathbf{x}_2\in[-500,500]^d
\end{aligned}
$$

任务 1的全局最优解是$\mathbf{x}_1^*=[0,...,0]\in\mathbb{R}^{50}$，任务 2的全局最优解是$\mathbf{x}_2^*=[420.9687,...,420.9687]\in\mathbb{R}^{50}$。$\mathbf{x}_1^*$ 的统一表示为$[0.5,...,0.5]\in\mathbb{R}^{50}$，$\mathbf{x}_2^*$ 的统一表示为$[0.9209687,...,0.9209687]\in\mathbb{R}^{50}$。这两个统一表示是显著不同的。