控制与决策
Control and Decision

基于多场景建模的动态鲁棒多目标进化优化算法

徐标, 吕修豪, 李文姬, 范衡, 巩敦卫, 贺杰

引用本文:
徐标, 吕修豪, 李文姬, 等. 基于多场景建模的动态鲁棒多目标进化优化算法[J]. 控制与决策, 2024, 39(12): 3997-4006.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2023.1641

基于多场景建模的动态鲁棒多目标进化优化算法

徐 标[1,2†], 吕修豪[1], 李文姬[1], 范 衡[3], 巩敦卫[4], 贺 杰[2]

(1. 汕头大学 工学院，广东 汕头 515063；
2. 梧州学院 广西机器视觉与智能控制重点实验室，广西 梧州 543002；
3. 电子科技大学(深圳)高等研究院，广东 深圳 518110；
4. 青岛科技大学 自动化与电子工程学院，山东 青岛 266100)

摘 要: 为了解决实际生产中的动态多目标优化问题,提出一种基于多场景建模的动态鲁棒多目标进化优化算法. 首先,所提出算法将不同环境下的问题视为不同场景,并通过相似度计算和场景聚类建立多个场景;然后,利用改进的多场景多目标进化优化算法求解各场景的折中解,当环境发生变化时,根据新问题所属的场景类,直接应用该场景类的折中解作为新问题的最优解,从而加快算法的响应速度;最后,通过对场景类中问题的约减,保留最具代表性的问题,逐步提高算法的鲁棒性,并降低解的切换成本. 实验结果表明,所提出算法能够快速响应环境变化,并提高解的鲁棒性.

关键词: 动态优化；多场景；多目标；鲁棒优化；进化算法

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2023.1641

引用格式: 徐标, 吕修豪, 李文姬, 等. 基于多场景建模的动态鲁棒多目标进化优化算法 [J]. 控制与决策, 2024, 39(12): 3997-4006.

Dynamic robust multi-objective evolutionary optimization algorithm based on multi-scenario modeling

XU Biao[1,2†], LV Xiu-hao[1], LI Wen-ji[1], FAN Zhun[3], GONG Dun-wei[4], HE Jie[2]

(1. College of Engineering，Shantou University，Shantou 515063，China；2. Guangxi Key Laboratory of Machine Vision and Intelligent Control，Wuzhou University，Wuzhou 543002，China；3. Shenzhen Institute for Advanced Study，UESTC，Shenzhen 518110，China；4. College of Automation and Electronic Engineering，Qingdao University of Science and Technology，Qingdao 266100，China)

Abstract: This paper proposes a dynamic robust multi-objective evolutionary optimization algorithm based on multi-scenario modeling, aiming to address dynamic multi-objective optimization problems in practical production. The algorithm treats problems in different environments as different scenarios and establishes multiple scenarios through similarity calculation and scenario clustering. Subsequently, it utilizes an improved multi-scenario multi-objective evolutionary optimization algorithm to find compromise solutions for each scenario. When the environment changes, the algorithm directly applies the compromise solution of the corresponding scenario class as the optimal solution for the new problem, thus speeding up the algorithm's response rate. Through reducing the number of problems in scenario classes and retaining the most representative ones, the algorithm gradually improves its robustness and reduces solution switching costs. Experimental results demonstrate that the proposed algorithm can rapidly respond to environmental changes and enhance solution robustness.

Keywords: dynamic optimization；multiple scenarios；multi-objective；robust optimization；evolutionary algorithm

0 引 言

工程中常见的问题,如调度和路径优化问题,具有动态变化的特点,包括目标函数、约束条件个数、参数以及决策变量的维度可能随环境改变而变化,导致最优解发生变化. 若问题涉及两个或以上目标函数,且这些目标函数间存在一定冲突,则称之为动态多目标优化问题(dynamic multi-objective optimization problems, DMOPs). 本文主要考虑目标函数参数随环

收稿日期: 2023-11-25；**录用日期:** 2024-03-05.

基金项目: 国家自然科学基金项目 (61961036, 62162054)；广东省基础与应用基础研究项目 (2023B1515120020, 2024A1515012450)；汕头大学科研启动项目 (NTF20009)；广西自然科学基金项目 (2020JJA170007)；广西科技基地和人才专项项目 (桂科 AD20297148)；广西创新驱动发展项目 (科技重大专项)(桂科 AA18118036)；广西机器视觉与智能控制重点实验室开放课题项目 (2022B04).

责任编委: 毛志忠.

†通讯作者. E-mail: xubiao@stu.edu.cn.

境动态改变的一类动态多目标优化问题. 不失一般性,该问题描述为

$$\min\ F(x,t)=(f_1(x,t),f_2(x,t),\ldots,f_m(x,t));$$
$$\text{s.t. } x\in\Omega. \tag{1}$$

其中: $x=(x_1,x_2,\ldots,x_n)\in\Omega\subset R^n$ 为决策向量; Ω 为 n 维决策空间; $F=(f_1,f_2,\ldots,f_m)\in\Lambda\subset R^m$ 为目标函数向量, Λ 为 m 维目标空间; t 为环境参数.

解决该类问题的常用方法为跟踪最优解方法(track the moving optima, TMO)[1-2]. 该方法的核心思想是在发现环境变化后立即重新进行优化,以快速、准确地找到适应新环境的Pareto最优解集(Pareto optimal set, PS(t), $t=1,2,\ldots,K$),其相应的最优解称为Pareto最优前沿(Pareto optimal front, PF(t), $t=1,2,\ldots,K$). 这类方法的核心在于如何在环境变化后快速响应,加速算法的收敛. 如文献[3-4]中提到的方法使用不同的预测技术来解决动态多目标优化问题. TMO方法旨在快速追踪当前环境下的最优解,但是面临环境变化快、切换成本高等挑战. 尤其是在环境变化频繁或问题复杂度高时,很难在有限时间内获得每个环境下的最优解,因此,频繁追踪新解通常并不切实可行.

Yu等[5]首次提出了基于时间的动态鲁棒最优解(robust optimization over time, ROOT)的概念,旨在降低操作切换代价,其核心思想是找到一组适用于未来多个连续动态变化问题的鲁棒解集;陈美蓉等[6]引入了新型鲁棒Pareto最优解(robust pareto optima over time, RPOOT),在动态多目标优化问题中描述了解集的生存时间和平均适应度,将目标函数转化为兼顾多个动态环境下解的平均适应度函数,并通过新型学习预测策略实现动态多目标鲁棒最优解集的求解. 尽管这些鲁棒优化算法在评估Pareto解集时表现出了优秀的性能,但是对于环境随机变化的适应性较弱,应用于实际问题时不理想.

