Knee Point-Based Imbalanced Transfer Learning for Dynamic Multiobjective Optimization

Jiang, Min and Wang, Zhenzhong and Hong, Haokai and Yen, Gary G.

摘要

动态多目标优化问题 (Dynamic multiobjective optimization problems，DMOPs) 是具有多个冲突优化目标的优化问题，这些目标随着时间的推移而变化。基于迁移学习的方法已被证明是可行的，然而，缓慢的求解速度是这些方案的主要障碍之一。运行速度慢的原因之一是低质量的个体占据了大量的计算资源，这些个体可能会导致负迁移。将高质量的个体（例如拐点）与迁移学习相结合是该问题的可行解决方案。然而，高质量个体的数量通常非常小，因此很难使用传统的迁移学习方法获得实质性的改进。在本文中，我们提出了一种基于拐点的迁移学习方法，称为 KT-DMOEA，用于解决 DMOP。在所提出的方法中，开发了一个趋势预测模型 (trend prediction model,TPM) 来生成估计的拐点。然后，提出了一种不平衡迁移学习方法，通过使用这些估计的拐点来生成高质量的初始种群。这种方法的优势在于，通过无缝集成少量高质量个体和不平衡迁移学习技术，可以极大地提高计算效率，同时保持解决方案的质量。与选择的一些最先进的算法进行的实验结果和性能比较表明，所提出的设计能够显著提升动态优化的性能。

关键词： 领域自适应；进化动态多目标优化；拐点；预测；迁移学习；

1 引言

当涉及到 DMOPs 时，这类问题包含多个相互冲突的优化目标，这些目标随着时间推移而变化。许多工程问题都可以归结为动态多目标优化问题。例如，对于机器人运动规划 [1]，需要路径规划算法根据当前任务（如最小化能耗和最大化稳定性）和机器人在动态环境中的运行来持续生成最优的动作。另一个显著的例子是工业生产中的动态调度 [2]；决策者需要调整策略以应对不断变化的电力需求、老化设备或故障子系统等环境，以同时实现提高生产效率、降低成本和减少能源消耗等竞争目标。因此，高效、有效地解决动态多目标优化问题具有重要的现实意义。

多年来，已经提出了各种方法来解决动态多目标优化问题 [3]。例如，静态多目标优化算法 (static multiobjective optimization algorithms，SMOAs) [4] 可以直接用于解决动态多目标优化问题，当检测到环境变化时，该方法简单地重置种群。然而，种群重置方法可能会无意中导致搜索过程偏离正确的方向，因此这类算法的效率通常较低。另一方面，多种群方法 [5–7] 已经被引入到进化算法（evolutionary algorithms，EAs）中，以确保种群具有足够的多样性。

例如，Jin 等人 [8] 提出了一种基于参考点的策略，将种群划分为多个子种群，以增强种群的多样性。多样性确实是解决动态多目标优化问题的关键，但仅通过使用多个种群来增强多样性的方法仍然效率较低。例如，当发生较大变化时，生成多个种群仍然是一个主要障碍。

近年来，基于预测的动态多目标优化算法引起了广泛关注。这些方法通过利用不同的机器学习技术来预测不断变化的环境状态，从而使算法能够提前适应变化。然而，大多数基于预测的方法都假设“过去”和“未来”状态是相关的，这可能在处理较慢或微小变化的动态时是有效的。然而，在发生更剧烈变化时，预测模型往往表现不佳，难以达到最初的预测目标。

基于迁移学习 [9–11] 的预测模型被认为是很有潜力的。这种方法的优点在于它们可以利用现有的“经验”来获取高质量的初始种群。例如，在 [9] 和 [10] 中，引入了域自适应方法来预测新环境下的 Pareto 最优前沿（Pareto optimal front，POF），大量的实验结果证实了所获得的解的质量。然而，现有的方法通常需要较长的训练时间，这是某些动态多目标优化问题的主要障碍之一。运行速度慢的原因之一是现有方法在环境选择过程中没有仔细选择个体，这不仅会导致大量计算资源被浪费在低质量的个体上，而且大大增加了负迁移的可能性 [12]。因此，我们认为，如果能够将那些拥有高质量个体的知识进行迁移（从收敛性和多样性的角度），就可以为动态多目标优化问题建立更高效、更准确的预测模型，从而应对各种实际复杂情境的挑战。

2 相关工作

近年来，在求解 DOMPs 方面取得了许多进展，现有的大多数方法可以分为三大类:

2.1 基于多样性维护的方法

当检测到更改时，多样性维护方法倾向于通过使用某种技术向解决方案添加变化。例如，在 [13] 中提出的 SGEA 方法中，参数 用于控制被随机创建的解替换的种群部分。每次发生更改时，重新初始化的种群由 50% 旧解、η% 随机生成和 $(50-\eta)$% 预测引导解组成，以提高种群多样性。DNSGA-II 由 Deb 等人提出的 [14]，也是一种保持多样性的方法。DNSGA-II 有两个版本，称为 DNSGA-II-A 和 DNSGA-II-B。在第一个版本中，人口被一些随机创建的个体解决方案所取代，而在第二个版本中，通过用突变的解决方案替换一部分人口来增强多样性。

Jin 等人 [8] 提出了一种基于参考点的策略。该方法根据参考点将种群划分为几个子种群。同一子种群的中心用于预测新环境下同一子种群的中心，增加区域内解的多样性。Yazdani 等人 [15] 提出了一种用于解决大规模动态优化问题的协同协同进化多种群框架。该框架将大规模动态优化问题分解为一组低维分量，并控制预算分配给组件来跟踪多个移动最优值。曾等人 [16] 提出了一个一个通用框架，称为 DCMOEA，用于解决动态约束优化问题。DCMOEA 通过减少约束边界来处理约束难度。Chen 等人 [17] 开发了一种动态双存档 EA (DTAEA)，用于处理目标数量变化 dops。DTAEA 同时维护两个亚群。一个子种群专注于为 POS 提供有竞争力的选择压力，而另一个子种群保持有希望的多样性。

