分类号：　　单位代码：10560

密　级：

汕頭大學
硕士学位论文

中文题名：不平衡约束多目标优化方法及其在井眼轨迹优化中的应用

英文题名：**Imbalanced constrained Multi-objective Optimization Methods and Its Application in Well Trajectory Optimization**

姓　　名　王柳　　学　　号　152009072

所在学院　工学院　　导　　师　范衡

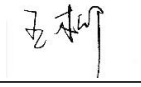
专　　业　计算机技术　　合作导师

入学日期　2020 年 10 月 9 日　　答辩日期　2023 年 5 月 21 日

论文提交日期　二〇二三 年 四 月 二十 日

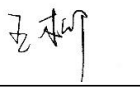
学位论文原创性声明

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

作者签名：  日期：2023 年 5 月 21 日

学位论文使用授权声明

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

作者签名：  导师签名： 

日期：2023 年 5 月 21 日 日期：2023 年 5 月 21 日

硕 士 学 位 论 文

论文中文题目：不平衡约束多目标优化方法及其在井眼轨迹优化中的应用

论文英文题名：Imbalanced constrained Multi-objective Optimization Methods and Its Application in Well Trajectory Optimization

指 导 教 师：范衡
申 请 人：王柳

论文答辩委员会成员

主席：肖刚 教授 （韩山师范学院数学与统计学院）

委员：陈耀文 教授 （汕头大学）
李兵 副教授 （汕头大学）
崔岩 教授 （汕头大学）
李恪 副教授 （汕头大学）

摘要

约束多目标优化问题（CMOPs）广泛存在于现实世界各种研究中，解决这类问题需要同时优化多个目标函数和满足不同的约束条件，而且要同时优化的多个目标之间往往互相冲突、此消彼长。进化算法作为一种元启发式的随机搜索算法，凭借着自身的优势，被广泛地应用于解决 CMOPs，同时促进了约束多目标进化算法（CMOEAs）的发展。然而对于目标或约束不平衡的 CMOPs，现有的研究还不够深入。因此，本文针对不平衡的 CMOPs 进行分析和设计研究，提出了一种基于分解的混合约束多目标优化方法，主要内容如下：

本文采用基于分解的思想，结合改进的ε约束处理机制，设计了一种基于分解的混合约束多目标进化算法——εM2M。在该算法中，首先使用基于分解的策略将种群分为多个子种群，以此来保证算法的多样性；然后对每个子种群应用一种改进的约束处理机制，帮助种群在搜索空间中尽可能地跨越不可行区域，从而促进种群的收敛。两种机制的融合大大提高了算法的性能。

除此之外，为了验证所提算法的有效性，本文设计了一套符合不平衡 CMOPs 特性的测试问题集，命名为 IM-CMOPs。该测试问题集的特点是目标不平衡，且约束同时具有多样性和收敛性困难。实验结果表明，在 IM-CMOPs 测试问题集上，εM2M 算法显著优于其他 8 种具有代表性的经典算法。在另一套不平衡的约束多目标测试问题集 UCMOPs 上，εM2M 仍然明显优于其他对比算法。

最后，为了检验所提算法在实际应用中的效果，将εM2M 运用到井眼轨迹的优化问题中。通过实验与其他算法比较，结果表明，所提算法在实际工程优化问题上也取得了不错的效果。

关键词：多目标优化；进化算法；约束处理机制；井眼轨迹优化

Abstract

Constrained multi-objective optimization problems (CMOPs) are widely found in various real-world research. Solving such problems requires optimizing multiple objective functions and satisfying different constraints simultaneously, and the multiple objectives to be optimized simultaneously are often in conflict with each other. As a metaheuristic random search algorithm, evolutionary algorithms have been widely applied to solve CMOPs with their own advantages, and facilitating the development of constrained multi-objective evolutionary algorithms (CMOEAs). However, for CMOPs with imbalanced objectives or constraints, the existing research is not deep enough. Therefore, in this paper, an analysis and design study is conducted for imbalanced CMOPs, and a decomposition-based hybrid constrained multi-objective optimization method is proposed, which is mainly described as follows:

In this paper, a decomposition-based hybrid constrained multi-objective evolutionary algorithm, εM2M, is designed using decomposition-based ideas combined with an improved ε constraint handling mechanism. In this algorithm, a decomposition-based strategy is first used to divide the population into multiple subpopulations as a way to ensure the diversity of the algorithm; and then an improved constraint handling mechanism is applied to each subpopulation to help the population cross as many infeasible regions as possible in the search space, thus facilitating the convergence of the population. The integration of the two mechanisms greatly improves the performance of the algorithm.

In addition, to verify the effectiveness of the proposed algorithm, this paper designs a test problem set, named IM-CMOPs, that conforms to the properties of imbalanced CMOPs. the test problem set is characterized by imbalanced objectives and constraints with both diversity and convergence difficulties. Experimental results show that the εM2M algorithm significantly outperforms eight other representative classical algorithms on the IM-CMOPs test problem set. On another imbalanced set of constrained multi-objective test problems, UCMOPs, εM2M still significantly outperforms the other comparative algorithms.

Finally, to test the effectiveness of the proposed algorithm in practical

applications, εM2M is applied to the optimization of well trajectories problem. By comparing the experiments with other algorithms, the results show that the proposed algorithm also achieves good results in practical engineering optimization problems.

Keywords: Multi-objective Optimization, Evolutionary Algorithm, Constraint handing Technique, Well Trajectory Optimization

目 录

第 1 章 绪论

约束多目标优化问题（Constrained Multi-objective Optimization Problems，CMOPs）是我们在现实生活和工程应用中最常见的优化问题。这类问题普遍存在多个目标甚至约束条件，并且各个目标之间通常相互冲突、相互竞争，一个目标改善的同时往往会引起其他目标性能的降低。也就是说，想要同时使各个目标达到最优是不可能的，只能对它们进行协调和折中处理。因此，解决 CMOPs 是非常具有挑战性的。最初，研究者们采用运筹学中的传统方法解决了一系列的简单优化问题，比如线性规划、非线性规划等，但这些方法依然存在诸多限制，如多个约束条件和目标的存在，增加了优化问题的难度，导致传统方法不能独立解决这些问题，因此进化算法（Evolutionary Algorithms，EAs）应运而生。进化算法是一种随机搜索算法，可以从随机出现的个体中选出最优解，从而有效地解决复杂问题。进化算法凭借其基于种群的全局搜索能力、高鲁棒性、自组织、自适应等特征，在各种智能优化方法中脱颖而出，广泛应用于 CMOPs 的求解。

1.1 研究背景与意义

带约束的多目标优化问题广泛存在于众多现实工程实践和科学研究中[1-4]。解决这类问题需要同时优化多个目标函数以及满足不同的约束条件，而且要同时满足的多个目标之间往往互相冲突、此消彼长。因此，在多目标优化问题中，由于多个目标之间的相互矛盾，使得问题不能得到唯一解，而是产生一组可选的折中解集[5]。在求解多目标优化问题时，常用的解决办法有约束法、混合法、权重法等[6, 7]。这些方法将多个目标组合成一个单一的函数，不足之处在于，各个目标的权重分布带有较强的主观性，进化过程中无法改变它们的发展趋势，也不能对它们进行任何操作等。因此，传统方法对求解复杂问题（如高维数、非线性等）具有较大的局限性。遗传算法是一种自组织、自适应的人工智能技术，是模拟自然界中生物的演化过程和机理的一种方法，适合用来求解优化和搜索问题。它是一种对整个种群进行的演化运算，关注的是个体的集合，而对多目标优化问题的求解通常也是得到一组非支配解的集合。因此，遗传算法凭借自身固有的特点在求解多目标优化问题上展现出极大的优势，多目标进化算法（Multi-objective Optimization Evolutionary Algorithms，MOEAs）也从而成为解决多目标优化问题的一种非常流行的选择[8-10]。

然而，对于目标和约束不平衡的约束多目标优化问题，即不平衡 CMOPs，还未得到足够的关注。现有的 MOEAs 在优化多目标优化问题时，主要执行两个任务：一是将其种群推向帕累托前沿，以确保收敛；二是保持种群的多样性，从而保证能找到各种各样的解。尽管这两个任务的重要性相同，但大多数最先进的算法都会优先考虑收敛性，其次才去维护多样性。这意味着在下一代种群中，收敛性优于多样性的解将更容易被选择。另外，约束条件的限制使搜索空间的变得更加复杂，例如，当不可行区域占据了极大部分空间，在这种情况下，算法应该考虑如何快速高效地探索到仅存的小部分可行区域；同时，由于约束条件的存在会改变搜索区域的地形，导致不可行区域，故而也不能忽视解的可行性。众所周知，设计约束多目标进化算法（Constrained Multi-objective Optimization Evolutionary Algorithms，CMOEAs）的关键是要同时考虑收敛性、多样性、可行性，并在三者之间进行有效的权衡。所以大部分现有的 CMOEAs 从某种意义上来说并不能充分反映其有效性，尤其是在处理约束和目标不平衡的 CMOPs 时。同时，测试问题的设计也是 CMOEAs 研究中不可或缺的一部分，不同特征的测试问题可以帮助检验和评估不同算法的性能。目前已有一系列不平衡的约束多目标测试问题被提出[2]，可以用于评估不平衡 CMOEAs 的效果。但更多的是强调由于约束不均而导致的不平衡，忽略了目标空间搜索不均时的情况，考虑条件不够全面。因此，对不平衡的约束多目标测试问题集的研究也要加强，这对 CMOEAs 的发展具有重要意义。

1.2 进化算法基本概念

1.2.1 进化算法的基本思想和流程

进化算法是一种基于生物进化过程的优化方法，它通过模拟群体自然选择和遗传演化的过程来寻找最优解。该方法具备很强的鲁棒性和全局搜索能力，在诸多领域中被广泛应用，包括机器学习、数据挖掘、工业制造以及交通运输等。

进化算法的基本思想是通过不断迭代和优胜劣汰的方式，逐步提高染色体个体的适应度，以达到求解复杂问题的目的[3]。其中，染色体是表示问题解空间中的一个可行解，个体则为染色体在群体中的表现形式。在进化算法中，每个个体都有自己的适应度值，该值反映了个体与其他个体之间的竞争力。进化算法通过选择、交叉和变异等遗传操作改变个体的基因组合，以期望获得更优的解。

进化算法的流程[3]从初始化种群开始，逐步进行选择、交叉、变异等操作，生成新的种群，并通过对比新旧两代种群的适应度值，筛选出优秀的个体，使其

参与下一代的繁殖。整个过程重复多次，直到达到最大迭代次数或者满足某一终止条件为止。具体的流程如下：

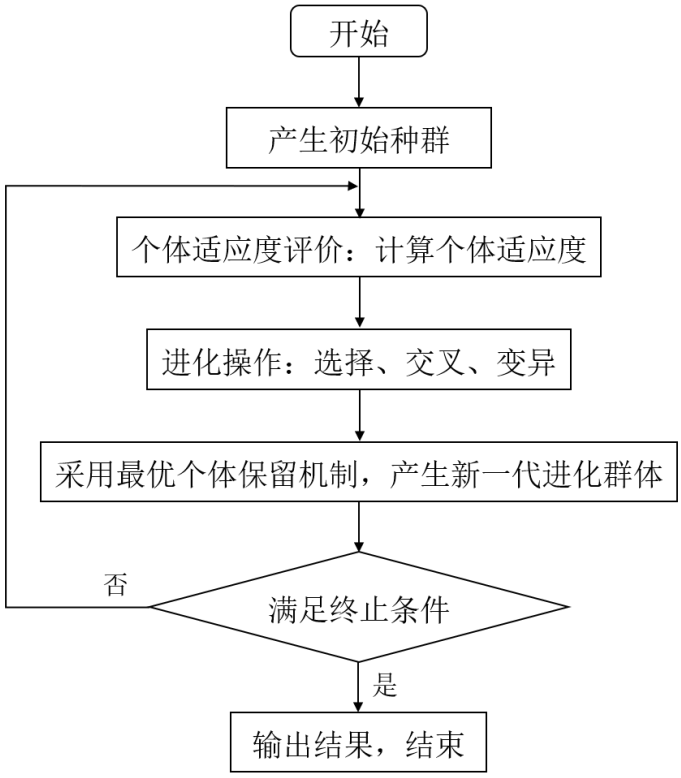


图 1-1 进化算法流程图

首先随机生成一定数量的个体，每个个体都有一组基因表示其解空间中的位置。然后对每个个体计算它们的适应度值，即目标函数的值。适应度值通常越小越好，也可以通过转换使适应度值越大越好。接着进行选择操作，进化算法中的选择操作是按照每个个体的适应度值来选择优秀的个体，并将其复制到下一代种群中。传统的选择操作方法有轮盘赌、锦标赛和精英策略等。在经过选择操作后，优秀的个体会参与交叉操作。交叉操作是将两个个体的染色体进行交叉，以产生新的染色体个体。交叉操作可以采用单点交叉、多点交叉和均匀交叉等方法。变异操作是在交叉操作完成之后，向部分个体的染色体中引入一定程度的随机变异。变异操作可以采用位变异、插入变异和互换变异等方法。通过选择、交叉和变异等操作生成新的种群，并计算每个个体的适应度值。根据设定的终止条件，判断进化算法是否达到停止条件。当满足某一终止条件时，算法停止，输出当前最优解集合。若算法未达到停止条件，则重新从第 2 步开始进行迭代操作。

总的来说，进化算法是一种全局搜索算法，它通过模拟生物进化过程，不断迭代地寻找最优解。其运行流程如图 1-1 所示，通过不断重复上述步骤，进化算法能够逐渐搜索到问题的最优解。在此基础上，还有一些进化算法的改进和扩展方法，例如粒子群优化（PSO）、差分进化算法（DE）和蚁群算法（ACO）等。

这些算法都是基于自然界现象的仿真，每个算法都有其自身的优缺点，在不同场景下可以灵活应用。

1.2.2 进化算法在多目标优化问题中的应用

多目标优化问题通常涉及到多个目标函数的优化，这些目标函数之间通常存在相互制约的关系。传统的单目标优化算法难以有效地解决这类问题，因此进化算法在解决多目标优化问题中被广泛应用。在进化算法中，通过模拟生物进化机制来实现对多个目标函数的最优解集合（即帕累托前沿）进行搜索。不同于单目标优化问题中只有一个最优解的情况，多目标优化问题中可能存在多个非支配解，也就是说这些解不能同时被其他解所支配。这些非支配解即构成了帕累托前沿。

多目标进化算法能够有效地搜索帕累托前沿，以提供多个帕累托最优解供决策者选择。常见的多目标进化算法包括 NSGA-II、SPEA2、MOEA/D 等。这些算法都是基于自然进化过程的全局搜索方法，将遗传、交叉、变异等操作应用于多目标优化问题中，从而得到一组非支配解。

进化算法在解决多目标优化问题中的应用涉及到许多领域，例如工程设计、生产调度、资源分配等[11, 12]。在工程设计中，需要考虑多个目标函数的优化，如构件强度和重量的平衡、稳定性和响应时间的平衡等。利用进化算法可以搜索一组帕累托最优解，以提供多个可供选择的设计方案。生产调度问题涉及到多个目标函数，如最小化生产成本和最大化生产效率之间的平衡。通过基于进化算法的调度策略，可以求解出一组帕累托最优解，以供制造商选择最适合其需求的调度计划。在资源分配问题中，需要考虑多个目标函数，如最大化收益和最小化资源消耗之间的平衡。通过利用进化算法，可以快速搜索到一个帕累托最优解集合，为决策者提供多个选项。

总之，进化算法在解决多目标优化问题中具有重要的应用价值。通过搜索帕累托最优解集合，可以为决策者提供多个可行方案，以适应不同的需求和限制条件。随着计算机技术的不断发展和多目标优化问题的越来越广泛，进化算法在未来仍将继续发挥其重要作用。

1.3 约束多目标进化算法研究现状

根据不同的约束处理机制可将现有的 CMOEAs 大致分为以下几类[10]：基于惩罚函数[13, 14]的方法；基于目标与约束分离的方法[15, 16]；多目标方法[17, 18]；其他方法（基于多种群协同进化的方法[19, 20]、双阶段方法[19]、混合方法等[21, 22]）。

1.3.1 基于惩罚函数的方法

基于惩罚函数的方法主要是根据约束的违反程度来构造惩罚项，并将其加入目标函数中，从而把带约束的优化问题转化为对应的无约束版本。在此类方法中，如何设置惩罚系数是最重要的一个环节，也是影响算法性能的重要因素。惩罚方法可以根据不同的设置方式分为静态、动态和自适应三种模式。

静态法意味着在进化过程中惩罚系数不会发生变化，但在进化的早期和后期，目标和约束之间的偏好保持不变会影响目标与约束之间的平衡。因此，静态惩罚函数的方法更适用于解决简单的问题，对于复杂问题则达不到理想的效果。

动态法中的惩罚系数会随着进化代数或其他指标的变化而变化。惩罚系数的动态变化，导致在不同的进化阶段中，目标和约束条件的权重也有所不同，从而实现两者之间的平衡。但是，这种方法的困难在于要设计出合适的变更规则，需要对不同的问题设置不同的变更规则。

自适应方法通过保持群体在进化过程中的相关信息，并利用这些信息对群体进行修正，从而实现对惩罚因子的调节。相对于以上两种方法，由于反馈信息能够引导算法的演化，使得算法能够更有效地处理更复杂的问题。

罚函数方法在理论上是可行的，但是在实践中还存在着一定的缺陷。由于对限制条件的处罚太重或者太轻，都会使演化算法无法找到最佳的结果。当惩罚因子设定太大时，会使搜索空间变小，虽然这种方法可以使算法更快地收敛至最优解，但是，当一个问题的求解空间上有多个连续的较小的可行域时，该方法仅能收敛至较少的局部，而无法得到所有的最优解集。如果惩罚系数设定得太小，则搜索空间就太大，这会导致算法不能高效地搜索到可行区域内的可行解，而会将大部分的计算资源浪费在对不可行解的搜索上，特别是在极其狭窄的可行区域中，算法找到可行解的可能性更低。

1.3.2 基于目标与约束分离的方法

在当前的进化算法中，目标与约束分离是一种常见的处理办法。不同于惩罚函数的方法，约束与目标分离的方法是将目标和约束分开进行处理。这样，在构建新的惩罚函数时可以很好地解决参数设定的难题，同时也优化了目标函数，有效地解决了约束的问题。这些方法分别比较了目标和约束条件，主要包括约束支配准则（CDP）[23]、ε约束方法[24, 25]和随机排序（SR）[26]。

1）CDP

CDP 由 Deb 等人[23]提出，由于其简单、易于实现而最常用。CDP 采用以下标准对个体 A 和 B 进行比较：当 A 和 B 都是可行解时，选择适应度值更好的解；

当 A 是可行解，B 是不可行解时，选择 A；当 A 和 B 都是不可行解时，比较两个解的约束违反值，然后选择违反程度较小的那个解。CDP 的操作相对简单，但更倾向于可行解，当 CMOP 存在离散可行区域或不可行障碍时，会导致种群进入一些局部可行区域。

2）ε约束方法

该方法由 Takahama[25]等人提出，它通过使用一个参数ε来放松约束。ε逐渐减小，当个体的约束违反度小于ε时，认为是可行解。显然，当ε减少为 0 时，ε约束方法与 CDP 相同。ε约束方法使用了以下标准来比较个体 A 和 B：如果 $\mathrm{CV(A)} \leq \varepsilon$、$\mathrm{CV(B)} \leq \varepsilon$和 A 支配 B，则选择 A；如果 $\mathrm{CV(A)} \leq \varepsilon$、$\mathrm{CV(B)} > \varepsilon$，则选为 A；如果 $\mathrm{CV(A)} > \varepsilon$、$\mathrm{CV(B)} > \varepsilon$和 $\mathrm{CV(A)} < \mathrm{CV(B)}$，则选为 A。Fan 等人[27]改进了ε约束处理方法，并将其整合到 MOEA/D 框架中，采用当前总体中可行解的比例来动态调整ε参数水平。Yang 等[28]提出了一种动态约束处理机制，该机制将搜索过程分为两种模式，无约束搜索和有约束搜索。在约束搜索模式下，运用了改进的ε约束方法，从而提高了种群的多样性。

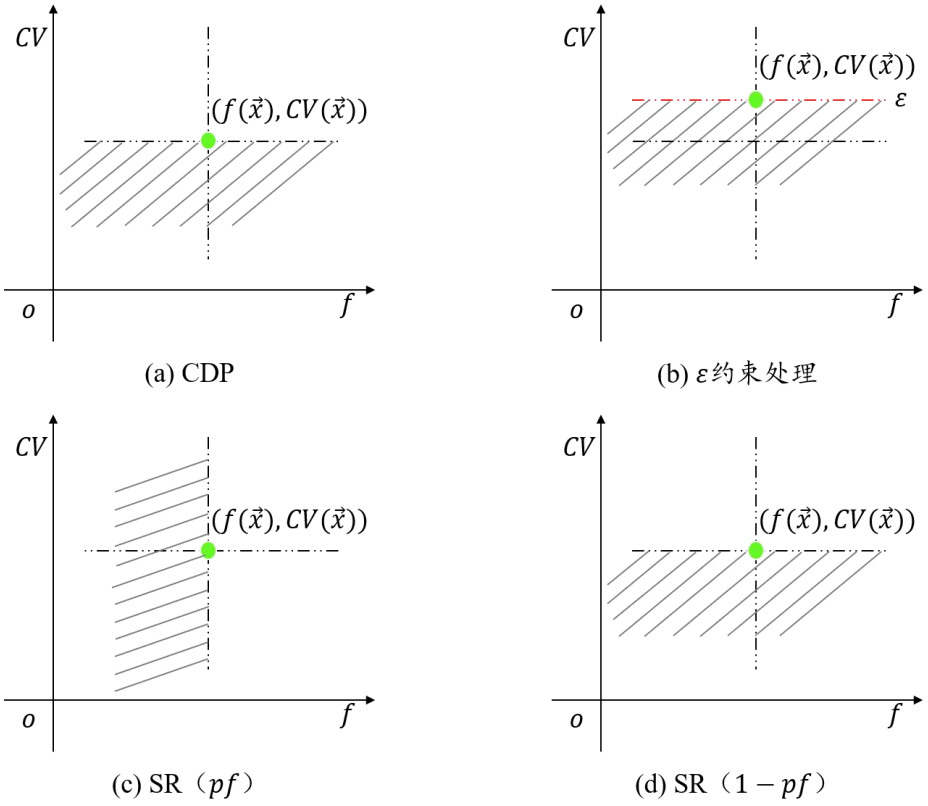
3）SR

该方法引入了一个概率参数P_f。 对任意一对相邻的个体，若二者均是可行的，将二者的适应度进行比较，适应度较好的个体保存到下一代。反之，也就是，在两个个体中，其中一个不是可行的，或者两者都是不可行的，那么就设定一个概率P_f，根据目标适应值的大小，用概率P_f来对解进行排序，将具有较好适应值的个体保留下来；根据约束违反程度，用概率（$1-P_f$）对解进行排序，筛选出约束违反程度较低的个体进行保留。该方法在一定程度上能够利用目标函数的信息。

图 1-2 为 CDP 法、ε约束法和 SR 法图解。$\vec{x}$表示一个个体，其目标值和约束违反度分别为$f(\vec{x})$和$CV(\vec{x})$。当将其他个体与$\vec{x}$进行比较时，如果它们位于灰色阴影中，这意味着它们比$\vec{x}$更好。显然，当使用 CDP 来选择个体时，它将优先考虑可行解。这样，种群的收敛速度非常快，但也会使种群很容易陷入局部最优状态。而ε约束法和 SR 法都在一定程度上弥补了 CDP 的缺陷，因为一定程度上考虑了不可行解的信息。然而，ε约束方法和 SR 中的参数设置也具有挑战性，因为该参数可能依赖于问题。

以上三种目标与约束分离的方法是将种群划分为可行和不可行区域，然后运用各自的筛选机制来保留精英个体至下一代。然而当遇到不可行区域较小的 CMOPs 时，这些算法却不能取得很好的效果，因为它们没有尽可能地处理好不可行解中携带的信息，忽略了这些信息对种群的进化也起着推动作用。因此，这

三种方法都需要根据具体情况进行调整，以确保最终结果达到更高的准确性和可靠性。



(a) CDP　　(b) ε约束处理

(c) SR（pf）　　(d) SR（$1-pf$）

图 1-2 CDP 法, ε约束法和 SR 法图解

1.3.3 基于多目标的方法

在多目标方法中，将约束条件视为一个附加目标或多个附加目标，将 CMOP 转换为无约束对应版本，然后利用 MOEAs 来解决转换后的问题。变换后的目标函数如下：

$$\min \vec{F}(\vec{x}) = \left(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}), CV(\vec{x})\right)^T \tag{1-1}$$

或者

$$\min \vec{F}(\vec{x}) = \left(f_1(\vec{x}), \ldots, f_m(\vec{x}), cv_1(\vec{x}), \ldots, cv_k(\vec{x})\right)^T \tag{1-2}$$

式中$\vec{F}(\vec{x})$代表目标向量，$CV(\vec{x})$代表约束违反，m和k分别表示它们的维数。文章[28]使用了此种方法，分别基于惩罚函数和违反约束的数量将约束条件转化为两个新的目标[29]。Ray 等人[30]也采用了多目标的思想，在约束违反的基础上增加了一个新的优化目标，设计了一种以不可行为驱动的方法，使种群从不可行区

域不断地逼近其约束边界。Isaacs 等人[31]同样将约束等价转换为一个新目标，将原本m个目标的约束优化问题转化为具有$m+1$个目标的无约束问题。Long 等人[32]构造了一个新的求解 CMOPs 的约束处理机制，并将所得解的收敛性、多样性和可行性作为多目标子问题的三个新目标。Peng 等人[18]为求解 CMOPs，以约束违反度为新目标，设计了一种基于定向权值的约束处理机制，采用分别分布在可行和不可行区域的两种权值来引导种群向有希望的区域进行搜索。

1.3.4 其他方法

1）基于多种群的协同优化问题

为了更高效地解决约束多目标优化问题，许多研究者利用转化的思想，将 CMOPs 转换为协同优化问题或者双阶段优化问题。多方面的研究表明此类方法可以帮助种群更好地探索到搜索空间，使群体能够发掘出有用的潜在信息，并最终得到完整的约束帕累托前沿（Constrained Pareto Front，CPF）。