文献[7-8]提出了多场景多目标进化优化算法,其核心是同时优化同一对象在不同场景下的多个优化模型,以获得适用于所有场景的最优解. 该算法能够找到分布性和收敛性俱佳的基于场景的非支配解,这些解代表了各场景的折中结果,而不仅仅是某一单一场景的最优解. 因此,这种多场景优化算法为解决鲁棒优化问题提供了新的思路.

本文将动态多目标优化问题看作不同场景的优化问题,并通过场景聚类建立多个代表性的场景类. 然后,使用多场景优化算法解决各场景类的优化问题. 当环境发生变化时,通过评估新问题与已有场景类的相关性,选择相关度最高的场景类的最优解作为新问题的解,并更新该场景类的代表性场景及其最优解. 所提出方法能够节省搜索时间,提高算法的响应速度,获得更鲁棒的解决方案.

1 研究现状

多场景多目标优化问题(multi-scenario multi-objective, MSMO)是多目标优化问题的扩展,涉及同一问题在不同条件下的不同表达式. 其最优解在所有场景下均是可行的,且是所有场景下问题的折中解. 虽然所得解不是每个场景下的最优解,但是能够满足所有场景下的优化要求. 若每个场景中的决策空间相同,且每个场景均包含一个多目标优化问题(multi-objective optimization problems, MOPs),则该问题称为多场景多目标优化问题. 其具体表达式为

$$\begin{aligned}&\min: F^{(1)}(x)=(f_1^{(1)}(x),f_2^{(1)}(x),\ldots,f_m^{(1)}(x));\\&\qquad\vdots\\&\min: F^{(S)}(x)=(f_1^{(S)}(x),f_2^{(S)}(x),\ldots,f_m^{(S)}(x)).\\&\text{s.t. } g_y^{(i)}(x)\leqslant 0;\\&\quad y=1,2,\ldots,Y;\\&\quad i=1,2,\ldots,S;\\&\quad x\in\Omega.\end{aligned} \tag{2}$$

其中: S 为场景总规模,且每个场景均包含 m 个目标和 Y 个约束.

Deb等[7]首次提出了多场景多目标优化的概念,并引入了一种新的支配排序方法——场景支配,提出了参考点选择策略,通过整合各场景信息来指导种群的进化方向,应用于进化过程中,选择满足多个场景需求的解. 表1为不同场景下,点 x 与点 y 的场景支配关系. 由表1可见:在场景1中,当 x ε-支配 y 时,若在场景2中满足 x ε-支配 y 或互不支配的情况,则称 x 基于场景支配 y;否则,需比较二者的拥挤距离. 另外,若在两个场景中 x 与 y 均互不 ε 支配,则也需要进一步计算它们的拥挤距离进而确定它们之间的支配关系.

表1 基于场景的支配原则

SC_1	SC_2		
	x ε-dominates y	ε-non dominates	y ε-dominates x
x ε-dominates y	选择 x	选择 x	比较拥挤距离
ε-non dominates	选择 x	比较拥挤距离	选择 y
y ε-dominates x	比较拥挤距离	选择 y	选择 y

2 基于多场景建模的动态鲁棒多目标进化优化算法

为了避免频繁搜寻动态问题的最优解,本文提出了以下解决方案:首先,将初始时刻的问题单独归为一类. 当环境变化时,将新问题的特征与已有多场景

类中问题的相关性进行比较,比较过程中出现两种情况:若新问题与已有场景类中问题的相关性较强(类型1),则将其加入相关性最大的场景类中,并使用多场景优化方法求解,或直接应用该场景类的最优解;若新问题与已有聚类中心相关性不强(类型2),则将其单独归为一类,并使用多目标优化算法求解. 随着时间推移,场景类中的问题逐渐增多,达到一定数量后对其进行约减,保留最具代表性的问题,从而使得问题的场景类逐步达到稳定状态.

在此基础上,所提出基于多场景建模的动态鲁棒多目标进化优化算法(dynamic robust multi-objective evolutionary optimization algorithm based on multi-scenario modeling, DMS)的流程如图1所示.

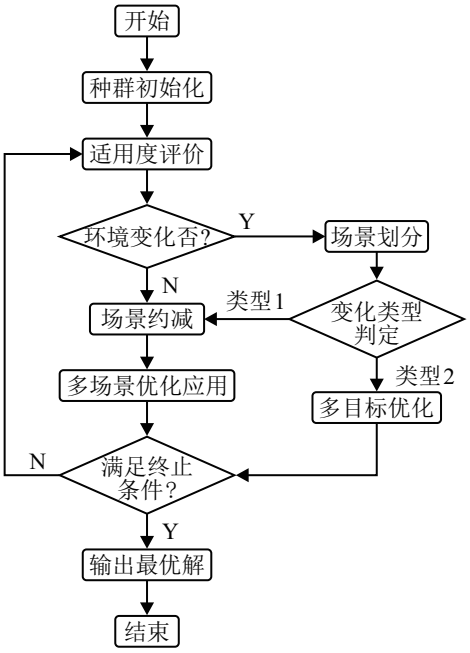


图 1　本文算法流程

由图1可见:所提出算法实现的关键步骤在于如何对场景进行划分,包括场景相似度的计算、场景约减、多场景应用、多场景优化以及应用策略. 下文将分别介绍.

2.1　场景划分

为了提高所提出算法的鲁棒性和响应速度,首先对历史场景问题进行聚类,形成多个多场景问题,并对每个多场景问题进行优化. 由于多场景问题的最优解是各场景问题的折中解,当新问题与已有场景问题相关性较高时,可直接应用已有场景问题的最优解,从而加快算法的响应速度. 同时,若新问题和历史问题属于同一类时,这种方法还增强了解的鲁棒性,降低了决策切换代价. 然而,如何对历史场景问题进行划分以及如何对问题规模进行约减来加快求解速度,仍然是待解决的关键问题.

2.1.1　格贴近度的定义

根据图1的框架,算法需要对场景进行划分. 每个时刻的目标函数均为多目标问题,因此在场景划分和后续的场景判断中,需要对问题进行描述,提取各时刻或类别的问题特征并进行比较,以判断不同场景间的关系. 传统的机器学习方法提取问题的特征向量会增加算法的时间复杂度,而模糊数学可直接比较两个模糊集,确定问题间的相似程度,避免了大量计算资源的占用.

所提出算法将场景类视为模糊范围,不采用统一特征描述. 它将新场景视为识别对象,在计算中,通过采点取样对场景类与新问题的贴近度进行比较,以判定场景. 若贴近度在一定范围内,则视为相同场景类;否则,视为新的场景类. 根据分类结果采用不同求解方式.

