角度修正和分级多种群的动态多目标进化算法

杨 乐,马永杰*,平镐羽,杨 岳
(西北师范大学物理与电子工程学院,甘肃兰州 730070)

摘 要: 为更好地应对动态多目标优化中的环境变化,提出了一种对差分向量进行角度修正以及分级多种群协同进化(Angle Correction and Hierarchical Multi-Population, ACHMP)的进化算法. 根据历史信息,使用无迹卡尔曼滤波模型来预测种群的中心点,通过不同时刻的中心点产生不同的差分向量,再使用无迹卡尔曼滤波对差分向量进行角度修正;提出的多种群协同进化模式将种群分为三部分并使其沿不同的方向进化,子种群监督主种群进化,在提升了算法性能的同时,也保证了种群的多样性. 与10种对比算法在不同测试问题上的实验结果显示,ACHMP算法的性能总体优于其他算法,证明了本文提出的角度修正和分级多种群方法在处理动态多目标优化问题时具有较强的竞争力.

关键词: 动态多目标优化;差分向量角度修正;分级多种群;无迹卡尔曼滤波;预测策略
基金项目: 国家自然科学基金(No.62066041)
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112(2024)09-3278-13
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20221330

Dynamic Multi-Objective Evolutionary Algorithm Based on Angle Correction and Hierarchical Multi-Population

YANG Le, MA Yong-jie*, PING Hao-yu, YANG Yue
(*School of Physics and Electronic Engineering*, *Northwest Normal University*, *Lanzhou*, *Gansu* 730070, *China*)

Abstract: In order to better cope with the environmental changes in dynamic multi-objective optimization, an evolutionary algorithm with angular correction of difference vectors and hierarchical multi-population co-evolution (ACHMP) is proposed. According to the historical information, use the unscented Kalman filter model to predict the population centroids, generate different difference vectors through different centroids at different times, and then use the unscented Kalman filter to correct the angle of the difference vectors. A multi-population coevolution model is proposed, which divides the population into three parts to evolve in different directions. The sub-population supervises the evolution of the master population, which not only improves the performance of the algorithm, but also ensures the diversity of the population. Experimental results with 10 comparison algorithms on different test problems show that the ACHMP algorithm performs better than the other algorithms in general, which proves that the angle correction and hierarchical multi-population method proposed in this paper has strong competitiveness in dealing with dynamic multi-objective optimization problems.

Key words: dynamic multi-objective optimization; difference vector angle correction; hierarchical multi-population; unscented Kalman filter; prediction strategy
Foundation Item(s): National Natural Science Foundation of China (No.62066041)

1 引言

多目标(Multi-objective Optimization Problems, MOPs)优化问题是一种同时含有多个互相冲突的目标函数的优化问题[1]. 目标函数、约束条件以及相关参数随时间或环境改变而动态变化的问题则称为动态多目标优化问题(Dynamic Multi-objective Optimization Problems,DMOPs). 对于DMOPs,环境发生变化后存在多个初始点,而进化算法(Evolutionary Algorithms,EAs)基于种群的属性使每次搜索均得到一组解[2],非常适合处理这类问题,因此动态多目标优化进化算法(Dynamic Multi-objective Optimization Evolutionary Algorithm, DMOEA)应运而生[3]. DMOEA采用种群收敛的方式搜

收稿日期:2022-11-21;修回日期:2023-07-17;责任编辑:梅志强
*通讯作者:马永杰

索最优解,然而多样性的丢失使算法在进化后期难以快速应对环境变化. 因此,如何快速收敛种群以及维持种群多样性是处理DMOPs的棘手问题.

DMOPs中的环境变化主要是指Pareto最优解集(Pareto optimal Set,PS)和Pareto最优前沿(Pareto optimal Front,PF)随时间的变化,可分为以下4类[4]:(1)PS随时间变化,而PF不随时间变化;(2)PS和PF都随时间变化;(3)PS不随时间变化,而PF随时间变化;(4)问题环境发生改变,但PS和PF都不变化. 环境检测方法主要有2种,即重新评估解和目标函数值的分布估计[5]. 前者通过对一部分个体重新评估,观测相邻两次迭代目标函数值差异是否超过某个规定值;后者则判断相邻两次迭代的目标解集是否属于同一种统计分布. 检测到环境变化后,如何响应环境变化尤为重要.