2.2 基于记忆的方法

基于记忆的方法使用额外的存储来隐式或显式地记录先前生成的有用信息，以指导未来的搜索。这些方法可以分为多个子类。

例如，Goh 和 Tan [18] 提出了一种协同进化的多目标算法。该算法将种群划分为不同的种群子集，每个子集代表多目标优化问题的一个子组件。各子群之间进行竞争以找到最佳代表，最终获胜者合作进化以获得更优解。

Azzouz 等人 [19] 则提出了一种自适应的混合种群管理策略。该策略利用记忆、局部搜索和随机策略来有效处理动态多目标优化问题中的环境变化。特别之处在于该算法根据环境变化的严重程度调整要使用的记忆和随机解的数量，以适应不同情况下的问题动态性。

2.3 基于预测的方法

基于预测的方法 [20]、 [21] 跟踪移动最优解引起了越来越多的兴趣，并提出了各种基于预测的方法。Zhou 等人 [22] 提出了一种基于种群的预测策略 (PPS)。在 PPS 中，使用自回归模型来预测种群。建立了一个预测模型来预测 POS 中心。一旦预测了新的 POS 中心，POS 流形就会直接转移到新的 POS 模型中。Rong 等人 [23] 提出了一种多方向预测策略来预测移动 POS 的位置。在该方法中，采用聚类算法将种群划分为不同的组，根据环境变化的强度调整聚类的数量。Mururanantham 等人 [24] 提出了一种基于卡尔曼滤波的预测动态多目标 EA (MOEA/DKF)。当识别出环境变化时，卡尔曼滤波器用于通过生成新的初始种群来引导搜索到新的 POS。彭等人 [25] 提出了一个探索算子和利用算子来预测新的最优解。在预测策略中，采用了重用先前确定的精英解决方案的最佳解决方案保存机制。Ruan 等人 [26] 提出了一种混合多样性维护方法，称为预测和记忆策略 (PMS)，以提高预测精度。基于中心点运动方向，PMS 使用预测模型重新定位了许多接近新 POS 的解决方案。Gee 等人 [27] 提出了基于逆建模的多目标算法 (IM-MOEA/D) 来求解 dops。IMMOEA/D 使用逆模型来引导搜索走向有希望的决策区域。Bu 等人 [28] 开发了一种基于动态物种的粒子群优化算法，并提出了一种定位和跟踪可行区域策略的集合来处理约束中的不同类型的动态。Gong 等人 [29] 为动态区间多目标优化问题提供了一个协同协同进化优化框架。该策略根据每个决策变量和区间参数之间的区间相似度将决策变量分为两组，并利用这两个子种群协同优化决策变量。Rong 等人 [30] 提出了一种多模型预测方法 (MMP)，该方法检测变化类型，然后选择合适的预测模型生成初始种群。

基于迁移学习的方法可以被认为是预测方法的一个分支。Jiang 等人 [9] 指出，不同环境下的解分布可能并不总是独立且相同 (IID)，因此提出了一种基于迁移学习的 Tr-DMOEA 框架来预测求解 dops 的有效初始种群。Tr-DMOEA 的基本思想是将过去环境中的 POF 映射到潜在空间，然后使用这些映射的解决方案构建一个种群，用于在新的环境下搜索 POF。基于迁移学习的方法确实显着提高了解决方案的质量，但需要解决的这种方法的一个问题是如何在提高解决方案质量的同时加快搜索过程。如上所述，现有方法在“平庸个体”上花费了大量的计算资源，大量的转移增加了负迁移的可能性。因此，我们相信如果我们可以转移“高质量”个体所拥有的知识，我们可以获得更有效的 DMOP 算法。

3 本文方法

3.1 区域拐点

拐点是非支配解的子集，如果没有其他特定于问题或特定于用户的偏好，POF 中拐点的选择作为解决方案自然是首选。这些个体具有更高的超体积 (HV) 值。研究 [31] 表明，当一组解完全支配另一组解时，前者的 HV 大于后者的 HV。同时，HV 指数可以同时衡量解集的收敛性和多样性。

在 [32] 中提出了一种寻找拐点的简单方法，该方法通过计算点到极值线或超平面的距离来确定一个点是否为拐点。对于双目标最小化问题，如图 1 所示，极值线 L 定义如下:

$$L : a \cdot f_1 + b \cdot f_2 + c = 0 \tag{1}$$

其中 a、b 和 c 可以由非支配集中的两个极值点确定。因此，从解 S(xs, ys) 到极值线 L 的距离可以计算为:

$$d(S, L) = \begin{cases} \frac{|ax_s + by_s + c|}{\sqrt{a^2+b^2}}, & ax_s + by_s + c < 0 \\ -\frac{|ax_s + by_s + c|}{\sqrt{a^2+b^2}}; & \text{otherwise.} \end{cases} \tag{2}$$