Chafekar 等人[33]将一个 CMOP 转化为多个单目标优化问题，然后使用不同的遗传算法优化每一个目标，最后交换其目标信息。Wang 等人[34]也提出了一个基于多种群的协同进化框架，该框架将种群分为m个子种群，每个子种群优化一个带约束的目标，即将问题转换成单约束优化。此外，为了接近 CPF，使用了一个归档集来保存所获得的约束非支配解。Liu 和 Wang[35]也是将一个大问题分解成几个小问题，再为每个小问题建立各自的子种群和临时归档集，最终实现对每个子问题的协同进化。随后，他们又提出了一种基于边界搜索和归档[36]的约束处理方法，在之前的基础上加强了对边界的搜索，算法的性能因此大有改善。为了保证种群的多样性，Yang 等人[37]同样是使用将一个问题分解为多个子问题的方法，并且对多个子问题使用不同的约束处理机制。Liu 等人[38]将一个 CMOP 转化为基于双种群的协同优化问题，其中一个种群负责优化约束，另一个种群负责优化目标。两个种群相互作用，同时二者之间还伴随着知识和信息的迁移。Tian 等人[39]提出了一个求解 CMOPs 的共同进化框架（CCMO），其中一个种群负责求解原始的 CMOP，另一个种群求解与原始 CMOP 对应的无约束版本。这意味着两个种群分别搜索到带约束的PF和无约束的PF，从而相互作用实现协同进化。Li 等人[40]设计了一种基于双存档集的算法（C-TAEA），两个存档集一个面向收敛性一个面向多样性。面向收敛的存档旨在推动种群趋向帕累托前沿，面向多样性的存档用于探索前者未开发到的区域从而维持种群的多样性。Wang 等人[41]让一个种群在进化的早期不考虑约束，而后期只考虑约束。另一个种群则正常工作，最终将两个种群融合。Liu 等人同时采用一个主种群和一个归档种群[42]，具体来说，主种群保持了可行性并从可行侧转移到 CPF，存档种群保持种群多样性并从

不可行侧接近 CPF。

2）基于双阶段的优化问题

Fan 等人[19]提出了一个推拉搜索（PPS）框架，将整个搜索过程分为推和拉两个阶段。在推阶段不考虑约束，得到无约束的 PF。在拉阶段，采用约束处理机制将种群拉回到可行区域。而后，Wang 等人[43]又在此基础上添加了一种基于分解的方法，每个子问题对应一个子群体，将 PPS 应用于每个子群体来解决相关的子问题。Wang 等人[44]提出了一种目标空间修正机制，使有前途的不可行解能够更有效地寻找最优解。此外，利用 PPS 搜索框架来调整 PF 的位置，防止种群落入局部最优状态，从而降低了时间复杂度。Tian[45]等人提出了一个基于双阶段的进化算法（CMOEA-MS），两个阶段间的切换会根据设定的条件自适应触发。其中一个阶段协助种群到达可行区域，另一个阶段使种群沿可行边界扩散。对于在决策空间和目标空间中都有约束的 CMOPs，Liu 和 Wang 等人[46]在第一阶段着眼于单目标问题以便寻找有希望的可行区域，第二阶段则探索真实的 PF。Xiang 等人[47]同样如此，第一阶段只考虑目标寻找 UPF，第二阶段强调逐渐接近 CPF。在 Yu 等人[48]设计的方法中，第一阶段旨在平衡收敛性和多样性，第二阶段维持可行性和多样性，从而覆盖分布良好的帕累托前沿。Ming 等人[49]同样是双阶段的方法，第一阶段寻找 UPF，第二阶段集中探索 CPF。

3）混合方法

由于约束问题的复杂多变，仅使用一种处理技术很难达到预期的结果，因此，构建一种高效的、自适应的、可混合的、可组合的控制技术成为当前约束处理技术的发展趋势。混合式算法是指将两个或多个不同的约束处理技术联合起来处理一个约束问题，它可以充分发挥这些技术各自的优势，从而更加高效地求解一个约束最优问题，然而，在这种算法中，如何将这些技术进行合理地组合，是其关键所在。

虽然 EA 可以从全局的角度找到有希望的领域，但它在局部区域的搜索可能不如数学规划，特别是面对等式约束，通常会减少搜索空间的维数。许多约束优化算法将等式约束放宽为不等式约束，这并不能严格保证所得到解的可行性。一些研究人员已经将 EA 和数学规划结合起来来处理 CMOPs。Kelner 等[21]提出了一种将遗传算法与基于内点法的局部搜索策略相结合的混合优化技术。Datta 等人[50]提出了一种新的多目标优化技术，该技术将内点法与非凸径向边界交点相结合。径向边界相交将 CMOPs 分解为几个子问题，从而找到沿等距线径向向外的最近参考点的解。Lara 等人[22]提出了搜索过程中基于子空间的运动的构造方法，其中使用多目标随机局部搜索来引导个体。Uribe 等人[51]用特殊的局部搜索机制杂交 MOEAs，提出了一种双目标优化问题的下降方向计算方法，在局部搜索中

考虑约束信息。Cuate 等[52]将 MOEA 与类延续技术相结合，获得了一个快速、可靠的数值求解器。Schutze 等[53]将梯度子空间近似与 EA 相结合来求解 CMOPs，这允许根据给定的邻域信息以最佳拟合的方式计算下降方向。

在搜索过程中怎样合理有效地处理不可行解是创造 CHT 的关键。对于可行解比例较低的 CMOPs，可以将搜索过程分为两个部分，无可行解阶段和有可行解阶段。一个有效的 CHT 应当充分利用不可行解的优势，使不可行解中携带的信息协助种群穿越不可行障碍到达可行区域，并且能够尽可能快速地接近最优可行区域。这使得 CHT 在整个过程中都要考虑所有目标和约束。然而，现有的 CHT 在无可行解阶段常常忽视了个体间的空间分布，从而容易陷入局部最优导致无法找到可行解。因此，使用有效的 CHT 对一个算法高效地在搜索空间内找到最优解具有非常重要的意义。

本节讨论了 CHT 和 MOEAs 之间的关系。一般来说，一个 CMOEA 是由 CHT 和 MOEA 组成的。具体来说，使用 CHT 来处理约束条件，确保解的可行性，而使用 MOEA 作为主要的演化能力，用一组收敛良好且分布良好的可行解来逼近 PF。自然地，通过将 MOEAs（即基于支配的[54]、基于分解的[55]和基于指标的算法[56]）与 CHTs 集成来设计 CMOEAs 是一种可行的、简单的方法，因为它们都已经被广泛地单独研究。根据目前的研究，我们观察到 CHT 经常与基于显性和基于分解的 MOEAs [57，58]结合，因为它们比基于指标的 MOEAs 具有更好的灵活性和泛化性。特别是对于基于分解的 MOEA，它们将一个 CMOP 转化为多个单目标优化问题。因此，许多应用于约束单目标优化的 CHT 可以直接嵌入。对于

表 1-1 各种方法的优点和局限性

种类	优点	不足
基于惩罚函数的方法	简单易行，操作灵活，应用范围广泛。	惩罚系数难以调整，在处理更复杂的问题时，算法的性能会不满足。
基于目标和约束条件分离的方法	简单、易于实现，收敛速度快。	约束条件和目标难以平衡，相关参数也难以设置。
基于多目标的方法	有效地平衡约束和目标，使种群具有良好的多样性。	很难设计出附加的目标函数。此外，该方法可能会导致多目标优化问题，使算法效率低下。
将 CMOP 转化为其他问题的方法	有助于种群更好地利用搜索空间中的信息，避免了直接解决原始 CMOP 时遇到的困难。	很难设计出一种有效的转换技术来保证与原问题的等价性。如果转换机制不合理，由于计算资源效率低或与原问题的偏差，算法性能会急剧恶化。

使用质量指标来定义选择机制的基于指标的模型模型，它们是处理由于选择压力[56]的增加而导致的多目标优化问题的一种很有前途的方法。因此，将基于指标的MOEAs 与 CHT 相结合是解决约束多目标优化问题的可行替代方案。

此外，基于上述对现有 CMOEAs 的综述，表 1-1 中总结了各类别的优点和局限性。虽然现有算法在解决 CMOPs 方面表现出了良好的性能，但仍需要考虑一些研究差距。

1.4 主要研究内容和论文章节安排

本文主要针对目标和约束不平衡的 CMOPs 进行研究，采用基于分解的思想，结合改进的ε约束处理机制，设计了一种基于分解的混合约束多目标进化算法——εM2M。

在该算法框架中，首先使用基于分解的策略将种群分为多个子种群，以此来保证算法的多样性；然后对每个子种群应用一种改进的约束处理机制，帮助种群在搜索空间中跨越不可行区域，从而能够较好地收敛到真实的帕累托前沿。两种方法的融合大大提高了算法的性能。

除此之外，为了证明所提算法的有效性，本文设计了一组符合不平衡 CMOPs 特性的测试问题集，命名为 IM-CMOPs。该测试问题集的特点是目标不平衡，且约束同时具有多样性和收敛性困难。实验结果表明，在 IM-CMOPs 测试问题集上，εM2M 显著优于其他 8 种具有代表性的经典算法。在另一套不平衡的测试问题集 UCMOPs 上，εM2M 也是明显优于其他对比算法。

最后，为了检验所提算法在实际应用中的效果，将εM2M 运用到井眼轨迹的优化问题中。通过实验对比其他算法，发现本文提出的算法在工程优化上也取得了不错的效果。

主要的结构安排如下：

第一章是绪论部分，这部分首先介绍了研究背景与意义，接着总结了约束多目标进化算法和约束处理机制的研究现状，对现有的约束处理方法进行了详细的阐述。

第二章是相关理论的介绍。本章围绕本文的研究内容，详细解释和分析了不平衡的约束多目标优化问题；其次，介绍了不平衡 CMOPs 对应的测试问题，以及本文所提测试问题的设计来源。最后介绍了对于井眼轨迹优化问题模型求解的经典方法。

第三章提出了一种基于分解的混合约束多目标进化算法用于求解不平衡约束多目标优化问题。首先设计了符合不平衡 CMOPs 特性的测试问题集

IM-CMOPs，详细描述了设计背景和问题细节；然后介绍了分解和改进的ε 处理机制的核心思想，提出了εM2M 算法框架。最后与现有的 8 种 CMOEAs 一起在两类具有不平衡特性的测试问题集上进行对比实验，并对结果进行分析。

第四章是实际问题的应用。首先进行了井眼轨迹优化问题模型的建立，然后将εM2M 和上述对比算法运用到该模型中进行对比实验，最后对实验结果进行分析。

第五章是总结与展望，主要围绕着不平衡的约束多目标优化问题和测试问题集的设计以及算法的改进开展了总结，同时提出了未来的研究方向。

第 2 章 相关理论问题介绍

2.1 约束多目标优化问题

2.1.1 CMOPs 基本概念

一般来说，一个约束多目标优化问题（Constrained Multi-objective Optimization Problem，CMOP）在数学形式上可表示为[9, 59]：

$$\begin{aligned}&\min \vec{F}(\vec{x})=\left(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})\right)^T \\ &\text{s.t.} \begin{cases} g_j(\vec{x}) \leq 0, j=1, \ldots, l \\ h_j(\vec{x})=0, j=l+1, \ldots, k \\ \vec{x}=\left(x_1, x_2, \ldots, x_D\right)^T \in \mathbb{S} \end{cases}\end{aligned} \tag{2-1}$$

其中，$\mathbb{S}$是决策空间，$\vec{x}$是$\mathbb{S}$中具有D维的决策向量。$\vec{F}(\vec{x})$是目标向量，其中包含m个需要解决的目标。$g_j(\vec{x}) \leq 0$ 表示第j个不等式约束。$h_j(\vec{x})=0$ 是第$(j-l)$个等式约束。l和$(k-l)$分别表示不等式约束和等式约束的个数。

对于决策向量$\vec{x}$，其对第j个约束的约束违反程度计算为：

$$cv_j(\vec{x})=\begin{cases} \mathrm{m}\left(0, g_j(\vec{x})\right), & j=1, \ldots, l \\ \mathrm{m}\left(0,\left|h_j(\vec{x})\right|-\delta\right), & j=l+1, \ldots, k \end{cases} \tag{2-2}$$

其中，δ的存在是为了放宽等式约束。通过在每个约束上添加$\vec{x}$的约束违反度，可以得到其总的约束违反度。

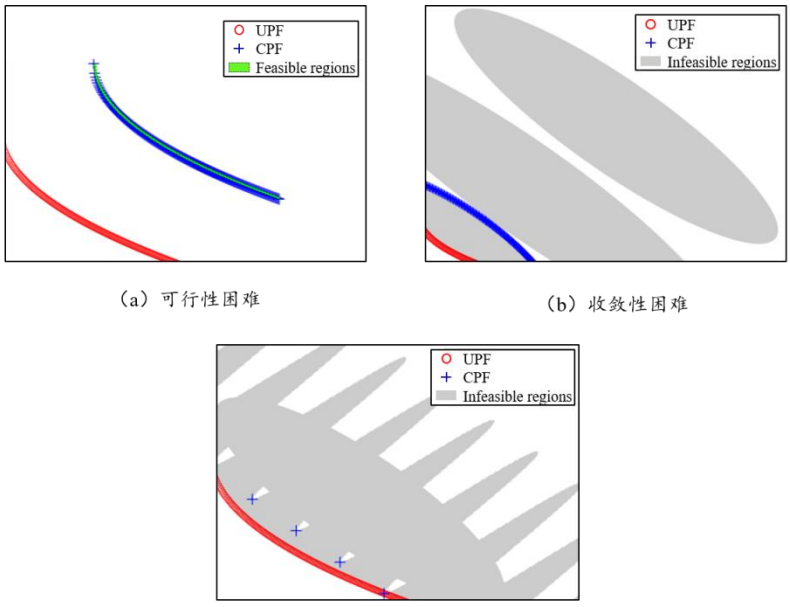
$$CV(\vec{x})=\sum_{j=1}^{k} cv_j(\vec{x}) \tag{2-3}$$

根据总约束违反CV，可以判断一个解的可行性。如果一个解的CV值为零，称为可行解；否则，则为不可行解。

对于两个可行的解决方案$\vec{x}_1$和$\vec{x}_2$，如果每一个$i \in \{1, \ldots, m\}$都存在$f_i(\vec{x}_1) \leq f_i(\vec{x}_2)$，并且在$j \in \{1, \ldots, m\}$至少有一个$f_j(\vec{x}_1) < f_j(\vec{x}_2)$，则称为$\vec{x}_1$支配$\vec{x}_2$。如果一个解不被任何解所支配，则称这个解为帕累托最优解。由所有帕累托最优解组

成的集合是帕累托最优解集（Pareto optimal Set，PS），PS 在目标空间上的映射称为帕累托前沿（Pareto Front，PF）。

约束条件可以将可行空间划分为窄的不连通区域，使大部分搜索空间不可行，或使无约束 MOP 的 PF 部分或完全不可行。为了便于区分，通常使用约束帕累托前沿（Constrained Pareto Front，CPF）和无约束帕累托前沿（Unconstrained Pareto Front，UPF）来表示有约束和无约束多目标优化问题[19, 47, 60, 61]的 PFs。显然，无约束多目标优化问题关注的是 UPF，而在 CMOPs 中需要覆盖具有良好收敛性和分布性的 CPF。具体来说，良好的收敛性意味着解可以准确地接近 PF，良好的分布代表了解在目标空间中具有良好的扩散和均匀性。



（a）可行性困难　　（b）收敛性困难

（c）多样性困难

图 2-1 不同类型的困难示例

具体来说，在处理 CMOPs[62]时，将会遇到以下主要挑战：

1）可行性难度：如图 2-1(a)所示，约束会使搜索空间中的很大一部分变成不可行区域。所以，要找到一个非常小的可行区域也是非常困难的。

2）收敛性难度：如图 2-1(b)所示，不可行区域会阻碍通往 CPF 的道路。一个算法很容易被困在可行的外部区域，因此要找出真实的 CPF 就没那么简单了。

3）多样性难度：如图 2-1(c)所示，断开的可行区域将使真正的 CPF 由多个不连通的片段组成，覆盖它们是一项具有挑战性的任务。

鉴于上述挑战，实现收敛性、多样性和可行性之间的平衡是解决 CMOPs 的重要和关键因素。

2.1.2 不平衡的 CMOPs

一般来说，一个 MOEA 在优化一个多目标优化问题（Multi-objective Optimization Problem，MOP）时，主要执行两个任务：其一，将其种群推向 PF，从而确保收敛；其二，保持种群的多样性，以保证能找到各种各样的解。因此，多目标进化算法的设计必须能够有效地完成上述两种任务。尽管这两个任务的重要性相同，但大多数最先进的算法都遵循“收敛性第一，多样性第二”[2]的原则。这意味着在选择下一代种群时，收敛性优于多样性的解将更受青睐。例如，Elitist Non-dominated Sorting Genetic Algorithm（NSGA-II）[23]和 Strength Pareto Evolutionary Algorithm 2（SPEA2）[63]首先强调非支配解的收敛，接着为了维持种群的多样性去处理不那么拥挤的非支配解。而基于分解的方法，如基于分解的多目标进化算法（MOEA/D），也会优先选择非支配解，而不是最多样化的解。据观察，尽管这些 EMO 算法在优化许多现实世界的 MOP 方面取得了很大的成就，但它们在优化某一类问题时遇到了困难，这些问题需要更加重视多样性的保存。我们称这些问题为不平衡问题，因为这些问题在强调收敛和多样性之间产生了不平衡，需要给予多样性同等或更多的重视。

不平衡问题是指在 PFs 前面的不同区域具有不同程度搜索难度的 MOP。PF 的某些部分比其他部分更容易搜索，而且 PF 上容易找到的部分往往在搜索空间中占据相对较大的区域，因此在 PF 其余部分很难找到有利的解。研究表明，如果有一个收敛和多样性平衡的机制，这种不平衡的问题可以得到有效优化，如基于分解的进化算法（M2M）[64]，研究表明 M2M 在多样性和收敛性两方面都取得了不错的效果。

Liu 等人给出了不平衡问题的定义[65]，假设一个 MOP 的某个前沿子集（称为“有利子集”）所对应子问题的计算难度明显低于其他不受欢迎前沿子集（“不利子集”）所对应子问题的计算复杂度；并且，有利子集的解在可行变量空间中比不利子集的更加优越，即支配了更多的可行空间。则这个 MOP 被定义为一个不平衡的 MOP。

虽然上述定义没有明确量化一个 PF 子集对另一个 PF 子集的青睐程度，但很明显，在这样的问题中，如果被青睐的 PF 相对容易找到，并且大部分可行搜索空间都被相应的 PS 所支配，这两个条件使得种群很难收敛到整个真实的 PF。在这类问题中，多样性的保护必须比收敛更重要，这样尽管一个 PF 子集天生有利，但多样性保护迫使不利的 PF 被发现和保持。一个子集对另一个子集的偏爱程度，会导致算法的差异性，可能取决于算法的搜索能力，因此很难量化。Peng 和 Liu 等人进而提出了一组不平衡的测试问题套件[2]，并对其进行分析。这组测

试套件也是评价本文提出算法的主要测试问题集之一。

在 PF 中对一个子集的偏好可以通过以下方式来引入。

1）不平衡映射：PF 的有利部分可以很容易在可行搜索空间找到，而 PF 的不利部分只能在相对较小的搜索区域找到。实现此目标的一种方法是，从搜索空间到有利的 PF 进行多对一的映射，以便可以很容易地得到有利的 PF。

2）变量关联：变量关联可能导致 PF 中的某些部分很难找到。某些不平衡问题可以这样构造，即，有利的解集来源于相互关联较弱的变量，而不利的解集则来源于变量高度关联的部分。这些特点会使算法更容易找到 PF 的有利部分。

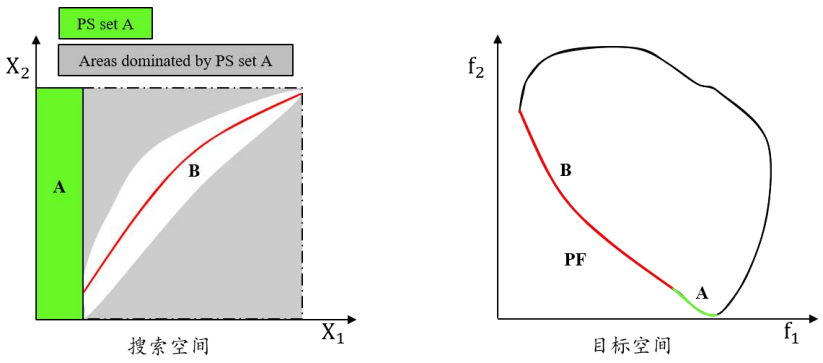
3）约束隔离：不利的 PS 子集可能靠近约束边界，而有利的 PS 子集则通常位于约束边界内部，这会导致算法难以找到和维护不利子集的可行解。

图 2-2 说明了上述三个场景。图 2-2（a）说明了一个问题，在这个问题中，完整区域 A 映射到了 PF 的一个子集（用 A 标识）。因此，在搜索空间中，A 区域内的任意一点自动成为帕累托最优点，并位于 PF 的有利子集中。此外，PS 区域 A 还支配了灰色阴影区域。因此，可能会存在一个狭窄的搜索区域（用白色标出），其可能会导致 PF 子集 B（不利子集）的出现。这两个方面都会使算法难以找到整个 PF（A 和 B）。

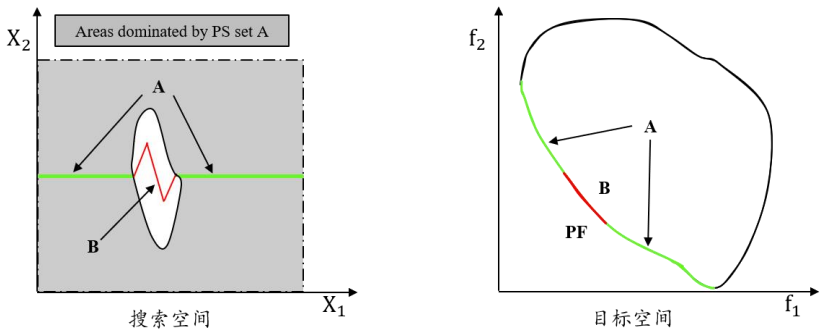
图 2-2(b)所示问题的特点是，PF 的有利部分来自于搜索空间中的两条线段，但支配着大部分搜索空间。更糟糕的是，PF 的不利部分（标记为 B）需要变量之间遵循特定的非线性关联。

图 2-2(c)展示了一个因约束而导致难以到达不利 PF 的问题。不利子集（B）周围的搜索空间受到强烈的约束，这使得算法更难以找到并保持 B 中的 PS 点。与此相反，有利子集（A）处的 PS 点远离约束边界，一旦这些点被发现，它们便会占据大部分的搜索空间，因此导致搜索过程不平衡，使算法难以同时保持 A 和 B 中的多样性 PS 点。

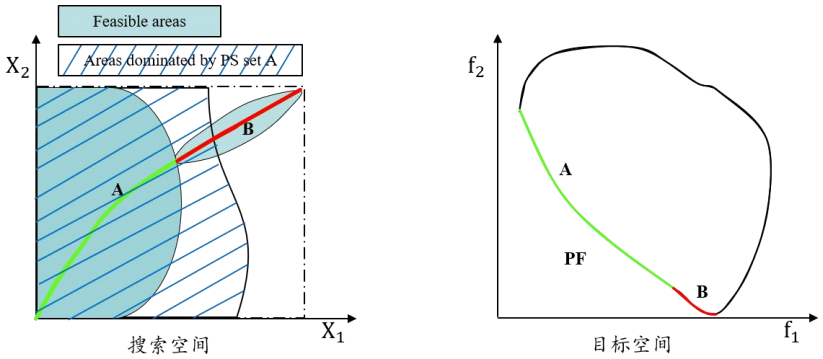
这些问题的性质可以总结如下。1）PS A 相对容易找到。2）PS A 支配了大部分的搜索空间。3）PS B 很难找到。



（a）不平衡映射



（b）变量关联



（c）约束隔离

图 2-2 不平衡问题图解

2.2 不平衡 CMOPs 测试函数

一般来说，在一个有约束的多目标优化问题中，约束条件将搜索空间划分为几个区域。在某些情况下，约束条件使得找到 PS 的某个特定子集比找到 PS 的其余部分更加容易。这个特定的 PS 子集被称为 PS 的有利子集。而更难找到的部分则是 PS 的不受欢迎的子集，被称为不利子集。它们在 PF 中的对应部分被称为有利 PF 和不利 PF。

UCMOPs 测试问题集是 Peng 等人[2]在近两年提出的一系列约束不平衡的 CMOPs 测试套件，这组测试套件包含三种不同类型的约束。此外，测试套件的不平衡程度可以通过调整相应的参数加以扩展。

UCMOPs 目标函数框架如下：

$$\begin{aligned}
&\min f_i(\mathbf{x}) = \alpha_i\left(\mathbf{x}^I\right) + \beta_i\left(\mathbf{x}^{II} - S\left(\mathbf{x}^I\right)\right)\\
&\text{s.t.}\\
&g_j(\mathbf{x}) \le 0 \quad j = 1,2,\ldots,q\\
&h_j(\mathbf{x}) = 0 \quad j = q+1,\ldots,l\\
&\mathbf{x} \in \mathbf{D}
\end{aligned} \tag{2-4}$$

其中$i = 1,2,...,m$，α_i定义了 PF 的形状，β_i是距离函数，$S(\mathrm{x}^I)$是一个从D_1^r到D_{r+1}^n的函数，其中$\mathrm{D}_{\mathrm{k}}^{\mathrm{p}}(k \le p)$表示由一个变量的第$k$维到该变量的第$p$维组成的决策空间。$g_j(x)$和$h_j(x)$分别是不等式和等式约束。一个$n$维的变量$x$由两部分组成，即$\mathrm{x}^I = (x_1, x_2, ..., x_r)$ 和 $\mathrm{x}^{II} = (x_{r+1}, x_2, ..., x_n)$。

2. 2. 1 I 型 UCMOP

在 I 型 UCMOP（UCMOPI-X）中，目标空间被划分为两部分。一部分是可行的，另一部分是不可行的，除了 PF 附近的一个小区域。第一类型的约束条件如下：

$$\begin{aligned}
&g_1(\mathbf{x}) = \begin{cases} 15\sin\left(0.5\pi\sum_{i=1}^{m-1} f_i(\mathbf{x})\right)\sum_{j=1}^{k}\frac{|t_j|}{1+e^5|t_j|} - a \le 0 & \text{if } d \ge 0\\ 0 & \text{otherwise}\end{cases}\\
&g_2(\mathbf{x}) = \begin{cases} 10\log_2\left(\sum_{i=1}^{m-1} f_i(\mathbf{x}) + 1\right)\sum_{i=1}^{m}\beta_i\left(\mathbf{x}^{II}\right) - a \le 0 & \text{if } d \ge 0\\ 0 & \text{otherwise}\end{cases}\\
&d = \sum_{i=1}^{m-1} f_i(\mathbf{x}) - \frac{3}{7} f_m(\mathbf{x}),\\
&t_j = x_{l_j}^{II} - \sin\left(\frac{0.5\pi}{r}\sum_{i=1}^{r} x_i^I\right),\\
&n - k < l_j \le n.
\end{aligned} \tag{2-5}$$