根据环境响应策略,可以将现有的DMOEA分为基于记忆、多种群、多样性以及预测的方法. 记忆的方法是将历史信息重新调用到当前种群,并对其进行筛选以便于获取最优解,以此来提高适应度. Azzouz等[6]提出了一种基于记忆、局部搜索和随机策略的自适应混合种群管理方法,根据变化的严重程度来调整随机解和内存的大小. 多种群的方法是将种群分解成多个子种群,同时跟踪子种群的变化,能够增加多样性. Yang等[7]提出了一种用于动态多目标优化的生物启发自学习协作算法,通过共同进化多个子种群来快速逼近Pareto最优解的位置. 多样性的方法是通过变异、重新初始化或随机插入新个体的方式来维持种群多样性. Li等[8]通过引入高突变的方式增强种群的多样性,Ruan等[9]提出了一种混合多样性维护方法,提高了预测精度. 预测的方法是使用历史信息设计一个预测模型来预测新种群,更好地适应环境变化. Zhou等[5]提出了一种基于种群预测策略的算法,结合中心点预测和流形预测来初始化种群;郑金华等[10]通过记录每次环境变化初始时和自主进化后种群中心点位置的前后变化预测最优解的所在方向;马学敏等[11]提出了一种基于多区域中心点预测的动态多目标优化算法求解复杂的DMOPs;Jiang等[12]则引入了迁移学习提高算法运行效率;李二超等[13]根据PS的变化类型,采取不同的预测策略有针对性的进行预测;Chen等[14]使用混合预测策略协调基于中心点的预测和指导基于个体的预测提高预测准确度,并结合精确可控的突变策略控制解的变异程度提高种群多样性.

目前,基于预测的方法应用相对广泛,但预测准确度是引导种群向正确方向进化的关键因素. 由于卡尔曼滤波(Kalman Filter,KF)具有及时跟踪系统变化的优点,而且计算复杂度低、抗干扰性强,因此引入KF处理DMOPs是一种重要的尝试[15,16]. 其优势体现在巧妙的融合了观察数据与估计数据,并将误差限定在一定范围内,但仅适用于处理一些较为简单的问题. 对于DMOPs,非线性是问题的常见特征之一,同时无迹卡尔曼滤波(Unscented Kalman Filter,UKF)的最大优势之一就是处理非线性问题[17],因此将其应用于DMOPs中的效果令人期待. 本文提出了一种基于角度修正和分级多种群的动态多目标进化算法(Angle Correction and Hierarchical Multi-Population,ACHMP). 其基本思想是,将种群分为三部分,分别沿着不同的差分向量方向进行进化. 当环境发生变化时,使用UKF模型来预测下一时刻的中心点,同时通过使用近似理想Pareto最优解集的中心点修正预测的中心点,使用两种不同的中心点预测方式产生不同的中心点和差分向量,并使用无迹卡尔曼滤波对其中一个差分向量进行优化修正. ACHMP的主要贡献体现在:(1)提出了一种角度修正预测种群中心点的方法,并利用UKF模型对差分向量进行角度修正,减小预测模型的误差;(2)采用分级多种群协同进化,子种群监督主种群的进化方向,以避免种群陷入偏好区域,增强了种群的多样性;(3)检测到环境变化之后,采用UKF模型对种群中心点进行优化.