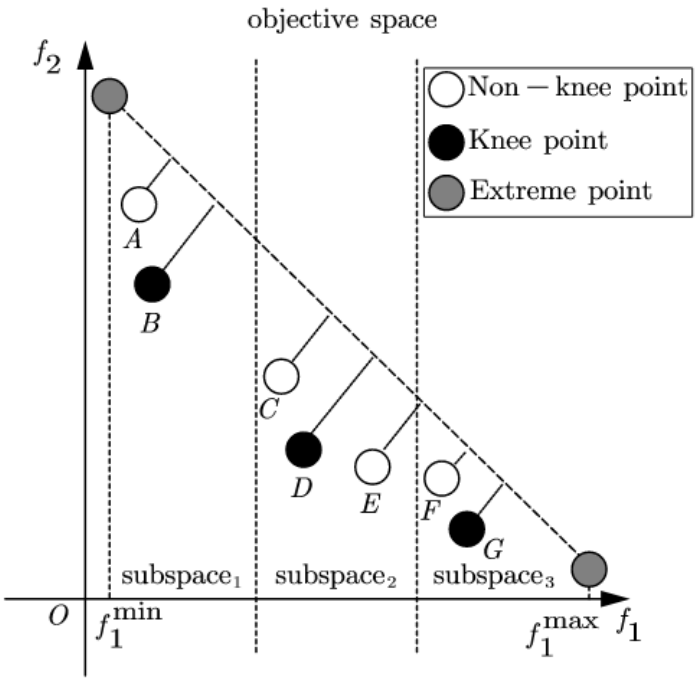

图 1. 对于双目标最小化问题，点 D 与极值点 L 的距离最大，因此被选为拐点。目标空间分为三个子空间，因此有三个区域拐点 B、D 和 G。

拐点表示非支配解集中与极值线 L 距离最大的解。在图 1中，点 D 与 L 的距离最大，因此该点被认为是当前种群下的全局拐点。该算法将目标空间划分为几个子空间，每个子空间都有一个各自的拐点，即区域拐点。然后，我们将区域拐点输入 TPM，得到新环境下估计的拐点，使这些估计的拐点可以作为目标域的样本来辅助迁移学习。

为了获得区域拐点，我们随机选择第 m 个目标函数，并将整个目标空间均匀地分成 p 个子空间。假设在时刻 t，第 m 个函数的最大值为 $f_{m,t}^{\max}$，最小值为 $f_{m,t}^{\min}$，那么在时刻 t，每个

子空间的大小为

$$\text{size}_{m,t,i} = \frac{f_{m,t}^{\max} - f_{m,t}^{\min}}{p}. \tag{3}$$

每个子空间的下界可以计算为

$$LB_{m,t,i} = f_{m,t}^{\min} + (i-1) \times \text{size}_{m,t} \tag{4}$$

而上界则根据以下公式计算：

$$UB_{m,t,i} = f_{m,t}^{\min} + i \times \text{size}_{m,t} \tag{5}$$

其中 $i = 1, \ldots, p$。

一旦我们确定了每个子空间，就可以确定该子空间的区域拐点。简单地说，我们使用 Das 方法 [32] 来寻找离极值线最远的点，图 1给出了这种方法的一个简单例子。在此示例中，目标空间分为三个子空间。点 B、D 和 G 分别是来自极值线 L 的三个子空间中最远的点，因此这三个点被视为区域拐点。

3.2 估计拐点

在我们的方法中，估计的拐点由 TPM 基于这些区域拐点得到，图 2显示了 TPM 方法的示意图。

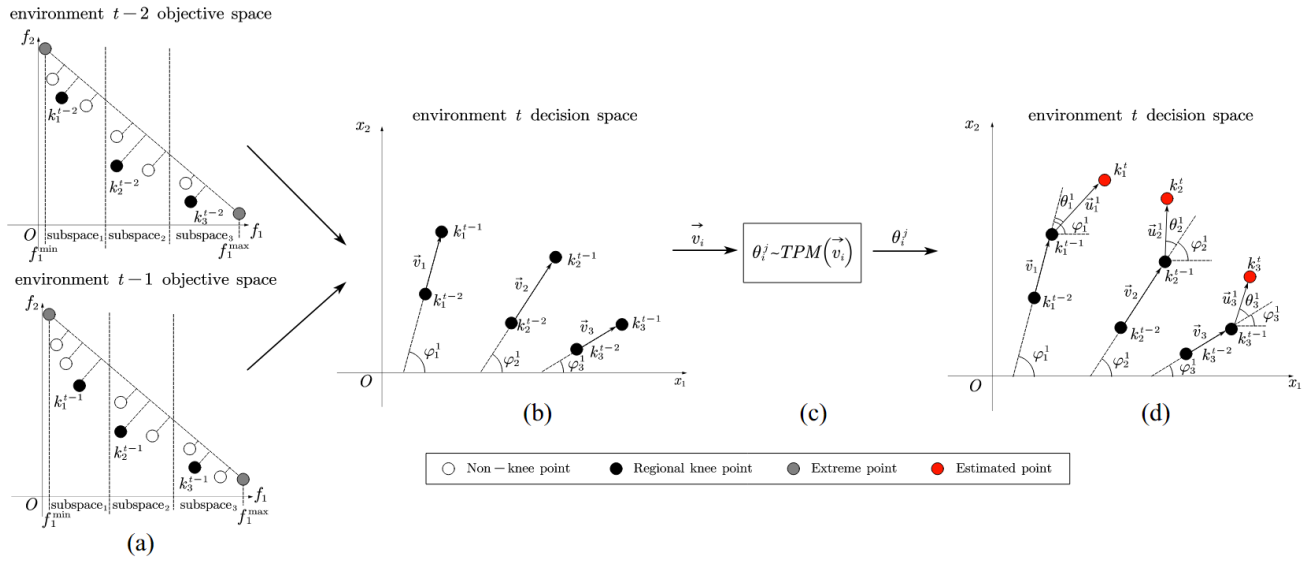

图 2. (a) 顶部和底部子图分别是 $t-2$ 和 $t-1$ 时刻的目标空间。每个目标空间被均匀地划分为三个子空间。对于每个子空间，识别一个区域拐点。例如，k_1^{t-1} 表示在 $t-1$ 时刻的子空间 $subspace_1$ 中的区域拐点。(b) 根据两个连续环境中的区域拐点计算运动向量 $\vec{v}_i = k_i^{t-1} - k_i^{t-2}$。(c) 对于给定的运动向量 $\vec{v}_i$，使用 TPM 计算样本偏转角度 θ_i^j。(d) 使用公式 (10) 生成估计的拐点。