这里n是决策变量x的维数，k是预定值。a是可调整的参数，控制不平衡程度。a的值越小，UCMOPI-X 的问题难度越大。如图 2-3 所示，UCMOPI-X 中的约束条件将整个目标空间分解为两部分。有利的 PF 被可行区域包围，而不利的 PF 被不可行区域包围。一个可调参数 a 用于控制不平衡的程度。a 的值越小，算法就越难找到不利的 PF。这可能导致可行区域和不可行区域之间的不平衡搜索，特别是对于使用基于偏好可行解而非不可行解的约束处理技术的 CMOEAs。当使用 MOEA/D 或 NSGA-II 求解 UCMOPI-X 时，他们将无法找到不利的 PF，因为它们在进化过程中无法找到一些不可行的解。

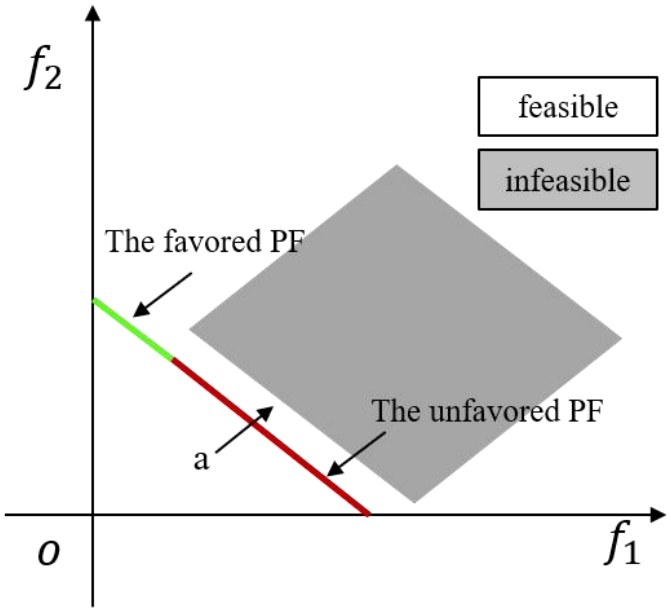


图 2-3 I 型约束图解

2. 2. 2 II 型 UCMOP

II 型 UCMOP（UCMOPII-X）是针对那些无法有效保持多样性的 CMOEAs，目的是使这些算法更加具有挑战性。目标空间通过一个不可行带（infeasible band）分解成两部分，而且有两个不可行带分别靠近 PF 中的两个部分。第二类约束条件表述如下：

$$
\begin{aligned}
g_1(\mathbf{x}) &= \begin{cases} 10\left(1-\sum_{i=1}^{m}\beta_i(.)\right)\left(\sum_{i=1}^{m}\beta_i(.)-0.5\right)\le 0 & \text{if } d<-0.25 \\ 10^9 & \text{if } |d|\le 0.25 \\ 10\left(a-\sum_{i=1}^{m}\beta_i(.)\right)\left(\sum_{i=1}^{m}\beta_i(.)-b\right)\le 0 & \text{otherwise} \end{cases} \\
g_2(\mathbf{x}) &= -\sin\left(\frac{\pi}{am}\sum_{i=1}^{m} f_i(\mathbf{x})\right)\le 0 \\
d &= \frac{\sum_{i=1}^{m-1} f_i(\mathbf{x}) - f_m(\mathbf{x})}{\sqrt{m(m-1)}}, \\
a &> b, \\
\beta_i(.) &= \beta_i\left(\mathbf{x}^{II} - S\left(\mathbf{x}^{I}\right)\right).
\end{aligned}
\tag{2-6}
$$

UCMOPII-X 的示例图如图 2-4 所示，图中有两个不相连且宽度不同的不可行带靠近 PF。UCMOPII-X 可以通过扩大靠近 PF 不利部分的不可行带（即减小 b 值，增大 a 值）来增加失衡程度。

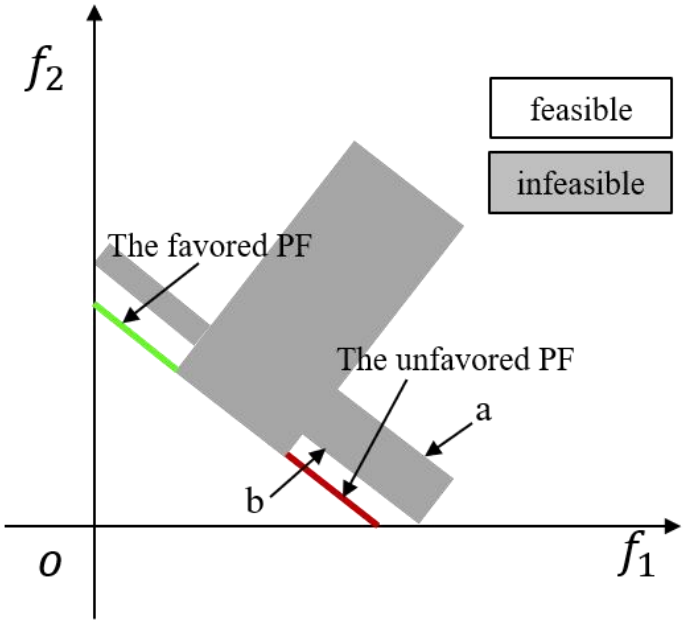


图 2-4 II 型约束图解

II 型约束引入了两个接近 PF 的不同宽度的断开的不可行带，如图 2-4 所示。这类问题的缺点在于，一些保留不可行解的 CMOEAs 可以很容易地找到有利 PF，而在有利 PF 中找到的解将消除大多数难以通过更大不可行带的解。因此，没有任何机制来维持不利 PF 周围的解的 CMOESs 最终倾向于只收敛到有利 PF。

2. 2. 3 III 型 UCMOP

在第三类 UCMOP（UCMOPIII-X）中，目标空间被分解成两部分：其中一部分与一个约束条件相关联，而另一部分则涉及多个约束条件。在同一区域内存在多个约束条件可能会使搜索复杂化，并使得尝试通过该不可行区域的 CMOEAs 的搜索过程变得更加困难。

如图2-5所示，一个区域内的一些约束重叠会导致该区域中更高的约束违反。同时，随着算法深入到相应的不可行带，约束违反程度会显得更大，这可能会给一些基于保留不可行解的 CMOEAs 引入差异。这些约束违反程度较大的不可行解更接近于不利的 PF，但这些可行解可以被距离较远的可行解所消除，除非约束处理技术可以保留它们，直到算法找到不利的 PF。基于分解的 CMOEAs 具有更强的维持种群多样性的能力。因此，在约束违反越大的不可行带中，不可行解的信息更有可能生存到下一代。第三类约束条件如下：

$$
g_1(\mathbf{x})=\begin{cases}\left(2-\sum_{i=1}^{m}\beta_i(.)\right)\left(\sum_{i=1}^{m}\beta_i(.)-0.2\right)\le 0 & \text{if } d>0.3\\ 10^9 & \text{otherwise}\end{cases}
$$

$$
g_2(\mathbf{x})=\begin{cases}20\left(1.5-\sum_{i=1}^{m}\beta_i(.)\right)\left(\sum_{i=1}^{m}\beta_i(.)-0.2\right)\le 0 & \text{if } d>0.3\\ 10^9 & \text{if } |d|\le 0.3\\ 0 & \text{otherwise}\end{cases}
$$

$$
g_3(\mathbf{x})=\begin{cases}20\left(1-\sum_{i=1}^{m}\beta_i(.)\right)\left(\sum_{i=1}^{m}\beta_i(.)-0.2\right)\le 0 & \text{if } d>0.3\\ 10^9 & \text{if } |d|\le 0.3\\ 0 & \text{otherwise}\end{cases}
$$

$$
g_4(\mathbf{x})=\begin{cases}20\left(0.5-\sum_{i=1}^{m}\beta_i(.)\right)\left(\sum_{i=1}^{m}\beta_i(.)-0.2\right)\le 0 & \text{if } d>0.3\\ 10^9 & \text{if } |d|\le 0.\\ 0 & \text{otherwise}\end{cases} \tag{2-7}
$$

$$
d=\frac{\sum_{i=1}^{m-1}f_i(\mathbf{x})-f_m(\mathbf{x})}{\sqrt{m(m-1)}},\quad \beta_i(.)=\beta_i\left(\mathbf{x}^{II}-S\left(\mathbf{x}^{I}\right)\right).
$$

2.3 不同约束困难的 CMOPs 测试函数

DAS-CMOPs 是一类难度可控的多目标测试问题集[62]，在此类问题中，目标和约束的数量是可以调节的，并且不同类型的约束可以自由组合，难度也可以分别调整。因此能够构成同时具有三种难度类型（可行性、收敛性、多样性）的约束多目标测试问题，从而全面评估 CMOEAs 在不同难度下的性能。

2.3.1 多样性困难

类型 I：多样性困难的约束

由于 MOPs 的 PF 主要受形状函数的影响，因此，控制形状函数的相关变量

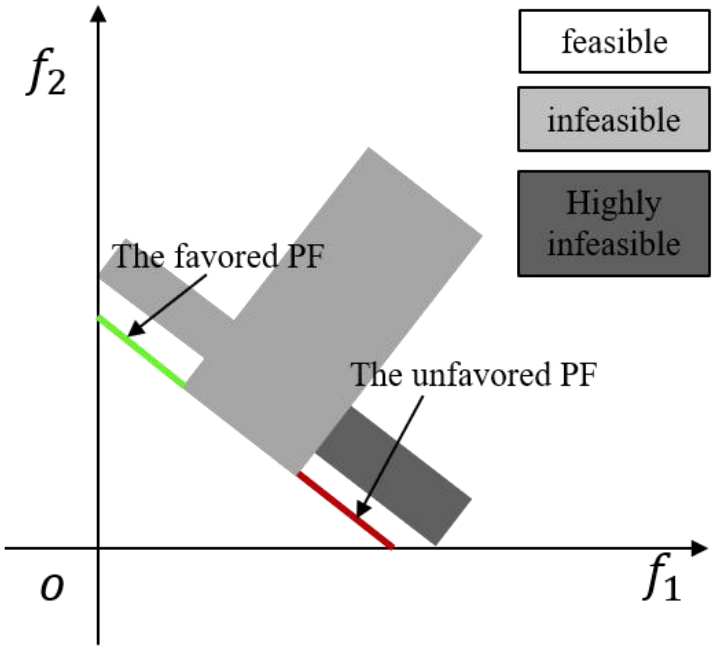


图 2-5 III 型约束图解

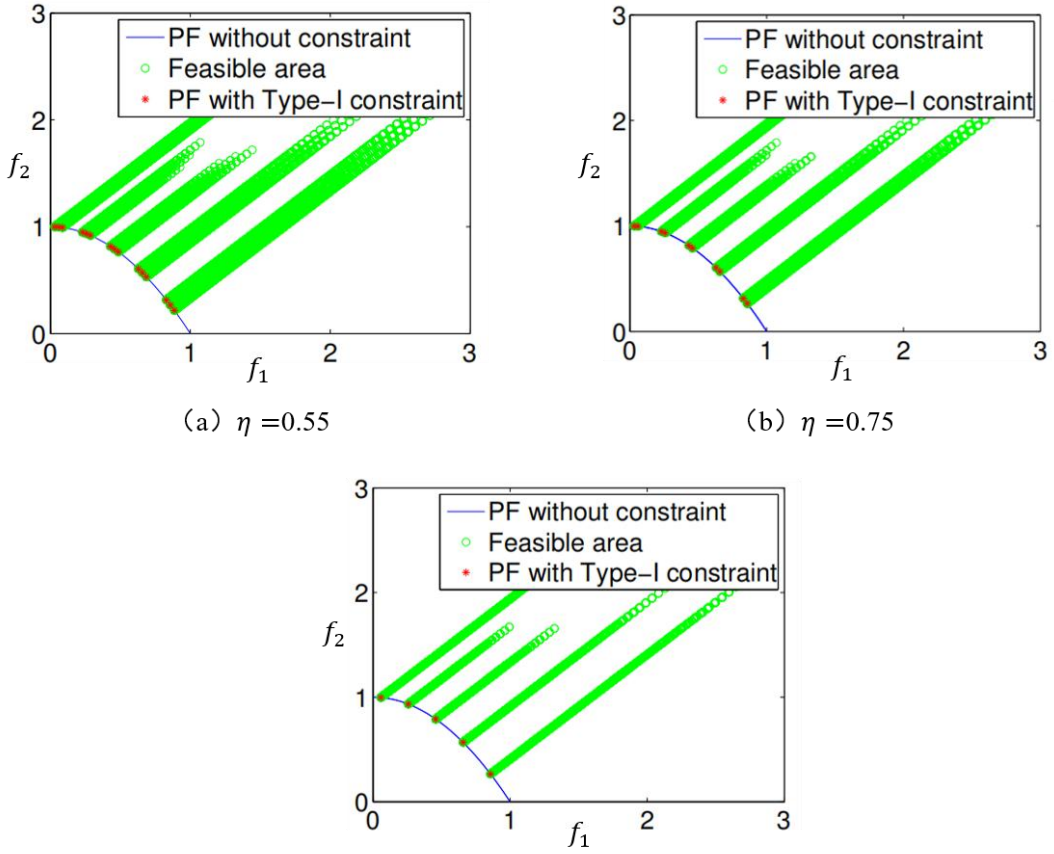
是设计多样性困难约束最直接的方法。此种类型约束的引入使得 PF 被分割成相互独立的非重叠区域，从而很难获得一组多样化的可行解（即多样性困难）。此时算法要想收敛到整个 PF 是非常困难的。具有多样性困难的 CMOPs 示例如下：

$$\begin{cases} \text{minimize} & f_1(x) = x_1 + g(x) \\ \text{minimize} & f_2(x) = 1 - x_1^2 + g(x) \\ & g(x) = \sum_{i=2}^{n} \left(x_i - \sin\left(0.5\pi x_1\right)\right)^2 \\ \text{subject to} & c(x) = \sin\left(a\pi x_1\right) - b \geq 0 \\ & x_i \in [0,1] \end{cases} \tag{2-8}$$

其中，a代表 PF 断开片段的数量，$a > 0$；b表示每个 PF 片段的宽度，$b \in [-1,1]$。η是多样性困难的难度系数，由b控制，b的值越大，难度等级η越高。图 2-6 画出了此种类型约束的 PF 变化。如图所示，测试问题 PF 上不连续的片段大小随着难度系数$\eta \in [0,1]$的变化而变化，导致问题的难度等级也随之变化。

2.3.2 收敛性困难

类型 II：收敛性困难的约束



（a）η =0.55　　（b）η =0.75

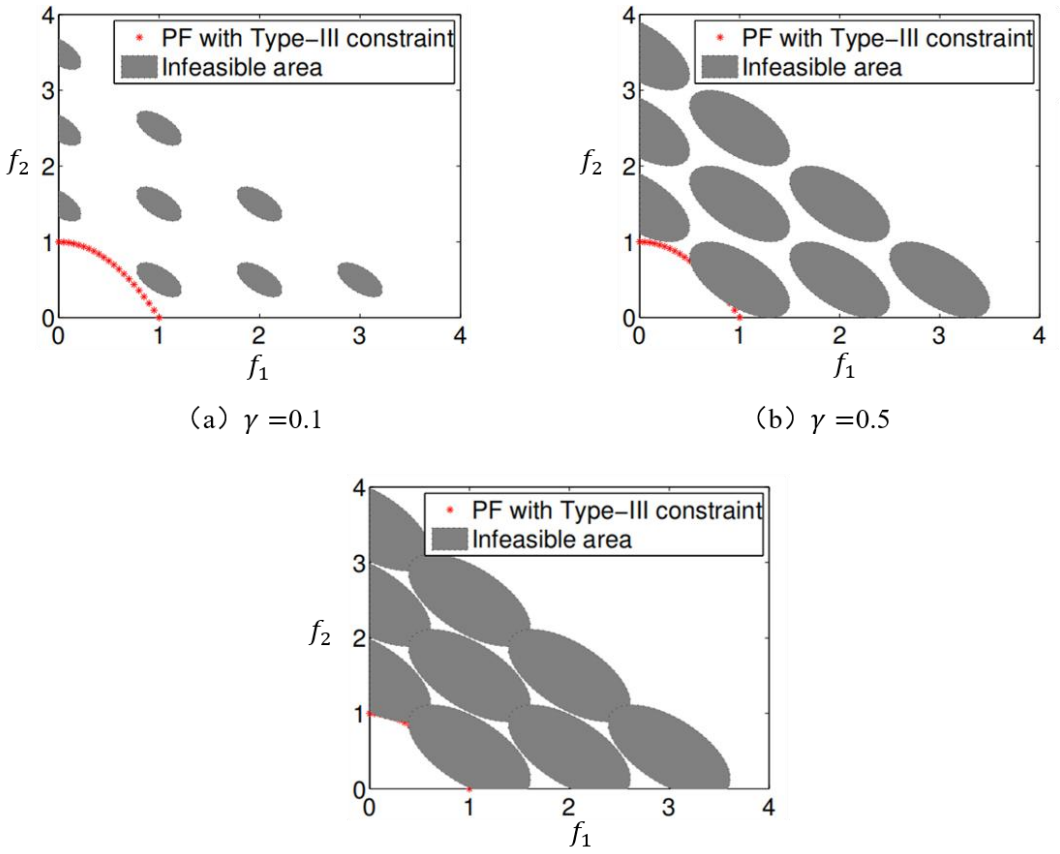
（c）η =1.0

图 2-6 多样性困难约束对 PF 的影响

II 型约束主要是对目标函数的可达范围进行了一定的限制，这类约束所形成的不可行区域为 CMOEAs 寻找最优解带来了诸多困难。由此， 此类约束导致了一个有收敛困难的多目标优化问题。具有收敛性困难的 CMOPs 的示例如下：

$$
\begin{cases}
\min & f_1(x) = x_1 + g(x) \\
\min & f_2(x) = 1 - x_1^2 + g(x) \\
\text{where} & g(x) = \sum_{i=2}^{n} \left(x_i - \sin\left(0.5\pi x_1\right) \right)^2 \\
\text{s.t.} & c_k(x) = \left(\left(f_1 - p_k \right)\cos\theta_k - \left(f_2 - q_k \right)\sin\theta_k \right)^2 / a_k^2 \\
& + \left(\left(f_1 - p_k \right)\sin\theta_k + \left(f_2 - q_k \right)\cos\theta_k \right)^2 / b_k^2 \geq r \\
& p_k = [0,1,0,1,2,0,1,2,3] \\
& q_k = [1.5,0.5,2.5,1.5,0.5,3.5,2.5,1.5,0.5] \\
& a_k^2 = 0.4, b_k^2 = 1.6, \theta_k = -0.25\pi \\
& c = 20, n = 30, x_i \in [0,1], k = 1,\ldots,9
\end{cases}
\tag{2-9}
$$

其中参数r控制着可行域的大小；在收敛性困难的约束中定义难度系数为$\gamma \in [0,1]$，γ由r控制。图 2-7 画出了此类型约束的 PF 变化。r越大，γ越大，不可行区域也会越大。



（a）γ =0.1　　（b）γ =0.5

（c）γ =0.75

图 2-7 收敛性困难约束对 PF 的影响

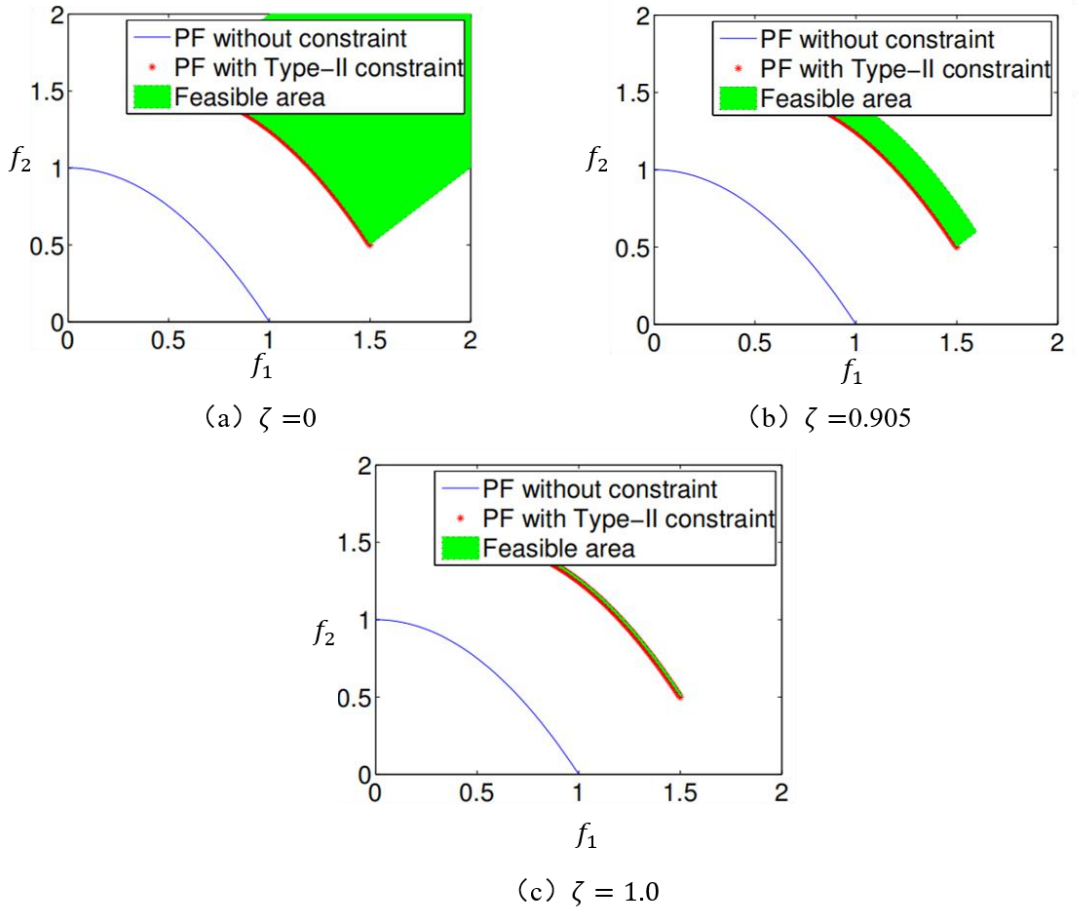
2.3.3 可行性困难

类型 III：可行性困难的约束

此种类型的约束主要用来控制当前种群中含有可行解的比例。具有可行性困难的 CMOPs 的示例如下：

$$\begin{cases} \text{minimize} & f_1(x) = x_1 + g(x) \\ \text{minimize} & f_2(x) = 1 - x_1^2 + g(x) \\ & g(x) = \sum_{i=2}^{n} \left(x_i - \sin\left(0.5\pi x_1\right) \right)^2 \\ \text{subject to} & c_1(x) = g(x) - a \geq 0; c_2(x) = b - g(x) \geq 0 \\ & n = 30, x_i \in [0,1] \end{cases} \tag{2-10}$$

具有可行性困难的约束的 PF 变化如图 2-8 所示。ζ越大可行区域越小，$\zeta \in [0,1]$。



（a）ζ =0　　（b）ζ =0.905

（c）ζ = 1.0

图 2-8 可行性困难约束对 PF 的影响

测试问题的设计是多目标优化问题的研究中不可或缺的一部分，设计一个多目标测试问题集的关键在于其特性是否符合现实需求。可扩展的测试问题可以帮助测试不同机制的算法并且验证其性能，但是不同测试问题的属性往往有所不同，使用多个测试问题可以更全面地了解算法的行为和性能。由于测试问题之间互相补充，因此使用不同测试问题进行充分评估和比较是很重要的。这样就可以利用 MOEAs 在同一个问题上进行任意特征的测试，最终获得一组最佳解决方案，以满足实际需求。

第 3 章 不平衡约束多目标测试问题设计及其算法研究

相较于单目标或无约束多目标优化问题，CMOPs 需要综合考虑多个相互冲突的目标和各种约束限制。约束条件的限制使搜索空间的变得更加复杂，例如，当不可行区域占据了极大部分空间，在这种情况下，算法应该考虑如何快速高效地探索到仅存的小部分可行区域；在已有的多目标进化算法中，大多是对收敛性和多样性的研究，通常忽略了约束条件。现有的 MOEAs 在优化 MOPs 时，主要执行两个任务：一是将其种群推向 PF，以确保收敛；二是保持种群的多样性，从而保证能找到各种各样的解。尽管这两个任务的重要性相同，但大多数最先进的算法都遵循“收敛性第一，多样性第二”的原则。这意味着在选择下一代种群时，收敛性优于多样性的解将更受青睐，从某种意义上来说并不能充分反映算法的有效性。总的来说，设计 CMOEA 的关键是要同时考虑收敛性、多样性、可行性，并在三者之间进行有效的权衡。

然而现实中存在的许多 CMOPs 都具有不平衡特征，即约束不平衡，或者目标不平衡。比如机械臂的设计、无线传感器网络等领域的实际工程和优化问题。解决这类问题是非常具有挑战性的，对于不平衡 CMOPs 的研究也仅在近几年才得到关注。基于此，本文针对具有不平衡特征的约束多目标优化问题提出一种基于分解的混合 CMOEA，命名为εM2M。该算法结合基于分解的策略和一种改进的约束处理机制，首先使用基于分解的方法将种群划分为多个子种群，以此保证种群的多样性；其次，运用改进的ε约束处理机制协助种群跨越不可行区域，搜索到真实的帕累托前沿，从而提高算法的收敛性；两种机制结合用来解决同时具有约束和目标不平衡 CMOPs，大大提高了算法的性能。此外，为了证明算法的有效性，还设计了一组具有不平衡 CMOPs 特性的测试问题集(IM-CMOPs)。与其他 8 种算法进行对比，结果显示，所提算法在解决不平衡 CMOPs 上具有显著的优越性。

