题目

Spatial-Temporal Knowledge Transfer for Dynamic Constrained Multiobjective Optimization

摘要

这篇论文研究了动态约束多目标优化问题（DCMOPs），这种问题具有随时间变化的目标和约束，解决起来极具挑战性。作者强调，在这种问题中保持种群多样性的重要性，不仅可以避免陷入局部最优解，还能帮助优化算法更好地应对约束。

为此，作者提出了两个任务：一个专注于探索无约束空间以维持多样性，另一个专注于约束空间以提高解的可行性。同时，为进一步提升算法性能，作者设计了两个知识迁移模块：一个模块在约束与无约束空间间传递知识，另一个模块利用历史知识提高搜索效率，即空间知识迁移模块和时间知识迁移模块。

为了验证算法，研究设计了 14 个测试问题和一个实际问题，实验结果表明，所提算法能够有效处理动态约束优化问题。

关键词：动态带约束的多目标优化，测试套件，进化多任务优化，知识迁移

1 引言

动态约束多目标优化问题（DCMOPs）广泛存在于动态电动车路径规划、物流分配和能源调度等实际应用中，其特点是多个冲突目标和动态变化的约束条件。由于环境的动态性和约束的复杂性，传统算法在保持解的多样性和快速跟踪 Pareto 最优解方面表现不足，尤其容易陷入局部最优。

研究现状：现有方法难以在动态环境中兼顾解的多样性与约束处理，尤其在可行解区域狭窄或不连续时效果有限。技术启发：知识迁移和多任务优化技术提供了新方向，通过时空相似性实现约束空间与非约束空间的协同优化，提高算法效率。实际需求：DCMOPs 在智能交通、能源管理等领域的应用要求更高效的优化算法，现有算法无法完全满足需求。

理论意义：提出基于时空知识迁移的优化算法（DC-MFEA），实现多样性和可行性的平衡，推进动态优化理论发展。实践意义：应用于交通调度、物流优化等领域，提高系统运行效率，为智能优化提供新工具。技术意义：设计时空知识迁移机制，提升算法在动态复杂环境中的适应性和鲁棒性，为相关领域提供参考。

2 相关工作

动态约束多目标优化问题（DCMOPs）处于动态多目标优化问题（DMOPs）和约束多目标优化问题（CMOPs）研究领域的前沿。在本小节中，我们将简要回顾 DMOPs 和 CMOPs 的最新研究文献，随后将重点评述专门针对 DCMOPs 的现有研究工作。

2.1 动态多目标优化算法

过去十年，针对动态多目标优化问题（DMOPs），已经开发了多种优化算法。这些算法主要分为三类：多样性增强与维护方法、基于记忆的方法和基于预测的方法。多样性增强与维护方法：这些方法旨在环境变化时缓解种群多样性丧失的问题。例如：DNSGA2 通过随机生成或变异部分解，同时保留一些历史解以重初始化 [11]。Jiang 等提出了一种稳态策略，重用过时解并通过变化响应策略维持多样性 [16]。Chen 等将种群分为两个子群体，一个提高收敛质量，另一个增强多样性 [10]。Rong 等利用聚类策略选择具有代表性的个体来维持环境变化下的多样性 [30]。基于记忆的方法：这些方法通过存储历史环境中的信息提升优化性能。例如：Branke 等提出了一种记忆机制，重用存档中找到的最优解 [4]。Jiang 等利用历史解构建地质流，以预测新环境中的解 [15]。Xu 等隐式保存过去环境中的解信息，通过这些信息预测新环境中的解 [35]。Chen 等设计了一个记忆池，用模糊模型存储和预测历史 Pareto 最优解 [6]。基于预测的方法：最近的研究利用机器学习为进化算法赋予学习能力。例如：Muruganantham 等使用卡尔曼滤波器生成初始种群 [22]。Zhou 等采用核化自编码生成解析解 [38]。Zhang 等结合逆高斯过程模型与进化算法，学习目标空间与决策空间的映射，从而生成解 [37]。TK-DMOEA 利用弱分类器预测新环境中的关键点，从而提高搜索效率 [14]。为了应对现实中的噪声问题，鲁棒优化逐渐受到关注。例如：Rambabu 等设计了一个基于“专家混合”的框架，通过门控网络在不同时间切换不同预测器以实现 Pareto 解的鲁棒预测 [28]。

2.2 带约束的多目标优化算法

针对约束多目标优化问题（CMOPs）的处理，研究者们提出了多种方法，这些方法可以归纳为以下几类：约束支配原则（CDP）、惩罚函数法、随机排序、 约束法、约束转换法以及混合技术。约束支配原则（CDP）：CDP 在环境选择过程中优先选择可行解。如果是不可行解，则通过约束违反程度（而非目标值）来决定支配关系 [12,19,36]。惩罚函数法：通过将约束条件作为目标函数的惩罚项来处理，惩罚权重可以是固定的，也可以在搜索过程中自适应调整 [2], [8], [3], [34]。例如：Ma 等提出了自适应惩罚函数 ShiP，帮助不可行解从不同方向接近可行区域 [20]。随机排序：通过约束违反程度或目标值比较两个解，具体选择标准由用户定义的概率决定 [31], [33]。例如：Tian 等通过目标和约束的评估确定支配关系，并根据可行解比例选择不同的适应度评估策略 [32]。 约束法：将松弛约束边界内的不可行解视为可行解，松弛程度由参数 控制。当 趋于 0 时，该方法退化为 CDP [13], [1]。最近，Qiao 等在多任务框架下提出改进的 约束法，通过保留多样性高、质量优的不可行解帮助主任务在早期阶段突破不可行障碍 [25]。约束转换法：将约束违反值视为附加目标，使用非支配排序方法对解进行排序 [23], [29]。混合技术：将上述几种约束处理技术结合使用 [27], [21], [17]。进化多任务优化方法：近年来，一些基于进化多任务的约束多目标优化算法出现。这些算法利用不可行解与可行解之间的潜在协同效应，提高可行解的收敛性。例如：引入辅助任务，探

索不可行区域并为原始 CMOP 补充多样性 [24], [26]。然而，这些方法缺乏完善的动态响应策略，未能有效利用连续环境中可行与不可行区域的知识，从而可能引入低质量解，阻碍收敛。不足与局限：尽管上述技术可以扩展到动态约束多目标优化问题（DCMOPs），它们仍存在选择偏差。当约束优先于目标时，可能导致种群多样性下降；当目标优先于约束时，可能导致收敛质量下降。这些问题与动态环境中快速获取 Pareto 最优解的目标相矛盾。总结：上述约束处理技术在 CMOPs 和动态约束优化问题中的具体缺陷已在文献 [18], [5] 中有详细讨论。