这意味着径向坐标 r 等于 $|\vec{v}|$。向量 $\vec{v}_i$ 的第 j 个相关角度计算如下：

$$\varphi_i^j = \arctan\left(\frac{\sqrt{\sum_{d=j+1}^{n}\left(\vec{v}_i^d\right)^2}}{\vec{v}_i^j}\right) \tag{6}$$

其中 $\vec{v}_i^j$ 表示向量 $\vec{v}_i$ 的第 j 个分量，n 表示决策变量的维数。

在第三步中，对于给定的运动向量 $\vec{v}_i$，所提出的算法为每个角度坐标生成多个偏转角度，然后基于以下概率密度函数选择具有最大概率的偏转角度：

$$\theta_i^j \sim \mathrm{TPM}\left(\vec{v}_i\right) = \frac{e^{-\operatorname{sign}\left(\theta_i^j\right)\cdot\frac{\theta_i^j}{|\vec{v}_i|}}}{\int_{-\pi}^{0} e^{\frac{\theta_i^j}{|\vec{v}_i|}} d\theta_i^j + \int_{0}^{\pi} e^{-\frac{\theta_i^j}{|\vec{v}_i|}} d\theta_i^j} \tag{7}$$

这里 θ_i^j 是一个偏转角度，$\vec{v}_i$ 是运动向量。我们认为对于特定的 DMOP，拐点很可能朝着某个方向移动的概率很高。在这个方程中，0° 的偏转角方向是拐点最有可能移动的方向。因此，靠近 0° 的偏转角具有较高的概率值。

在第四步 [图 2(d)] 中，当我们获得偏转角度时，可以通过以下方式计算时刻 t 子空间 i 的估计拐点的位置 k_i^t：

$$k_i^t = k_i^{t-1} + \vec{u}_i \tag{8}$$

其中 k_i^{t-1} 是时刻 $t-1$ 子空间 i 的估计拐点。向量 $\vec{u}_i$ 的计算公式如下：

$$\vec{u}_i^j = \begin{cases} r\cos\left(\varphi_i^1 + \theta_i^1\right), & j = 1 \\ r\prod_{d=1}^{j-2}\sin\left(\varphi_i^d + \theta_i^d\right)\cos\left(\varphi_i^{j-1} + \theta_i^{j-1}\right), & 1 < j < n \\ r\prod_{d=1}^{j-1}\sin\left(\varphi_i^d + \theta_i^d\right), & j = n \end{cases} \tag{9}$$

其中，$\vec{u}_i^j$ 表示向量 $\vec{u}_i$ 的第 j 个分量，θ_i^j 分别代表第 j 个对应的相关角度和偏转角度。n 表示决策变量的维数。

3.3 基于拐点的不平衡迁移学习

不同环境下拐点的分布可能有很大差异，因此仅使用 TPM 很难准确预测下一时刻的拐点。因此，我们建议使用迁移学习技术，称为 TrAdaboost [33]，以提高预测准确性。TrAdaboost 的基本思想是利用增强学习技术来过滤源域数据中与目标域数据不相似的数据。该方法使用基础分类器获取一组弱分类器。每个弱分类器将样本 X 映射到标签 $Y(X) \in 0, 1$，其中 $X = X_{\text{source}} \cup X_{\text{target}}$。这些弱分类器的输出被组合成一个强分类器。

然而，与其他类型的解决方案相比，例如非支配解和支配解，拐点的数量非常少，因此这个问题被认为是典型的不平衡学习问题。当数据不平衡时，少数样本很可能被弱分类器误分类，导致源域中权重的快速收敛。因此，在训练的下一个迭代中，这些少数解对模型训练的影响很小，因此预测模型对少数样本的识别率非常低。因此，直接使用 TrAdaboost 预测新环境中的拐点并不合适。

我们提出了一种解决这个问题的不平衡版本的 TrAdaboost 方法。该方法利用上一个时间步的获取的部分优势解（POS）和一些随机生成的解来形成源域 X_{so}，将新环境的估计拐点和一些随机生成的解视为目标域 X_{ta}。然后，通过迭代地使用分类器从 $D = x, y(x)$ 中学习一系列弱分类器 $h_1, \ldots, h_{N_{\max}}$，其中 $x \in X_{ta} \cup X_{so}$，$y(\cdot)$ 将拐点标记为 1，非拐点标记为 0。

为了防止少数类样本权重值快速减少，我们从一开始就为源域中的拐点设置更大的初始

权重，同时设置源域和目标域样本的权重如下：

$$w_1(x) = \begin{cases} \frac{1}{p}, & \text{if } x \in X_{so} \text{ and } y(x) = 1 \\ \frac{1}{n-p}, & \text{if } x \in X_{so} \text{ and } y(x) = 0 \\ \frac{1}{m}, & \text{if } x \in X_{ta} \end{cases} \tag{10}$$

其中，p 是子空间的数量，n 和 m 分别是源域和目标域中样本的数量。

引入因子 σ 来创建更新样本权重的规则，σ 可以表示为：

$$\sigma = \frac{c_k \cdot I_k + 0.001}{c_{nk} \cdot I_{nk} + 0.001} \tag{11}$$

这里，c_{nk} 和 c_k 分别代表源域中非拐点和拐点误分类的成本，I_k 和 I_{nk} 分别是拐点和非拐点的误分类数量。

通过 σ 更新样本权重的基本思想是，在训练集中只有少数拐点。当数据不平衡时，少数样本很可能被弱分类器误分类，导致源域中权重的快速收敛。为解决此问题，引入了 σ 因子。当少数类的误分类成本大于多数类时，σ 设定为大于 1，这样乘以 σ 因子可以防止少数类样本权重值的快速减少，并提高预测的性能。在这项研究中，我们设定 c_{nk} 等于 1，$c_k = ((|X_{so}| - p) \cdot c_{nk}) / p$，其中 p 是子空间的数量。