本章首先介绍测试问题集的设计背景和细节；然后详细介绍εM2M 的算法设计和框架；最后描述了实验设置和各种算法在不同测试问题集上的结果分析。

3.1 IM-CMOPs 测试问题集设计

3.1.1 研究背景

测试问题是一个多目标进化算法设计和评估的重要环节。一个理想的约束多目标测试问题要重视“约束”和“目标”这两个关键点，意味着不仅要考虑目标空间的复杂性，还要兼顾 PF 附近的不可行障碍。然而，大多数现有的设计要么关注目标空间的复杂性，要么着重于 PF 附近的不可行区域，很少将二者同时考虑。例如，经典的 DTLZ[66]、MaF[67]等问题仅考虑了约束边界，CTP[68]和 CF[69]等问题的目标数量和目标空间的维度都是不可改变的。因此，综合考虑目标和约束对 CMOPs 的设计至关重要。

DAS-CMOPs 是一类难度可控的多目标测试问题集[62]，在此类问题中，目标和约束的数量是可以调节的，并且不同类型的约束和其难度都可以自由组合和分别调整。因此能够构成同时具有多种难度特征（可行性、收敛性、多样性）的约束多目标测试问题，从而全面评估 CMOEAs 的性能。

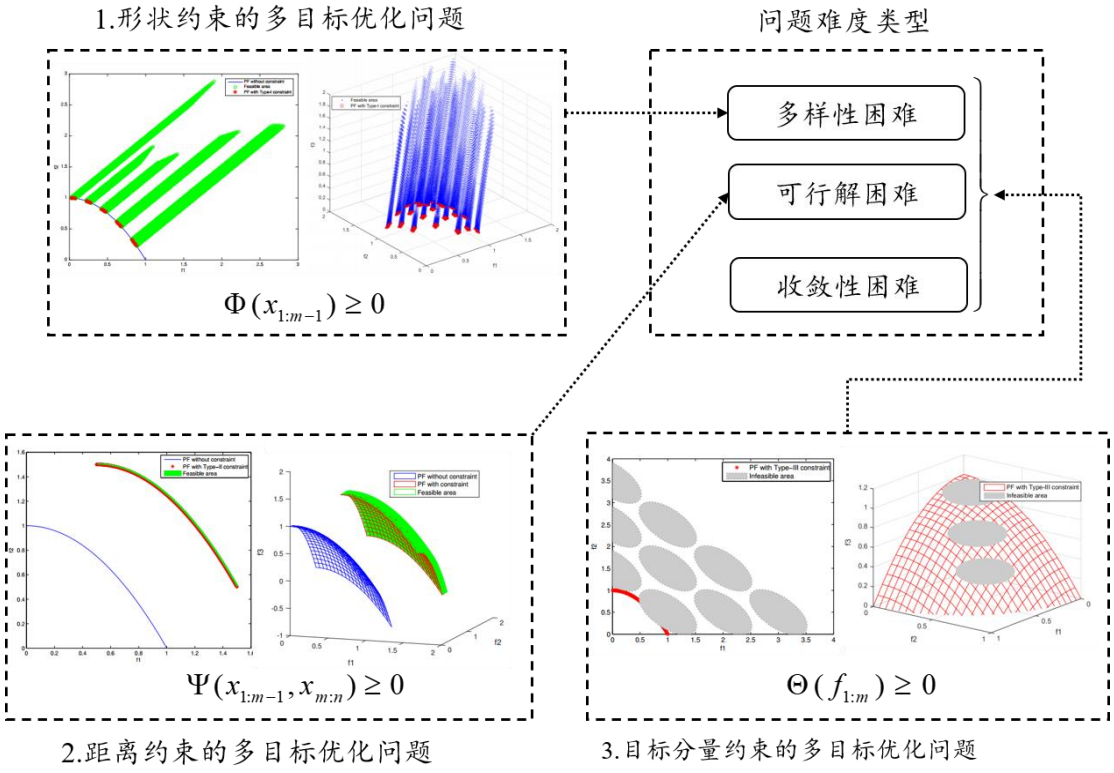


图 3-1 约束多目标测试问题设计背景

现有研究中不平衡的约束多目标测试问题集的设计仍然较少，因此，为了让

所提算法能够得到有效的评估，设计了一组目标不平衡，且约束存在着多样性和收敛性困难的问题函数，将εM2M 和 8 种现有的经典算法在此类问题上进行测试。

第二章中已经介绍过，假设 PF 的某个有利子集对应的子问题的复杂度明显低于不利子集，并且有利子集中的解支配了更大的可行空间。这种问题我们将它称为不平衡问题[65]。在进化过程中，种群中的个体往往倾向于找到某段容易满足约束帕累托前沿，即算法更容易找到有利的 PF，并且与其相对应的 PS 占据了大部分可行空间，从而导致种群很难收敛到整个真实的 PF。

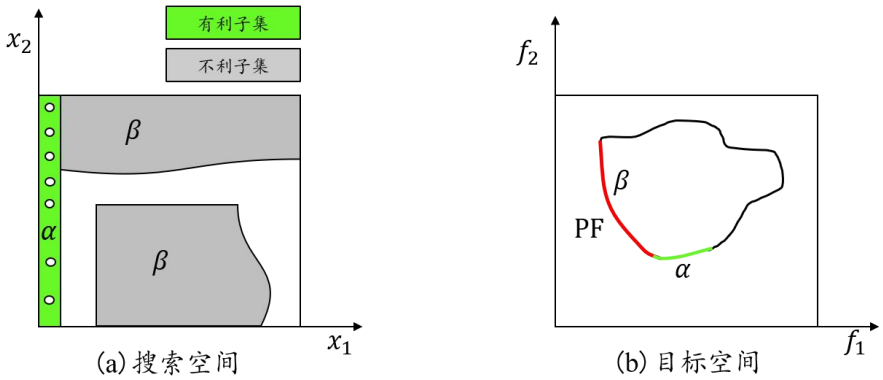


图 3-2 不平衡搜索空间示例

图 3-2（a）简单描绘了一个不平衡多目标优化问题的搜索空间，3-2（b）对应其目标空间。绿色区域α映射到（b）中对应的是绿色线段，灰色区域β映射到（b）中对应着红色线段。其中，绿色阴影α中的每个解都会自动成为帕累托最优解，不仅支配着β中的任意一个解，并且位于 PF 的有利子集中。同时区域α也支配着区域β。因此，大部分 MOEAs 都更容易搜索到区域α中，也就意味着更多的解会收敛到 PF 的有利子集α中。这就导致在搜索过程中，算法难以有效地寻找到所有的 PF，从而使其缺乏多样性。

3.1.2 问题设计

本节主要设计了一组具有不平衡特征（包括目标不平衡，约束具有收敛性、多样性困难等）的约束多目标测试问题集，命名为 IM-CMOPs。其目标函数是在 ZDT 和 DTLZ 测试问题集的基础上加以修改得到的[64]，因此符合目标不平衡的特点。修改后的 g(x)函数与原始版本中的不同，主要表现在其搜索空间设置为 $[0,1]^n$。具有多样性困难和收敛性困难的约束条件的设计则来源于 DAS-CMOPs[62]。η、ξ和γ分别代表了多样性、可行性、和收敛性等三种难度等级，取值可以根据实际需求手动调整，但范围须保持在 0 到 1 之间。鉴于 IM-CMOPs 的设计着重于多样性和收敛性，ξ的值暂且设置为 0，η和γ设置为 0.5。

综上，本节构造了五个约束测试函数，各自的难度等级相同。五个测试函数的约束都是不平衡的，即具有多样和收敛性困难。下面以 IM-CMOP4 和 IM-CMOP5 为例，介绍 IM-CMOPs 系列测试函数的细节。IM-CMOP4 的目标函数表达式如下：

$$\text{IM-CMOP4}\begin{cases}\min & f_1(\mathbf{x})=(1+g(\mathbf{x}))x_1\\ \min & f_2(\mathbf{x})=(1+g(\mathbf{x}))\left(1-x_1^{0.5}\cos^2\left(2\pi x_1\right)\right)\\ \text{where} & g(\mathbf{x})=10\sin\left(\pi x_1\right)\sum_{i=2}^{n}\left(\frac{|t_i|}{1+e^{5|t_i|}}\right)\\ & t_i=x_i-\sin\left(0.5\pi x_1\right)\\ & n=10,\mathbf{x}\in[0,1]^n\end{cases}\tag{3-1}$$

对应的 PS 是$\{(x_1,\cdots,x_n)|0<x_1<1,x_j=\sin(0.5\pi x_1);\ j=2,\ldots,n;\ \text{or}\ x_1=0,1.\}$，另外，其 PF 是不连续的。

约束函数表达式为:

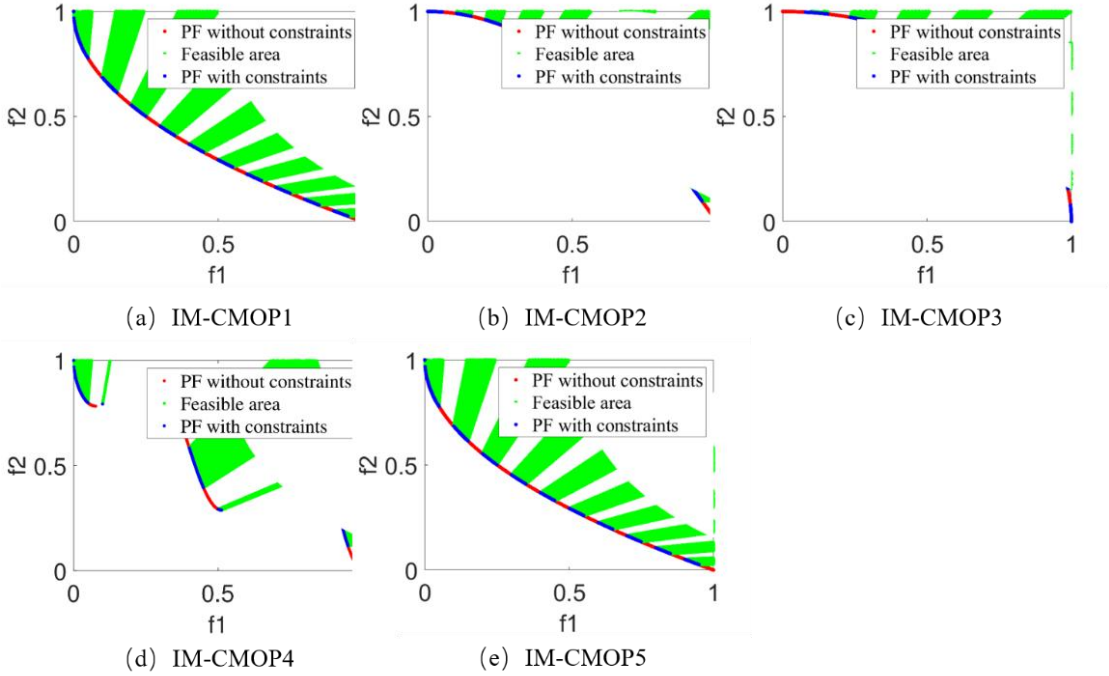
$$\begin{cases}c_1(\mathbf{x})=\sin\left(a\pi x_1\right)-b\geq 0\\ c_k(\mathbf{x})=\left(\left(f_1-p_k\right)\cos\theta_k-\left(f_2-q_k\right)\sin\theta_k\right)^2\Big)/a_k^2\\ +\left(\left(f_1-p_k\right)\sin\theta_k-\left(f_2-q_k\right)\cos\theta_k\right)^2\Big)/b_k^2\geq r\\ a=20,b=0\\ p_k=[0,1,0,1,2,0,1,2,3]\\ q_k=[1.5,0.5,2.5,1.5,0.5,3.5,2.5,1.5,0.5]\\ a_k^2=0.4,b_k^2=1.6,\theta_k=-0.25\pi\\ c=20,n=30\\ x_i\in[0,1],k=1,\ldots,9\end{cases}\tag{3-2}$$

其中 $c_1(\mathbf{x})$和$c_k(\mathbf{x})$分别表示多样性约束和收敛性约束。a 代表 PF 断开的段数，这里 a 的值设为 10，即 PF 由不连续的 10 段组成。本文中我们将多样性难度等级η定为 0. 5，因此由$b=2\eta-1$ 可得$b=0$。γ表示收敛性约束难度等级，此时$\gamma=0.5$。η和γ的变化会造成多样性和收敛性难度的变化，从而导致与之对应的 PF 也会受到不同程度的影响，比如断开的段数越来越多、不可行区域越来越大等。

IM-CMOP5 的目标函数表达式如下：

$$\text{IM-CMOP5}\begin{cases}\min & f_1(\mathbf{x})=(1+g(\mathbf{x}))x_1\\ \min & f_2(\mathbf{x})=(1+g(\mathbf{x}))\left(1-\sqrt{x_1}\right)\\ \text{where} & g(\mathbf{x})=2\left|\cos\left(\pi x_1\right)\right|\sum_{i=2}^{n}\left(-0.9t_i^2+\left|t_i^{0.6}\right|\right)\\ & t_i=x_i-\sin\left(0.5\pi x_1\right)\\ & n=10,\mathbf{x}\in[0,1]^n\end{cases} \tag{3-3}$$

对应的 PS 为$\{(x_1,\cdots,x_n)|0\le x_1\le 1, x_j=\sin(0.5\pi x_1);\ j=2,...,n;\ \text{or}\ x_1=0.5\}$，PF 是$f_2=1-\sqrt{f_1}, 0\le f_1\le 1$。约束函数同 IM-CMOP4，实际上 5 个测试函数共用一个约束，并且难度等级都相同。5 个测试问题函数在目标空间中的 PF 分布情况展示在图 3-3 中。



(a) IM-CMOP1　(b) IM-CMOP2　(c) IM-CMOP3

(d) IM-CMOP4　(e) IM-CMOP5

图 3-3 测试问题的 PF 分布图

为了证明εM2M 在处理不平衡 CMOPs 上的优越性，以 IM-CMOP2 为例进行一个具体过程的分析。首先用 CM2M 算法（未使用ε约束处理）在 IM-CMOP2 上进行测试，图 3-4（a）是种群的初始状态分布图，此时个体几乎都分布在不可行区域，与真实的 PF 距离甚远。理论上来讲，没有约束机制的处理，这些个体几乎不可能跨越这些障碍以到达可行区域；随着实验的进行，结果如图 3-4(b)所示。实际情况与预测的一样，绝大部分个体仍然徘徊在不可行区域之中，很难收敛到真实的帕累托前沿。

接着采用 MOEA/D-IEpsilon（未使用 M2M 策略）在 IM-CMOP2 上进行测试，图 3-5 展示了其搜索过程。其中图 3-5(a)画出了 MOEA/D-IEpsilon 算法种

群的初始状态，与 CM2M 相似，大多数个体位于不可行区域且与真实的 PF 距离较远。没有 M2M 策略的加持，个体在搜索空间中可能会陷入约束容易满足的地方，进而导致其在目标空间只能找到小部分的帕累托前沿，因此造成了目标的不平衡，并且大大降低了算法的多样性。诚然，如图 3-5(b)所示，小部分个体收敛到小段的 PF 上，大部分个体仍然留在不可行区域当中。

最后，将 M2M 和 IEpsilon 互相融合，图 3-6 展示了εM2M 的搜索过程。种群的初始状态与上述相同，仍然是大多数个体位于不可行区域之中，与真实的 PF 距离甚远。但在使用了εM2M 方法之后，结果如图 3-6(b)所示，绝大多数个体跨越了不可行区域，并且有更多的解收敛到帕累托前沿。基于 M2M 的分解方法，使用 5 个方向向量（$v^1, ..., v^j$）将目标空间划分为五个子区域 $P_1, ..., P_J$，从而保证了个体在每个子区域中的均匀分布，即维持了种群的多样性；基于 IEpsilon 的机制帮助种群穿过不可行区域，收敛到整个 PF，从而保证了收敛性。综上，融合之后的算法性能明显优于分开使用的效果。

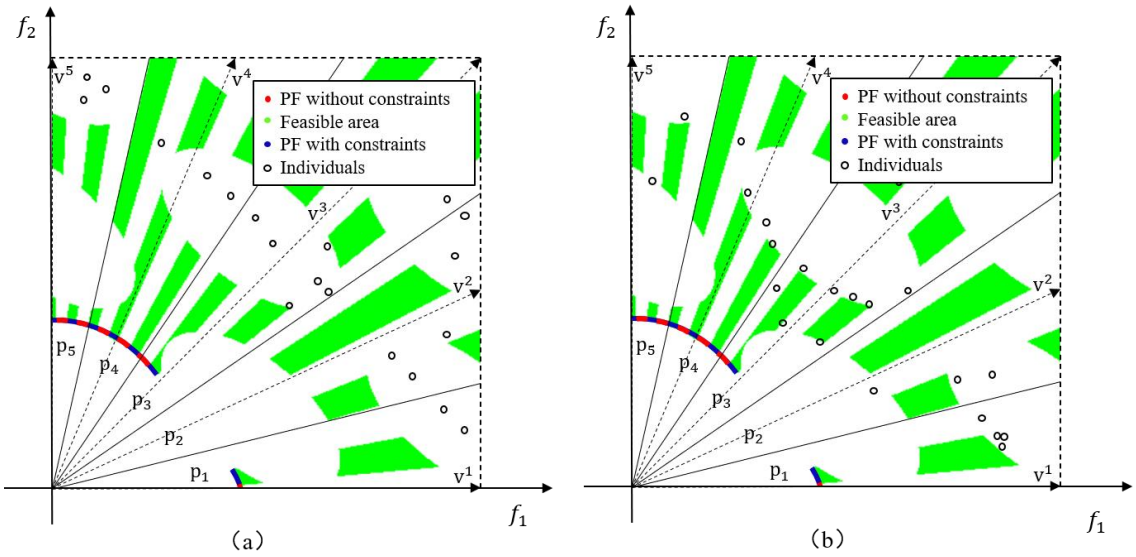


图 3-4 CM2M 方法搜索过程

3.2 εM2M 算法框架

3.2.1 分解机制

M2M 是 Liu 等人[64]在近些年提出的一种基于分解的进化算法。与大多数现有的进化多目标算法相同，M2M 分解策略遵循“多样性第一、收敛性第二”的原则。换句话说，M2M 策略具有很强的能力来保持种群的多样性，特别是在解决不平衡的约束多目标优化问题时。需要注意的是，与 MOEA/D[70]不同，M2M

是将一个 CMOP 分解为多个简单的多目标优化子问题，而不是单目标优化问题。

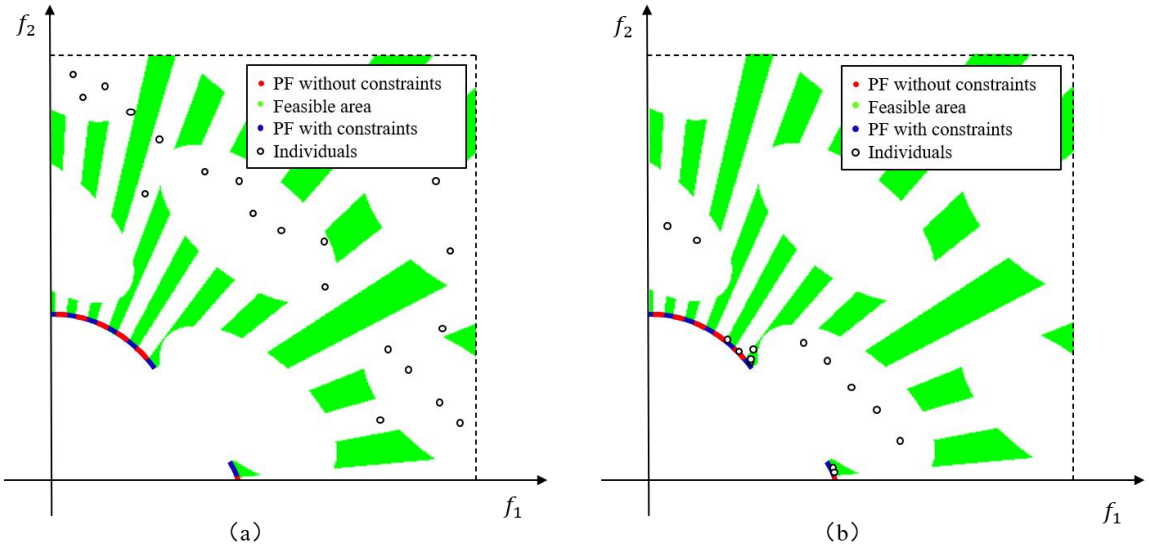


图 3-5 MOEA/D-IEpsilon 方法搜索过程

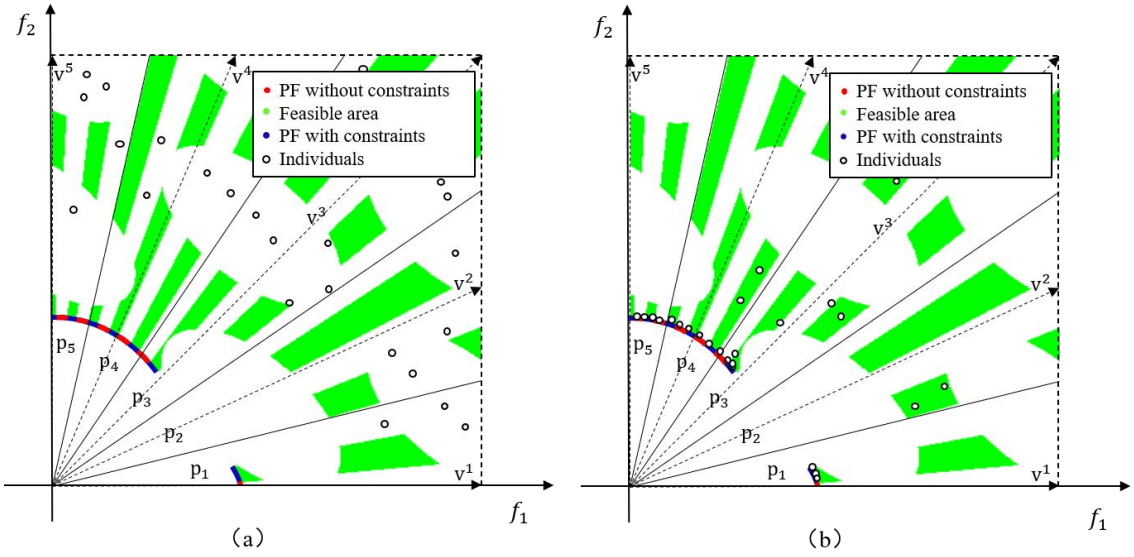


图 3-6 εM2M 方法搜索过程

在初始化阶段，εM2M 将目标空间分解为多个子区域。每个子区域对应一个子问题，这些子问题被协调解决，以提高种群的多样性。为了简单起见，我们设置K个方向向量（$v^1, v^2, ..., v^K$）对目标空间$\mathbf{R}_+^m$进行均匀分解，然后生成K个非相邻子区域（$\Omega_1, \Omega_2, ..., \Omega_K$）。一个子区域$\Omega_k$的定义如下：

$$\Omega_k = \left\{ u \in \mathbf{R}_+^m \middle| \left\langle u, v^j \right\rangle \le \left\langle u, v^i \right\rangle \text{ for any } i = 1,2,\ldots,K \right\} \tag{3-4}$$

其中u是目标向量，v^i是第i个方向向量，$\langle u, v^i \rangle$是u和v^i之间的锐角。为了在每个子区域找到最优解，将整个种群划分为K子种群。子问题 k 的定义可以表述如

下：

$$\begin{cases} \text{minimize} & \mathbf{F}(x)=\left(f_1(x),\ldots,f_m(x)\right) \\ \text{subject to} & x\in\prod_{i=1}^{n}[a_i,b_i] \\ & F(x)\in\Omega_k \end{cases} \tag{3-5}$$

对于每个子问题k，子种群P_k需要独立优化。每个子种群在每一代需要保持S个个体。子种群P_k的更新方法如下：

1）如果P_k中的个体小于S，则从整个种群中随机选择 $S-|P_k|$ 个体并添加到P_k中。

2）如果P_k有超过S个个体，则采用非支配排序和约束处理方法选择S个个体。

因此，利用等式将一个受约束的多目标优化问题分解为K个简单的子问题。这个过程在算法 3-1 中被描述。算法 3-1 将每一代的种群被分解为 K 个子种群，以此保证种群的多样性。

算法 3-1 分解策略

1： **function** Result = $DecompositionPop(Q,K)$

2： // Q: all individual solutions.

3： // K: K sub-populations.

4： // F-value: objective function values of each individual solution.

5： // P_k: kth sub-population.

6： **for** $i=1,2,...,K$ **do**

7： Initialize P_k as the solutions in Q whose F-values are in Ω_k according to Eq. ;

8： **end for**

9： return $P_1,...,P_K$;

10： end function

3.2.2 约束机制

为了在满足约束条件的情况下找到好的目标，范等人[71]提出了MOEA/D-Epsilon。在 MOEA/D-Epsilon 中，ε随进化代数的增加而动态变化。值得注意的是，在进化过程中ε是下降的。尽管如此，MOEA/D-Epsilon 还是难以

解决具有较大的不可行区域的 CMOPs。为了克服这个缺点，又提出了一种改进的约束处理方法，即 MOEA/D-IEpsilon[71]。它利用当前种群中的可行解的比例来动态调整 MOEA 水平，具有探索可行和不可行区域的能力。对约束处理方法的具体描述如下：

$$\varepsilon(g)=\begin{cases}\text{rule 1}:\phi\left(\mathbf{x}^{\theta}\right),\ \text{if } g=0\\ \text{rule 2}:(1-\tau)\varepsilon(g-1),\ \text{if } r_g<\alpha \text{ and } g<T_c\\ \text{rule 3}:(1+\tau)\phi_{\max},\ \text{if } r_g\geq\alpha \text{ and } g<T_c\\ \text{rule 4}:0,\ \text{if } g\geq T_c\end{cases} \tag{3-6}$$

其中g为当前代数；$\phi\left(\mathbf{x}^{\theta}\right)$表示种群初始时，前$\theta$个个体的总体约束违反；$r_g$为第$g$代中可行个体的比例。$\tau$是 0 到 1 之间的正数，有两个函数：(1)可以控制减少约束松弛的速度；(2)还可以控制比例因子乘以总体约束的最大值违反。α被用于调整算法在可行和不可行区域之间的搜索偏好。T_c表示控制生成，$\phi_{\max}$是已发现的最大总体约束违反。

为了处理目标和约束不平衡的 CMOP，εM2M 方法采用了改进的ε约束机制。通过动态调整ε的水平，在εM2M 中保持了一组不可行的解决方案，这有助于指导搜索到有希望的区域。该方法有四个规则，如等式中定义.