2.3 动态带约束多目标优化

尽管动态多目标优化问题（DMOPs）和约束多目标优化问题（CMOPs）的研究已有大量成果，但对动态约束多目标优化问题（DCMOPs）的关注却较少。现有的研究包括以下方法及其局限性：部落划分方法：Chen 等提出了一种部落划分方法，将种群分成若干部落，并对每个部落采用不同策略，引导其向可行区域移动 [9]。局限性：该方法使用一维搜索策略，需要额外的函数评估，增加计算开销。子空间知识迁移算法（SKTEA）：Chen 等提出的 SKTEA 算法通过子空间知识迁移来解决 DCMOPs [7]。局限性：其知识迁移效果严重依赖于可行子空间的可用性。当 DCMOPs 中不可行区域较大时，SKTEA 可能产生负迁移，导致初始种群质量下降。动态 NSGA2 及其改进版本：Azzouz 等采用动态 NSGA2 作为基础求解器，并设计了自适应惩罚函数来引导可行区域 [3]。问题：对于需要跨越某些不可行区域的问题，该方法表现较差。改进：Azzouz 等进一步提出改进版本 DC-MOEA，利用自适应惩罚函数和可行性驱动策略，帮助优化向新的可行区域搜索 [1]。自适应惩罚函数方法（dCMOEA）：Chen 等提出了 dCMOEA 算法，利用自适应惩罚函数平衡可行解和不可行解，以避免过早收敛 [8]。局限性：这些方法通过记忆机制重新初始化部分解来应对环境变化，但在可行区域不连续时，可能导致种群多样性丧失和过早收敛。

3 本文方法

3.1 本文方法概述

动态约束多任务进化算法（DC-MFEA）遵循多任务进化算法的框架，通过在统一搜索空间中同时优化两个任务来解决动态约束多目标优化问题（DCMOPs）。其主要工作流程如下：

任务划分与优化：

主任务：在约束搜索空间内搜索，即优化原始 DCMOP。辅助任务：移除 DCMOP 中的约束，在非约束搜索空间内进行搜索以维持多样性。两个任务共享一个统一搜索空间，通过遗传操作实现隐式的空间知识迁移。时空知识迁移：

空间知识迁移（SKT）：通过遗传操作（如交叉、变异）在约束和非约束搜索空间之间共享知识，加速两个任务的探索。时间知识迁移（TKT）：当环境发生变化时，将主任务和辅助任务中的历史解分别存储在存档（AC 和 AU）中，根据连续环境之间的相似性，从存档中重用历史解，指导新环境的搜索。整体工作流程：

初始阶段，分别为主任务种群 P_0 和辅助任务种群 Q_0 随机初始化。环境不变时，通过 SKT 优化两个种群，优化结果存储到相应的存档（AC 和 AU）。环境变化时，通过 TKT 从存档中提取历史解（P_t 和 Q_t），根据环境相似性指导搜索，然后由 SKT 继续优化。每次迭代输

出的解集中标识动态约束 Pareto 最优解（$DCPOS_t$）图示与算法：在图 1中，对 DC-MFEA 的结构进行了详细说明，算法的伪代码呈现在 Algorithm 1 中，展示了主任务和辅助任务的初始化、优化及其与环境变化的互动。

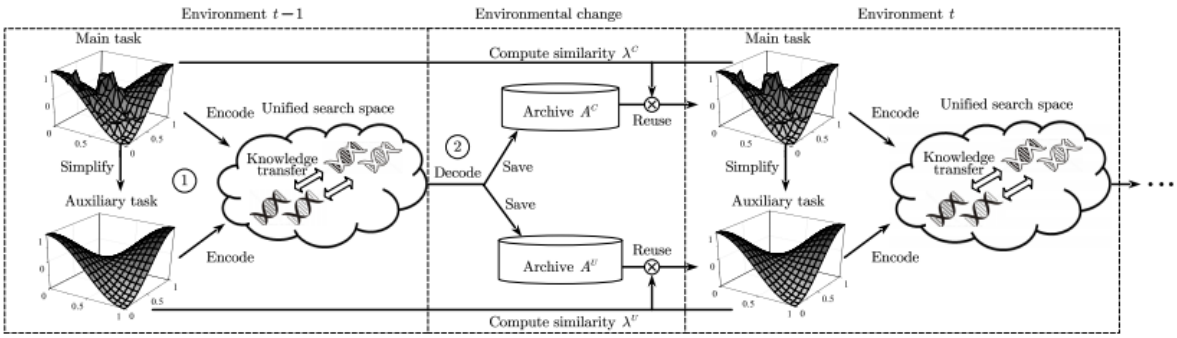

图 1. DC-MFEA 算法的整体框架图

Algorithm 1: DC-MFEA

Input: the DCMOP $F_t(\cdot)$, the number of individuals for main task N_{T1}, the number of individuals for auxiliary task N_{T2}
Output: the $DCPOS_t$ of the DCMOP at the the environment step t

Set environment step $t = 0$;
Initialize a population P_0 for the main task, $|P_0| = N_{T1}$;
Initialize a population Q_0 for the auxiliary task, $|Q_0| = N_{T2}$;
while *stopping criterion is not met* **do**
  if *a change is detected* **then**
    $t = t + 1$;
    P_t, Q_t=**TKT**(A^C, A^U);
  end
  P_t, Q_t=**SKT**$(F_t(\cdot), P_t, Q_t)$;
  $A^C = P_t, A^U = Q_t$;
  Identify the $DCPOS_t$ in P_t;
  return $DCPOS_t$;
end

3.2 空间知识迁移模块

空间知识迁移 (SKT) 是 DC-MFEA 框架中的核心模块之一，通过基于任务之间的空间相似性，实现主任务和辅助任务之间的隐式知识共享与迁移。

SKT 的主要功能在于协同优化：主任务聚焦于约束搜索空间，目标是将种群引导至可行区域；辅助任务则移除约束，在无约束搜索空间中进行搜索，从而维持种群的多样性。SKT 的实现基于多任务进化算法 (MFEA)，通过遗传操作（尤其是交叉操作）实现隐式知识迁移。

为了统一主任务和辅助任务的搜索空间，个体的决策变量 x_d 被重新编码为无量纲变量 x'_d，公式如下：

$$x'_d = \frac{x_d - x_d^{lb}}{x_d^{ub} - x_d^{lb}}, \tag{1}$$

其中 x_d^{lb} 和 x_d^{ub} 分别为 x_d 的下界和上界。这种编码方式保证了主任务和辅助任务能够在相同的搜索空间中优化。

在统一的搜索空间中，计算以下关键特性以支持优化过程：

1. **任务因子排名 (Factorial Rank)：** 对于解 x，其在第 k 个任务上的因子排名 r_x^k 表示其在该任务中基于非支配排序的排名（$k=1$ 表示主任务，$k=2$ 表示辅助任务）。

2. **标量适应度 (Scalar Fitness)：** 解 x 的标量适应度 ϕ_x 定义为其在所有任务中最好排名的倒数：
$$\phi_x = \frac{1}{\min\{r_x^1, r_x^2\}} \tag{2}$$

3. **技能因子 (Skill Factor)：** 解 x 的技能因子 ℓ_x 表示其在哪个任务中表现最好：
$$\ell_x = \arg\min_{j\in\{1,2\}} \{r_x^j\} \tag{3}$$

针对不同任务，SKT 采用了两种不同的环境选择策略：

- **主任务：** 使用基于可行性驱动的非支配排序 (CDP) 来选择解。可行解优先于不可行解；对于不可行解，根据其约束违反值 (CV) 进行排序。约束违反值的定义为：
$$CV_k = \begin{cases} \max(0, h_k(x,t)), & k = 1, ..., N_H, \\ \max(0, g_k(x,t)), & k = 1+N_H, ..., N_H+N_G, \end{cases} \tag{4}$$
其中，$h_k(x,t)$ 和 $g_k(x,t)$ 分别表示等式约束和不等式约束的违反值。