在进行场景刻画时,首先,在决策空间中随机采集一组样本点 $X=\{x_1,x_2,\dots,x_p\}$, p 为样本点的规模;然后,计算各场景在上述样本点下的目标函数值,将第 s $(s=1,2,\dots,S)$ 个场景问题的目标函数值记为 $(f_1^{(s)},f_2^{(s_i)},\dots,f_m^{(S)})$. 设 $\mathrm{SC}_n=\{F^{(n_1)},F^{(n_2)},\dots,F^{(n_k)}\}$ 为第 n 个场景类. 其中: $F^{(n_i)}(i=1,2,\dots,k)$ 为该类中的第 i 个场景, k 为该类中所包含场景的数量. 设新时刻的场景为 $F^{(t)}$,则场景类 SC_n 中场景问题的目标值矩阵为

$$U_{\mathrm{SC}_n}=\begin{bmatrix}(f_1^{(n_1)}(x_1),\dots,f_m^{(n_1)}(x_1)),\dots,\\(f_1^{(n_k)}(x_1),\dots,f_m^{(n_k)}(x_1))\\\vdots\\(f_1^{(n_1)}(x_p),\dots,f_m^{(n_1)}(x_p)),\dots,\\(f_1^{(n_k)}(x_p),\dots,f_m^{(n_k)}(x_p))\end{bmatrix},$$

新时刻的场景 $F^{(t)}$ 的目标函数值为

$$U_{F^{(t)}}=[(f_1^{(t)}(x_1),\dots,f_m^{(t)}(x_1)),\dots,(f_1^{(t)}(x_p),\dots,f_m^{(t)}(x_p))],$$

在后续计算过程中,使用模糊数学中格贴近度的定义进行场景相似度的判断,通过求取与场景类 SC_n 中所有场景的格贴近度的均值 $\bar{n}(U_{\mathrm{SC}_n},U_{F^{(t)}})$ 作为判断新时刻场景与该场景类的相似指标,并以此来判别相似程度(详见下文第2.1.2节场景划分).

在处理模糊的对象和模式库时,需要计算它们之间的距离或贴近度来衡量它们的关系. 下面给出格贴近度的定义[9].

定理1　设 A、$B\in F(U)$,则 $(A,B)=(A\odot B)\bigwedge(A\otimes B)^c$ 为模糊集 A、B 的贴近度,称为 A、B 的格贴近度,记作 $N(A,B)$,其在本文具体形式如下所示:

$$N(A,B)=\frac{1}{2}[A\odot B+(1-A\otimes B)],\qquad(3)$$

其中$(A\odot B)=\bigvee\limits_{u\in U}(A(u)\bigwedge B(u))$和$(A\otimes B)=\bigwedge\limits_{u\in U}(A(u)\bigvee B(u))$分别为模糊集$A$、$B$的内积和外积. 根据贴近度的性质发现,给定模糊集$A$,令模糊集$B$靠近$A$,会使得内积$(A\odot B)$增大而外积$(A\otimes B)$减少. 换言之,当$(A\odot B)$较大且$(A\otimes B)$较少时,$A$与$B$比较贴近,因此,采用内积与外积相结合的格贴近度$N(A,B)=(A\odot B)\bigwedge(A\otimes B)^c$来刻画两个模糊集的贴近程度是合理的[9].

模糊数学中模式识别的基本方法有最大隶属度原则和择近原则,要从一群模糊集$A_1,A_2,\ldots,A_n$中判定B归属于A_i中的哪一类,即当识别对象是模糊集的集合不是单个模糊集时,使用择近原则. 择近原则定义[9]如下.

定义1 (择近原则) 设A_i、$B\in F(U)(i=1,2,\ldots,n)$,若存在$i_0$,使得$N(A_{i_0},B)=\max|N(A_1,B),N(A_2,B),\ldots,N(A_n,B)|$,则认为$B$与$A_{i_0}$最贴近,即判定$B$与$A_{i_0}$为一类. 该原则成为择近原则.

根据上述格贴近度的相关理论,给出如下多场景问题的判别方法.

2.1.2 场景判别

在本文所研究的动态多目标优化问题中,为了方便表达,将当前时刻的各场景类SC_i记作$A_i=\{A_i(j),j=1,2,\ldots,P_i\}(i=1,2,\ldots,Q)$,即当前共有$Q$个场景类,每个类中包含$P_i$个场景,视$A_i$为模糊子集,环境变化后的新问题即新时刻场景$F^{(t)}$,记为$B$,表示待判别的场景,它们共同构成标准模型库,并判断新时刻场景所属的场景类. 具体的场景判别方法如下:首先,由式(3)计算场景B与第i个场景类中每个场景j的格贴近度$n_i(j)=N(A_i(j),B)(i=1,2,\ldots,Q;j=1,2,\ldots,P_i)$. 然后,由下式求出场景$B$与第$i$个场景类中所有场景的格贴近度的平均值:

$$\bar{n}_i=\frac{1}{P_i}\sum_{j=1}^{P_i}n_i(j),\ i=1,2,\ldots,Q. \tag{4}$$

若$n_k=\max\limits_{i=1}^{Q}\bar{n}_i$的值大于给定阈值$\delta$时,表明$B$与场景类$A_k$间的相关度较大,则将$B$并入场景类$A_k$中;否则,表明$B$与所有场景间均存在较大差异,则将$B$划分为一个单独的新场景. 上述过程的伪代码如算法1所示.

算法1 场景划分.

input: 已有场景类$A_i=\{A_{ij},j=1,2,\ldots,P_i\},i=1,2,\ldots,Q$;新环境下的问题$B$.

1. for $i=1:Q$
2. for $j=1:P_i$
3. $n_{ij}=N(A_{ij},B)$ //由式(3)计算B与A_{ij}的格贴近度
4. end for
5. $\bar{n}_i=\dfrac{1}{P_i}\sum\limits_{j=1}^{P_i}n_{ij}$
6. end for
7. if $(\max\limits_{l=1}^{i}(\bar{n}_l)\geqslant\delta)$
8. $A_l=A_l\bigcup\{B\}$
9. else
10. $A_{Q+1}=\{B\}$
11. end if

output: 输出$\{B\}$所在场景类.

本文利用模糊数学的择近原则来判断模糊标准库中的模型. 为了解决动态多目标优化问题中不同时刻目标问题的大幅变化,本文在择近原则上增加了限制条件:只有在满足较大阈值δ时才使用择近原则. 这确保了某一场景类内问题的相似性较高,提高了折中解对类内问题的收敛性. 当$\delta=1$时,每个场景问题均单独归为一类,每次环境变化后均需要重新寻优新问题,这与传统的TMO方法一致. 有关阈值设定的详细分析见后文第3.3.2节.

2.2 场景类规模约减

为了防止场景类中场景数量无限制增加从而影响求解效果,需要对规模较大的场景类进行约简,以保留最具代表性的问题,降低计算复杂度,提高问题求解的精度.