算法 3-2 ε值的设定

```
function ε(g) = SETEPSILON (τ, r_g, α, φ_max, T_c, g)
    if g ≥ T_c then
        ε(k) = 0
    else
        if r_g < α then
            ε(g) = (1 − τ)ε(g − 1)
        else
            ε(g) = (1 + τ)φ_max
        end if
    end if
end function
```

在规则 1 中，$\varepsilon(0)$被设置为$\phi\left(\mathbf{x}^{\theta}\right)$。这就确保了$\varepsilon(0)$在初始化阶段不等于零。利用生成计数器$g$和可行个体的比值$r_g$这两个参数来控制$\varepsilon(g)$的值。如果$g<T_c$

和$r_g < \alpha$，εM2M 应用规则 2 来加强对可行区域的搜索。否则，εM2M 将应用规则 3 来加强对不可行区域的探索。值得注意的是，规则 2 使$\varepsilon(g)$具有显著的下降率，这可以有效地生成可行的解决方案。

此外，εM2M 应用规则 3 加强了对不可行区域的探索，为促进种群的趋同和多样性提供了最佳方向。采用规则 2 和规则 3 动态调整$\varepsilon(g)$的值，具有同时探索可行区域和不可行区域的能力。如果$g \geq T_c$，M2M 应用规则 4 对可行区域施加最高的选择压力。3.2.3 εM2M

基于多目标分解方法和改进的约束处理技术，本节提出一种基于分解的混合 CMOEA，命名为εM2M。该算法旨在解决具有不平衡约束和目标的约束多目标优化问题。εM2M 的基本思想，首先通过分解方法将问题拆成若干个更小的子问题，然后利用改进的约束处理机制优化每个子问题。同时，它还通过采用基于多样性和收敛性的新评价函数来平衡多样性和收敛性难度。此外，该算法还引入了一种自适应机制，可以根据不同问题的复杂性，自动调整算法参数，从而获得最优解。

3.2.4 算法框架

算法 3-3 εM2M 框架

输入：

K：子问题的数量；

$v^1, \ldots, v^k$：K个单位方向向量；

S：每个子种群的大小；

Q：一组个体解和对应的目标函数值；

$g_{\max}$：最大代数；

T_c：$\varepsilon(k)$的控制代数；

输出：一组非支配的可行解。

1： 初始化

2： 根据算法 3-1 将种群分解为K个亚种群（$P_1 \ldots P_k$），

3： 根据$DecompositionPop(Q, K)$，每个亚种群包含S个个体；

4： 根据等式初始化$\varepsilon(0)$，r_g，$\phi_{\max}$，τ，α，和T_c；

5： **while** $gen \leq g_{\max}$ **do**

6： **for** $i = 1,2, \ldots K$ **do**

7： **for** each $x \in P_k$ **do**

8： 从 P_k 中随机选择个体 y;

9： 个体 x 和个体 y 统一应用遗传操作生成新的解 z;

10： $R: = R \cup \{z\}$;

11： end for

12： $Q: = R \cup \left(\cup_{k=1}^{K} P_k\right)$

13： 根据等式和使用Q来设置 $P_1, ..., P_K$;

14： **end for**

15： ε(g) = SetEpsilon (τ， r_g， α， T_c， g， $\phi_{\max}$)；

16： **for** $i = 1,2, ...K$ **do**

17： **if** $|P_k| \leq S$ **then**

18： 随机选择 $|S - P_k|$ 加入 P_k 中;

19： **else**

20： 根据改进的 Epsilon 机制，从P_k中选择S个解。

21： **end if**

22： **end for**

23： gen = gen + 1；

24： end while

25：输出：一组非支配的可行解

本节详细描述εM2M 算法的框架和步骤，其整体框架流程如算法 3-3 所示。代码的第 1 行到第 4 行是算法的初始化阶段。在第 2 行中，根据等式将种群分解为K个子种群， $v^1, ..., v^k$代表k个方向向量，在目标空间中均匀分布。第 3 行将每个子种群的大小设置为S。第 4 行初始化了算法中包含的各个参数，包括$\varepsilon(0)$, r_g， $\phi_{\max}$ ， τ， α， 和 T_c等。

第 5 到 24 行是算法的核心部分。算法在当前代数小于$g_{\max}$的情况下重复执行，直到运行到最大进化次数方可跳出循环。第 6 行到第 14 行显示了产生新的子代个体的过程。首先从P_k中随机选择个体y，根据操作算子生成新的子代z，然后计算对应的目标函数值，然后根据等式和使用Q来设置 $P_1, ..., P_K$。第 15 行根据算法 3-2 的框架设置$\varepsilon(g)$的值。第 16 到 22 行执行了K个子种群的更新过程。更新过程分为两部分： (1) 如果子种群P_k的大小小于或等于子种群S的大小，则

从所有单独的解 Q 中随机选择$S-P_k$个解并添加到P_k中；(2)如果P_k的数量大于子种群S，则根据改进的ε机制从P_k中选择S个解。具体来说，如果选择了一个解决方案，那么就必须满足这两个基本规则中的一个。对于任意两个解x和y，如果它们的总体约束违反小于或等于$\varepsilon(g)$，并且x的总体约束违反小于y的约束违反，则选择x。如果x和y有相同的总体约束违反，并且x在目标方面优于y，则选择x。第 23 行更新生成计数器，并在第 25 行输出最终的非支配可行解。

3.3 实验设置

3.3.1 参数设置

为了验证所提算法的有效性，本文使用εM2M 与其他 8 个经典且具有代表性的 CMOEAs（包括 CCMO[39]，CMOEA-MS[45]，cDPEA[61]，ToP[46]，MOEA/D-CDP[72]，MOEA/D-DW[2]，CM2M2[18]，和 CM2M[36]）在两套具有不平衡特性的 CMOPs 测试函数（IM-CMOPs 和 UCMOPs）上进行实验。所有算法在每个测试问题上独立运行 30 次。其中有些 CMOEAs 包含了相同的处理机制，所以这些算法在使用相同处理机制时的参数设置将保持一致。各个算法具体参数设置如下:

1） 种群大小：$N=200$ 。

2） 变异、交叉：突变概率$P_m=1/n$，交叉概率$CR=1.0$。

3） 终止条件：当评价次数达到 300000 次时停止。

4） εM2M、CM2M 、CM2M2： 在这些算法中，一个目标空间被分解为K（$K=10$）个子区域。其中，CM2M2 算法中的不可行权重$N_1=60$，可行权重 $N_2=140$；εM2M 算法中的T_c设为 800，α为 0.95N， τ设置为 0.1，θ设为 0.05 N。

5） MOEA/D-CDP:邻域大小T定为 30，$n_r=2$。

6） 其余算法的参数都与其原文中保持一致。

3.3.2 对比算法

本文使用了 8 种对比算法和εM2M 一起在两类不平衡测试问题集上进行测试，包括 CCMO[39]， CMOEA-MS[45]， cDPEA[61]， ToP[46]， MOEA/D-CDP[72]，MOEA/D-DW[2]，CM2M2[18]，和 CM2M[36]。各个算法的简要描述如下：

1）CCMO：该算法的基本思想是利用一个简单问题去辅助复杂问题进行优化，其中一个种群负责求解原始的 CMOP，另一个种群求解与原始 CMOP 对应的无约束版本。这意味着两个种群分别搜索到带约束的 PF 和无约束的 PF，从而

相互作用实现协同进化。这样既可以实现简单问题和复杂问题之间的相互协作，又可以提升算法的搜索效率。

2）CMOEA-MS：该方法将进化过程分为两个阶段，并在这两个阶段使用不同的适应度评价策略来调整目标和约束的优先级，以缓解使用单个约束处理机制解决不同约束多目标问题的局限性。第一阶段为了可行性将目标和约束分配了同样的优先级，第二阶段则令目标的优先级低于约束，以便维持种群的多样性。

3）cDPEA：c-DPEA 是一个基于双种群的进化算法，它利用双种群的概念，分别考虑两个种群，以达到优化的目的。在此基础上，提出了一个新的自适应惩罚函数，以保证种群 1 中不可行解的竞争力；而在种群 2 中采用面向可行性的方法。该算法能够有效地解决优化问题，并获得良好的结果。

4）ToP：其第一阶段采用加权的方法将 CMOP 转化为一个简单的单目标优化问题，第二阶段则使用现有的约束多目标进化算法对第一步优化结果进行二次优化，此时优化的是带约束的多目标优化原问题。

5）MOEA/D-CDP：该算法将 CDP 嵌入到 MOEA/D 中，以实现对约束的有效处理。 MOEA/D 是一种经典的多目标优化算法，它采用分解策略将多目标问题分解为多个单目标子问题，并通过协作搜索来解决整个多目标问题。该算法的核心思想是将多目标问题转化为多个单目标问题，每个单目标问题都可以使用现有的单目标优化算法求解，然后通过协作搜索获得全局最优解。在 MOEA/D 算法中，用权重矢量来定义子问题，并利用权重矢量计算其欧氏距离，从而决定子问题间的邻居关系。而 CDP 则被用来决定两个个体之间的优劣。

6）MOEA/D-DW：该方法采用两种权值，即可行区域上的可行权值和不可行区域上的不可行权值，将搜索引向可行区域。其中不可行权重是为了保持那些携带有用信息的不可行个体。同时，它们也会随着进化而动态变化，趋向于具有较好目标值和较小约束违反的不可行个体。该算法能保持不可行个体的多样性，覆盖尽可能多的非支配解，有利于种群的进化。

7）CM2M2：CM2M2 在 M2M 的基础上采用了有向加权的方法来解决 CMOPs 问题。该算法提出了两类有向权重，即分布在可行区域内的可行权重和分布在不可行区域内的不可行权重。通过可行权重寻找分布性良好的可行解，而不可行权重来维持多样性良好的不可行解。

8）CM2M: CM2M 把一个多维的最优问题分解为若干个子问题，并在同一时间内对它们进行优化。其中每个子问题都对应一个临时寄存器和一个子种群，临时寄存器用来记录之前发现的个体，子种群则存放一些目标值较好且违反约束较少的个体。

3.3.3 性能指标

本文使用两种通用指标，反转世代距离（IGD）和超体积（HV），来衡量获得的非支配解集的收敛性和多样性。

IGD [73]是非支配解集中所有个体到帕累托解集的平均距离的逆向映射。它用帕累托最优解集中的个体到由该算法所求得的非支配解集的平均距离来表示。其计算公式如下：

$$\begin{cases} \mathrm{IGD}\left(P^*, A\right) = \dfrac{\sum y^* \in P^* d\left(y^*, A\right)}{\left|P^*\right|} \\ d\left(y^*, A\right) = \min_{y \in A}\left\{\sqrt{\sum_{i=1}^{m}\left(y_i^* - y_i\right)^2}\right\} \end{cases} \tag{3-7}$$

其中，P^*表示真实 PF 的一组代表性解。$d(y^*, A)$表示P^*中从帕累托最优曲面上的点y_i^*到单个点y_i的最小欧氏距离。IGD 值越小，代表算法的性能越好。

HV [74]反映了由 CMOEA 所获得的非支配解集与真实 PF 的接近程度。通过计算非支配解集与参考点围成的空间的超体积的值来评价 CMOEA 的综合性能。其计算公式如下：

$$HV(S) = \mathrm{VOL}\left(\cup_{x \in S}\left[f_1(x), z_1^r\right] \times \ldots \left[f_m(x), z_m^r\right]\right) \tag{3-8}$$

式中，VOL (·)为勒贝格测度，m表示目标数，$\mathbf{z}^r = (z_1^r, ..., z_m^r)^T$ 表示目标空间中用户定义的参考点。HV 值越大，说明其多样性和收敛性的性能越好。

3.4 实验结果与分析

3.4.1 IM-CMOPs 测试问题实验结果

在本小节，我们先分析 9 种算法在第一个测试问题集 IM-CMOPs 上的性能；然后在下一小节分析它们在第二个测试问题集 UCMOPs 上的表现，同时使用弗里德曼测试来说明各个算法的性能差异。

表 3-1 和表 3-2 显示了εM2M 和 CCMO[39]、CMOEA-MS[45]、cDPEA[61]、ToP[46]、MOEA/D-CDP[72]、MOEA/D-DW[2]、CM2M2[18]、CM2M[36]等 8 种对比算法在 IM-CMOP1~IM-CMOP5 上独立运行 30 次后 IGD 和 HV 指标的平均值（mean）和标准差（std）。表格中黑色加粗的数据代表最佳结果。根据弗里德曼检验，εM2M

获得的性能度量值显著优于其他 8 种对比算法。五个测试问题中，只有在 IM-CMOP4 上没有达到最好的效果，CM2M2 在 IM-CMOP4 上的效果略微胜过 εM2M，但也无明显差异。对于 IM-CMOP3 和 IM-CMOP5，εM2M 以压倒性的优势领先于其他对比算法。在 IM-CMOP1 和 IM-CMOP2 问题上，εM2M 获得的性能指标值略优于 CM2M 和 CM2M2，明显优于 CCMO、CMOEA-MS、cDPEA、ToP、MOEA/D-CDP、MOEA/D-DW。总体而言，εM2M 仍然以显著的优势胜于其他算法。

图 3-7 和图 3-8 显示了每个算法在 IM-CMOP4 和 IM-CMOP5 上获得的 HV 中位值的非支配解集。图中红色空心圆圈代表算法获得的解，蓝色实点代表真实的帕累托前沿。从图 3-7 和 3-8 中可以清楚地看到，εM2M 算法可以找到几乎全部的真是帕累托前沿，而其他算法只能收敛到部分。不仅如此，εM2M 始终能找到分布更好的非支配解。可能原因是εM2M 将种群分解为许多简单的子种群，因此可以保证种群的多样性，每个子种群都采用 IEpsilon 约束处理方法，进而能够动态调整约束违反的松弛度。因此，εM2M 可以找到所有的帕累托最优解，而其他算法只能找到真正的 PF 中的一小部分。结果表明，εM2M 算法在求解具有收敛性和多样性特征的 CMOP 方面取得了很好的效果。

表 3-1　εM2M 和其他 8 个 CMOEAs 在 IM-CMOP1~IM-CMOP5 上的 IGD 结果。最佳结果用黑色加粗字体突出显示

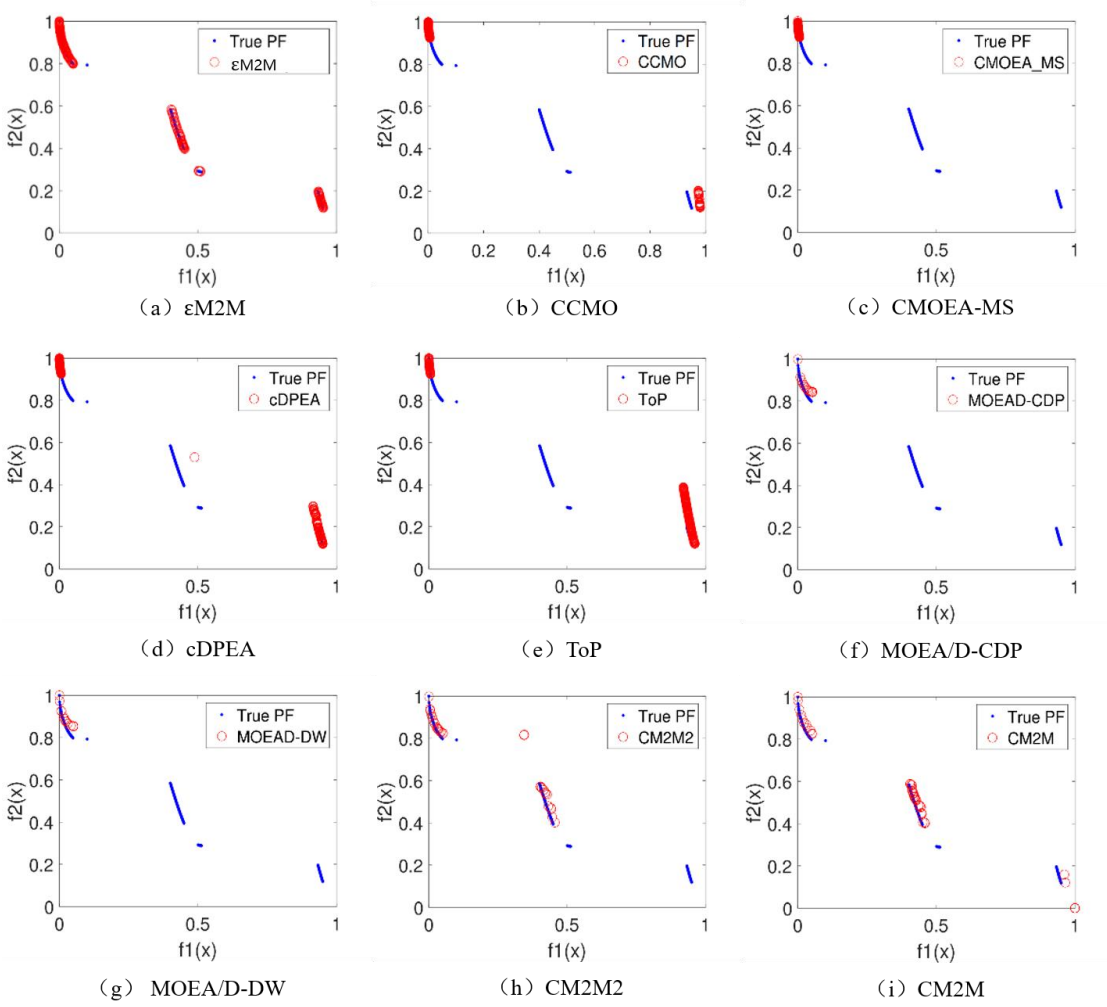
Test Instance		εM2M	CCMO	CMOEA-MS	cDPEA	ToP	MOEA/D-CDP	MOEA/D-DW	CM2M2	CM2M
IM-CMOP1	mean	**0.03750**	0.42567	0.72918	0.30854	0.36398	0.33271	0.42510	0.20507	0.09095
	std	0.01895	0.01622	0.00819	0.05301	0.04891	0.01665	0.07122	0.04467	0.02258
IM-CMOP2	mean	**0.15108**	0.26735	0.32341	0.25558	0.29512	0.32341	0.33189	0.16681	0.22885
	std	0.10097	0.00223	6.03005	0.01157	0.02906	6.40005	0.00264	0.06107	0.05649
IM-CMOP3	mean	**0.11469**	0.48192	0.41241	0.32502	0.59398	0.55310	0.51577	0.38134	0.34059
	std	0.07090	0.06758	0.03932	0.04477	0.02496	0.05654	0.07293	0.06365	0.06059
IM-CMOP4	mean	0.17339	0.31451	0.50361	0.23815	0.31415	0.35911	0.44058	**0.14578**	0.23233
	std	0.11054	0.00546	0.00064	0.07272	0.01254	0.00716	0.02871	0.07880	0.06818
IM-CMOP5	mean	**0.02648**	0.26141	0.26521	0.15691	0.30582	0.30355	0.3166	0.16961	0.12033
	std	0.00650	0.03691	0.03622	0.02137	0.03073	0.03069	0.02939	0.04744	0.03870
Friedman test		1.2	6	7.3	3.4	6.8	6.9	8	2.8	2.6

表 3-2　εM2M 和其他 8 个 CMOEAs 在 IM-CMOP1~IM-CMOP5 上的 HV 结果。最佳结果用黑色加粗字体突出显示

Test Instance		εM2M	CCMO	CMOEA-MS	cDPEA	ToP	MOED/D-CDP	MOED/D-DW	CM2M2	CM2M
IM-CMOP1	mean	**0.97381**	0.33481	0.32839	0.52119	0.44468	0.49874	0.41550	0.82248	0.90562
	std	0.02165	0.01887	0.01051	0.07817	0.07854	0.03546	0.04742	0.02576	0.02731
IM-CMOP2	mean	**0.49430**	0.34535	0.22799	0.39292	0.28908	0.22799	0.22799	0.34149	0.43405
	std	0.06201	0.01601	1.69E-16	0.01340	0.06266	1.69E-16	1.69E-16	0.05063	0.03169
IM-CMOP3	mean	**0.45889**	0.23926	0.23926	0.25727	0.23926	0.23926	0.23926	0.26553	0.25723
	std	0.05580	8.47E-17	8.47E-17	0.02879	8.47E-17	8.4E-17	8.47E-17	0.02907	0.02866
IM-CMOP4	mean	0.62277	0.42519	0.31325	0.51471	0.42091	0.37248	0.37271	**0.62905**	0.57176
	std	0.14871	0.01231	0.00107	0.06780	0.01583	0.01164	0.01455	0.12918	0.07193
IM-CMOP5	mean	**0.98601**	0.63929	0.63993	0.78486	0.59753	0.6027	0.59368	0.81168	0.83639
	std	0.01042	0.05659	0.05165	0.02913	0.03881	0.04946	0.05019	0.04845	0.04986
Friedman test		1.2	6	7.6	3.6	6.6	7	7.6	2.8	2.6

如图 3-7 所示，εM2M 找到了所有的帕累托最优解。其次是 cDPEA 和 ToP，收敛到三分之二的 PF。对于 CCMO、CMOEA-MS、MOEA/D-CDP、MOEA/D-DW，只有少量的解能收敛到很小一段的 PF，可能原因是这些算法不适用于解决不平衡的约束多目标优化问题。对于 CM2M2 和 CM2M，部分解找到了真实的 PF，可能是因为虽然 CM2M2 和 CM2M 都采用了 M2M 分解方法，但它们没有特定的约束处理机制来解决同时存在多样和收敛困难约束的 CMOPs，导致个体在搜索过程中无法穿过不可行区域，因此只有少部分的个体能够找到真实的帕累托前沿。

在图 3-8 中，εM2M 以压倒性的优势在一众算法中取得了最佳结果。对于 CM2M2 和 CM2M，少量非支配解收敛到真正的 PF。原因还是在于没有 IEpsilon 机制的辅助，大量个体在搜索空间中无法跨越障碍以到达可行区域。对于其他的六种算法，几乎没有个体可以找到真正的 PF。由此可见，分解和约束处理是解决具有收敛性和多样性困难的不平衡 CMOPs 的核心，是两个至关重要、不可或缺的步骤。



（a）εM2M　（b）CCMO　（c）CMOEA-MS

（d）cDPEA　（e）ToP　（f）MOEA/D-CDP

（g） MOEA/D-DW　（h）CM2M2　（i）CM2M

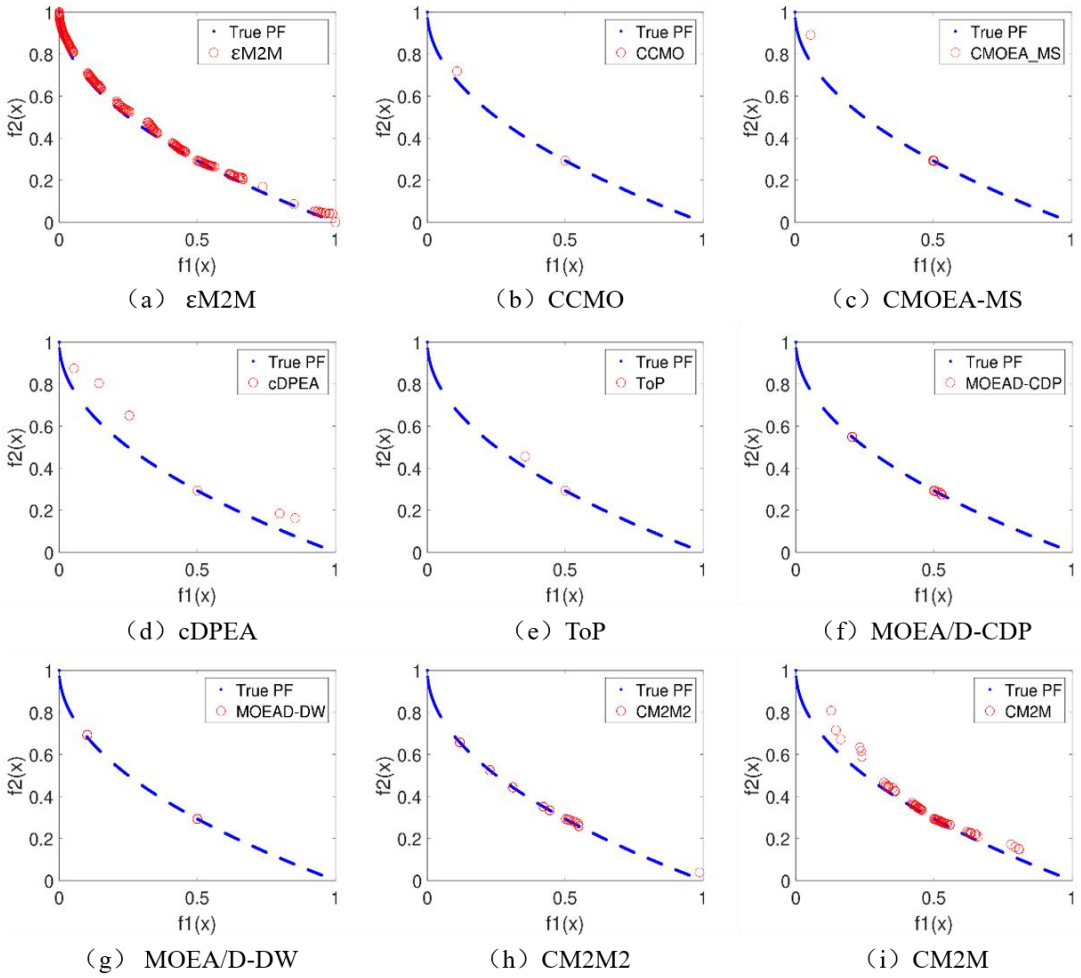
图 3-7 各个算法在 IM-CMOP4 上获得的非支配解集

3.4.2 UCMOPs 测试问题实验结果

εM2M、CCMO、CMOEA-MS、cDPEA、ToP、MOEA/D-CDP、MOEA/D-DW、CM2M2 以及 CM2M 等 9 种对比算法在 UCMOP1-1~UCMOP3-3 上独立运行 30 次后 IGD 和 HV 指标的平均值和标准差如表 3-3 和表 3-4 所示，最佳结果以粗体显示。

如表 3-3 所示，εM2M 的总体效果最好。对于 UCMOP1-1，使用 CM2M2 和 CM2M 获得的性能度量值略低于εM2M。对于 UCMOP1-2，MOEA/D-DW 的 IGD 结果与εM2M 近似。对于 UCMOP1-3，εM2M 效果略弱于 CM2M2, 但仍显著优于其他 7 种算法。在剩下的 6 个 UCMOPs 中，εM2M 总是领先于其他算法。从表 3-4 中，我们可以看到εM2M 的性能始终优于所有其他评估算法。对于 UCMOP1-2 问题，MOEA/D-DW、CM2M2 和 CM2M 的效果略低于εM2M。而对于 UCMOP1-3，CM2M2 的 HV 结果与所提出的εM2M 的效果相似。对于剩余