源域和目标域样本的权重可以根据以下规则进行更新：

$$w_{i+1}(x) = \begin{cases} w_i \sigma \beta^{|h_i(x) - y(x)|}, & x \in X_{so} \text{ and } y(x) = 1 \\ w_i \beta^{|h_i(x) - y(x)|}, & x \in X_{so} \text{ and } y(x) = 0 \\ w_i \beta_i^{-|h_i(x) - y(x)|}, & x \in X_{ta} \end{cases} \tag{12}$$

其中，

$$\beta = 1 / \left(1 + \sqrt{2 \ln |X_{so}| / N_{\max}}\right) \tag{13}$$

β_i 是弱分类器 h_i 的错误率，$N_{\max}$ 是最大迭代次数。β_i 可以计算为

$$\beta_i = \frac{\epsilon_i}{1 - \epsilon_i} \tag{14}$$

其中，ϵ_i 是目标域中的加权错误率

$$\epsilon_i = \sum_{x \in X_{ta}} \frac{w_i(x) \cdot |h_i(x) - y(x)|}{\sum_{x \in X_{ta}} w_i(x)}. \tag{15}$$

每当更新样本权重时，我们获得一个新的弱分类器 h_i。当训练完成后，我们可以将 $N_{\max}$ 个弱分类器组合成一个强分类器 $h(\cdot)$：

$$h(\cdot) = \text{sign}\left(\sum_{i=1}^{N_{\max}} \ln\left(1/\beta_i\right) h_i(\cdot)\right). \tag{16}$$

$h(\cdot)$ 可以预测解是否为拐点。我们可以随机生成大量解，并将这些解输入到 $h(\cdot)$ 进行分类。

3.4 整体算法

在初始阶段，种群经过随机初始化，然后通过基于种群的 SMOA 进行优化，得到对应的 POS。

当环境发生变化时，我们识别了最近两个环境的拐点。基于这两个环境中拐点的运动趋势，TPM 用于估计新环境中的拐点。$K_{estimated}$ 中的预估拐点标记为 1，而一些随机支配的非拐点 Pt 被标记为 0。接着，将 $K_{estimated}$ 和 P_t 合并到目标域训练集 X_{ta} 中。同样地，最近一个环境中的随机支配的非拐点 P_{t-1} 和拐点 k_{t-1} 合并到源域训练集 X_{so} 中。

接下来，拐点不平衡转移技术预测了一些新的拐点 $K_{predicted}$。这些预测的拐点 $K_{predicted}$ 可以引导搜索到真正的 Pareto 最优前沿（POF），然后利用基于种群的 SMOA 算法对 $K_{predicted}$ 进行优化。需要注意的是，由于拐点的数量远远小于种群大小，因此种群中的其他个体是通过向 $K_{predicted}$ 中添加高斯噪声生成的。

Algorithm 1 KT-DMOEA

Input: The Dynamic Optimization Function $F_t(\cdot)$; A SMOA; The number of subspaces p. · The Pareto Optimal Set (POS) of $F_t(\cdot)$ at different moments.

Initialization
while the environment has changed **do**
  $t = t + 1$
  if $t == 1$ or $t == 2$ **then**
    Initialize randomly the population $initPop$
    $POS_t = SMOA(F_t(\cdot), initPop)$
    Generate randomly dominated solutions Pt
  else
    Identify regional knee points k_{t-2} and k_{t-1} in POS_{t-2} and POS_{t-1}, respectively
    $K_{estimated} = TPM(F_t(\cdot), k_{t-2}, k_{t-1}, p)$
    Generate randomly dominated solutions Pt
    $X_{ta} = K_{estimated} \cup Pt$
    $X_{so} = k_{t-1} \cup Pt_{-1}$
    $K_{predicted} = KPIT(X_{so}, X_{ta}, p)$
    $POS_t = SMOA(F_t(\cdot), K_{predicted})$
  end if
end while
return POS_t

4 复现细节

4.1 与已有开源代码对比

本论文并没有开源代码，其中 MOEA/D 模块参考了现在已有的代码，请点击这里访问 MOEA/D 示例网站，其余部分由自己独立实现。

将论文里的模块例如 TPM，KPIT（Knee Points Imbalanced Transfer）一一实现，并且最后使用自己实现的测试问题对这个算法进行了测试。

4.2 各模块实现

4.2.1 获取区域拐点

首先通过 3.1 提到的方法分割子空间，然后使用 Das 方法 [32] 来寻找区域拐点吗，具体实现如图 3所示。

```
73  function [kneeF,kneeS]=getKneeGroup(Pareto,partNum)
74      [boundaryS,boundaryF]=getBoundary(Pareto.X,Pareto.F);
75      [posArr,pofArr]=partition(Pareto.X,Pareto.F,partNum,boundaryF);
76      for partNo=1:partNum
77          [kneeS,kneeF]=getKnees(posArr{partNo},pofArr{partNo});
78          kneeSArr{partNo}=kneeS;
79          kneeFArr{partNo}=kneeF;
80      end
81      kneeS=cell2mat(kneeSArr);
82      kneeF=cell2mat(kneeFArr);
83  end
```