场景规模约简的目的是用少量场景问题更全面地代表整个场景类,即选出几个分散的场景问题代表该类. 当场景类中的问题数量达到一定值时,将对该场景类进行约简,通过比较问题间的分散程度确定最终保留的代表性问题. 具体做法如下.

引入使用轮廓系数中的内聚度的概念[10]. 对于一个场景类A_z,其类内场景i的内聚度计算公式为

$$a(i)=\frac{1}{|A_z|}\sum_{j\neq i}^{|A_z|}\mathrm{dis}(i,j),\ i=1,2,\ldots,|A_z|. \tag{5}$$

其中:$|A_z|$为类内样本点总数;j为类内与样本i相异的其他样本点;$\mathrm{dis}(i,j)$为i与j间的距离,它反映了样本点i与类内元素的紧密程度. $a(i)$越大,表明该类内元素越分散,类内元素间的冗余性越小. 类内元素数量相同的情形下,$a(i)$越大,类内元素对该类特征的覆盖越全面. 因此,在接下来的场景约减过程中,将内聚度大作为场景问题的选择标准,内聚度越大,类内元素对该类特征的覆盖越全面,越能够代表该类场景. 本研究中,仍然使用格贴近度作为计算内聚度的距离算子.

算法2为场景类规模约减. 在算法2中,若新环境问题被纳入某一场景类后,首先检查场景类内问题

数量是否超过规定值k. 若场景类内问题的规模大于k(第1行),则需对场景类问题进行约减,确保问题在规模为k的条件下更全面地覆盖整个场景特征. 若原有场景类规模小于k,则直接将新时刻问题B添加至该场景类并输出(第17行). 若规模不小于k,则在约减过程中(第2行～第17行),先计算原有场景问题类的平均内聚度$\bar{a}$(第2行),再计算新问题逐一替代类内原问题后场景类的平均内聚度数组$\bar{a}'$(第4行～第9行),比较$\bar{a}'$数组内的值与$\bar{a}$的大小:若新问题替换后场景类的内聚度小于原有场景类的内聚度,则直接使用原有场景类的解作为新时刻问题B的解(第13行);否则,选择内聚度最大的场景类作为新的场景类A_z输出.

算法2　场景类规模约减.

input: 新时刻场景问题B;类内其他场景问题$A_z=\{A_{zi}, i=1,2,\dots,|A_z|\}$;场景类内最大场景数量$k$.

1. if $|A_z|\geqslant k$

2. 计算原有平均内聚度$\bar{a}=\dfrac{1}{|A_z|}\sum\limits_{i=1}^{|A_z|}a(i)\Big/\!\Big/a(i)$由式(5)得到

3. $A'_z=A_z$

4. for $q=1:|A_z|$

5. $A'_{zq}=B$

6. 计算$\bar{a}'_q=\dfrac{1}{|A_z|}\sum\limits_{i=1}^{|A_z|}a'_q(i)\Big/\!\Big/$由式(5)得到

7. $A'_z=A_z$

8. end for

9. if $\max\limits_{q=1}^{|A_z|}\bar{a}'_q>\bar{a}$

10. $A_{zm}=B//m=\arg\max\limits_{q=1,2,\dots,|A_z|}\bar{a}'_q$

11. else

12. $\mathrm{PS}_B=\mathrm{PS}_W//$直接调用该场景类的解作为$B$的解

13. end if

14. else

15. $A_z=A_z\bigcup\{B\}$

16. end if

output: 该场景类下的代表性场景A_z.

根据算法2,场景类规模约减后有两种求解方法:直接使用现有场景类下的解或应用多场景多目标优化算法. 若新时刻B无需更新已有场景类A_z,则可直接使用其下的解作为新时刻B的解;若新时刻B替换或添加至场景类A_z,则需使用多场景优化算法进行寻优. 综上所述,本文提出了3种优化策略:1) 直接使用普通多目标优化算法求解;2) 调用历史时刻的多场景多目标优化解作为新时刻解集;3) 使用多场景多目标优化算法求解新时刻解集. 通过这3种优化策略,可根据新时刻问题与已有场景类的相似度和需求情况,选择不同的求解方法和策略,以提高求解效率和解的鲁棒性.

2.3　改进的多场景多目标优化算法

在上述步骤中,通过对新时刻问题进行场景判别和规模约简,得到了动态优化问题的多个场景类. 然后,采用多场景优化方法对每个场景类下的问题集进行求解,旨在获得各场景类问题的折中解. 本文采用文献[7]中提出的算法框架,并对其参考点选择策略进行改进,以更好地综合各场景的信息. 接下来,将详细介绍这一改进的多场景算法.

本文集成了一组场景共享参考解,这些解代表各场景的重要信息. 具体做法如下:首先,利用现有的多目标优化方法求解每个场景下的Pareto最优解集;然后,从每个解集中选择H个解,在各自的目标空间内均匀分布. 这些候选参考解被用于多目标优化问题的进化过程中,提供收敛的方向. 对于每个候选解集,计算其拥挤距离排序,并确定其决策变量的平均值,从而得到一组场景共享参考解,这些解在不同场景问题下具有不同的目标值.

在这项研究中,本文采用多场景多目标优化方法来处理动态多目标优化问题,具体步骤如下.

step 1: 将每个时刻的动态多目标优化问题视为一个独立的场景下的多目标优化问题;

step 2: 当检测到环境发生变化时,将当前时刻的场景与已有的场景类进行相似度比较,并确定当前时刻所属的场景类;

step 3: 若当前时刻的场景与某个已有场景类的相似度高于设定的阈值,则进行场景合并,并进行场景规模约简;

step 4: 对合并后的场景采用策略3)的多场景多目标优化方法求解,或直接调用该场景类下的解作为新场景问题的解;

step 5: 若当前时刻的场景与已有场景的相似度低于设定的阈值,则将该时刻作为一个新的场景类,并采用策略1),即普通的多目标优化方法进行求解;

step 6: 所有得到的场景类保存为一个场景类模型库,用于与接下来的新场景问题进行相似度判别,以获得满足多个场景的鲁棒解.

所提出方法充分利用已有场景信息,并有效地进行场景合并和求解选择,以获得满足多个场景要求的鲁棒解.

2.4 算法复杂度分析

为了方便分析,本文将总体规模、目标数量和相关总体子集规模分别记为N、m、T. 总体大小N由用户设定,目标个数m根据具体问题而定. 在算法1主循环中,最大复杂度为$O(N^2)$;在算法2中,最大复杂度为$O(N)$;而对于所提出算法,主要的计算复杂度来自多场景多目标优化算法,虽然已经做出改进,但其本身的复杂度为$O(N^3)$. 尽管如此,所提出算法通过场景判别等步骤,在整个动态问题中选择3种不同的计算策略,从而减少计算次数,降低计算资源消耗并减少解的切换代价.