的问题，εM2M 仍然以压倒性的优势取得最佳结果。综上，在绝大多数测试实例中，εM2M 始终比其他 8 种算法的性能更好，获得了更好的种群收敛性和多样性。为进一步展示εM2M 算法在求解 UCMOPs 上的优势，图 3-9 画出了每个算法在 UCMOP2-3 上获得的 HV 中位值的非支配解集。



（a） εM2M　（b）CCMO　（c）CMOEA-MS

（d）cDPEA　（e）ToP　（f）MOEA/D-CDP

（g） MOEA/D-DW　（h）CM2M2　（i）CM2M

图 3-8 各个算法在 IM-CMOP5 上获得的非支配解集

CCMO、CNOEA-MS 和 cDPEA 可以很好地搜索第一段真实 PF，但在第二段真实 PF 上的表现不如 ε M2M。ToP 的结果与 ε M2M 相当，无明显差异。对于 MOEA/D-CDP、MOEA/D-DW、CM2M2 和 CM2M，这四种算法得到的非支配解只找到了 PF 的一部分，而在另一部分上几乎没有一个解。这可能是因为，虽然 CM2M 和 CM2M2 采用了 M2M 分解策略，但它们并没有一种有效的约束处理机制来帮助种群中的个体跨越大型不可行区域。结果表明，改进的 IEpsilon 策略致力于保留种群中一些不可行的解，而 M2M 分解方法保持了种群的多样性，这两种策略的结合可以完美地解决不平衡的约束多目标问题。基于以上分析，我们可以得出结论，在解决不平衡的 CMOPs 上，εM2M 显著优于其他 8 种对比算法。得到以上结果的原因在于：

UCMOP1-1~UCMOP1-3 中的约束条件将整个目标空间分解为两部分，有利的 PF 被可行区域包围，而不利的 PF 被不可行区域包围。a 的值越小，可行区域和不可行区域之间的不平衡搜索就越大，导致算法就越难找到不利的 PF。特别是对于使用基于偏好可行解而非不可行解的约束处理技术的 CMOEAs，他们将无法找到不利的 PF，因为它们在进化过程中无法找到一些不可行的解。

表 3-3　εM2M 和其他 8 个 CMOEAs 在 UCMOPs 上的 IGD 结果。测试问题的比较算法的最佳平均值用黑体突出显示。

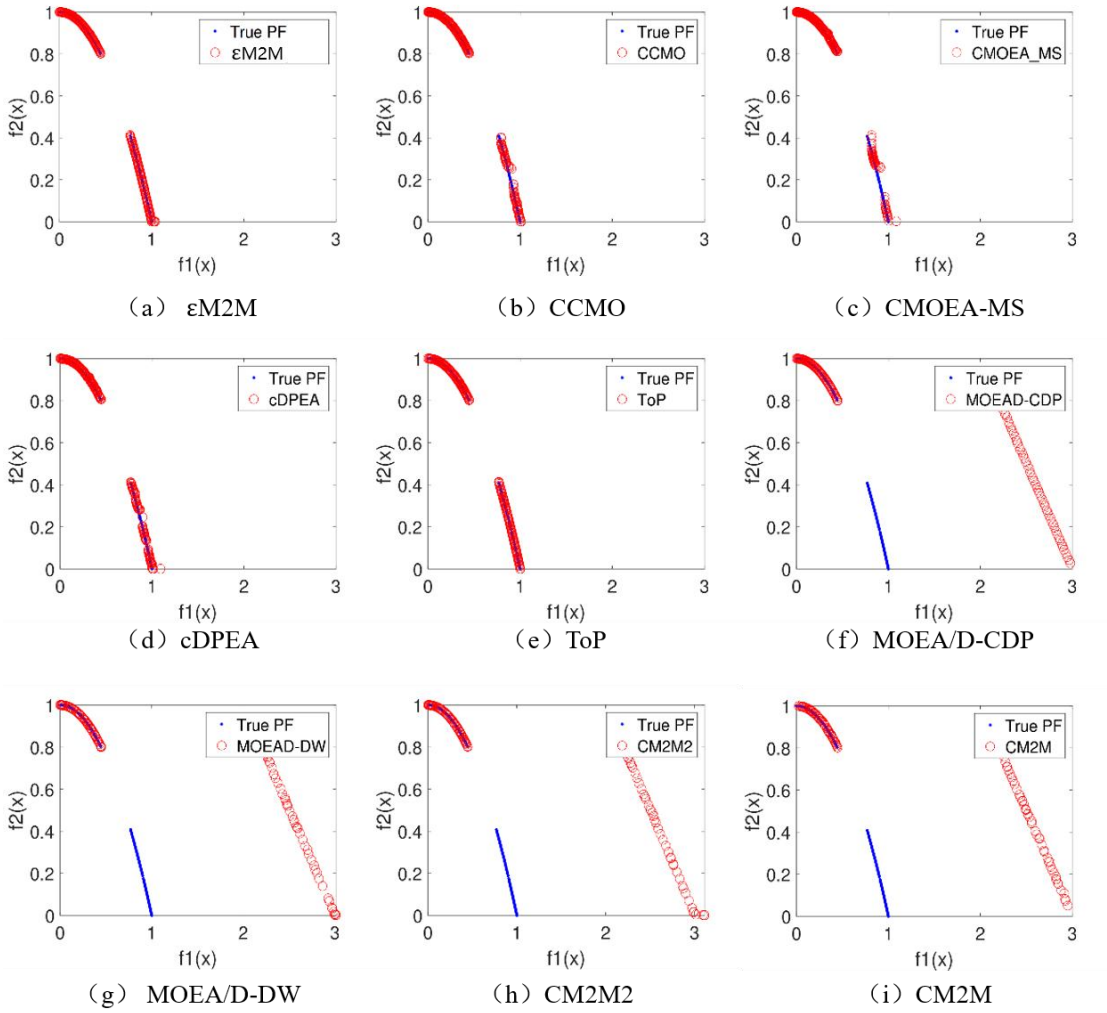
Test Instance		εM2M	CCMO	CMOEA-MS	cDPEA	ToP	MOEAD-CDP	MOEAD-DW	CM2M2	CM2M
UCMOP1-1	mean	**0.00265**	0.33807	0.34680	0.34680	0.34685	0.34693	0.05813	0.00769	0.00977
	std	0.00076	0.04237	0.00036	0.00028	0.00124	0.00366	0.12678	0.00031	0.00090
UCMOP1-2	mean	**0.00252**	0.36890	0.37742	0.37739	0.37640	0.37758	0.00299	0.00769	0.01045
	std	0.00035	0.04650	0.00071	0.00042	0.00562	0.00382	0.00143	0.00069	0.00139
UCMOP1-3	mean	0.00849	0.39272	0.39272	0.39272	0.39276	0.39279	0.27605	**0.00642**	0.00841
	std	0.01185	0.00141	0.00159	0.00153	0.00250	0.00352	0.18207	0.00093	0.00030
UCMOP2-1	mean	**0.01378**	0.21204	0.28035	0.20621	0.33001	0.38244	0.22098	0.34160	0.27064
	std	0.06644	0.15486	0.16315	0.13748	0.11116	0.09382	0.18040	0.09162	0.16315
UCMOP2-2	mean	**0.08586**	0.21655	0.24646	0.23204	0.35162	0.38737	0.23004	0.38363	0.37322
	std	0.10894	0.11623	0.09573	0.10585	0.12728	0.14283	0.18035	0.08993	0.10831
UCMOP2-3	mean	**0.00167**	0.16051	0.25521	0.15927	0.36810	0.40165	0.26451	0.40195	0.27963
	std	0.00010	0.13032	0.14976	0.10466	0.13435	0.07543	0.20233	0.07540	0.19754
UCMOP3-1	mean	**0.00154**	0.31457	0.36561	0.33531	0.14057	0.85701	0.08325	0.11724	0.11750
	std	0.00012	0.16480	0.14236	0.17397	0.19119	0.54149	0.16290	0.00329	0.00805
UCMOP3-2	mean	**0.00458**	0.19031	0.26001	0.23195	0.07817	1.00360	0.14842	0.39017	0.09741
	std	0.00012	0.11161	0.06260	0.09389	0.18250	0.52754	0.23685	0.35520	0.19925
UCMOP3-3	mean	**0.00155**	0.25682	0.33096	0.23976	0.26864	1.16490	0.04181	0.08204	0.08239
	std	0.00010	0.18923	0.19140	0.20925	0.24078	0.38538	0.11845	0.00779	0.00013
Friedman test		1.2222	4.6667	6.6111	5.1667	6.4444	8.8889	3.4444	4.3333	4.2222

而 UCMOP2-1~UCMOP2-3 中的约束含有两个接近 PF 的不同宽度的断开的不可行带。一些保留不可行解的 CMOEAs 可以很容易找到有利 PF，而在有利 PF 中找到的解将消除大多数难以通过更大不可行带的解。因此，没有任何

机制来维持不利 PF 周围的解的 CMOESs 最终倾向于只收敛到有利 PF。

表 3-4　εM2M 和其他 8 个 CMOEAs 在 UCMOPs 上的 HV 结果。测试问题的比较算法的最佳平均值用黑体突出显示。

Test Instance		εM2M	CCMO	CMOEA-MS	cDPEA	ToP	MOEAD-CDP	MOEAD-DW	CM2M2	CM2M
UCMOP1-1	mean	**0.93591**	0.56308	0.55476	0.55476	0.55466	0.55457	0.87419	0.92809	0.92116
	std	0.00112	0.04058	0.00028	0.00026	0.00164	0.00526	0.14142	0.00638	0.00243
UCMOP1-2	mean	**1.10330**	0.77945	0.77347	0.77356	0.77440	0.77310	1.10280	1.09600	1.08840
	std	0.00041	0.03224	0.00205	0.00042	0.00510	0.00010	0.00445	0.00704	0.00284
UCMOP1-3	mean	0.76291	0.37034	0.37034	0.37034	0.37029	0.37020	0.48989	**0.76343**	0.75639
	std	0.01417	0.00020	0.00019	0.00019	0.00108	0.00226	0.18656	0.00752	0.00278
UCMOP2-1	mean	**0.86519**	0.63259	0.57521	0.64086	0.60084	0.55573	0.69365	0.59357	0.64843
	std	0.05509	0.12729	0.14104	0.12128	0.09238	0.10328	0.14888	0.07466	0.13591
UCMOP2-2	mean	**1.00801**	0.72984	0.67431	0.69499	0.78731	0.75087	0.88878	0.76416	0.76583
	std	0.08931	0.20290	0.17401	0.17254	0.10045	0.15945	0.14630	0.07242	0.08847
UCMOP2-3	mean	**0.71458**	0.54488	0.46203	0.52806	0.44686	0.42898	0.52700	0.42865	0.51069
	std	0.00015	0.10424	0.10064	0.08836	0.09439	0.05392	0.14454	0.05385	0.13937
UCMOP3-1	mean	**0.84831**	0.58839	0.55247	0.56225	0.73678	0.29620	0.78606	0.82229	0.81727
	std	0.00014	0.11651	0.10592	0.13108	0.14715	0.29806	0.12380	0.00706	0.00033
UCMOP3-2	mean	**1.00600**	0.70624	0.60947	0.63469	0.96188	0.23348	0.92380	0.75783	0.94705
	std	0.00018	0.17735	0.10119	0.14852	0.10397	0.37374	0.13348	0.25361	0.11126
UCMOP3-3	mean	**0.69194**	0.52116	0.49500	0.53766	0.46976	0.07913	0.66157	0.68122	0.67618
	std	0.00011	0.09056	0.10281	0.10746	0.19714	0.18579	0.09076	0.00011	0.00049
Friedman test		1.3333	5.3333	7.3889	5.9444	6.1111	8.5556	3.4444	3.5556	3.3333



（a） εM2M　（b）CCMO　（c）CMOEA-MS

（d）cDPEA　（e）ToP　（f）MOEA/D-CDP

（g） MOEA/D-DW　（h）CM2M2　（i）CM2M

图 3-9 各个算法在 UCMOP2-3 上获得的非支配解的中值 HV 值

UCMOP3-1~UCMOP3-3 问题的特点是一个区域内的一些约束重叠会导致该区域中更高的约束违反。同时，随着算法深入到相应的不可行带，约束违反程度会显得更大，这可能会给一些基于保留不可行解的 CMOEAs 引入差异。这些约束违反程度较大的不可行解更接近于不利的 PF，但这些可行解可以被距离较远的可行解所消除，除非约束处理技术可以保留它们，直到算法找到不利的 PF。基于分解的 CMOEAs 具有更强的维持种群多样性的能力。因此，在约束违反越大的不可行带中，不可行解的信息更有可能生存到下一代。

第 4 章 井眼轨迹的设计与优化

作为世界范围内的主要能源之一，石油和天然气已经成为促进产业革命的重要动力。随着我国对石油资源需求的不断增长，对钻井工程的研究和发展提出了更为详尽和严格的要求，包括钻井工程的设计、井身质量控制、钻井效率的提高以及相关环保问题的解决等方面。这些要求的实现，对保证国家石油产业发展的健康和持续具有十分重要的意义[75]。为此，需要在设计环节进行全面考虑，提升钻井工程的安全性和可靠性；在井身质量上，要求确保钻井过程中井身的稳定性和完整性；在效率方面，要求在降低成本和缩短钻井周期方面进行有效的优化。在钻井工程中，井眼轨道规划是技术研究的核心内容，其好坏不仅关系到油井的生产效率，而且也关系到整个钻探过程的成败。井眼轨道规划包含井眼轨道设计和井眼轨道优化。在实际应用中，因轨道轮廓的复杂性，使得常规设计方法在工程约束条件下难以实现轨道的精确入靶。因此，井眼轨道规划实质是根据油气勘探开发要求来构建最优的井眼轨道的空间形态。

本文的动机是在满足勘探开发与钻井技术条件的基础上，尽可能设计形状简单、易施工的井眼轨道，以减少井眼轨迹控制的难度和工作量。通过这种方式，实现钻井安全、高效、优质的目标。这样可以减少不必要的设备投入，提高钻井效率，同时也可以降低钻井成本。此外，应重视对井眼轨道的维护和管理工作，以确保井眼轨道控制的有效性，确保钻井过程中的安全性和可靠性。然而，在设计井眼轨道的过程中，由于各种因素的制约，常常会出现一些冲突的情况。比如，井倾斜角度过小，虽然对完井和开采工作有利，但也增加了在钻进过程中的定向控制难度。因此，在井眼轨道设计时，应优先解决突出问题，对井眼中的问题进行深入的分析，以权衡利弊，综合考虑各种因素，比如地质条件、环境状况、经济成本以及其他因素等，并结合实际情况提出最佳方案，从而设计出最优的井眼轨道。

井眼轨道的长度是影响钻井成本和时间的最基本因素之一。为此，本文在可接受的操作约束和地质限制等约束条件下，利用曲率半径法在七段式井眼轨道基本类型上构建出三维井眼轨道长度模型，该模型可以模拟钻井过程中的井眼轨迹，旨在为钻井提供参考依据。由于井眼轨道长度受多种约束条件限制，传统的井眼轨道设计很大程度上依赖于设计师的经验和判断，以至于很难获得的最优的设计方案。为此，本文采用 CMOEAs 对构建的三维井眼轨道长度模型进行求解，从而提供一套最优的井眼轨道规划方案。

此外，在井眼轨道规划时，不仅要考虑钻井成本，还要兼顾井眼轨道的安全，即在钻井作业过程中最小化钻柱上的扭矩，从而降低钻具的故障率和减少完井问题。因此，为了建立相应的扭矩模型，需要对钻井过程中产生的扭矩和阻力进行分析。在此基础上，利用带约束的多目标进化算法对其进行优化，从而给出一套最优的井眼轨道规划方案。

4.1 基于空间形态的井眼轨道模型

4.1.1 圆弧段轨道求解

本文采用曲率半径法[76, 77]来测量圆弧段的井眼轨道。该方法首先选择曲线上的两个点，计算它们之间的弧长和夹角。然后基于曲率半径公式，计算出这两点之间的平均曲率半径。最后，根据该曲率半径评估曲线的弧形情况和大小。在测量过程中，假设曲线为一条等变螺旋角的圆柱螺旋线，该曲线与井眼方向线在起点和终点处相切。并且，该曲线在水平和垂直方向上分别投影为一个圆弧，其半径分别为R_v和R_h。曲率半径法的模型示意图如下所示：

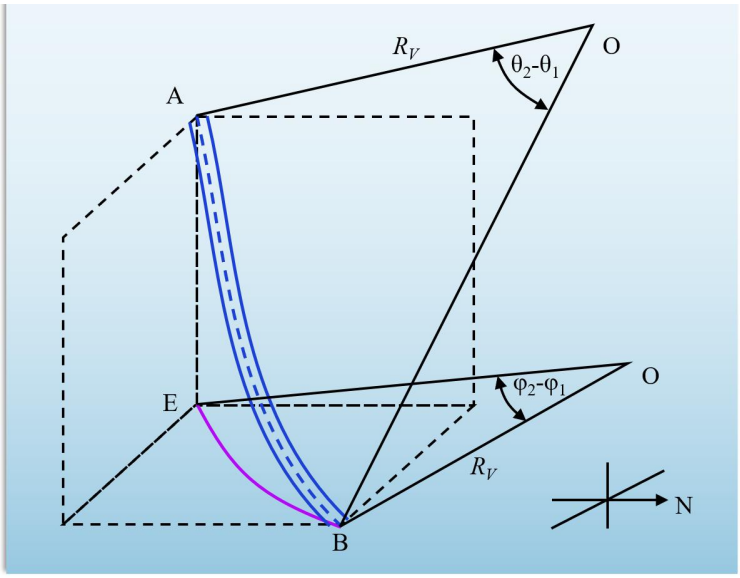


图 4-1 曲率半径法模型

R_v和R_h分别表示垂直面和水平面上的曲率半径:

$$\frac{(\theta_2-\theta_1)}{360}=\frac{TMD}{2\pi R_V}\leftrightarrow R_V=\frac{TMD}{\theta_2-\theta_1}*\left(\frac{180}{\pi}\right) \tag{4-1}$$

$$R_h=\frac{\Delta H}{360}*\left(\frac{180}{\pi}\right) \tag{4-2}$$

ΔV和ΔH分别代表垂直和水平方向上的增量:

$$\Delta V = R_V * (\sin\theta_2 - \sin\theta_1) = \frac{TMD}{(\theta_2 - \theta_1)} * \left(\frac{180}{\pi}\right) * (\sin\theta_2 - \sin\theta_1) \tag{4-3}$$

$$\Delta H = R_V * (\cos\theta_2 - \cos\theta_1) \tag{4-4}$$

向北方向的增量ΔN为:

$$\Delta N = \frac{TMD}{(\theta_2 - \theta_1)} * \left(\frac{180}{\pi}\right) * \frac{(\cos\theta_2 - \cos\theta_1)}{\varphi_2 - \varphi_1} (\sin\varphi_2 - \sin\varphi_1) \tag{4-5}$$

向东方向的增量ΔE为:

$$\Delta E = \frac{TMD}{(\theta_2 - \theta_1)} * \left(\frac{180}{\pi}\right) * \frac{(\cos\theta_2 - \cos\theta_1)}{\varphi_2 - \varphi_1} (\cos\varphi_2 - \cos\varphi_1) \tag{4-6}$$

4.1.2 直线段轨道求解

正切法[78]是一种利用切线与水平或垂直线的夹角来确定曲线坡度或曲率的方法，是最简单的井眼轨道计算模型。本文采用正切法测量直线段的井眼轨道，如图 4-2 所示，计算出的井眼轨道用红色表示，蓝色代表实际井眼轨道。

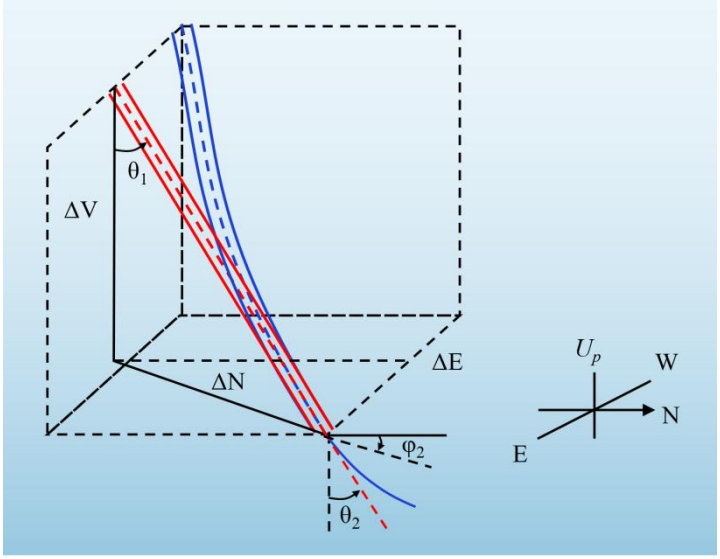


图 4-2 切线法模型示意图

正切模型的计算为：

$$\Delta V = TMD * \cos\theta_2 \tag{4-7}$$

$$\Delta N = TMD * \cos\theta_2 * \cos\varphi_2 \tag{4-8}$$

$$\Delta E = TMD * \cos\theta_2 * \cos\varphi_2 \tag{4-9}$$

式~中：ΔV、ΔN和ΔE分别代表垂直、向北、向东方向上的增量；TMD为两个测量点之间的垂直距离；θ_2为下方测量站点相对于上方测量站点的倾斜角度，常用度数或百分比度量；φ_2为下方测量站点相对于上方测量站点的方向角度，通常用角度（以度、弧度或坐标系表示）来度量。

4.1.3 模型建立

本文的三维井眼轨迹设计与优化模型是以七段式井眼轨道模型[79]为基础框架构建的。在钻井过程中，有正切法、最小曲率法、曲率半径法等几种常见的求解方法。为了更精确地计算井眼的轨迹，一般将其看作一条空间曲线或弧线。其中，直线模型是最简单的，球体或圆柱体模型则比较复杂，用于描述大位移、定向井等复杂井眼轨迹。对于垂直井，采用直线模型能获得较高的精确度。使用简单模型得到复杂井眼轨迹的计算结果误差很大，因此需要更加复杂的计算模型来描述。

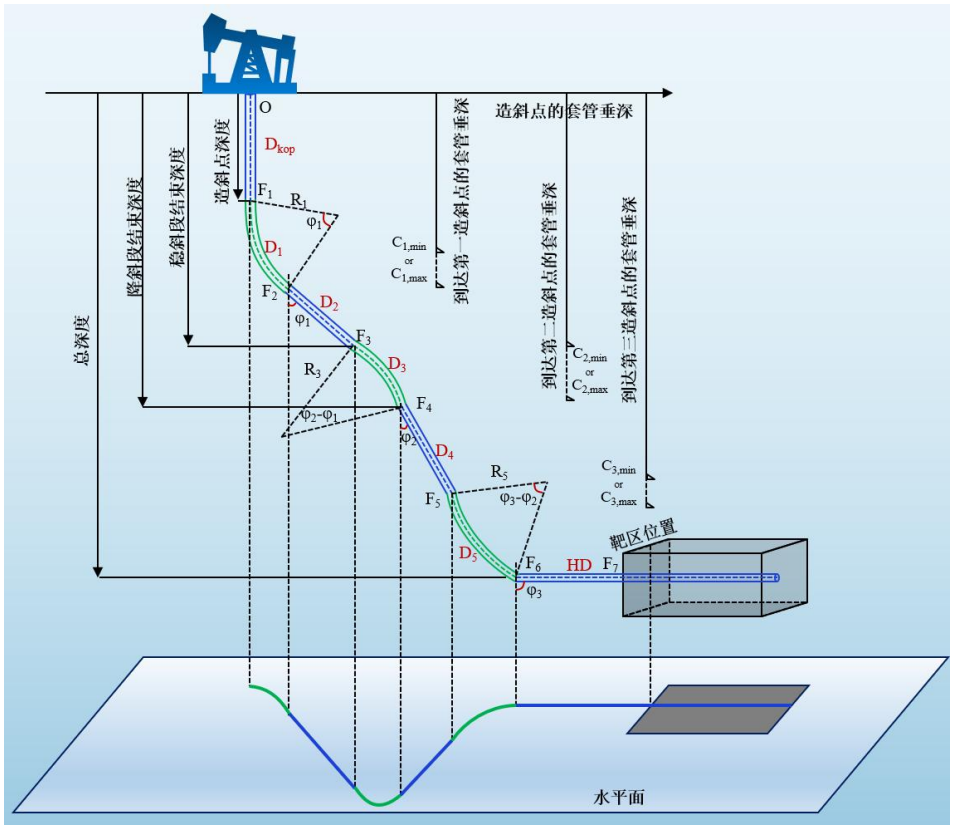


图 4-3 七段式井眼轨道设计模型

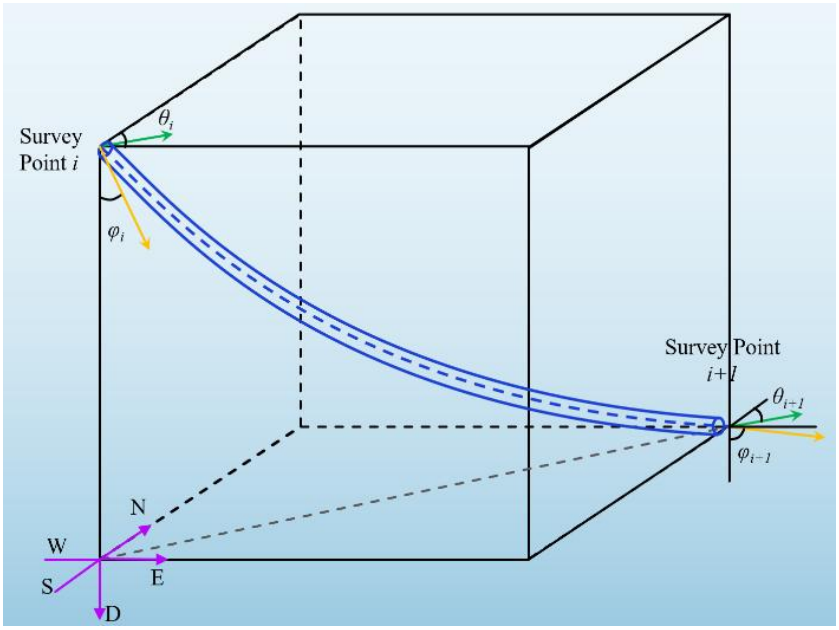


图 4-4 圆弧段的三维示意图

通过对七段式井眼轨道模型进行分析，我们发现该模型是由圆弧段和直线段所构成的，因此，本文分别利用曲率半径法和正切法来构建圆弧段和直线段的井眼轨道模型。

七段式井眼轨道设计模型如图 4-3 所示。主要由七个部分组成：垂直段（D_{KOP}）、增斜段（D_1）、稳斜段（D_2）、降斜段（D_3）、稳斜段（D_4）、增斜段（D_5）和水平段（HD）。圆弧段的三维示意图如图 4-2 所示。计算真实测量深度（TMD）的主要参数包括方位角、垂直倾斜保持角、真实垂直深度、横向长度、狗腿度等。