- **辅助任务：** 使用经典的非支配排序方法，侧重选择多样性丰富的解。

通过交叉操作的空间知识迁移，SKT 实现了主任务和辅助任务搜索结果的整合：

- 提升种群多样性，帮助解跨越不可行区域的障碍，接近可行区域。
- 改善收敛性能，避免种群陷入局部最优。

SKT 的设计有效解决了动态约束优化问题中种群多样性丧失和局部最优陷阱的问题，同时提升了主任务的可行性和算法的整体性能。有关 SKT 的具体流程，请参见 Algorithm 2。

3.3 时间知识迁移组件

时间知识迁移 (TKT) 是 DC-MFEA 中的另一关键模块，旨在通过利用任务间的环境相似性来重用主任务和辅助任务的历史解，以指导新环境下的优化。

时间知识迁移的核心原理在于通过主任务存档 A^C 和辅助任务存档 A^U 中的历史解进行知识迁移。环境相似性越高，两个存档中的解被重用的概率越大，从而为新生成的环境提供指导。

主任务和辅助任务的环境相似性分别定义为 λ^C 和 λ^U，它们是依据解的排名变化计算的。每个解 x 在 t-th 环境中的排名为 $\rho_{x,t}^{(\cdot)}$，其中：

- $\rho_{x,t}^C$：主任务存档中解的排名（通过非支配排序和 CDP 方法获得）。

Algorithm 2: Spatial Knowledge Transfer (SKT)

Input: the DCMOP $F_t(\cdot)$, the population of the main task P_t, the population of the auxiliary task Q_t
Output: P_t, Q_t

1 Remove constraints from the $F_t(\cdot)$ to create an auxiliary task $F_t^{'}(\cdot)$;
2 Encode $P_t \cup Q_t$ in the unified space;
3 Calculate the skill factor, factorial rank, and scalar fitness of each solution in $P_t \cup Q_t$;
4 Perform assortative mating on $P_t \cup Q_t$ to obtain the offspring O_t;
5 Determine the task for each solution in O_t by vertical cultural transmission;
6 Calculate $CV(x)$, where $\{x|x \in P_t \cup Q_t \cup O_t, \ell_x := 1\}$;
7 Perform environmental selection for $F_t(\cdot)$ by using CDP to obtain P_t ;
8 Perform environmental selection for $F_t^{'}(\cdot)$ by using nondominated sorting to obtain Q_t;
9 **return** P_t, Q_t;

- $\rho_{x,t}^{U}$：辅助任务存档中解的排名（通过经典非支配排序获得）。

排名根据拥挤距离 $CD(x)$ 进行修正，以提高多样性：

$$CD(x) = \sum_{x^* \in R_{x,t}} \sum_{m=1}^{M} |f_m(x) - f_m(x^*)|, \tag{5}$$

其中 $R_{x,t}$ 为排名与 x 相同的解集合，计算公式为：

$$R_{x,t} = \{x^* | x^* \in A^{(\cdot)} \setminus x, \rho_{x,t} = \rho_{x,t}^{(\cdot)}\}. \tag{6}$$

拥挤距离 $CD(x)$ 被归一化到 $[0, 0.5]$：

$$\overline{CD}(x) = \frac{CD(x) - CD_{\min}}{2(CD_{\max} - CD_{\min})}, \tag{7}$$

其中 $CD_{\min}$ 和 $CD_{\max}$ 分别为 $CD(x)$ 的最小值和最大值。

根据归一化的拥挤距离，对解的排名进行修正：

$$\rho_{x,t}^{(\cdot)} = \rho_{x,t}^{(\cdot)} - \overline{CD}(x). \tag{8}$$

使用 Spearman 排名相关系数计算连续两个环境 $t-1$ 和 t 之间的相似性：

$$\lambda^{(\cdot)} = \frac{\mathrm{cov}(\{\rho_{x,t}^{(\cdot)}\}, \{\rho_{x,t-1}^{(\cdot)}\})}{\mathrm{std}(\{\rho_{x,t}^{(\cdot)}\}) \cdot \mathrm{std}(\{\rho_{x,t-1}^{(\cdot)}\})}, \tag{9}$$

其中 $\mathrm{cov}(\cdot)$ 为协方差，$\mathrm{std}(\cdot)$ 为标准差。当 $\lambda^{(\cdot)} < 0$ 时，设 $\lambda^{(\cdot)} = 0$，以消除负相关。

根据相似性 λ^C 和 λ^U，分别从存档 A^C 和 A^U 中选出排名靠前的解直接用于新环境。主任务和辅助任务的未重用解根据需要重新初始化，以补充种群 P_t 和 Q_t 的规模。新环境中的优化过程由 Algorithm 3 描述。

Algorithm 3: Temporal Knowledge Transfer (TKT)

Input: the archive for the main task A^C, the archive for the auxiliary task A^U

Output: the population P_t of the main task, the population Q_t of the auxiliary task

Get the historical ranking $\rho^{(\cdot)}_{x,t-1}$ for each solution from $A^{(\cdot)}$;
Re-evaluate solutions from $A^{(\cdot)}$ to get the current ranking $\rho^{(\cdot)}_{x,t}$;
Compute the crowding distance CD;
Modify the rankings according to Eq. (11);
Compute Spearman's rank correlation coefficient according to Eq. (12);
$\lambda^C = \max\{\lambda^C, 0\}$, $\lambda^U = \max\{\lambda^U, 0\}$;
Select the top $\lambda^C N_{T1}$ ranking solutions from A^C and denote as P_t;
Select the top $\lambda^U N_{T2}$ ranking solutions from A^U and denote as Q_t;
Randomly generate $N_{T1} - \lambda^C N_{T1}$ solutions and add them to P_t;
Randomly generate $N_{T2} - \lambda^U N_{T2}$ solutions and add them to Q_t;
return P_t, Q_t;

4 复现细节

4.1 与已有开源代码对比

本文的算法实现由于并未提供源代码，故参考了 platEMO 平台上的 DCNSGAIII 的代码部分动态和约束处理的代码处理方式。引用部分：

Listing 1: EnvironmentalSelection MATLAB Code

```
function Population = EnvironmentalSelection(Population, N, Z, Zmin,
    initialE, epsn)
    [~, nCon] = size(Population.cons);
    ConVio    = max(0, Population.cons);
    if sum(sum(ConVio<=epsn, 2)==nCon) > N
        %% Selection among epsilon-feasible solutions
        tmp        = sum(ConVio<=epsn, 2)==nCon;
        Population = Population(1:end, tmp);
        CV         = sum(max(0,Population.cons)./initialE, 2)./nCon;
        %% Non-dominated sorting based on objectives and constraint
            violation
        [FrontNo, MaxFNo]    = NDSort([Population.objs, CV], N);
        Next                 = false(1, length(FrontNo));
        Next(FrontNo<MaxFNo) = true;
        %% Select solutions in the last front
        Last                 = find(FrontNo==MaxFNo);
        pop1                 = [Population(Next).objs];
        pop2                 = [Population(Last).objs];
        Choose               = LastSelection(pop1, pop2, N-sum(Next), Z,
```

```
            Zmin);
        Next(Last(Choose)) = true;
        Population         = Population(Next);
    else
        %% Selection including epsilon-infeasible solutions
        CV          = sum(max(0, Population.cons)./initialE, 2)./nCon;
        [~, rank]   = sort(CV);
        Population = Population(rank(1:N));
    end
end
```