2　相关工作

2.1　问题描述

DMOP中一个含有n维决策变量、m个目标函数的最小化问题可定义为[4]:

$$\begin{cases}\min\limits_{k\in\Omega} \boldsymbol{F}(\boldsymbol{x},t)=\left[f_1(\boldsymbol{x},t),f_2(\boldsymbol{x},t),\cdots,f_m(\boldsymbol{x},t)\right]^{\mathrm{T}}\\ g_i(\boldsymbol{x},t)\leqslant 0,i=1,2,\cdots,p;\ h_j(\boldsymbol{x},t)=0,j=1,2,\cdots,q\end{cases}\tag{1}$$

其中,t表示时间变量,Ω表示决策空间,$\boldsymbol{x}$表示决策变量,f表示目标函数,$g_i(\boldsymbol{x},t)\leqslant 0$和$h_j(\boldsymbol{x},t)=0$分别表示不等式约束条件和等式约束条件,$p$和$q$则是对应约束条件的数量.

假设$\boldsymbol{F}$为目标函数,$\boldsymbol{x}$为决策变量,对一个给定的多目标优化问题,其定义如下:

定义1　Pareto支配:当且仅当对$\forall i=\{1,2,\cdots,m\}$有$f_i(\boldsymbol{x}_1)\leqslant f_i(\boldsymbol{x}_2)$且$\exists j\in\{1,2,\cdots,m\}$满足$f_j(\boldsymbol{x}_1)<f_j(\boldsymbol{x}_2)$时,就可定义$\boldsymbol{x}_1$支配$\boldsymbol{x}_2$,记为$f(\boldsymbol{x}_1)\prec f(\boldsymbol{x}_2)$.

定义2　Pareto最优解集(Pareto optimal Set, PS):决策空间中非支配解的集合,即:

$$\mathrm{PS}=\left\{\boldsymbol{x}\in\Omega\mid\neg\exists\boldsymbol{x}^*\in\Omega,\boldsymbol{F}(\boldsymbol{x}^*)\prec\boldsymbol{F}(\boldsymbol{x})\right\}\tag{2}$$

定义3　Pareto最优前沿(Pareto optimal Front, PF):目标空间中非支配解的集合,即:

$$\mathrm{PF}=\left\{\boldsymbol{y}=\boldsymbol{F}(\boldsymbol{x})\mid\boldsymbol{x}\in\mathrm{PS}\right\}\tag{3}$$

2.2　无迹卡尔曼滤波模型

无迹变换(Unscented Transform,UT)不依赖非线性的具体形式,通过在预测点的邻域搜索采样点,然后使

用这些搜索的采样点(sigma points)来替代高斯密度近似状态的概率密度函数. UKF 在处理非线性滤波时是通过 UT 变换的方式,令获取的 sigma points 均值及协方差等于原状态分布,再通过非线性映射便可以求出近似的状态概率密度函数. 该模型与扩展的 Kalman 滤波最大的不同之处在于,它是对非线性问题的一种近似概率密度分布,这种方法计算精度高、稳定性好.

3 角度修正和分级多种群的 DMOEA

3.1 差分向量角度修正

本文提出了一种减小预测误差的方法,使用历史信息种群中心点构建的差分向量进行角度修正,其基本思想如图 1 所示. 角度修正是对差分向量进行修正. 其原理是,利用历史角度信息及其权重,推测出下一时刻差分向量的角度变化范围,并对下一时刻的差分向量进行修正. 其具体实现过程是,使用这一时刻的差分向量减去前一时刻的差分向量,形成角度差值,将形成的所有角度差值一一记录,并赋予不同的权重,以此预测出下一时刻应发生的角度变化,并使用预测的角度对已产生的下一时刻的差分向量进行修正,使其所指的种群迁移方向更接近理想状态.

图 1 展示了预测以及修正 $t+1$ 时刻的种群中心点的过程. 已知 t 时刻种群中心点预测 $t+1$ 时刻的中心点,若仅以 t 时刻种群中心点减去 $t-1$ 时刻种群中心点得到的差分向量 $\boldsymbol{d}_t$ 对下一时刻的种群中心点进行预测,得到的预测中心点可能存在很大的误差. 为了避免这种情况,采用以往历史信息的所有种群中心点构建的差分向量修正预测的种群中心点. 即在 t 时刻预测 $t+1$ 时刻的种群中心点时,使用 t 时刻种群中心点减去 $t-1$ 时刻种群中心点得到的差分向量 $\boldsymbol{d}_t$,结合 $t-1$ 时刻种群中心点减去 $t-2$ 时刻种群中心点得到的差分向量 $\boldsymbol{d}_{t-1}$,以此类推直至得到所有相邻时刻的种群中心点的差分向量,共同预测 $t+1$ 时刻的种群中心点.