由曲率半径法可得：

$$a = \frac{1}{\Delta m}\sqrt{\left(\theta_{i+1}-\theta_i\right)^2 \sin^4\left(\frac{\varphi_{i+1}+\varphi_i}{2}\right)+\left(\varphi_{i+1}+\varphi_i\right)^2} \tag{4-10}$$

$$r = \frac{1}{a} = \frac{180*100}{\pi * T} \tag{4-11}$$

$$\Delta m = r\sqrt{\left(\theta_{i+1}-\theta_i\right)^2 \sin^4\left(\frac{\varphi_{i+1}+\varphi_i}{2}\right)+\left(\varphi_{i+1}+\varphi_i\right)^2} \tag{4-12}$$

Δm 是两点之间的井眼轨道，a是曲率常数，T是狗腿度，r是曲率半径，θ_i是方位角，φ_i是井斜角（图 4-3 所示，其中“S”，“E”、“N”和“W”代表“南”、“东”、“北”和“西”）。

实际测量长度优化模型计算：

$$TMD = D_{KOP} + D_1 + D_2 + D_3 + D_4 + D_5 + HD \tag{4-13}$$

其中，

$$\mathrm{D_1} = \mathrm{R_1}\sqrt{(\theta_2 - \theta_1)^2 \sin^4\left(\frac{\varphi_1 + \varphi_0}{2}\right) + (\varphi_1 + \varphi_0)^2}, (\varphi_0 = 0) \tag{4-14}$$

$$\mathrm{D_3} = \mathrm{R_3}\sqrt{(\theta_4 - \theta_3)^2 \sin^4\left(\frac{\varphi_2 + \varphi_1}{2}\right) + (\varphi_2 + \varphi_1)^2} \tag{4-15}$$

$$\mathrm{D_5} = \mathrm{R_5}\sqrt{(\theta_6 - \theta_5)^2 \sin^4\left(\frac{\varphi_3 + \varphi_2}{2}\right) + (\varphi_2 + \varphi_3)^2} \tag{4-16}$$

$$\mathrm{D_2} = \mathrm{D_D} - \mathrm{D_{KOP}} - \mathrm{D_1} \times (\sin\varphi_1 - \sin\varphi_0) / ((\varphi_1 - \varphi_0) \times \cos\varphi_1) \tag{4-17}$$

$$\mathrm{D_4} = \mathrm{D_B} - \mathrm{D_D} - \mathrm{D_3} \times (\sin\varphi_2 - \sin\varphi_1) / ((\varphi_2 - \varphi_1) \times \cos\varphi_2) \tag{4-18}$$

上式中：TMD代表了实际测量长度；$\mathrm{D_1}$~$\mathrm{D_5}$分别表示第一段增斜段、正切段、降斜段、稳斜段、第二段增斜段轨道的测量长度；HD表示水平段井身的长度；θ_i和φ_i分别代表方位角和井斜角；$cas_{\max}$和$cas_{\min}$代表管深度约束的上下限；$\mathrm{R_1}$、$\mathrm{R_3}$、和$\mathrm{R_5}$分别表示$\mathrm{D_1}$、$\mathrm{D_3}$、和$\mathrm{D_5}$三段的曲率半径。

4.2 基于摩阻扭矩的井眼轨道模型

本文通过建立扭矩模型来反应钻井过程中阻力对井眼轨道的影响。首先对井眼轨道每一段进行受力分析，然后再从井眼轨道的末端钻杆所受实际控制转矩逐步向上延伸到井口。

为了方便计算钻头的扭矩，井筒内采用重索简化钻柱，同时忽略钻杆引起的管状刚度影响。对于向下作用的重力，采用井筒倾斜方式。井筒倾斜处直管的力平衡如图 4-5 所示。因此，计算直线段扭矩的公式如下：

$$F_2 = F_1 + Bw\Delta L\cos\varphi \tag{4-19}$$

$$T = \mu\frac{D}{2}w\Delta L\sin\varphi \tag{4-20}$$

式中，F_1为元件底部的轴向载荷，F_2为元件顶部的轴向载荷，B为浮力系数，w为单位管长的重量，ΔL为间隔长度，μ为摩擦系数，D为管道直径。

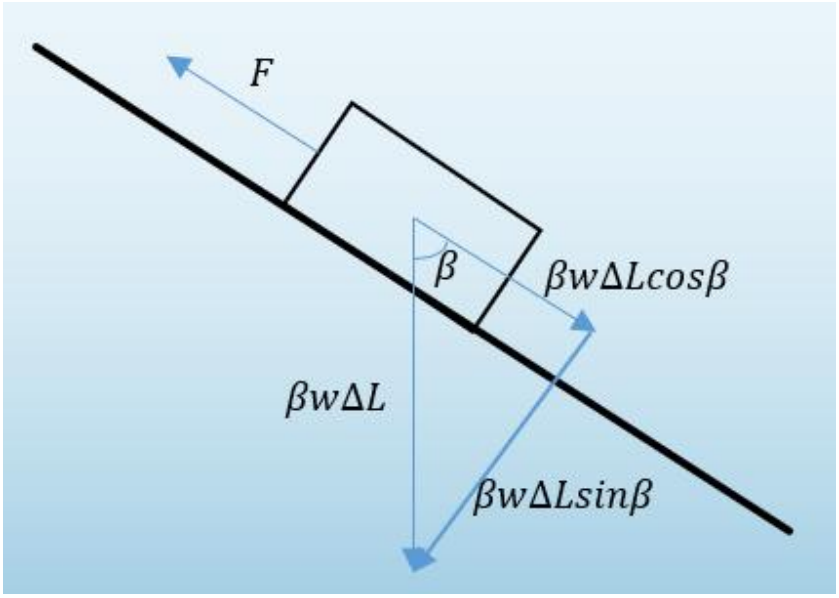


图 4-5 管道沿直线表面拉伸时的力平衡

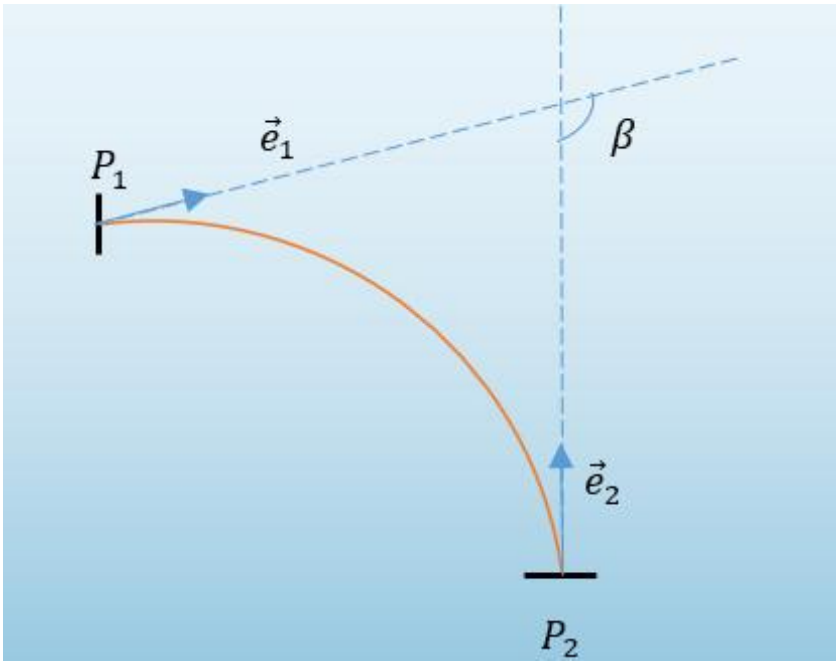


图 4-6 在点P_1和P_2处的总方向变化和单位向量$\vec{e}_1$、$\vec{e}_2$。

至于弯曲的部分，管道中的轴向载荷影响弦与孔之间的正常接触力。因此，管道内的张力对于计算力矩是至关重要的。通过公式和中给出的关系可以推导出，确定总方向变化的角度是扭矩计算中其他不可缺少的参数。图 4-6 和图 4-7 展示了当两点之间的总方向变化（记作β）时，曲线截面上的轴向载荷计算如下：

$$\mathbf{e}_1 \cdot \mathbf{e}_2 = |\mathbf{e}_1||\mathbf{e}_2|\cos\beta = \cos\beta \tag{4-21}$$

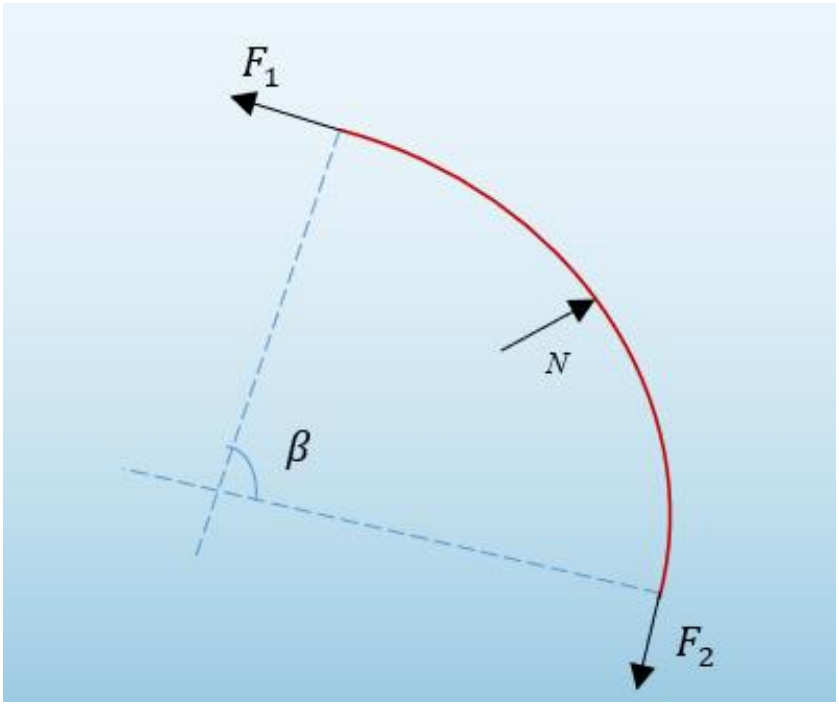


图 4-7 在方向上总改变为β期间，曲线截面上的轴向载荷。

$$\cos\beta = \sin\varphi_1 \sin\varphi_2 \cos(\theta_1 - \theta_2) \tag{4-22}$$

式中，β为总方向变化，$\mathbf{e}_1$、$\mathbf{e}_2$为井筒方向的单位矢量。由公式可计算出轴向载荷，从而通过公式得出扭矩:

$$F_2 = F_1 + Bw\Delta L\left(\frac{\sin\varphi_2 - \sin\varphi_1}{\varphi_2 - \varphi_1}\right) \tag{4-23}$$

$$T = \mu\frac{D}{2}F_1\beta \tag{4-24}$$

对于假设由七个部分组成的井筒轨道（如图 4-3 所示），将分别计算每个部分，然后相加得到最终扭矩。扭矩目标函数如下：

$$T = T_{\text{vertical}} + T_1 + \cdots + T_5 + T_{\text{horizontal}} \tag{4-25}$$

假设钻柱的钻头上没有重量（即$F_7 = 0$），并且它在旋转时没有任何轴向向上或向下运动。

$$\begin{cases} F_7 = 0 \\ F_6 = F_7 + Bw\Delta L_7 \cos\varphi_3 \\ F_5 = F_6 + Bw\Delta L_6 \left(\dfrac{\sin\varphi_3 - \sin\varphi_2}{\varphi_3 - \varphi_2} \right) \\ F_4 = F_5 + Bw\Delta L_5 \cos\varphi_2 \\ F_3 = F_4 + Bw\Delta L_4 \left(\dfrac{\sin\varphi_2 - \sin\varphi_1}{\varphi_2 - \varphi_1} \right) \\ F_2 = F_3 + Bw\Delta L_3 \cos\varphi_1 \\ F_1 = F_2 + Bw\Delta L_2 \left(\dfrac{\sin\varphi_0 - \sin\varphi_1}{\varphi_0 - \varphi_1} \right), (\varphi_0 = 0) \end{cases} \tag{4-26}$$

每个部分的扭矩的计算方法如下：

$$\begin{cases} T_7 = \mu \dfrac{D}{2} w\Delta L_7 \sin\varphi_3 \\ \cos\beta_6 = \sin\varphi_3 \sin\varphi_2 \cos(\theta_5 - \theta_6) + \cos\theta_5 \cos\theta_6 \\ T_6 = \mu \dfrac{D}{2} F_6 \beta_6 \\ T_5 = \mu \dfrac{D}{2} w\Delta L_5 \sin\varphi_2 \\ \cos\beta_4 = \sin\varphi_1 \sin\varphi_2 \cos(\theta_3 - \theta_4) + \cos\theta_3 \cos\theta_4 \\ T_4 = \mu \dfrac{D}{2} F_4 \beta_4 \\ T_3 = \mu \dfrac{D}{2} w\Delta L_3 \sin\varphi_1 \\ \cos\beta_2 = \sin\varphi_1 \sin\varphi_0 \cos(\theta_1 - \theta_2) + \cos\theta_1 \cos\theta_2 \\ T_2 = \mu \dfrac{D}{2} F_2 \beta_2 \\ T_1 = \mu \dfrac{D}{2} w\Delta L_1 \sin\varphi_0, (\varphi_0 = 0) \end{cases} \tag{4-27}$$

在上述计算中，浮力系数B为 0.7，单位管长重量w为 0.3 kN/ft，摩擦系数μ为 0.2，管道半径D为 0.2 ft。

4.3 约束多目标井眼轨迹优化

早期的井道优化模型大多是单目标优化模型，往往存在一定的局限性。井道优化模型通常有多个目标。与单目标优化模型相比，具有两个以上目标的多目标优化模型更难求解。在本研究中，所制定的模型有两个相互冲突的目标，这使得搜索空间非光滑和非线性。此外，约束条件提高了求解所提出的井道轨迹优化模

型的复杂性。

本文制定的井眼轨道优化模型包含两个目标：井眼轨道的长度和实际控制扭矩。此外，制定的优化模型包含许多约束条件。例如，套管设置点应位于一个合适的地层中，以便在固井后为套管鞋提供压力完整性。在油井轨迹设计过程中，通常会考虑与定置套管深度和真实垂直深度（TVD）有关的约束条件的变化。然而，一些操作性的钻井参数，如建孔角度率、建孔角度下降率和保角井段等，则不能考虑负值，而这些参数对井轨迹设计是至关重要的，以确保正常作业。通过4.1节和4.2节的分析，将制定的优化模型定义如下：

$$
\begin{aligned}
&obj\_function = \min\left(f_1(x), f_2(x)\right)\\
&s.t. \begin{cases}
x_{i,min} \le x_i \le x_{i,max}\\
C_{1,\min} \le C_1 \le C_{1,\max}\\
C_{2,\min} \le C_2 \le C_{2,\max}\\
C_{3,\min} \le C_3 \le C_{3,\max}\\
cas_{j,min} \le cas_j \le cas_{j,max}\\
TVD_{min} \le TVD \le TVD_{max}\\
AMI_{min} \le D_k + TVD_1 + TVD_2 \le AMI_{max}\\
DS_{min} \le D_k + TVD_1 + TVD_2 + TVD_3 \le DS_{max}\\
D_i > 0 (i=1,2,3,4,5)
\end{cases}
\end{aligned}
\tag{4-28}
$$

式中，$f_1(x)$和$f_2(x)$代表两个目标：井眼轨道实际长度TMD和实际扭矩大小T。其中，$TMD = D_k + HD + \sum_{i=1}^{5} D_i$，$D_k$为造斜点处轨道测量长度，$D_i(i = 1,2,3,4,5)$为各段轨道的实际长度；$T = \sum_{i=1}^{7} T_i$，$T_i(i = 1,2,3,4,5,6,7)$为各段轨道的实际扭矩大小。在约束条件中，$C_1$、$C_2$和$C_3$分别表示第一、第二和第三个套管设置的深度；$cas_j$为第$j$个套管深度，其中$j = 1,2,3$；$AMI_{min}$和$AMI_{max}$分别为稳斜段结束深度的上限和下限；$TVD_1$、$TVD_2$和$TVD_3$分别为第一、第二和第三段的深度；$DS_{min}$和$DS_{max}$分别为降斜段结束深度的上限和下限。

根据实际的井眼轨道优化过程的思想，井眼轨道的长度越短越好，并且控制扭矩也要尽可能地小。所制定的优化模型的目标存在冲突，比如较短的井道长度会导致在该点处有更深的踢腿和更高的狗腿严重程度，最终增加扭矩。因此，有必要系统地分析影响井道的诸多因素，以达到最佳的技术和经济条件。

4.4 实验结果与分析

在本研究中，第一个目标$f_1(x)$代表井筒长度TMD，第二个目标$f_2(x)$代表扭

矩T。通过最小化井筒长度和扭矩而设计的井筒可能比其他轨迹更快、更便宜，而且这两个目标相互冲突。

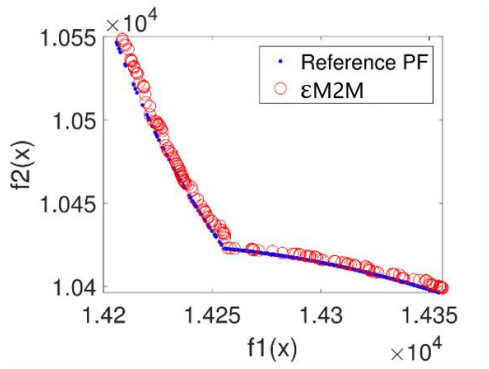
采用εM2M、CCMO[39]、CMOEA-MS[45]、cDPEA[61]、ToP[46]、MOEA/D-CDP[72]、CM2M2[18]和 CM2M[36]等八种算法对井眼轨迹优化问题进行了测试。其中每个算法的参数设置与上述实验中保持一致，仍然使用 HV 和 IGD 两个指标来评估 8 种 CMOEAs 在井眼轨迹优化问题上的性能。

表 4-1 列出了εM2M 和另外 7 种 CMOEAs 在井眼轨迹优化问题上取得的结果。很明显，εM2M 在 8 种算法中获得了最好的效果。每个 CMOEA 所获得的非支配解绘制在图 4-8 中。如图 4-8 所示，εM2M 总体上取得了更好的效果，CM2M 在井筒轨迹优化问题上与εM2M 具有相似的收敛性和多样性。ToP 具有良好的收敛性，但多样性较差，它只收敛于 PF 的一部分。对于 CMOEA-MS 和 cDPEA，它们获得的非支配解只找到了 PF 的一小部分，这反映了两种算法的多样性较差。对于 CCMO，该算法得到的解没有收敛到真实的 PF 上。表现最差的是 MOEA/D-CDP 和 CM2M2，在实验过程中，我们观察到 MOEA/D-CDP 和 CM2M2 在不同的分区中甚至没有获得可行非支配解。这可能是因为种群在搜索过程中陷入了不可行区域，无法跨出，从而导致算法性能不佳。

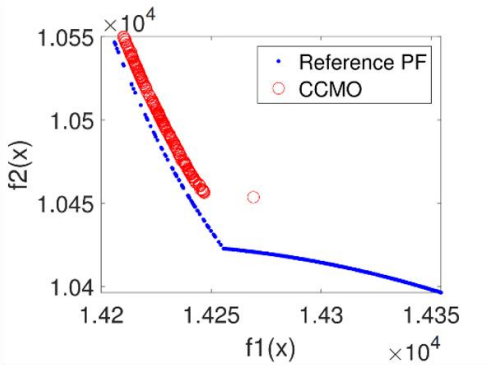
表 4-1 εM2M 和其他 7 个算法在井眼轨迹优化问题上的 HV 和 IGD 结果。

Test Instance		εM2M	CCMO	CMOEA-MS	cDPEA	ToP	MOEAD-CDP	CM2M2	CM2M
HV	mean	**6.81E+06**	6.63E+06	6.74E+06	6.61E+06	6.79E+06	6.45E+06	6.68E+06	6.68E+06
	std	8.70E+05	1.50E+05	1.41E+05	1.30E+05	3.07E+05	1.23E+06	1.92E+05	2.27E+05
IGD	mean	**11.616**	56.954	26.366	60.735	21.639	332.27	88.512	51.492
	std	29.696	45.349	42.314	39.102	15.471	1304.3	46.065	67.016

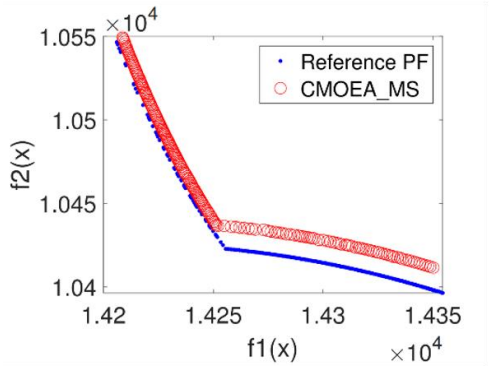
图 4−9 画出了εM2M 算法设计的井眼轨迹图。参考 PF 由每个算法所得到的非支配解组成。D 轴、N 轴和 E 轴分别表示深度、向北和向东方向上坐标。s1、s2 和 s3 代表三条轨迹。S1 对应井眼长度$f_1(x)$最小时的轨迹，S2 对应扭矩$f_2(x)$最小时的轨迹，S3 对应 PF 拐点处的轨迹。图 4-10~图 4-12 分别画出了三条井眼轨迹的正面图、俯视图和左视图。