图 3. 区域拐点 matlab 实现

4.2.2 TPM 实现

具体实现基本参考 3.2 节提到的算法，matlab 代码如图 4所示。

```
function pop = TPM(kneeArray,T)
    vec=kneeArray{T-1}-kneeArray{T-2};
    step=vec+normrnd(0,0.1);
    for j=1:size(vec,2)
        for i=1:size(vec,1)-1
            kjl=kneeArray{T-1}(i+1:end,j);
            fenzi=sqrt(sum(kjl.^2));
            pietheta(i)=atan(fenzi/kneeArray{T-1}(i,j));
        end
    end
    num=1;
    while num<11
        theta=ones(1,size(vec,1)-1).*(randi([-1,1],1,1)/10990);
        fi= pietheta+ theta;
        for i=1:size(vec,1)
            if i==1
                u(i)=vec(i)*cos(fi(i));
            elseif i<size(vec,1)
                temp=1;
                for j=1:size(vec,1)-2
                    temp=temp*sin(fi(j));
                end
                u(i)=vec(i)*temp*cos(fi(i));
            else
                temp=1;
                for j=1:size(vec,1)-1
                    temp=temp*sin(fi(j));
                end
                u(i)=vec(i)*temp;
            end
        end

        samplePop(:,num)=u;
         num=num+1;
    end
    testppp=vec;
     pop= mod(abs(kneeArray{T-1}+samplePop(:,10)),1);

pop1=mod(abs(kneeArray{T-1}+step.*(kneeArray{T-1}-kneeArray{T-2})),1);
end
```

图 4. TPM 实现

4.2.3 KPIT 实现

KPIT 具体实现基本参考 3.3 节提到的算法，matlab 代码如图 5所示。

```
        for checkp=1:n
            if checkp>length(predict)||checkp>length(tdY)
                break;
            end

            if predict(checkp)~=tdY(checkp) && tdY(checkp)==1
                Ik=Ik+1;
            end
            if predict(checkp)~=tdY(checkp) && tdY(checkp)==-1
                Ink=Ink+1;
            end
        end
        sigama=(ck*Ik+0.001)/(cnk*Ink+0.001);
        sW = sum(w(n+1:m+n));
        et = sum(w(n+1:m+n).*(predict(n+1:m+n)~=tsY)/sW);
        if et >= 0.5
            et = 0.499;
        elseif et == 0
            et = 0.001;
        end
        bT = et/(1-et);
        beta(t) =bT;
        b = 1/(1+sqrt(2*log(n/T)));
        wUpdate = [(b*ones(n,1)).*sigama.^(predict(1:n)~=tdY) ; (bT*ones(m,1)).^(-(predict(n+1:m+n)~=tsY)) ];
        w = w.*wUpdate;
    end
```

图 5. KPIT 实现

4.3 创新点

较为完整的复现算法的各个部分，并且实验结果与文章中的结果基本相同。另外对实验的参数如 n_t 以及 τ_t 也进行了调整并对比实验，对子区域的个数也进行了实验，探究子区域数量 p 对实验结果的影响，最后将其他的测试问题对此算法进行了测试。