3.2 分级多种群

本文将原始种群按照不同比例分为三个子种群,其中,一级主源种群由未经 UKF 修正的差分向量方向产生,对主种群的进化方向起决定性的作用;二级原始种群由中心点预测生成的差分向量方向产生,以保持种群多样性;二级修正种群由 UKF 修正后的差分向量方向产生,防止种群进化陷入偏好区域.

3.2.1 中心点的计算

首先建立基础中心点预测模型,为了产生不同的差分向量,本算法采用了两种不同的中心点计算方式,其一为大多数算法如定向搜索策略(Directed Search Strategy,DSS)常用的中心点计算方式,即:

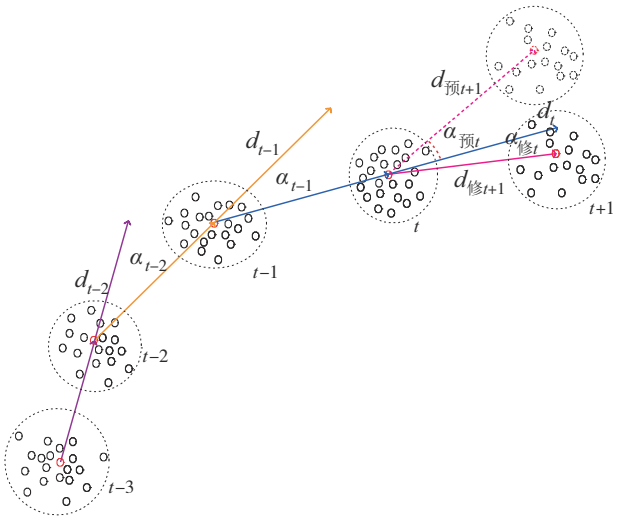


图 1 差分向量角度修正原理图

$$\boldsymbol{C}_1^t=\frac{1}{\left|P^t\right|}\sum_{\boldsymbol{x}\in P^t}\boldsymbol{x} \tag{4}$$

其中,P^t 表示的是当前 t 时刻所得到的非支配解集,$|P^t|$ 表示的是这个非支配解集的大小. 这种方式的本质是通过从决策空间中挑出 Pareto 最优解集,然后对该最优解集中的个体进行运算,求出所有个体的平均值,然后使用这个平均值当作中心点. 这种中心点计算方式的优点在于计算方式简单,在算法进化后期,所求 Pareto 解集与理想最优解距离较近的情况下,中心点计算较为准确.

但是这种求中心点的计算方式在使用过程中会带来以下几个问题:(1)在种群进化前期,算法所求出的 Pareto 最优解集与实际的 Pareto 的最优解集相差甚远. (2)使用这种方式求出的中心点存较大的误差,当遇到复杂场景时,极有可能陷入持续偏向动态偏好区域的问题,造成算法性能的严重下降.

上述问题会严重影响无迹卡尔曼滤波的预测准确度,为减小这种影响,本文还采用了另外一种中心点计算方案,即找出当前时刻的 PS,通过计算求解出其对应目标空间的目标函数值,然后求得其至原点的欧氏距离,找出距离原点最近的一个点,把其当作计算的中心点,也是卡尔曼滤波的观测值和真实值,记为 $\boldsymbol{C}_2^t$.

3.2.2 中心点的预测

使用 UKF 模型对 3.2.1 节中计算所得的中心点 $\boldsymbol{C}_2^t$ 进行跟踪,将其作为无迹卡尔曼滤波的观测值代入模型. 具体步骤如算法 1 所示.