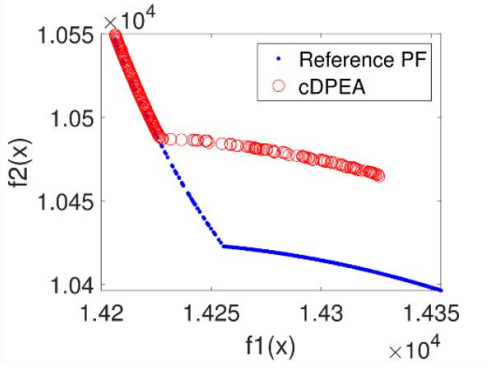
（a） εM2M



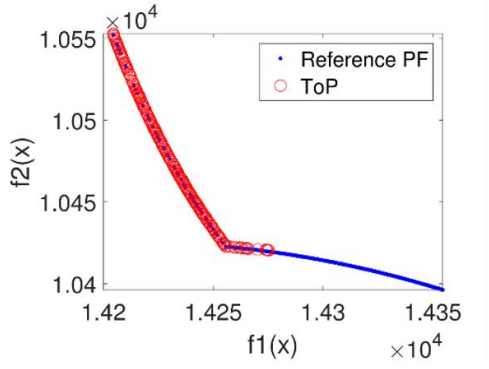
（b）CCMO



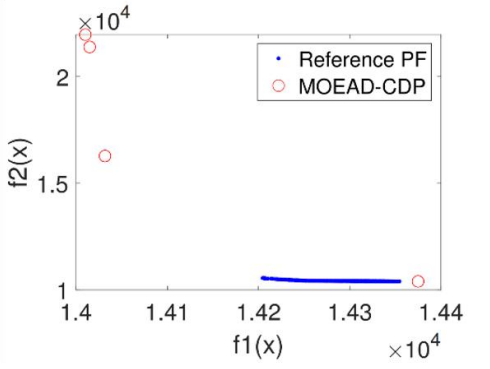
（c）CMOEA-MS



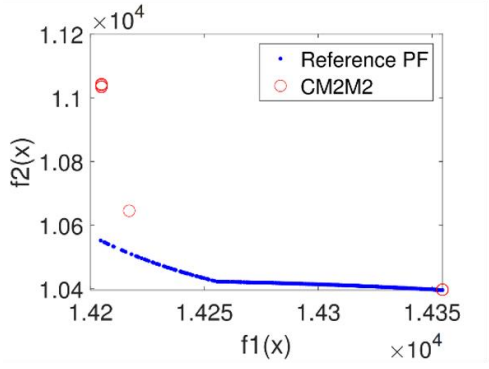
（d）cDPEA



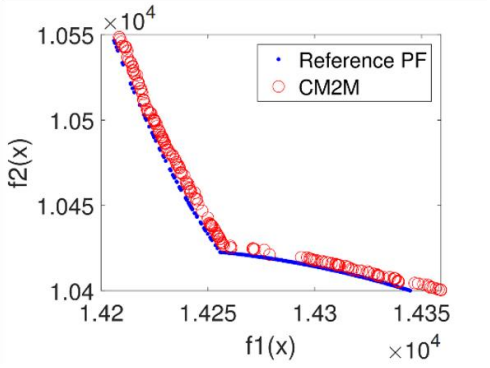
（e）ToP



（f）MOEA/D-CDP



（g）CM2M2



（h）CM2M

图 4-8 各算法对井筒轨迹优化问题的非支配解绘制在(a)-(h)中。参考 PF 由每个算法所得到的非支配解组成。

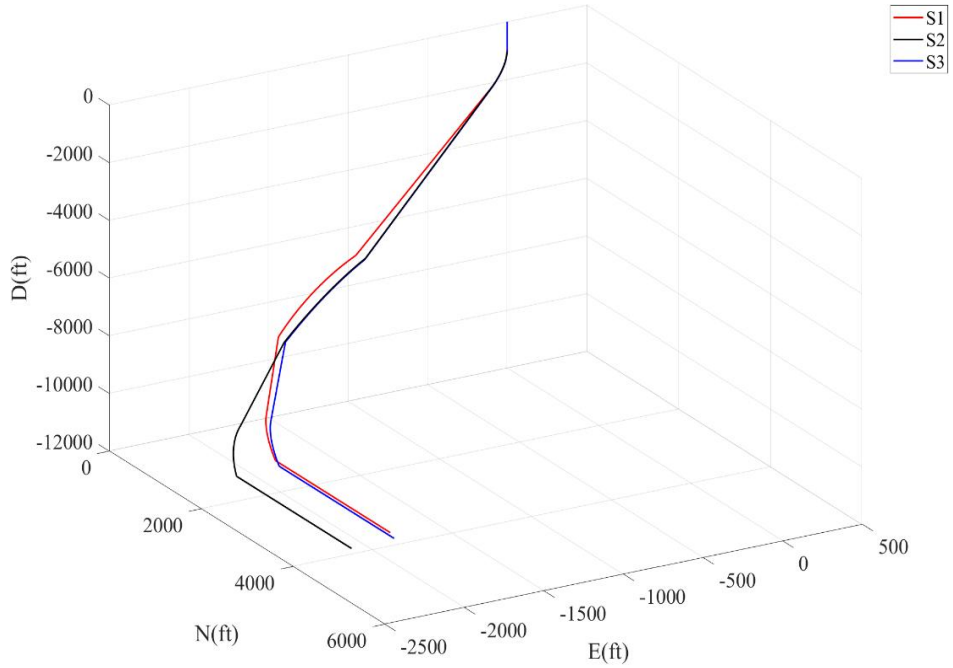


图 4-9 εM2M 算法设计的井眼轨迹空间视角。s1、s2、s3 分别对应井眼轨道长度最短、扭矩最小和 PF 拐点处的三条轨迹。

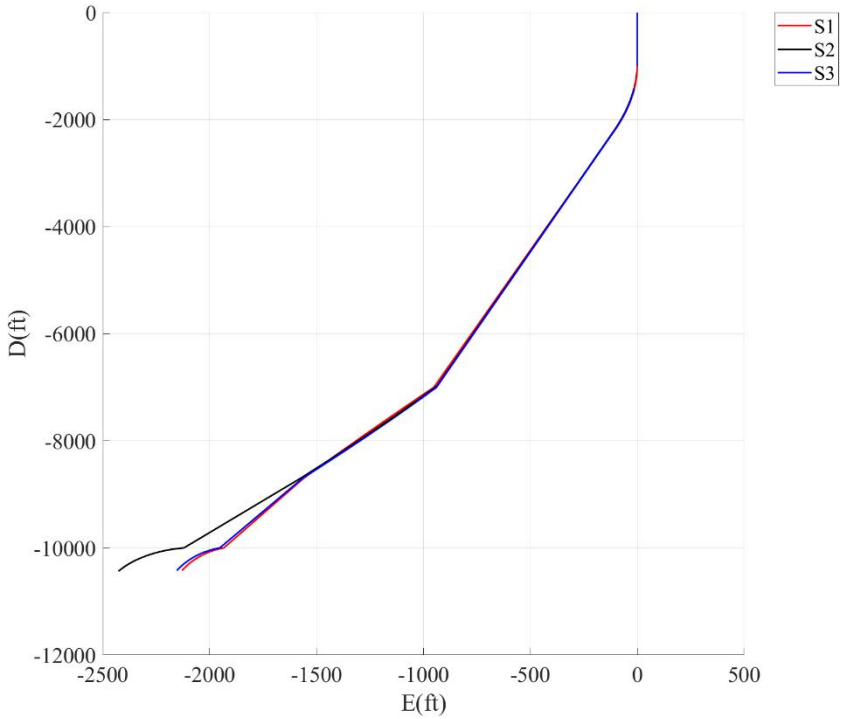


图 4-10 井眼轨迹正面视角

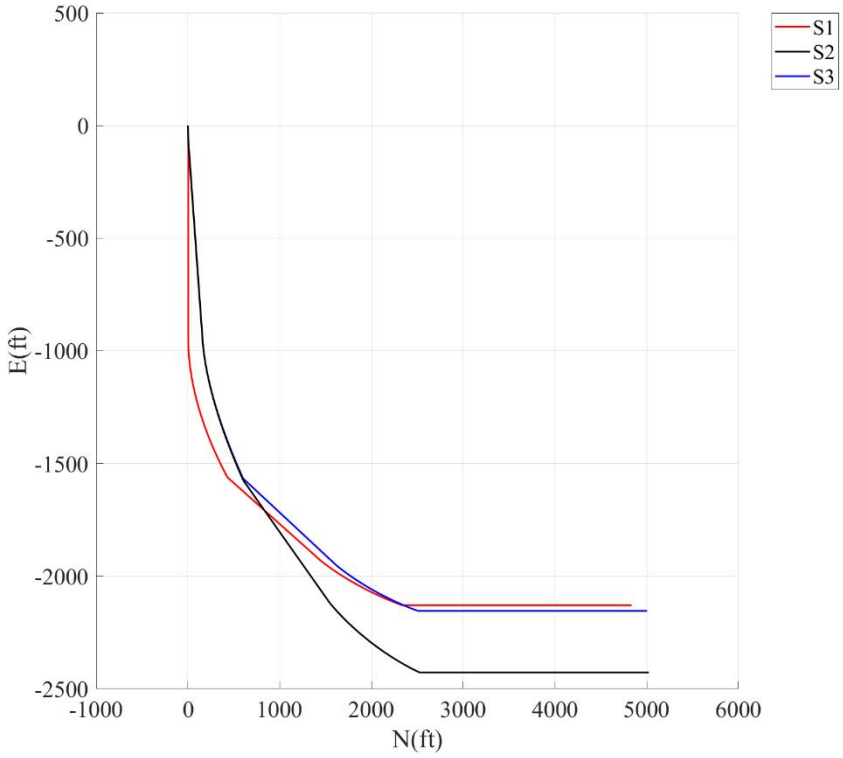


图 4-11 井眼轨迹俯视图

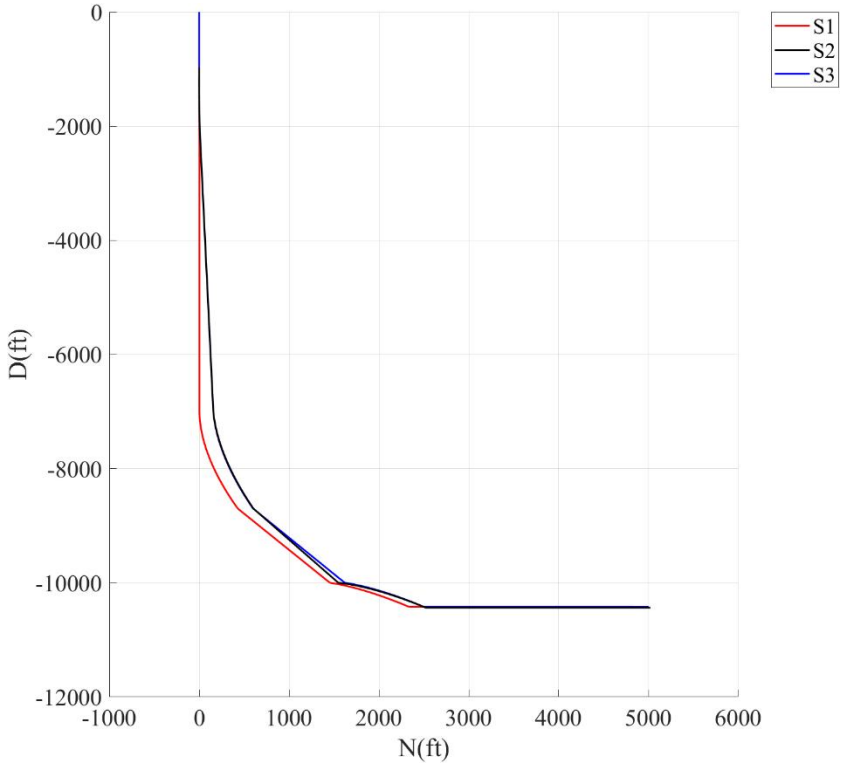


图 4-12 井眼轨迹左视图

第 5 章 总结与展望

5.1 研究总结

现实中存在的许多 CMOPs 都具有约束或者目标不平衡的特征，解决此类不平衡的问题比一般的约束多目标优化问题更具有挑战性，在本研究中，我们提出了一种有效的混合算法εM2M，它可以有效地处理目标或者约束不平衡的 CMOPs。该算法在基于分解的基础上融入改进后的ε约束处理机制，增强了种群在整个进化过程中的收敛性和多样性。具体来说，εM2M 算法首先通过基于分解的策略将一个完整的种群分解为一组子种群，每个子种群都被用来解决相应的子问题。这种方法在一定程度上保证了进化过程中种群的多样性。此外，使用改进的约束处理技术来增强种群的收敛性，帮助种群跨越大的不可行区域从而能够快速地收敛到真实的帕累托前沿。换句话说，我们通过动态调整ε的值来控制约束的松弛，保留了一些不可行解的信息，从而增强了种群的收敛性。两种机制的融合使得算法的性能得到显著提升，种群的多样性和收敛性也因此得到改善，在处理不平衡的约束多目标优化问题上有很大突破。同时提出了一组具有不平衡 CMOPs 特性的约束多目标测试问题集（IM-CMOPs），该测试问题不仅有着不平衡的目标，同时包含收敛性和多样性困难的约束条件。使用εM2M 和 8 种经典算法对 IM-CMOPs 和另一组现有的不平衡测试问题集（UCMOPs）进行测试，综合实验表明，εM2M 在处理不平衡测试问题上的效果要显著优于另外 8 种算法。

此外，井眼轨迹的设计是钻井工程的基础，直接影响到整个施工的过程。然而，油井的轨迹设计是一个复杂的模型。在设计和优化过程中，只考虑最小轨迹长度或井深的设计目标，并不一定会设计出扭矩最小、目标误差最小的井道模型。最小的轨迹长度或井深往往会导致复杂的轨迹结构和较大的目标误差，从而导致钻井过程中的事故和误差。因此，本文设计了一种新的井眼轨迹优化模型，该模型考虑了不同约束条件下的井道长度、扭矩等相互冲突的目标。并将提出的εM2M 算法运用到井眼轨迹的设计与优化问题中。通过实验对比其他算法，发现本文提出的算法在工程优化问题上也取得了不错的效果。

5.2 研究展望

大多数用于井眼轨迹优化模型的算法都是单目标或多目标优化算法。这些算

法可以很容易地为单目标或多目标优化模型找到一个具有代表性的帕累托最优解集。然而，对于高维目标优化模型，随着目标空间维数的增加，总体中非支配解的比例显著增加。尽管εM2M 方法在处理不平衡的多目标优化问题上取得了良好的效果，但本文主要研究对象是两个目标，在更多目标的复杂问题上仍然存在一定的局限性，因此下一步研究准备将问题扩展到高维，开发更具有效性和高效性的算法。此外，除了进化算法，还可以将不平衡多目标优化问题与其他技术结合，如神经网络、深度学习等，以提高算法的鲁棒性和性能。将不平衡多目标优化算法和技术应用到更多实际问题中，如智能交通、金融投资等领域，以提高生产效率和社会效益。另外实际问题的特点往往比较复杂，而现有的研究多集中于简单的问题模型。因此，未来的研究需要更多考虑实际问题的特点，如不确定性、动态性等，以提高算法的适应性和实用性。总之，不平衡的约束多目标优化问题是一个具有挑战性和应用前景的研究方向，未来的研究需要结合实际问题的特点，开发更加有效性和高效性的算法，并将其应用于更多领域中。

此外，考虑研究大规模变量的融合机器学习的约束多目标进化算法。实际工程领域中，还存在一类优化问题，其特点是决策变量的数量特别多，如神经网络参数的优化，大规模调度问题等等。如何采用机器学习的办法对大规模的变量进行合理有效的分类，以降低整个搜索空间大小并提高搜索效率是一个值得深入研究的课题。

参考文献

[1] Y Wu, H Ding, B Xiang, et al. Evolutionary Multitask Optimization in Real-World Applications: A Survey[J]. Journal of Artificial Intelligence and Technology, 2023, 3(1): 32-38.

[2] C Peng, H L Liu, and E D Goodman. Handling multi-objective optimization problems with unbalanced constraints and their effects on evolutionary algorithm performance[J]. Swarm and Evolutionary Computation, 2020, 55: 100676.

[3] 郑金华, 邹娟. 多目标进化优化[M]. 北京: 科学出版社, 2017.

[4] J Liang, X Ban, K Yu, et al. A Survey on Evolutionary Constrained Multiobjective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2023, 27(2): 201-221.

[5] Gang C, Dan Z. Novel Adaptive Simulated Annealing Algorithm for Constrained Multi-Objective Optimization[J]. China Communications, 2012, 9(9): 68-78.

[6] F Ming, W Gong, L Wang, et al. A tri-population based co-evolutionary framework for constrained multi-objective optimization problems[J]. Swarm and Evolutionary Computation, 2022, 70: 101055.

[7] H Alsouly, M Kirley, and M A Muñoz. An Instance Space Analysis of Constrained Multi-Objective Optimization Problems[J]. IEEE Transactions on Evolutionary Computation, 2022, 1-12.

[8] Q Deng, Q Kang, L Zhang, et al. Objective Space-Based Population Generation to Accelerate Evolutionary Algorithms for Large-Scale Many-Objective Optimization[J]. IEEE Transactions on Evolutionary Computation, 2023, 27(2): 326-340.

[9] C Wang, J Zhao, L Li, et al. A Multi-Transformation Evolutionary Framework for Influence Maximization in Social Networks[J]. IEEE Computational Intelligence Magazine, 2023, 18(1): 52-67.

[10] J Liang, X Ban, K Yu, e al. A survey on evolutionary constrained multi-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2022, 27(2):201-221.

[11] W Gao, W Xu, M Gong, et al. A decomposition-based evolutionary algorithm using an estimation strategy for multimodal multi-objective optimization[J]. Information Sciences: An International Journal, 2022, 606: 531-548.

[12] T Zhang, F Li, X Zhao, et al. A convolutional neural network-based surrogate model for multi-objective optimization evolutionary algorithm based on decomposition[J]. Swarm and Evolutionary Computation, 2022, 72: 101081.

[13] Z Xia, Y Liu, J Lu, et al. Penalty method for constrained distributed quaternion-variable optimization[J]. IEEE Transactions on Cybernetics, 2020, 51(11): 5631-5636.

[14] Z Ma and Y Wang. Shift-based penalty for evolutionary constrained multiobjective optimization and its application[J]. IEEE Transactions on Cybernetics, 2021, 53(1): 18-30.

[15] B C Wang, H X Li, J P Li, et al. Composite differential evolution for constrained evolutionary optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 49(7): 1482-1495.

[16] C Wang and R Xu. An angle based evolutionary algorithm with infeasibility information for constrained many-objective optimization[J]. Applied Soft Computing, 2020, 86: 105911.

[17] C Segura, C A C Coello, G Miranda, et al. Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization[J]. Annals of Operations Research, 2016, 240: 217-250.

[18] C Peng, H L Liu, and F Gu. An evolutionary algorithm with directed weights for constrained multi-objective optimization[J]. Applied Soft Computing, 2017, 60: 613-622.

[19] Z Fan, W Li, X Cai, et al. Push and pull search for solving constrained multi-objective optimization problems[J]. Swarm and evolutionary computation, 2019, 44: 665-679.

[20] H Zhang, K Tao, L Ma, et al. Handling Constrained Multi-Objective optimization with Objective Space Mapping to Decision Space Based on Extreme Learning Machine. in 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2020: 1-7.

[21] V Kelner, F Capitanescu, O Léonard, et al. A hybrid optimization technique coupling an evolutionary and a local search algorithm[J]. Journal of Computational and Applied Mathematics, 2008, 215(2): 448-456.

[22] A Lara, L Uribe, S Alvarado, et al. On the choice of neighborhood sampling to build effective search operators for constrained MOPs[J]. Memetic Computing, 2019, 11: 155-173.

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

[24] K Qiao, J Liang, K Yu, et al. Self-adaptive resources allocation-based differential evolution for constrained evolutionary optimization[J]. Knowledge-Based Systems, 2022, 235: 107653.

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

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

[27] Z Fan, H Li, C Wei, et al. An improved epsilon constraint handling method embedded in MOEA/D for constrained multi-objective optimization problems. in 2016 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2016: 1-8.

[28] Y Yang, J Liu, and S Tan. A constrained multi-objective evolutionary algorithm based on decomposition and dynamic constraint-handling mechanism[J]. Applied Soft Computing, 2020, 89: 106104.

[29] D A Vieira, R L Adriano, J A Vasconcelos, et al. Treating constraints as

objectives in multiobjective optimization problems using niched Pareto genetic algorithm[J]. IEEE Transactions on Magnetics, 2004, 40(2): 1188-1191.

[30] T Ray, H K Singh, A Isaacs, et al. Infeasibility driven evolutionary algorithm for constrained optimization[J]. Constraint-handling in evolutionary optimization, 2009: 145-165.

[31] A Isaacs, T Ray, and W Smith. Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems. in 2008 IEEE Congress on Evolutionary Computation. IEEE, 2008: 2780-2787.

[32] Q Long. A constraint handling technique for constrained multi-objective genetic algorithm[J]. Swarm and Evolutionary Computation, 2014, 15: 66-79.

[33] D Chafekar, J Xuan, and K Rasheed. Constrained multi-objective optimization using steady state genetic algorithms. in Genetic and Evolutionary Computation—GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, 2003 Proceedings, Part I. 2003. Citeseer.

[34] J Wang, G Liang, and J Zhang. Cooperative differential evolution framework for constrained multiobjective optimization[J]. IEEE transactions on cybernetics, 2018, 49(6): 2060-2072.

[35] H L Liu and D Wang. A constrained multiobjective evolutionary algorithm based decomposition and temporary register. in 2013 IEEE Congress on Evolutionary Computation. IEEE, 2013: 3058-3063.

[36] H L Liu, C Peng, F Gu, et al. A constrained multi-objective evolutionary algorithm based on boundary search and archive[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2016, 30(01): 1659002.

[37] Y Yang, J Liu, and S Tan. A partition-based constrained multi-objective evolutionary algorithm[J]. Swarm and Evolutionary Computation, 2021, 66: 100940.

[38] B Liu, H Ma, X Zhang, et al. A memetic co-evolutionary differential evolution algorithm for constrained optimization. in 2007 IEEE Congress on Evolutionary Computation. IEEE, 2007: 2996-3002.

[39] Y Tian, T Zhang, J Xiao, et al. A coevolutionary framework for constrained multiobjective optimization problems[J]. IEEE Transactions on Evolutionary

Computation, 2020, 25(1): 102-116.

[40] K Li, R Chen, G Fu, et al. Two-archive evolutionary algorithm for constrained multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 23(2): 303-315.

[41] J Wang, Y Li, Q Zhang, et al. Cooperative multiobjective evolutionary algorithm with propulsive population for constrained multiobjective optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, 52(6): 3476-3491.

[42] Z Z Liu, B C Wang, and K Tang. Handling constrained multiobjective optimization problems via bidirectional coevolution[J]. IEEE Transactions on Cybernetics, 2021, 52(10): 10163-10176.

[43] Z Fan, Z Wang, W Li, et al. Push and pull search embedded in an M2M framework for solving constrained multi-objective optimization problems[J]. Swarm and Evolutionary Computation, 2020, 54: 100651.

[44] H Wang, T Cai, K Li, et al. Constraint handling technique based on Lebesgue measure for constrained multiobjective particle swarm optimization algorithm[J]. Knowledge-Based Systems, 2021, 227: 107131.

[45] Y Tian, Y Zhang, Y Su, et al. Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization[J]. IEEE Transactions on Cybernetics, 2021, 52(9): 9559-9572.

[46] Z Z Liu and Y Wang. Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(5): 870-884.

[47] Y Xiang, X Yang, H Huang, et al. Balancing constraints and objectives by considering problem types in constrained multiobjective optimization[J]. IEEE Transactions on Cybernetics, 2021, (99): 1-14.

[48] K Yu, J Liang, B Qu, et al. Purpose-directed two-phase multiobjective differential evolution for constrained multiobjective optimization[J]. Swarm and Evolutionary Computation, 2021, 60: 100799.

[49] F Ming, W Gong, H Zhen, et al. A simple two-stage evolutionary algorithm

for constrained multi-objective optimization[J]. Knowledge-Based Systems, 2021, 228: 107263.

[50] S Datta, A Ghosh, K Sanyal, et al. A radial boundary intersection aided interior point method for multi-objective optimization[J]. Information Sciences, 2017, 377: 1-16.

[51] L Uribe, A Lara, and O Schütze. On the efficient computation and use of multi-objective descent directions within constrained MOEAs[J]. Swarm and Evolutionary Computation, 2020, 52: 100617.

[52] O Cuate, A Ponsich, L Uribe, et al. A New Hybrid Evolutionary Algorithm for the Treatment of Equality Constrained MOPs[J]. Mathematics, 2020, 8(1): 7.

[53] O Schütze, L Uribe, and A Lara. The gradient subspace approximation and its application to bi-objective optimization problems. in Advances in Dynamics, Optimization and Computation: A volume dedicated to Michael Dellnitz on the occasion of his 60th birthday. 2020: 355-390.

[54] S Dutta and K N Das. A survey on pareto-based eas to solve multi-objective optimization problems[J]. Soft Computing for Problem Solving: SocProS 2017, Volume 2, 2019: 807-820.

[55] A Trivedi, D Srinivasan, K Sanyal, et al. A survey of multiobjective evolutionary algorithms based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2016, 21(3): 440-462.

[56] J G Falcón-Cardona and C A C Coello. Indicator-based multi-objective evolutionary algorithms: A comprehensive survey[J]. ACM Computing Surveys (CSUR), 2020, 53(2): 1-35.

[57] Z Ma, Y Wang, and W Song. A new fitness function with two rankings for evolutionary constrained multiobjective optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 51(8): 5005-5016.

[58] Z Fan, Y Fang, W Li, et al. MOEA/D with angle-based constrained dominance principle for constrained multi-objective optimization problems[J]. Applied Soft Computing, 2019, 74: 621-633.

[59] Y Feng, L Feng, S Kwong, et al. A multivariation multifactorial evolutionary

algorithm for large-scale multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2021, 26(2): 248-262.

[60] Z Ma and Y Wang. Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(6): 972-986.

[61] M Ming, A Trivedi, R Wang, et al. A dual-population-based evolutionary algorithm for constrained multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2021, 25(4): 739-753.

[62] Z Fan, W Li, X Cai, et al. Difficulty adjustable and scalable constrained multiobjective test problem toolkit[J]. Evolutionary computation, 2020, 28(3): 339-378.

[63] E Zitzler, M Laumanns, and L Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm[J]. TIK-report, 2001, 103.

[64] H L Liu, F Gu, and Q Zhang. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems[J]. IEEE transactions on evolutionary computation, 2013, 18(3): 450-455.

[65] H L Liu, L Chen, K Deb, et al. Investigating the effect of imbalance between convergence and diversity in evolutionary multiobjective algorithms[J]. IEEE Transactions on Evolutionary Computation, 2016, 21(3): 408-425.

[66] K Deb, L Thiele, M Laumanns, et al. Scalable multi-objective optimization test problems. in Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600). IEEE, 2002, 1: 825-830.

[67] R Cheng, M Li, Y Tian, et al. A benchmark test suite for evolutionary many-objective optimization[J]. Complex & Intelligent Systems, 2017, 3: 67-81.

[68] K Deb, A Pratap. Constrained test problems for multi-objective evolutionary optimization. in Evolutionary Multi-Criterion Optimization: First International Conference, 2001 Proceedings 1. Springer Berlin Heidelberg, 2001: 284-298.

[69] Q Zhang, A Zhou, et al. Multiobjective optimization test instances for the CEC 2009 special session and competition[J]. University of Essex, Colchester, UK and Nanyang technological University, special session on performance assessment of

multi-objective optimization algorithms, technical report, 2008, 264: 1-30.

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

[71] Z Fan, W Li, X Cai, et al. An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions[J]. Soft Computing, 2019, 23: 12491-12510.

[72] K Deb. An efficient constraint handling method for genetic algorithms[J]. Computer Methods in Applied Mechanics and Engineering, 2000, 186(2-4): 311-338.

[73] P A Bosman and D Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(2): 174-188.

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

[75] 刘刚军，陈宇飞，吴家坤. 探讨钻井工程技术现状及发展趋势[J]. 中国石油和化工标准与质量, 2022, 42(10): 146-148.

[76] J Zheng, C Lu, and L Ga. Multi-objective cellular particle swarm optimization for wellbore trajectory design[J]. Applied Soft Computing, 2019, 77: 106-117.

[77] D A Wood. Hybrid bat flight optimization algorithm applied to complex wellbore trajectories highlights the relative contributions of metaheuristic components[J]. Journal of Natural Gas Science and Engineering, 2016, 32: 211-221.

[78] K Biswas, P M Vasant, J A G Vintaned, et al. A review of metaheuristic algorithms for optimizing 3D well-path designs[J]. Archives of Computational Methods in Engineering, 2021, 28: 1775-1793.

[79] V Mansouri, R Khosravanian, D A Wood, et al. 3-D well path design using a multi objective genetic algorithm[J]. Journal of Natural Gas Science and Engineering, 2015, 27: 219-235.

攻读学位期间主要研究成果

已投稿论文

[1] Zhun Fan, Liu Wang, Zhaojun Wang, Biao Xu and Wenji Li*. A Decomposition-based Evolutionary Multi-objective Optimization Method for Solving Constrained Optimization Problems with Imbalanced Objectives or Constraints, 外审中. (Natural Computing，第 3 章相关)

[2] Zhun Fan*, Zhaojun Wang, Liu Wang, Wenji Li, Guijie Zhu, Biao Xu, Dong Chen and Deli Gao. Well Trajectory Design Based on Constrained Many-objective Optimization Algorithms, 外审中. (Engineering Applications of Artificial Intelligence，第 4 章相关)

致谢

凤凰花开，骊歌轻响，又到一朝离别时。感谢这段岁月中帮助过我的老师、同学、家人和朋友们。正是有了你们的支持和帮助，我才能够成长，并收获许多宝贵的经验和人生财富。

在汕大求学期间，我在学习和成长方面有了很大的进步。首先由衷感谢我的导师范衡老师，您是一位学识渊博、敢于创新的学者，在科研上始终态度严谨，精益求精。感谢您在学习上的指导和生活上的关怀，您的支持让我能够克服迷茫，不断前行。

其次，感谢李文姬老师和王诏君师兄，二位不仅帮助我确定了科研方向，培养了科研的基本技能，还在论文分析和修改方面给予了我很多的指导，三年来无论是学业上的挫折还是生活中的问题都离不开二位的帮助，谢谢你们。感谢杨知师姐和周彬师兄在学习和生活上对我的照顾，怀念你们在校的那些日子。感谢我的室友和同门，还有 240 的小伙伴们，为三年的求学生涯带来了很多快乐。感谢我的朋友，在我最无助迷茫的时候给予我精神上的鼓励和无条件支持，爱你们。

最后，感谢我的父母和家人。你们的爱是支持我前进的最大动力。

作者：王柳

2023 年 4 月 1 日