3 实验部分

为了验证所提出算法的有效性,选择与已有动态鲁棒多目标优化算法[6]以及传统的动态多目标优化算法(如DNSGA-II[1])进行比较. 理由如下:它们均为基于NSGA-II[11]的算法框架,将它们作为对比算法,可以更好地分析新算法改进之处的效果. 另外,选择3种最先进的DMOEA作为比较算法:KT-DMOEA[12]利用基于Knee point的迁移学习;SEFS-DMOEA[13]算法关联最优解与权向量,构造时间序列训练在线预测模型;KTS-DMOEA[2]利用知识迁移,匹配迁移策略,生成新的初始种群. 因此,对比这两类算法可准确评估新算法中策略对性能的影响,避免算法框架差异引起的干扰.

本文根据决策者的使用对象和偏好设定场景相似度阈值,并测试不同参数取值以评估算法性能. 通过上述参数调整实验,为确定参数取值提供了指导.

3.1 基准测试函数和性能指标

所提出算法目前仅适用于两目标的多目标优化问题,并在12个常用的动态基准函数上进行了实验验证,包括FDA 1 ~ FDA 3[14]和DF 1 ~ DF 9[15]. 这些基准函数代表了最常见的动态多目标优化问题,并广泛应用于评估不同算法的性能. 这些问题涵盖了多种情况,如PS随时间变化而PF不变、PS和PF均随时间变化以及PS不变而PF随时间变化等,有助于全面评估所提出算法在不同情况下的表现和性能.

平均世代距离(mean inverted generation distance, MIGD)[16]是一种被广泛采用的度量,主要用于衡量算法得到的解的收敛性和多样性. 设PF_t^*为真实PF中的一组均匀分布的解,$\mathrm{PF}_t^{\mathrm{ob}}$为真实PF的一组近似解,则在$t$时刻,IGD可计算为

$$\mathrm{IGD}(\mathrm{PF}_t^*, \mathrm{PF}_t^{\mathrm{ob}}) = \sum_{g\in \mathrm{PF}_t^*} d(g, \mathrm{PF}_t^{\mathrm{ob}}) \Big/ |\mathrm{PF}_t^*|.$$

其中:$d(g, \mathrm{PF}_t^{\mathrm{ob}})$为$g$与$\mathrm{PF}_t^{\mathrm{ob}}$中各点间的最小欧氏距离,$|\mathrm{PF}_t^*|$为$\mathrm{PF}_t^*$中解的个数. 在DMOPs中,众多的环境表示存在不止一个IGD. 因此,本文提出一种改进的MIGD指标来估计DMOPs的性能,其为IGD的平均值. MIGD的定义如下所示:

$$\mathrm{MIGD} = \sum_{t\in T} \mathrm{IGD}(\mathrm{PF}_t^*, \mathrm{PF}_t^{\mathrm{ob}}) \Big/ |T|.$$

这里:T为离散时间点的集合,$|T|$为一次运行总的环境变化次数.

所提出算法通过问题聚类得到不同的场景类,其下的解适用于同一场景类中的所有问题. 本文引入解的适用度(average suitability, AS)来衡量解的适用性,即不同场景类下解适用的问题数量的平均值,有

$$\mathrm{AS} = |M_{\mathrm{PS}}|/|T|.$$

其中:M_{PS}为动态问题求解过程中去重后的解集合;$|M_{\mathrm{PS}}|$为在一次动态问题中去重后的解集数量,即算法求解的次数. 在实验中,计算每个解在每个场景类下的适用度,然后取这些适用度的平均值,得到解的平均适用度. 解的平均适用度越高,表示其在不同场景下的适应程度越好,具有越好的鲁棒性. 在动态鲁棒多目标优化问题中,解适用于问题的数量即为该组解的适用问题数. 使用TMO算法求解时,每个问题均分别优化,因此TMO方法所得解的适用问题数均为1,解的平均适用度也为1.

3.2 实验设置

1) 种群大小:在所有算法中种群大小N均被设置为100,从真正的PF中均匀地取约1 000个点,以计算性能指标.

2) 其他参数:比较算法中所有参数的设置与原始研究相同.

3) 停止标准和执行次数:对于每个测试问题,每种算法独立运行15次;动态环境变化频率$\tau_t = 20$或30;首次环境变化前算法共进化50代;动态环境变化强度$n_t = 10$;动态环境变化次数为100.

4) 相似度阈值δ:为了平衡所求解的收敛性与鲁棒性,本实验中所有算法的相似度阈值设定为0.8.

5) 多场景算法中场景数量k:多场景算法中场景数量k一般不超过3个,即目前只能解决2个或3个场景问题. 本实验中将k设置为3.

3.3 实验结果

3.3.1 所提出算法优化策略的具体表现

本节分析所提出算法在解决FDA 1问题时的IGD曲线,如图2所示. 由图2可见,本文将其变化趋势分为3种情形:1) 平滑下降:在新场景类下使用普通多目标优化算法求解,IGD值呈平滑下降趋势,如图2中点A与点B间的曲线. 2) 水平线:表示新场景类与已有场景类相似度高,直接调用已有场景类的解

作为问题解, 导致IGD值保持不变, 如点 D 与点 E 间的横线. 3) 小幅度扰动式变化: 表示对新场景问题进行场景聚类合并后使用多场景多目标优化算法进行寻优, 导致IGD值出现小幅度扰动变化, 如图2中点 B 与点 C 间的曲线部分. 综上所述, 直接调用已有场景类的解以及使用多场景优化算法寻求折中解能够提高该场景类下解的鲁棒性能, 使其适用于更多的场景问题数量.

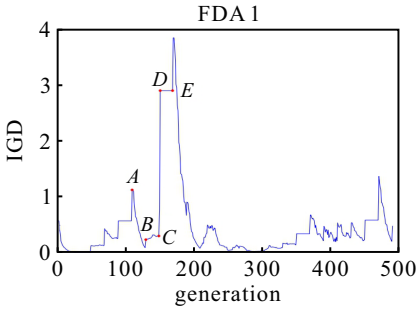


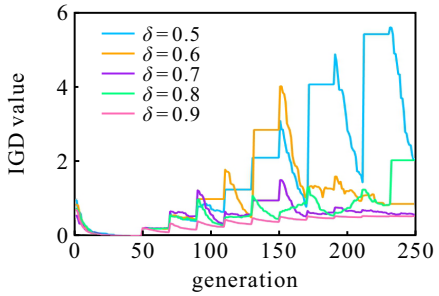
图 2　本文算法在FDA 1上的IGD示意图

值得说明的是, 图2中的IGD值有明显的跳跃点, 如点 A 和点 D 所示, 这是由于当环境发生变化后, 原场景问题的解不再适用于新问题, 无法归入上一时刻场景问题所属的场景类中, 上一时刻的解对于新时刻问题而言并非最优解.