这部分参考了如何用代码实现环境的变化后的选择，在面对动态的环境变化时候该如何编程进行环境变化之后的一系列选择。

Listing 2: ReduceBoundary MATLAB Code

```
function epsn = ReduceBoundary(eF, k, MaxK, cp)

    z        = 1e-8;
    Nearzero = 1e-15;
    B        = MaxK./power(log((eF + z)./z), 1.0./cp);
    B(B==0)  = B(B==0) + Nearzero;
    f        = eF.* exp( -(k./B).^cp );
    tmp      = find(abs(f-z) < Nearzero);
    f(tmp)   = f(tmp).*0 + z;
    epsn     = f - z;
    epsn(epsn<=0) = 0;
end
```

这部分参考了如何对约束条件进行处理以及计算约束条件的违反值

Listing 3: DCNSGAIII MATLAB 类

```
classdef DCNSGAIII < ALGORITHM

    methods
        function main(Algorithm,Problem)
            %% Parameter setting
            cp                      = Algorithm.ParameterSet(5);

            %% Generate the reference points and random population
            [Z,Problem.N]           = UniformPoint(Problem.N, Problem.M);
            Population              = Problem.Initialization();

            %% Calculate the initial dynamic constraint boundary
            [initialE, ~]           = max(max(0,Population.cons), [], 1);
```

```
            initialE(initialE==0) = 1;

            %% Optimization
            while Algorithm.NotTerminated(Population)
                % Reduce the dynamic constraint boundary
                epsn       = ReduceBoundary(initialE, ceil(Problem.FE/
                    Problem.N), ceil(Problem.maxFE/Problem.N)-1, cp);
                % Mating selection which prefers to select epsilon-
                    feasible solutions
                MatingPool = TournamentSelection(2, Problem.N,sum(max(0,
                    Population.cons-epsn), 2));
                Offspring  = OperatorGA(Problem,Population(MatingPool));
                % Update the reference points only consider the epsilon-
                    feasible solution set
                Zmin       = min([Population(sum(max(0, Population.cons)
                    <=epsn, 2)==size(Population.cons, 2)).objs; Offspring(
                    sum(max(0,Offspring.cons)<=epsn, 2)==size(Offspring.
                    cons, 2)).objs], [], 1);
                % Environment selection
                Population = EnvironmentalSelection([Population,
                    Offspring], Problem.N, Z, Zmin, initialE, epsn);
            end
        end
    end
end
```

这部分是实现 DCNSGAIII 的主程序结构，参考包括了动态环境变化检测，约束的处理，以及种群的选择交叉变异，以及环境变化后种群的处理的整体流程的框架实现。

本人对此文中算法的关键部分的复现包括：

Listing 4: MOMFEA_v1 MATLAB Code

```
function [POF,population]=MOMFEA_v1(t,mainTask,pop1,auxiliaryTask,pop2,
    rmp,gen,muc,mum,prob_vswap,initPop)
[L1,U1]=mainTask.getBound(t);
[L2,U2]=auxiliaryTask.getBound(t);

pop=pop1+pop2;
if mod(pop,2)~=0
    pop=pop+1;
    pop2=pop2+1;
end

dim1=length(L1);
dim2=length(L2);
```

```
dim=max([dim1,dim2]);

store=[];
if nargin==11
    for i=1:length(initPop)
        population(i)=Chromosome;
        population(i).rnvec=initPop(i).rnvec;
        population(i).skill_factor=initPop(i).skill_factor;
    end
    for i=length(initPop)+1:pop
        population(i)=Chromosome;
        population(i)=initialize(population(i),dim);
        if i<=pop1
            population(i).skill_factor=1;
        else
            population(i).skill_factor=2;
        end
    end
else
    for i=1:pop
        population(i)=Chromosome;
        population(i)=initialize(population(i),dim);
        if i<=pop1
            population(i).skill_factor=1;
        else
            population(i).skill_factor=2;
        end
    end
end

for i=1:pop
    population(i)=evaluate(t,population(i),L1,U1,mainTask,L2,U2,
       auxiliaryTask,dim1,dim2);
end

if nargin==11
    targetpopulation_T1=population([population.skill_factor]==1);
    targetpopulation_T2=population([population.skill_factor]==2);
    sourcrpopulation_T1=initPop([initPop.skill_factor]==1);
    sourcrpopulation_T2=initPop([initPop.skill_factor]==2);
    targetpopulation_T1=NDS_CV(targetpopulation_T1,mainTask);
    [targetpopulation_T2,~]=SolutionComparison.nondominatedsort(
       targetpopulation_T2,length(targetpopulation_T2),2);
```

```
    for j=1:length(sourcrpopulation_T1)
        T1xsource(j,:)=sourcrpopulation_T1(j).rnvec;
        T1ysource(j)=sourcrpopulation_T1(j).front;
        T1xtarget(j,:)=targetpopulation_T1(j).rnvec;
        T1ytarget(j)=targetpopulation_T1(j).front;
    end
    for j=1:length(sourcrpopulation_T2)
        T2xsource(j,:)=sourcrpopulation_T2(j).rnvec;
        T2ysource(j)=sourcrpopulation_T2(j).front;
        T2xtarget(j,:)=targetpopulation_T2(j).rnvec;
        T2ytarget(j)=targetpopulation_T2(j).front;
    end

    for iii=1:length(targetpopulation_T1)
        targetPopObjT1(iii,:)=targetpopulation_T1(iii).objs_T1;
    end
    for iii=1:length(sourcrpopulation_T1)
        sourcePopObjT1(iii,:)=sourcrpopulation_T1(iii).objs_T1;
    end
    for iii=1:length(targetpopulation_T2)
        targetPopObjT2(iii,:)=targetpopulation_T2(iii).objs_T2;
    end
    for iii=1:length(sourcrpopulation_T2)
        sourcePopObjT2(iii,:)=sourcrpopulation_T2(iii).objs_T2;
    end

    targetT1CrowdDis = CrowdingDistance(targetPopObjT1,T1ytarget);
    targetT2CrowdDis = CrowdingDistance(targetPopObjT2,T2ytarget);
    sourceT1CrowdDis = CrowdingDistance(sourcePopObjT1,T1ysource);
    sourceT2CrowdDis = CrowdingDistance(sourcePopObjT2,T2ysource);

    targetT1CrowdDis(find(isinf(targetT1CrowdDis)))=-inf;
    targetT2CrowdDis(find(isinf(targetT2CrowdDis)))=-inf;
    sourceT1CrowdDis(find(isinf(sourceT1CrowdDis)))=-inf;
    sourceT2CrowdDis(find(isinf(sourceT2CrowdDis)))=-inf;

    targetT1CrowdDis(find(isinf(targetT1CrowdDis)))=max(targetT1CrowdDis)
       ;
    targetT2CrowdDis(find(isinf(targetT2CrowdDis)))=max(targetT2CrowdDis)
       ;
    sourceT1CrowdDis(find(isinf(sourceT1CrowdDis)))=max(sourceT1CrowdDis)
       ;
    sourceT2CrowdDis(find(isinf(sourceT2CrowdDis)))=max(sourceT2CrowdDis)
```

```
        ;
94
95      [targetT1CrowdDis,~]=mapminmax(targetT1CrowdDis,0,0.5);
96      [targetT2CrowdDis,~]=mapminmax(targetT2CrowdDis,0,0.5);
97      [sourceT1CrowdDis,~]=mapminmax(sourceT1CrowdDis,0,0.5);
98      [sourceT2CrowdDis,~]=mapminmax(sourceT2CrowdDis,0,0.5);
99
100     T1ytarget=T1ytarget-targetT1CrowdDis;
101     T2ytarget=T2ytarget-targetT2CrowdDis;
102     T1ysource=T1ysource-sourceT1CrowdDis;
103     T2ysource=T2ysource-sourceT2CrowdDis;
104
105     T1sim=corr(T1ysource',T1ytarget','type','spearman');
106     T2sim=corr(T2ysource',T2ytarget','type','spearman');
```