另外也将算法中的 SVM 更换为集成决策树与 SVM 的集成模型，通过训练集成决策树和 SVM 两个弱分类器，将它们集成为一个强分类器，最后效果也显示良好。

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

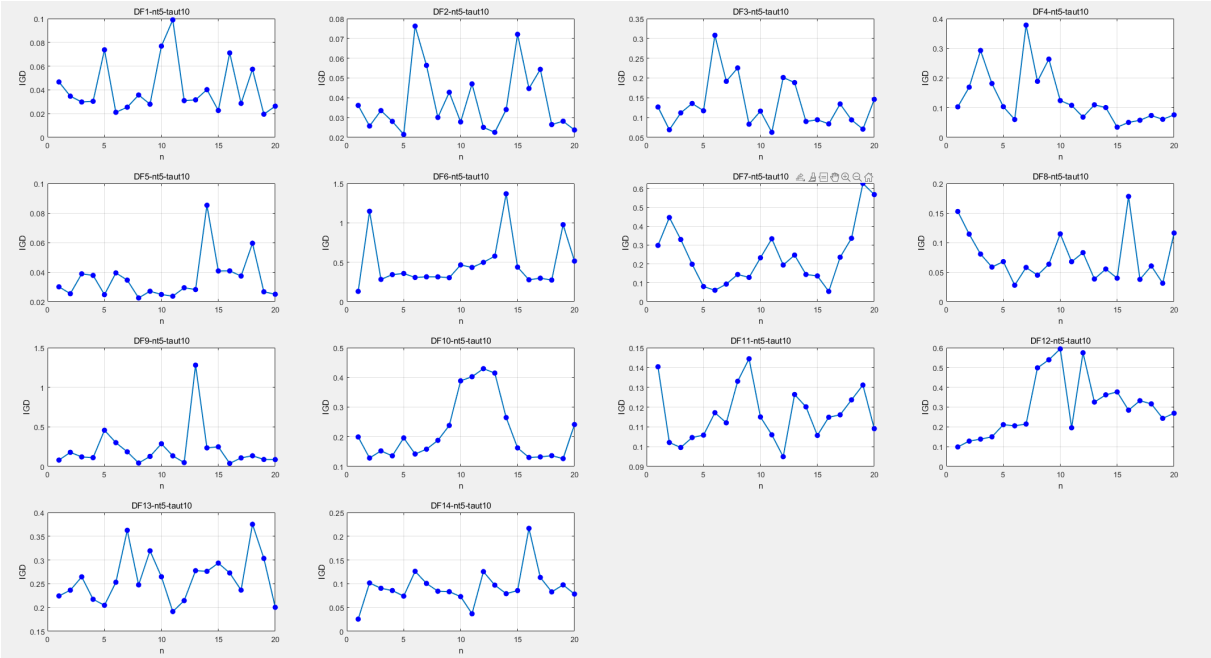

图 6. IGD 随环境的变化

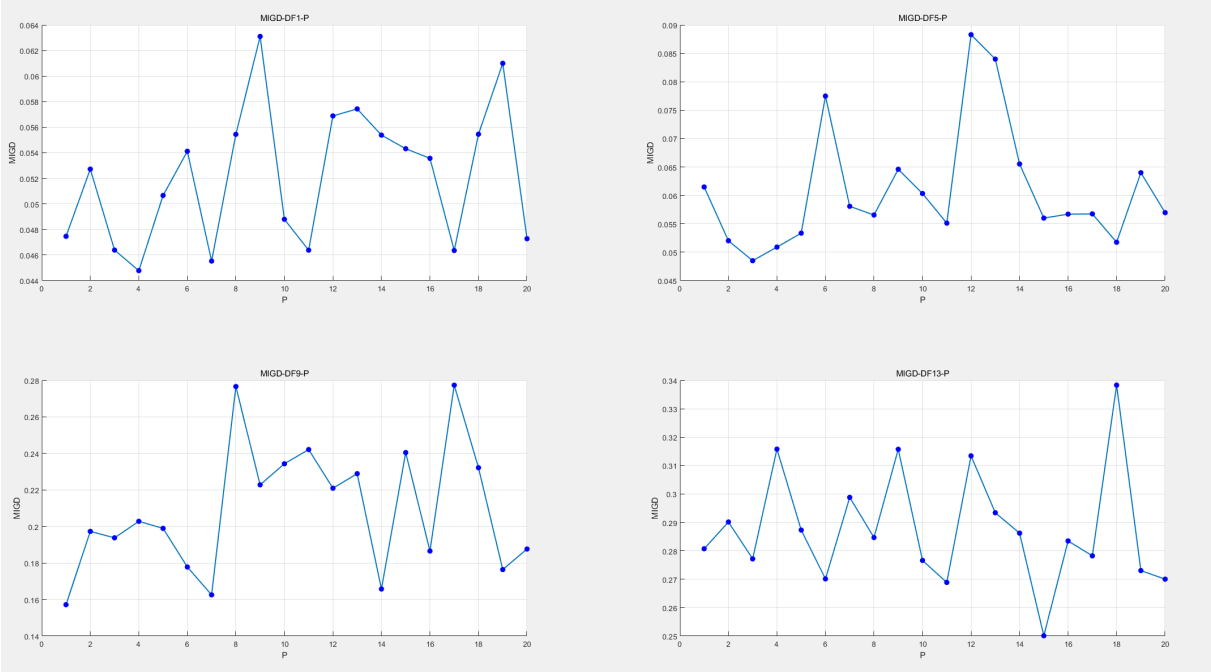

图 7. MIGD 随子区域大小的变化

与原论文不同的是，原论文默认将子区域的大小设置为 10，后续进行测试也是直接用 10，但是通过实验发现，分割子区域在 3 的时候性能最优，因此后续的实验步骤都是使用 3。

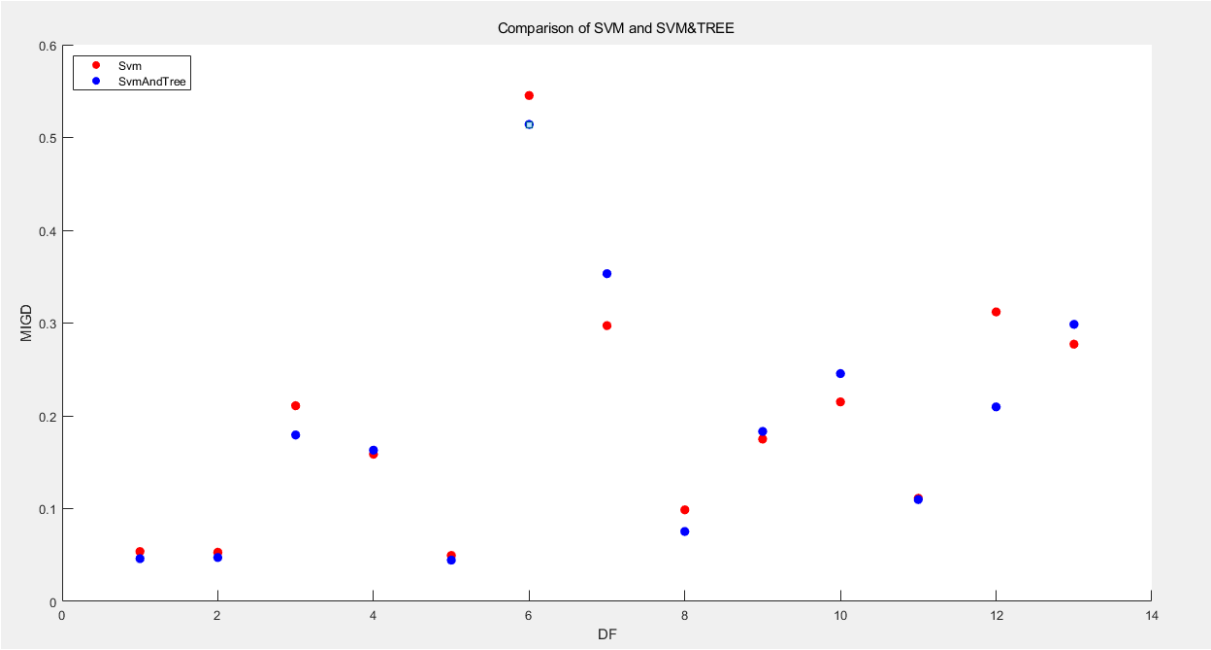

图 8. SVM 与 SVM 结合决策树在 DF 测试集上的性能比较

其中，蓝色点表示集成模型，红色点表示 SVM，从图中可以看出在 DF 测试集上，集成模型有八个指标优于 SVM，对性能有一定的提升。

6 总结与展望

利用迁移学习技术求解 DMOPs 已被证明是一种有效的方法。然而，现有方法往往较慢，运行速度慢的原因之一是低质量的个体消耗大量的计算资源，这些个体的性能不佳可能会导致负迁移。因此我们需要找到并利用高质量的个体，如拐点。本文提出了一种基于拐点的不平衡迁移学习方法 KT-DMOEA 来解决 DMOPs 问题。该方法将整个决策空间划分为几个区域，预测模型 TPM 为每个子区域生成一个估计的拐点。但是因为拐点的数量非常少。为了解决这个问题，我们提出了一种不平衡的迁移学习方法 KT-DMOEA，通过调整解决方案的权重来关注训练类不平衡问题。大量实验表明，所提出的预测方法比一些最先进算法性能更优。在未来的工作中，我们将继续探索如何减少负迁移并提出更有效的 DMOP 算法，同时探究能否引入深度学习来替换其中的机器学习模型部分。

参考文献

[1] M. Jiang, Z. Huang, G. Jiang, M. Shi, and X. Zeng, “Motion generation of multi-legged robot in complex terrains by using estimation of distribution algorithm,” in *2017 IEEE symposium series on computational intelligence (SSCI)*, pp. 1–6, IEEE, 2017.