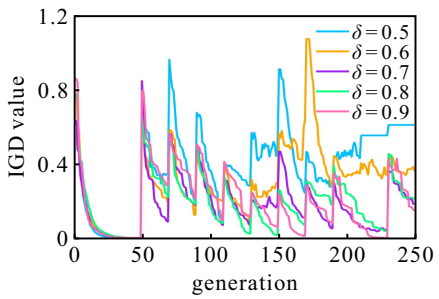
3.3.2　所提出算法中参数的敏感性分析

在所提出算法中, 需要通过比较问题间的相似度与相似度阈值的大小关系来判断新时刻对应的场景问题所属的类, 即相似度阈值对算法的精确性有直接影响. 故讨论相似度阈值的敏感性分析是有必要的.

在相似度阈值敏感性分析中, 考虑阈值较小时降低了问题间的区分度, 导致场景类中问题规模增大, 折中解收敛性变差. 本文讨论 δ 为0.9、0.8、0.7、0.6和0.5的情况, 而对于 $\delta = 1$ 的情况即为TMO方法, 不在此讨论范围内. 图3(a)和图3(b)分别为算法在测试问题FDA 3和DF 2上, 当相似度阈值取不同值且环境发生10次变化时, 所得到的IGD示意图. DF 2具有一个简单的动态, 其PF随时间保持不变, 而FDA 2的PS和PF随时间变化. 这两个测试问题可以很好地展示所提出算法在不同策略下的求解情况, 并显示参数设置对于求解动态多目标优化问题的影响.



(a)　FDA 3



(b)　DF 2

图 3　不同相似度阈值下本文算法在FDA 3与DF 2上的IGD比较

由图3(a)可得到如下结论: 1) 随着相似度阈值 δ 的减小, 算法DMS得到的解的IGD变化幅度增大. 这是因为较小的 δ 值要求场景类内问题的相似度更高, 导致多场景最优解偏离真实PF, 从而导致IGD值变差. 2) 曲线出现明显的转折点, 这是由于环境变化后, 问题间的相似度降低, 导致上一时刻场景类问题的解不再适用于下一时刻的新问题, IGD值增加. 3) 随着 δ 的增大, 算法求解FDA 3的IGD值减小, 表明其收敛性更好. 这是因为较大的 δ 值导致场景聚类方法失效, 仅使用普通多目标优化算法求解单一时刻问题, 提升了解的鲁棒性和适应性.

由图3(b)可得出如下结论: 1) 不同 δ 取值情况下, 环境变化后的IGD值呈现相似的平缓下降趋势. 这是因为新时刻问题与已有场景类的相似度未达到阈值 δ, 直接使用普通多目标优化方法求解, 导致IGD值保持稳定. 2) 尽管不同 δ 取值情况下, IGD值的变化趋势相似, 但是在同一问题上具体变化有所差异. 如当 $\delta = 0.7$ 时, IGD值可能出现小幅度扰动甚至逐渐增加, 这是因为新时刻问题与已有场景类合并, 并应用多场景优化算法寻求折中解. 3) 随着 δ 取值增大, 所提出算法在该测试问题上表现的收敛性越来越好. 但是当 δ 取值过大时, 所有时刻均被视为新的场景类, 无法充分利用历史时刻信息, 导致解的鲁棒性下降.

设定不同的问题相似度阈值对算法的效果至关重要. 当决策者需要高鲁棒性而对问题精度要求不高时, 可将相似度阈值调小; 反之, 若对问题的精度要求高, 则可适度增加相似度阈值来获取更高精度的解. 为了平衡解的收敛性与鲁棒性, 在接下来的实验中, 统一将 δ 值设置为0.8.

3.3.3　不同算法性能比较

本文验证了所提出算法的有效性, 通过与TMO和RPOOT进行实验比较, 并分析了不同算法的实验结果. 表2和表3为3种算法的MIGD值和AS值. 采用Mann-Whitney U检验, 在0.05显著水平下对不同