此部分实现了文中算法的关键组件，空间知识迁移组件 SKT 以及时间知识迁移组件 TKT。

Listing 5: Evolve MATLAB Code

```
1  classdef Evolve
2      methods(Static)
3          function [rnvec1,rnvec2]=crossover(p1,p2,muc,dim,prob_vswap)
4              rnvec1=zeros(1,dim);
5              rnvec2=zeros(1,dim);
6              randlist=rand(1,dim);
7              for i=1:dim
8                  if randlist(i)<=0.5
9                      k=(2*randlist(i))^(1/(muc+1));
10                 else
11                     k=(1/(2*(1-randlist(i))))^(1/(muc+1));
12                 end
13                 rnvec1(i)=0.5*(((1 + k)*p1(i)) + (1 - k)*p2(i));
14                 rnvec2(i)=0.5*(((1 - k)*p1(i)) + (1 + k)*p2(i));
15                 if rand(1) < prob_vswap
16                     tmp = rnvec1(i);
17                     rnvec1(i) = rnvec2(i);
18                     rnvec2(i) = tmp;
19                 end
20                 if rnvec1(i)<0
21                     rnvec1(i)=0;
22                 elseif rnvec1(i)>1
23                     rnvec1(i)=1;
24                 end
25                 if rnvec2(i)<0
26                     rnvec2(i)=0;
```

```
                elseif rnvec2(i)>1
                    rnvec2(i)=1;
                end
            end
        end

        function rnvec=mutate(p,mum,dim,prob_mut)
            rnvec=p;
            for i=1:dim
                if rand(1)<prob_mut
                    u=rand(1);
                    if u <= 0.5
                        del=(2*u)^(1/(1+mum)) - 1;
                        rnvec(i)=p(i) + del*(p(i));
                    else
                        del= 1 - (2*(1-u))^(1/(1+mum));
                        rnvec(i)=p(i) + del*(1-p(i));
                    end
                end
            end
        end
    end
end
```

这一部分是进化迭代的操作，对当前状态中的种群在对应状态下的约束条件和评价指标并在解空间内进行迭代优化。

Listing 6: DCMFEA MATLAB Code

```
function res=DCMFEA(Problem,popSize,MaxIt,T_parameter,group)
Nfor2=20;
    F(1) = struct('cdata',[],'colormap',[]);
    rmp=1;
    muc = 1;
    mum = 1;
    prob_vswap = 0;
    for T=1:T_parameter(group,3)/T_parameter(group,2)
        fprintf('␣%d',T);
        t = 2/T_parameter(group,1)*(T-1);   % next moment
        mainTask=Problem;
        auxiliaryTask=creatAuxiliaryTask(Problem);
        if T==1
            [POF,population]=MOMFEA_v1(t,mainTask,popSize-Nfor2,
               auxiliaryTask,Nfor2,rmp,MaxIt,muc,mum,prob_vswap);
        else
```

```
            % [POF,population]=MOMFEA_v1(t,mainTask,popSize-Nfor2,
                auxiliaryTask,Nfor2,rmp,MaxIt,muc,mum,prob_vswap);
            [POF,population]=MOMFEA_v1(t,mainTask,popSize-Nfor2,
                auxiliaryTask,Nfor2,rmp,MaxIt,muc,mum,prob_vswap,
                population);
        end
        CPOF=getFesiableSolutions(t,POF,mainTask);
        %F=drawSolutions(t,POF,mainTask,CPOF,T,F);
        res{T}.turePOF=Problem.getCPF(400,t);
        res{T}.POF=POF;
        res{T}.CPOF=CPOF;
    end
end
```

这一部分是构架对文中给定的 14 个动态约束测试问题的动态模拟测试框架

4.2 实验环境搭建

本实验在 MATLAB 中进行完成

5 实验结果分析

实验内容：在 14 个基准测试问题 DCF1-DCF14 中测试该动态约束处理方法的性能。基准问题的动态最优前沿是已知的，即对于每个不同状态下当前问题的最优解是预先知道的，所以通过执行算法得到的最优解与实际最优解进行比较评估，有以下两个评估指标：