[2] W. Du, W. Zhong, Y. Tang, W. Du, and Y. Jin, “High-dimensional robust multi-objective optimization for order scheduling: A decision variable classification approach,” *IEEE transactions on industrial informatics*, vol. 15, no. 1, pp. 293–304, 2018.

[3] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[4] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 838–856, 2015.

[5] Z.-H. Zhan, X.-F. Liu, H. Zhang, Z. Yu, J. Weng, Y. Li, T. Gu, and J. Zhang, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 704–716, 2016.

[6] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE transactions on evolutionary computation*, vol. 20, no. 4, pp. 590–605, 2015.

[7] Y. Wang, Z.-Z. Liu, J. Li, H.-X. Li, and G. G. Yen, "Utilizing cumulative population distribution information in differential evolution," *Applied Soft Computing*, vol. 48, pp. 329–346, 2016.

[8] Y. Jin, C. Yang, J. Ding, and T. Chai, "Reference point based prediction for evolutionary dynamic multiobjective optimization," in *2016 IEEE congress on evolutionary computation (CEC)*, pp. 3769–3776, IEEE, 2016.

[9] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 501–514, 2017.

[10] M. Jiang, L. Qiu, Z. Huang, and G. G. Yen, "Dynamic multi-objective estimation of distribution algorithm based on domain adaptation and nonparametric estimation," *Information Sciences*, vol. 435, pp. 203–223, 2018.

[11] A. T. W. Min, Y.-S. Ong, A. Gupta, and C.-K. Goh, "Multiproblem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 15–28, 2017.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[13] S. Jiang and S. Yang, "A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization," *IEEE Transactions on evolutionary Computation*, vol. 21, no. 1, pp. 65–82, 2016.

[14] K. Deb, U. B. Rao N, and S. Karthik, "Dynamic multi-objective optimization and decision-making using modified nsga-ii: A case study on hydro-thermal power scheduling," in *International conference on evolutionary multi-criterion optimization*, pp. 803–817, Springer, 2007.

[15] D. Yazdani, M. N. Omidvar, J. Branke, T. T. Nguyen, and X. Yao, "Scaling up dynamic optimization problems: A divide-and-conquer approach," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 1–15, 2019.

[16] S. Zeng, R. Jiao, C. Li, X. Li, and J. S. Alkasassbeh, "A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization," *IEEE transactions on Cybernetics*, vol. 47, no. 9, pp. 2678–2688, 2017.

[17] R. Chen, K. Li, and X. Yao, "Dynamic multiobjectives optimization with a changing number of objectives," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 157–171, 2017.

[18] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 103–127, 2008.

[19] R. Azzouz, S. Bechikh, and L. B. Said, "A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy," *Soft Computing*, vol. 21, pp. 885–906, 2017.

[20] M. Jiang, W. Hu, L. Qiu, M. Shi, and K. C. Tan, "Solving dynamic multi-objective optimization problems via support vector machine," in *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pp. 819–824, IEEE, 2018.

[21] J. Zhou, J. Zou, S. Yang, G. Ruan, J. Ou, and J. Zheng, "An evolutionary dynamic multiobjective optimization algorithm based on center-point prediction and sub-population autonomous guidance," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2148–2154, IEEE, 2018.

[22] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE transactions on cybernetics*, vol. 44, no. 1, pp. 40–53, 2013.

[23] M. Rong, D. Gong, Y. Zhang, Y. Jin, and W. Pedrycz, "Multidirectional prediction approach for dynamic multiobjective optimization problems," *IEEE transactions on cybernetics*, vol. 49, no. 9, pp. 3362–3374, 2018.

[24] A. Muruganantham, K. C. Tan, and P. Vadakkepat, "Evolutionary dynamic multiobjective optimization via kalman filter prediction," *IEEE transactions on cybernetics*, vol. 46, no. 12, pp. 2862–2873, 2015.

[25] Z. Peng, J. Zheng, J. Zou, and M. Liu, "Novel prediction and memory strategies for dynamic multiobjective optimization," *Soft Computing*, vol. 19, pp. 2633–2653, 2015.

[26] G. Ruan, G. Yu, J. Zheng, J. Zou, and S. Yang, "The effect of diversity maintenance on prediction in dynamic multi-objective optimization," *Applied Soft Computing*, vol. 58, pp. 631–647, 2017.

[27] S. B. Gee, K. C. Tan, and C. Alippi, "Solving multiobjective optimization problems in unknown dynamic environments: An inverse modeling approach," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4223–4234, 2016.

[28] C. Bu, W. Luo, and L. Yue, "Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 14–33, 2016.

[29] D. Gong, B. Xu, Y. Zhang, Y. Guo, and S. Yang, "A similarity-based cooperative co-evolutionary algorithm for dynamic interval multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 142–156, 2019.

[30] M. Rong, D. Gong, W. Pedrycz, and L. Wang, "A multimodel prediction method for dynamic multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 290–304, 2019.

[31] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[32] I. Das, "On characterizing the "knee" of the pareto curve based on normal-boundary intersection," *Structural optimization*, vol. 18, pp. 107–115, 1999.

[33] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, pp. 193–200, 2007.