表 2 算法在不同环境变化条件下的MIGD比较

problems	(τ_t, n_t)	DNSGAII	KT-DMOEA	SEFS-DMOEA	KTS-DMOEA	RPOOT	DMS
FDA 1	(20,10)	0.013+ (6.2e-03)	0.103 7+ (1.71e-02)	0.007 9+ (9.308e-04)	0.011 0+ (3.238e-04)	0.003 9+ (1.788 6e-04)	0.232 3 (1.512e-01)
	(30,10)	0.008 8+ (2.4e-03)	0.102 8+ (1.13e-02)	0.005 7+ (3.363e-04)	0.010 3+ (2.966e-04)	0.004 0+ (2.684e-04)	0.273 1 (2.416e-01)
FDA 2	(20,10)	0.083 3+ (8.1e-03)	0.475 0− (4.71e-02)	0.007 5+ (4.044e-04)	0.287− (2.18e-01)	0.270 3= (1.07e-02)	0.236 2 (1.071e-01)
	(30,10)	0.083 4+ (8.2e-03)	0.556 6− (5.49e-02)	0.007 7+ (4.381e-04)	0.215= (2.92e-01)	0.272 9= (9.090e-04)	0.275 3 (9.57e-02)
FDA 3	(20,10)	0.080 4+ (1.103e-01)	0.343 5+ (1.012e-01)	0.288 8+ (8.34e-02)	0.011 9+ (1.0e-03)	0.261 3+ (2.0e-01)	0.636 (4.262e-01)
	(30,10)	0.075 2+ (1.145e-01)	0.442 1+ (1.052e-01)	0.257 8+ (7.87e-02)	0.010 9+ (6.178e-04)	0.245 2+ (2.134e-01)	0.569 7 (3.221e-01)
DF 1	(20,10)	0.009 4+ (3.1e-03)	0.178 2+ (2.69e-02)	0.009 7+ (1.0e-03)	0.008+ (4.025e-04)	0.025 5+ (1.248e-04)	0.311 2 (1.944e-01)
	(30,10)	0.006 9+ (2.9e-03)	0.054 0+ (9.8e-03)	0.005 8+ (3.217e-04)	0.007 5+ (2.691e-04)	0.025 5+ (6.059 1e-05)	0.338 9 (2.015e-01)
DF 2	(20,10)	0.010 2+ (4.8e-03)	0.079 1+ (1.33e-02)	0.024+ (4.6e-03)	0.010 5+ (1.5e-03)	0.004 0+ (2.549e-04)	0.311 2 (1.944e-01)
	(30,10)	0.006 9+ (1.5e-03)	0.069 9+ (9.3e-03)	0.024 8+ (2.5e-03)	0.010 5+ (1.6e-03)	0.004 1+ (1.89e-04)	0.254 1 (1.964e-01)
DF 3	(20,10)	0.011 7+ (3.3e-03)	0.234 7= (5.23e-02)	0.009 7+ (1.4e-03)	0.273 7= (2.77e-02)	0.318 7= (9.17e-02)	0.347 3 (1.848e-01)
	(30,10)	0.008 5+ (2.5e-03)	0.398 0= (4.90e-02)	0.007 8+ (6.608e-04)	0.267 8= (3.07e-02)	0.340 0= (4.2e-02)	0.393 7 (1.697e-01)
DF 4	(20,10)	0.116 1+ (1.26e-01)	0.362 7+ (1.019e-01)	0.360 7+ (1.272e-01)	0.095 6+ (2.7e-03)	0.515 9+ (4.155e-04)	0.733 8 (4.424e-01)
	(30,10)	0.114 9+ (1.272e-01)	0.362 9+ (1.176e-01)	0.360 0+ (1.292e-01)	0.099 8+ (1.3e-03)	0.516 5+ (1.349e-04)	0.866 0 (6.558e-01)
DF 5	(20,10)	0.012 8+ (6.0e-03)	0.317 2+ (3.38e-02)	0.009 5+ (1.1e-03)	0.016 9+ (2.7e-03)	0.022 3+ (1.113 2e-04)	0.584 1 (6.327e-01)
	(30,10)	0.009 7+ (4.4e-03)	0.061 2+ (8.0e-03)	0.006 0+ (3.433e-04)	0.013 8+ (1.1e-03)	0.022 2+ (1.104e-04)	0.473 4 (5.619e-01)
DF 6	(20,10)	1.251 1+ (3.542 6)	3.026 7− (5.646e-01)	3.497− (3.468e-01)	0.476 8+ (1.167e-01)	0.064 3+ (1.46e-02)	1.835 5 (2.463 4)
	(30,10)	1.459 3+ (4.501 2)	2.727 9+ (5.184e-01)	3.012 6= (4.190e-01)	0.422 0+ (1.190e-01)	0.071 0+ (1.05e-02)	3.430 4 (3.950 4)
DF 7	(20,10)	0.012 1+ (2.8e-03)	1.454 9+ (2.662e-01)	0.147 5+ (3.91e-02)	0.046 5+ (3.0e-03)	0.023+ (2.7e-03)	2.195 8 (1.077 3)
	(30,10)	0.010 1+ (1.5e-03)	0.576 4+ (6.77e-02)	0.145 0+ (1.87e-02)	0.044 3+ (2.4e-03)	0.021 8+ (1.6e-03)	2.150 2 (1.129 2)
DF 8	(20,10)	0.010 7+ (4.0e-03)	0.578 8− (9.06e-02)	0.098 7+ (1.45e-02)	0.022 1+ (2.9e-03)	0.232 3− (7.26e-02)	0.153 9 (1.502e-01)
	(30,10)	0.008 7+ (3.4e-03)	0.961 4− (2.162e-01)	0.054 8+ (1.11e-02)	0.020 9+ (3.0e-03)	0.275 8− (7.42e-02)	0.166 6 (1.563e-01)
DF 9	(20,10)	0.030 2+ (1.46e-02)	1.065 2+ (2.811e-01)	0.093 8+ (1.25e-02)	0.101 9+ (9.7e-03)	0.016 5+ (2.0e-03)	4.275 (4.432 9)
	(30,10)	0.015 3+ (7.1e-03)	1.060 0+ (2.40e-01)	0.028 5+ (3.5e-03)	0.085 7+ (4.7e-03)	0.014 9+ (7.76e-04)	3.139 2 (3.174 4)
	+/=/−	24/0/0	17/2/5	22/1/1	20/3/1	18/4/2	

方法得到的指标值进行假设检验,以评估不同方法性能的差异显著性. 使用“+”“−”和“=”表示对比算法的性能优于、劣于或近似于所提出DMS算法. 每种指标的最佳平均值以灰色背景突显. 接下来,将分别对表2和表3进行分析.

根据表2中的MIGD值,TMO算法在收敛性方面表现最佳,大多数测试问题均能够收敛至真实PF;其次为RPOOT算法,DMS表现最差. 前两种算法针对单一问题进行优化,而DMS则综合多个时刻的问题信息,寻找满足多场景要求的折中解,因此在收敛性能上稍显不足. 尽管DMS相较于TMO算法表现较差,但是与RPOOT算法相比,其性能持平甚至更优,如DF 8问题. 对于未能达到理想结果的DF 6、DF 7和DF 9问题,进行如下分析:DF 6问题的相似性太高,多场景参考点限制了解向真实PF的收敛,可能导致算法在搜索空间受限;相反,DF 7问题的相似性较低,且在不同阶段相似度变化较大,可能导致算法在求解过程中发生较大波动,影响最终结果;至于DF 9问题,不同时刻PS和PF的不连续性是挑战之一,可能导致算法无法准确捕捉PF的演化过程,从而影响结果质量.

表 3　算法在不同环境变化条件下的MAS比较

problems	(τ_t, n_t)	DNSGAII	KT-DMOEA	SEFS-DMOEA	KTS-DMOEA	RPOOT	DMS
FDA 1	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.921 1− (8.8e-03)	3.581 1 (1.751 7)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.921 1− (8.8e-03)	6.691 1 (4.967 1)
FDA 2	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	8.333+ (0)	8.031 6 (7.705e-01)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	4.980 4− (3.251 4)	7.169 0 (1.731 0)
FDA 3	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.253 3− (1.27e-02)	2.188 7 (0.278 2)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.047 1− (0)	2.114 3 (5.061e-01)
DF 1	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.621 7+ (1.27e-02)	1.443 7 (6.73e-02)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.031 6− (2.8e-03)	1.610 9 (5.18e-02)
DF 2	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.031 6− (2.8e-03)	3.222 8 (6.888e-01)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.031 6− (2.8e-03)	3.538 3 (6.161e-01)
DF 3	(20,10)	1− (0)	1− (0)	1− (0)	1= (0)	2.851 2+ (1.272e-01)	1.020 4 (0)
	(30,10)	1− (0)	1− (0)	1− (0)	1= (0)	1.521 5+ (1.67e-02)	1.010 (0)
DF 4	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.299− (0)	2.101 2 (1.474e-01)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.299− (0)	2.257 (6.193e-01)
DF 5	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.299− (0)	4.987 5 (3.366)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.041 7− (0)	3.990 7 (3.081 2)
DF 6	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.010− (0)	37.222 (8.253 6)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.010− (0)	41.667 (9.449 1)
DF 7	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.075 2= (0)	1.059 5 (1.388 9)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.010− (0)	1.069 3 (1.22e-02)
DF 8	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.020− (0)	4.403 6 (1.759 1)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.521 5− (1.67e-02)	4.175 (8.804e-01)
DF 9	(20,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.020− (0)	8.750 7 (1.591 4)
	(30,10)	1− (0)	1− (0)	1− (0)	1− (0)	1.021 8− (3.7e-03)	8.325 1 (1.940 7)
	+/=/−	0/0/24	0/0/24	0/0/24	0/0/24	4/1/19	