IGD，评估解的收敛性，即找到的最优解与实际最优解之间的平局距离，评估算法对于最优解的寻找能力。

HV，评估解的多样性，对于整个最优前沿，多样性用于评估算法找到解对于整个前沿的覆盖程度，越高，则多样性越好。

实验结果：

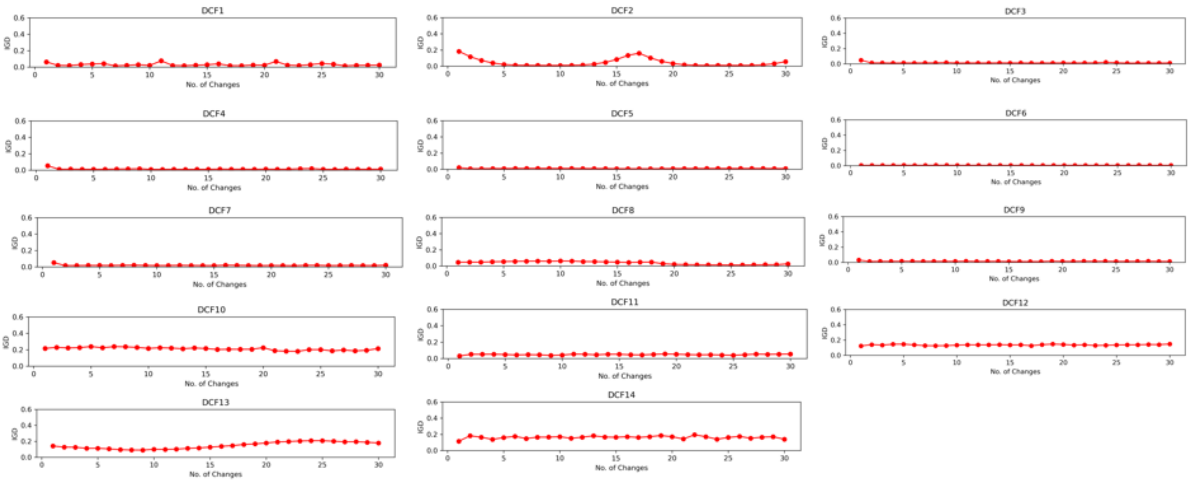

图 2. 在 14 个测试问题上平均 IGD 指标的结果

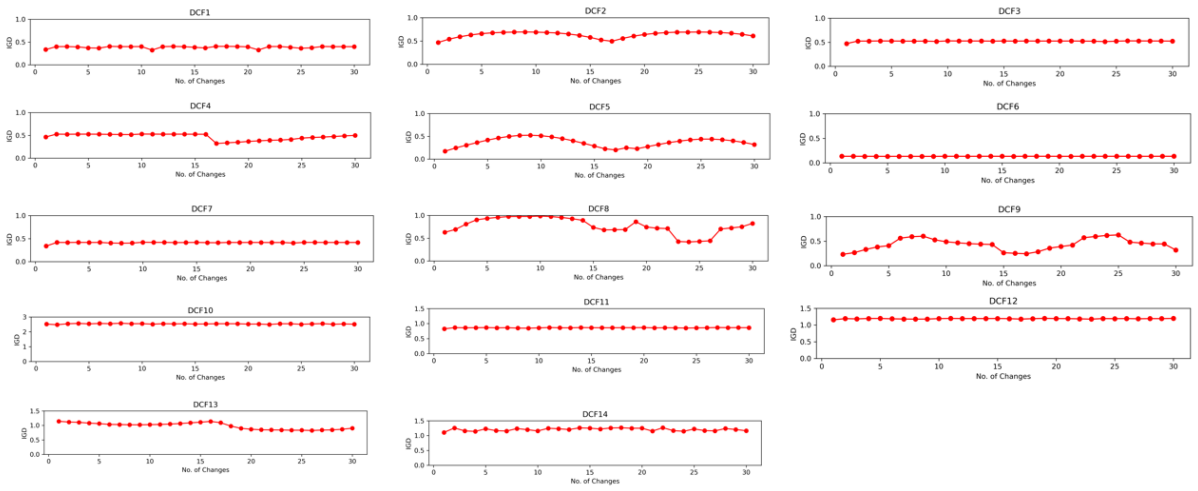

图 3. 在 14 个测试问题上平均 HV 指标的结果

Results Table

Problem	n_t, τ_t	DC-MFEA	
DCF1	1.80e-02(1.54e-02)	2.98e-02(1.56e-02)	4.56e-02(2.40e-02)
DCF2	4.95e-02(5.44e-02)	4.46e-02(4.90e-02)	4.54e-02(4.94e-02)
DCF3	2.36e-02(1.76e-02)	1.32e-02(6.91e-03)	2.07e-02(1.39e-02)
DCF4	2.35e-02(1.82e-02)	1.33e-02(8.10e-03)	1.81e-02(1.20e-02)
DCF5	1.04e-02(3.60e-03)	9.46e-03(2.59e-03)	1.12e-02(4.17e-03)
DCF6	6.87e-03(3.68e-03)	4.81e-03(5.81e-04)	5.82e-03(1.29e-03)
DCF7	3.00e-02(1.91e-02)	1.90e-02(6.29e-03)	2.39e-02(1.20e-02)
DCF8	4.04e-02(1.54e-02)	3.87e-02(1.86e-02)	3.88e-02(1.67e-02)
DCF9	1.48e-02(2.84e-03)	1.47e-02(3.50e-03)	1.57e-02(3.80e-03)
DCF10	2.06e-01(1.58e-02)	2.12e-01(1.70e-02)	2.04e-01(1.79e-02)
DCF11	4.54e-02(7.82e-03)	4.80e-02(5.58e-03)	4.73e-02(5.83e-03)
DCF12	1.32e-01(1.00e-02)	1.36e-01(6.68e-03)	1.32e-01(7.75e-03)
DCF13	1.45e-01(4.18e-02)	1.47e-01(4.16e-02)	1.45e-01(4.15e-02)
DCF14	1.49e-01(8.31e-03)	1.63e-01(1.64e-02)	1.63e-01(1.73e-02)

图 4. 在 14 个测试问题中三组不同参数下的 MIGD 均值和方差

Results Table

Problem	n_t, τ_t	DC-MFEA	
DCF1	4.03e-01(2.22e-02)	3.87e-01(2.35e-02)	3.67e-01(3.45e-02)
DCF2	6.29e-01(7.02e-02)	6.35e-01(6.40e-02)	6.33e-01(6.46e-02)
DCF3	5.08e-01(2.43e-02)	5.23e-01(9.99e-03)	5.12e-01(1.94e-02)
DCF4	4.57e-01(6.95e-02)	4.70e-01(6.85e-02)	4.63e-01(6.93e-02)
DCF5	3.62e-01(1.10e-01)	3.69e-01(1.01e-01)	3.67e-01(1.02e-01)
DCF6	1.32e-01(3.14e-03)	1.34e-01(5.81e-04)	1.33e-01(8.88e-04)
DCF7	3.89e-01(3.90e-02)	4.12e-01(1.48e-02)	4.00e-01(2.34e-02)
DCF8	7.70e-01(1.57e-01)	7.71e-01(1.77e-01)	7.61e-01(1.80e-01)
DCF9	4.18e-01(1.20e-01)	4.31e-01(1.21e-01)	4.29e-01(1.21e-01)
DCF10	2.53e+00(1.86e-02)	2.53e+00(2.10e-02)	2.53e+00(1.74e-02)
DCF11	8.51e-01(1.76e-02)	8.60e-01(8.52e-03)	8.55e-01(1.25e-02)
DCF12	1.18e+00(1.76e-02)	1.19e+00(9.08e-03)	1.18e+00(1.62e-02)
DCF13	9.80e-01(1.04e-01)	9.82e-01(1.12e-01)	9.80e-01(1.11e-01)
DCF14	1.20e+00(5.13e-02)	1.20e+00(4.64e-02)	1.20e+00(4.90e-02)

图 5. 在 14 个测试问题中三组不同参数下 MHV 的均值和方差

Results Table

Problem	n_t, τ_t	DC-MFEA	
DCF1	2.65e-02(1.69e-02)	3.65e-02(2.02e-02)	4.66e-02(2.45e-02)
DCF2	4.98e-02(5.47e-02)	4.46e-02(4.91e-02)	4.57e-02(4.96e-02)
DCF3	2.64e-02(1.97e-02)	1.52e-02(8.65e-03)	2.27e-02(1.39e-02)
DCF4	2.62e-02(1.93e-02)	1.43e-02(8.56e-03)	2.12e-02(1.34e-02)
DCF5	1.14e-02(5.09e-03)	9.87e-03(3.10e-03)	1.25e-02(4.45e-03)
DCF6	7.69e-03(3.92e-03)	5.56e-03(1.36e-03)	7.49e-03(2.19e-03)
DCF7	3.32e-02(1.97e-02)	2.06e-02(7.13e-03)	2.67e-02(1.32e-02)
DCF8	4.07e-02(1.47e-02)	3.88e-02(1.77e-02)	3.94e-02(1.47e-02)
DCF9	1.57e-02(3.60e-03)	1.47e-02(2.37e-03)	1.61e-02(3.41e-03)
DCF10	2.02e-01(1.44e-02)	2.08e-01(1.53e-02)	2.02e-01(1.56e-02)
DCF11	4.36e-02(8.18e-03)	4.62e-02(7.14e-03)	4.46e-02(6.77e-03)
DCF12	1.29e-01(1.17e-02)	1.35e-01(7.80e-03)	1.32e-01(8.25e-03)
DCF13	1.40e-01(4.17e-02)	1.44e-01(4.13e-02)	1.43e-01(4.20e-02)
DCF14	1.47e-01(8.46e-03)	1.63e-01(1.51e-02)	1.58e-01(1.58e-02)

图 6. 去除 SKT 组件后在 14 个测试问题中三组不同参数下 MIGD 的均值和方差

Results Table

Problem	n_t, τ_t	DC-MFEA	
DCF1	3.91e-01(2.50e-02)	3.78e-01(2.95e-02)	3.63e-01(3.54e-02)
DCF2	6.28e-01(7.06e-02)	6.35e-01(6.42e-02)	6.33e-01(6.49e-02)
DCF3	5.04e-01(2.74e-02)	5.20e-01(1.22e-02)	5.10e-01(1.85e-02)
DCF4	4.54e-01(7.00e-02)	4.69e-01(6.82e-02)	4.59e-01(6.95e-02)
DCF5	3.61e-01(1.11e-01)	3.68e-01(1.01e-01)	3.66e-01(1.03e-01)
DCF6	1.31e-01(3.33e-03)	1.33e-01(1.14e-03)	1.31e-01(1.77e-03)
DCF7	3.82e-01(3.96e-02)	4.08e-01(1.65e-02)	3.93e-01(2.58e-02)
DCF8	7.66e-01(1.57e-01)	7.67e-01(1.79e-01)	7.55e-01(1.79e-01)
DCF9	4.17e-01(1.20e-01)	4.31e-01(1.21e-01)	4.28e-01(1.22e-01)
DCF10	2.51e+00(2.39e-02)	2.53e+00(1.99e-02)	2.52e+00(1.37e-02)
DCF11	8.45e-01(2.08e-02)	8.55e-01(1.09e-02)	8.48e-01(1.49e-02)
DCF12	1.17e+00(2.41e-02)	1.18e+00(1.12e-02)	1.18e+00(1.52e-02)
DCF13	9.74e-01(1.06e-01)	9.77e-01(1.11e-01)	9.76e-01(1.10e-01)
DCF14	1.19e+00(5.30e-02)	1.20e+00(4.51e-02)	1.19e+00(4.17e-02)

图 7. 去除 SKT 组件后在 14 个测试问题中三组不同参数下 MHV 的均值和方差

Results Table

Problem	n_t, τ_t	DC-MFEA	
DCF1	7.28e-02(1.16e-02)	4.09e-02(2.45e-02)	5.46e-02(2.91e-02)
DCF2	4.95e-02(5.42e-02)	4.47e-02(4.93e-02)	4.58e-02(4.99e-02)
DCF3	2.30e-02(1.65e-02)	2.15e-02(1.47e-02)	3.57e-02(2.41e-02)
DCF4	2.07e-02(1.37e-02)	2.12e-02(1.50e-02)	3.24e-02(2.25e-02)
DCF5	1.05e-02(4.20e-03)	1.01e-02(3.13e-03)	1.28e-02(4.66e-03)
DCF6	8.17e-03(4.27e-03)	8.07e-03(4.23e-03)	1.29e-02(5.50e-03)
DCF7	3.17e-02(1.94e-02)	2.94e-02(1.86e-02)	3.80e-02(2.33e-02)
DCF8	4.02e-02(1.49e-02)	3.97e-02(1.49e-02)	4.05e-02(1.31e-02)
DCF9	1.47e-02(2.95e-03)	1.52e-02(2.85e-03)	1.68e-02(4.46e-03)
DCF10	2.04e-01(1.28e-02)	2.04e-01(1.39e-02)	1.91e-01(1.48e-02)
DCF11	4.52e-02(8.27e-03)	4.56e-02(7.40e-03)	4.33e-02(7.97e-03)
DCF12	1.31e-01(9.38e-03)	1.32e-01(8.84e-03)	1.29e-01(1.03e-02)
DCF13	1.36e-01(4.68e-02)	1.35e-01(4.73e-02)	1.27e-01(4.68e-02)
DCF14	1.42e-01(9.13e-03)	1.61e-01(1.77e-02)	1.50e-01(1.74e-02)

图 8. 去除 TKT 组件后在 14 个测试问题中三组不同参数下 MIGD 的均值和方差

Results Table

Problem	n_t, τ_t	DC-MFEA	
DCF1	3.24e-01(1.37e-02)	3.71e-01(3.64e-02)	3.55e-01(4.13e-02)
DCF2	6.29e-01(6.98e-02)	6.34e-01(6.44e-02)	6.33e-01(6.54e-02)
DCF3	5.09e-01(2.34e-02)	5.11e-01(2.07e-02)	4.92e-01(3.28e-02)
DCF4	4.61e-01(7.09e-02)	4.59e-01(6.94e-02)	4.44e-01(7.37e-02)
DCF5	3.62e-01(1.10e-01)	3.68e-01(1.02e-01)	3.65e-01(1.03e-01)
DCF6	1.31e-01(3.41e-03)	1.31e-01(4.30e-03)	1.26e-01(5.15e-03)
DCF7	3.87e-01(3.95e-02)	3.90e-01(3.74e-02)	3.73e-01(4.42e-02)
DCF8	7.70e-01(1.56e-01)	7.58e-01(1.80e-01)	7.50e-01(1.82e-01)
DCF9	4.18e-01(1.20e-01)	4.31e-01(1.22e-01)	4.28e-01(1.23e-01)
DCF10	2.53e+00(1.78e-02)	2.53e+00(2.19e-02)	2.47e+00(8.02e-02)
DCF11	8.50e-01(1.75e-02)	8.52e-01(1.58e-02)	8.37e-01(2.29e-02)
DCF12	1.18e+00(1.74e-02)	1.18e+00(1.53e-02)	1.16e+00(2.49e-02)
DCF13	9.68e-01(1.01e-01)	9.65e-01(1.06e-01)	9.48e-01(1.04e-01)
DCF14	1.18e+00(5.22e-02)	1.19e+00(4.52e-02)	1.18e+00(4.68e-02)

图 9. 去除 TKT 组件后在 14 个测试问题中三组不同参数下 MHV 的均值和方差

结果分析:

首先对于在 14 个测试问题在每个时间步上的指标显示，如图 2和图 3。

在测试问题 DCF1 和 DCF3-DCF9 上的收敛性和多样性的表现都比较好，在 DCF2 测试问题中显示，在大概第 15 个时间步变化之后起伏较大，并在接下来几个时间步内依然没有降低，表示算法对该问题在动态处理上的能力较差一些。

在 DCF10-DCF14 中，对于环境变化的处理比较稳定，但是收敛性相对较差，可能是的是函数的前沿过于复杂导致。

对于在 14 个测试问题的 MIGD 和 MHV 指标显示，如图 4和图 5。

展示了算法在三组不同参数（模拟不同环境变化的剧烈程度和变化的频率）中的表现都相对比较稳定，说明算法整体通过空间知识的迁移不断找到最优解以及通过时间知识迁移组件应对环境的变化的处理存在优势。

在图 6 图 9，展示了对两个关键组件做消融实验的结果。

结果显示在去除这两部分关键组件与之前相比，收敛性与多样性多明显变差，可以得出结论这两部分关键组件是有效的。

6 总结与展望

本部分对整个文档的内容进行归纳并分析目前实现过程中的不足以及未来可进一步进行研究的方向。本文档首先介绍了当前动态约束多目标优化相关的引言以及相关工作，从当前所有的动态多目标的研究方法和约束多目标的研究方法再到仅有的几个对于动态约束多目标的处理方法。对于实际的动态约束多目标问题的处理能力都不尽人意，于是本文创新的提出了一种时空知识迁移的策略来弥补原有一些处理方式的短板，实现了不错效果。