对比表3中的AS指标得出如下结论: TMO算法能够实时追踪问题的PF, 保证解仅适用于对应时刻的问题, 因此所有解的AS (解的适用度) 均为1. 相较之下, RPOOT算法和DMS算法中, 一些解不仅适用于一个时刻的问题, 还可能作为多个时刻的鲁棒解, 因此在这两种算法下, AS的值大于等于1. 分析表3数据发现, 所提出算法在AS指标上表现最佳. 具体而言, 在12个测试问题的24组实验中, 有19组实验表现明显优于RPOOT算法. 如在DF 8测试问题上, DMS算法所得解适用于的问题数量是RPOOT算法的3～4倍. 所提出算法在大部分测试问题上均取得了较高的AS值, 即较好的适用时刻数量.

通过比较表2与表3, 可得出以下结论: DMS算法在平衡解质量与鲁棒性方面取得了一定的成就, 显示出了在动态多目标优化问题中寻求鲁棒解的潜力. 尽管DMS方法得到的解并非最接近PF的解集, 但是从解的适用场景总数来看, DMS的解能够满足更多时刻的问题, 这简化了计算过程, 有效减少了解的切

换次数和成本.

4 结 论

针对动态多目标优化问题,传统的TMO方法需要实时追踪最优解集,增加了计算负担和解切换成本. 相比之下,本文提出了一种基于多场景建模的动态鲁棒多目标进化优化算法. 所提出算法通过综合问题相似度进行场景聚类,并采用多场景优化方法求解最优折中解. 在环境变化时,根据新问题与历史场景类的相似性,决定直接调用相关场景类的解或重新寻优. 实验结果表明,与TMO方法相比,所提出算法具有更强的鲁棒性且避免了高计算负担和解切换成本. 与基于RPOOT的方法相比,所提出算法不依赖准确的预测方法,提高了对环境变化的响应速度和准确性,但是在高维目标优化问题方面仍然有待加强. 未来的研究将进一步完善多场景多目标优化算法的框架,并探索与机器学习相结合来提高问题相似度判定速度和准确性.

参考文献(References)

[1] Deb K, Rao N U B, Karthik S. Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling[C]. International Conference on Evolutionary Multi-Criterion Optimization. Heidelberg, 2007: 803-817.

[2] Guo Y N, Chen G Y, Jiang M, et al. A knowledge guided transfer strategy for evolutionary dynamic multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2023, 27(6): 1750-1764.

[3] Jiang M, Huang Z Q, Qiu L M, et al. Transfer learning-based dynamic multiobjective optimization algorithms[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(4): 501-514.

[4] 呼子宇, 李紫晗, 孙浩, 等. 基于决策变量关系的动态多目标优化算法[J]. 控制与决策, 2024, 39(1): 78-86.
(Hu Z Y, Li Z H, Sun H, et al. A dynamic multi-objective optimization algorithm based on the relationship of decision variables[J]. Control and Decision, 2024, 39(1): 78-86.)

[5] Yu X, Jin Y C, Tang K, et al. Robust optimization over time — A new perspective on dynamic optimization problems[C]. IEEE Congress on Evolutionary Computation. Barcelona, 2010: 1-6.

[6] 陈美蓉, 郭一楠, 巩敦卫, 等. 一类新型动态多目标鲁棒进化优化方法[J]. 自动化学报, 2017, 43(11): 2014-2032.
(Chen M R, Guo Y N, Gong D W, et al. A novel dynamic multi-objective robust evolutionary optimization method[J]. Acta Automatica Sinica, 2017, 43(11): 2014-2032.)

[7] Deb K, Zhu L, Kulkarni S. Handling multiple scenarios in evolutionary multiobjective numerical optimization[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(6): 920-933.

[8] Zhao C L, Zhou Y R, Lai X S. An integrated framework with evolutionary algorithm for multi-scenario multi-objective optimization problems[J]. Information Sciences, 2022, 600: 342-361.

[9] 杨纶标, 高英仪. 模糊数学原理及应用[M]. 第4版. 广州: 华南理工大学出版社, 2005: 1-281.
(Yang L B, Gao Y Y. Principle and application of fuzzy mathematics[M]. The 4th edition. Guangzhou: South China University of Technology Press, 2005: 1-281.)

[10] 张朝, 郭秀娟, 张坤鹏. K-means算法聚类中心选取[J]. 吉林大学学报: 信息科学版, 2019, 37(4): 437-441.
(Zhang Z, Guo X J, Zhang K P. Clustering center selection on K-means clustering algorithm[J]. Journal of Jilin University: Information Science Edition, 2019, 37(4): 437-441.)

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

[12] Zhou A M, Jin Y C, Zhang Q F. A population prediction strategy for evolutionary dynamic multiobjective optimization[J]. IEEE Transactions on Cybernetics, 2014, 44(1): 40-53.

[13] Sun J, Gan X J, Gong D W, et al. A self-evolving fuzzy system online prediction-based dynamic multi-objective evolutionary algorithm[J]. Information Sciences, 2022, 612: 638-654.

[14] Goh C K, Tan K C. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(1): 103-127.

[15] Jiang S Y, Yang S X, Yao X, et al. Benchmark problems for CEC2018 competition on dynamic multiobjective optimisation[J]. Computer Science, 2018: 1-18.

[16] Zeng S Y, Jiao R W, Li C H, et al. A general framework of dynamic constrained multiobjective evolutionary algorithms for constrained optimization[J]. IEEE Transactions on Cybernetics, 2017, 47(9): 2678-2688.

作者简介

徐标(1981−), 男, 讲师, 博士, 主要研究方向为智能优化与应用、动态多目标优化, E-mail: xubiao@stu.edu.cn;

吕修豪(1999−), 女, 硕士生, 主要研究方向为智能优化与应用, E-mail: 21xhlv@stu.edu.cn;

李文姬(1988−), 男, 讲师, 博士, 主要研究方向为智能优化与应用、因果发现, E-mail: liwj@stu.edu.cn;

范衡(1974−), 男, 教授, 博士, 主要研究方向为智能优化与应用、机器人设计与控制, E-mail: zfan@stu.edu.cn;

巩敦卫(1970−), 男, 教授, 博士, 主要研究方向为智能优化与控制、软件测试, E-mail: dwgong@vip.163.com;

贺杰(1982−), 男, 教授, 博士, 主要研究方向为图形图像处理、三维可视化技术, E-mail: hnu.hejie@gmail.com.