然后对于算法的整体和关键部分进行了详细的描述并按照模块最后进行整体的复现，通过对实验结果的分析，验证了该算法的有效性，以及通过消融实验验证了关键组件的作用。

最后通过实验的结果分析可知，在部分相对没有这么复杂的测试问题中，该算法表现优秀且稳定性非常好。但是在一些可能有着复杂动态最优前沿的测试函数中表现非常不稳定。初步分析是由于这类测试问题带约束的最优前沿和去除约束的最优前沿之间相隔距离比较大，当算法在构建去除约束的辅助任务优化并通过 SKT 空间知识转移组件进行解的更新时候可能会误导优化的方向。

所以未来可能进一步的研究方向是针对不同问题自适应的调整主任务和辅助任务种群大小的比例。在检测到辅助任务与主任务之间最优前沿相距较近时候，可以加大辅助任务的种群比例，加速搜索更优解。在检测发现距离较大时，减小辅助任务的比例，防止对优化的方向产生影响，保证算法的稳定性。这样就可以在有着各种复杂前沿的实际问题中也可以稳定的发挥出算法应有的表现

参考文献

[1] M.-Y. Ameca-Alducin, M. Hasani-Shoreh, W. Blaikie, F. Neumann, and E. Mezura-Montes. A comparison of constraint handling techniques for dynamic constrained optimization problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018.

[2] R. Azzouz, S. Bechikh, L. B. Said, and W. Trabelsi. Handling time-varying constraints and objectives in dynamic evolutionary multi-objective optimization. *Swarm and Evolutionary Computation*, 39:222–248, 2018.

[3] R. Azzouz, S. Bechikh, and L. Ben Said. Multi-objective optimization with dynamic constraints and objectives: New challenges for evolutionary algorithms. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 615–622, New York, NY, USA, 2015. Association for Computing Machinery.

[4] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1875–1882 Vol. 3, 1999.

[5] 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*, 21(1):14–33, 2017.

[6] D. Chen, F. Zou, R. Lu, and X. Wang. A hybrid fuzzy inference prediction strategy for dynamic multi-objective optimization. *Swarm and Evolutionary Computation*, 43:147–165, 2018.

[7] G. Chen, Y. Guo, M. Jiang, S. Yang, X. Zhao, and D. Gong. A subspace-knowledge transfer based dynamic constrained multiobjective evolutionary algorithm. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–13, 2023.

[8] Q. Chen, J. Ding, S. Yang, and T. Chai. A novel evolutionary algorithm for dynamic constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 24(4):792–806, 2020.

[9] Q. Chen, J. Ding, G. G. Yen, S. Yang, and T. Chai. Multi-population evolution based dynamic constrained multiobjective optimization under diverse changing environments. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2023.

[10] R. Chen, K. Li, and X. Yao. Dynamic multiobjectives optimization with a changing number of objectives. *IEEE Transactions on Evolutionary Computation*, 22(1):157–171, Feb 2018.

[11] 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, 2007.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[13] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, and E. Goodman. An improved epsilon constraint-handling method in moea/d for cmops with large infeasible regions. *Soft Computing*, 23(23):12491–12510, 2019.

[14] M. Jiang, Z. Wang, H. Hong, and G. G. Yen. Knee point-based imbalanced transfer learning for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 25(1):117–129, 2021.

[15] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao, and K. C. Tan. A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning. *IEEE Transactions on Cybernetics*, 51(7):3417–3428, 2021.

[16] S. Jiang and S. Yang. A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 21(1):65–82, 2017.

[17] R. Jiao, B. Xue, and M. Zhang. A multiform optimization framework for constrained multiobjective optimization. *IEEE Transactions on Cybernetics*, 2022.

[18] J. Liang, X. Ban, K. Yu, B. Qu, K. Qiao, C. Yue, K. Chen, and K. C. Tan. A survey on evolutionary constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 27(2):201–221, 2023.

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

[20] Z. Ma and Y. Wang. Shift-based penalty for evolutionary constrained multiobjective optimization and its application. *IEEE Transactions on Cybernetics*, 53(1):18–30, 2023.

[21] A. Mani and C. Patvardhan. A novel hybrid constraint handling technique for evolutionary optimization. In *2009 IEEE Congress on Evolutionary Computation*, pages 2577–2583, 2009.

[22] A. Muruganantham, K. C. Tan, and P. Vadakkepat. Evolutionary dynamic multiobjective optimization via kalman filter prediction. *IEEE Transactions on Cybernetics*, 46(12):2862–2873, 2016.

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

[24] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, and C. Yue. An evolutionary multitasking optimization framework for constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 26(2):263–277, 2022.

[25] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, C. Yue, H. Lin, and K. C. Tan. Dynamic auxiliary task-based evolutionary multitasking for constrained multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2022.

[26] K. Qiao, K. Yu, B. Qu, J. Liang, H. Song, C. Yue, H. Lin, and K. C. Tan. Dynamic auxiliary task-based evolutionary multitasking for constrained multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2022.

[27] B. Y. Qu and P. N. Suganthan. Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods. *Engineering Optimization*, 43(4):403–416, 2011.

[28] R. Rambabu, P. Vadakkepat, K. C. Tan, and M. Jiang. A mixture-of-experts prediction framework for evolutionary dynamic multiobjective optimization. *IEEE Transactions on Cybernetics*, 50(12):5099–5112, 2020.

[29] T. Ray, H. K. Singh, A. Isaacs, and W. Smith. Infeasibility driven evolutionary algorithm for constrained optimization. In *Constraint-handling in Evolutionary Optimization*, pages 145–165. Springer, 2009.

[30] M. Rong, D. Gong, Y. Zhang, Y. Jin, and W. Pedrycz. Multidirectional prediction approach for dynamic multiobjective optimization problems. *IEEE Transactions on Cybernetics*, 49(9):3362–3374, 2019.

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

[32] Y. Tian, Y. Zhang, Y. Su, X. Zhang, K. C. Tan, and Y. Jin. Balancing objective optimization and constraint satisfaction in constrained evolutionary multiobjective optimization. *IEEE Transactions on Cybernetics*, 52(9):9559–9572, 2022.

[33] G. Toscano, R. Landa, G. Lárraga, and G. Leguizamón. On the use of stochastic ranking for parent selection in differential evolution for constrained optimization. *Soft Computing*, 21(16):4617–4633, 2017.

[34] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, 2009.

[35] D. Xu, M. Jiang, W. Hu, S. Li, R. Pan, and G. G. Yen. An online prediction approach based on incremental support vector machine for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2021.

[36] K. Yu, J. Liang, B. Qu, Y. Luo, and C. Yue. Dynamic selection preference-assisted constrained multiobjective differential evolution. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–12, 2021.

[37] H. Zhang, J. Ding, M. Jiang, K. C. Tan, and T. Chai. Inverse gaussian process modeling for evolutionary dynamic multiobjective optimization. *IEEE Transactions on Cybernetics*, pages 1–14, 2021.

[38] W. Zhou, L. Feng, K. C. Tan, M. Jiang, and Y. Liu. Evolutionary search with multi-view prediction for dynamic multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2021.