在算法 1 中,首先通过 UT 变换生成一组采样点及相应权值,从而求出采样点的一步预测值,获得系统的一步预测值和协方差. 然后将系统的一步预测值,再通过 UT 变换来生成新的采样点,由观测方程计算出新点集的预测观测值,使用加权求和的方式计算预测的均

算法1　预测中心点

输入:观测值$\boldsymbol{C}_2^t$,权值ω,协方差矩阵$\boldsymbol{Q}$和$\boldsymbol{R}$,理想中心点$\boldsymbol{c}_t$

输出:$\hat{\boldsymbol{C}}_3^t$

1. /*采样点及相应权值*/

2. $\boldsymbol{C}_2^{(i)}(t|t)=\left[\hat{\boldsymbol{C}}_2(t|t)\quad \hat{\boldsymbol{C}}_2(t|t)+\sqrt{(n+\lambda)\boldsymbol{P}(t|t)}\quad \hat{\boldsymbol{C}}_2(t|t)-\sqrt{(n+\lambda)\boldsymbol{P}(t|t)}\right]$;

3. /*其中,$\boldsymbol{P}$为方差,λ是放缩比例*/

4. /*采样点的一步预测*/

5. $\boldsymbol{C}_2^{(i)}(t+1|t)=f[t,\boldsymbol{C}_2^{(i)}(t|t)],i=1,2,\cdots,2n+1$;

6. /*系统的一步预测以及协方差*/

7. $\hat{\boldsymbol{C}}_2(t+1|t)=\sum_{i=0}^{2n}\omega^{(i)}\boldsymbol{C}_2^{(i)}(t+1|t)$;

8. $\boldsymbol{P}(t+1|t)=\sum_{i=0}^{2n}\omega^{(i)}[\hat{\boldsymbol{C}}_2(t+1|t)-\boldsymbol{C}_2^{(i)}(t+1|t)][\hat{\boldsymbol{C}}_2(t+1|t)-\boldsymbol{C}_2^{(i)}(t+1|t)]^{\mathrm{T}}+\boldsymbol{Q}$;

9. /*UT变换*/

10. $\boldsymbol{C}_2^{(i)}(t+1|t)=\left[\hat{\boldsymbol{C}}_2(t+1|t)\quad \hat{\boldsymbol{C}}_2(t+1|t)+\sqrt{(n+\lambda)\boldsymbol{P}(t+1|t)}\,\hat{\boldsymbol{C}}_2(t+1|t)-\sqrt{(n+\lambda)\boldsymbol{P}(t+1|t)}\right]$;

11. /*预测观测值*/

12. $\boldsymbol{Z}_{\boldsymbol{C}}^{(i)}(t+1|t)=h[\boldsymbol{C}_2^{(i)}(t+1|t)],i=1,2,\cdots,2n+1$;

13. /*预测的均值和协方差*/

14. $\bar{\boldsymbol{Z}}_{\boldsymbol{C}}(t+1|t)=\sum_{i=0}^{2n}\omega^{(i)}\boldsymbol{Z}_{\boldsymbol{C}}^{(i)}(t+1|t)$;

15. $\boldsymbol{P}_{z_t z_t}=\sum_{i=0}^{2n}\omega^{(i)}[\boldsymbol{Z}_c^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{C}}(t+1|t)][\boldsymbol{Z}_c^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{C}}(t+1|t)]^{\mathrm{T}}+\boldsymbol{R}$;

16. $\boldsymbol{P}_{c_t z_t}=\sum_{i=0}^{2n}\omega^{(i)}[\boldsymbol{C}_2^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{C}}(t+1|t)][\boldsymbol{Z}_{\boldsymbol{C}}^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{c}(t+1|t)]^{\mathrm{T}}$;

17. /*卡尔曼增益*/

18. $\boldsymbol{K}(t+1)=\boldsymbol{P}_{c_t z_t}\boldsymbol{P}_{z_t z_t}^{-1}$;

19. /*更新系统*/

20. $\hat{\boldsymbol{C}}_2(t+1|t+1)=\hat{\boldsymbol{C}}_2(t+1|t)+\boldsymbol{K}(t+1)[\boldsymbol{Z}_{\boldsymbol{C}}(t+1)-\hat{\boldsymbol{Z}}(t+1|t)]$;

21. $\boldsymbol{P}(t+1|t+1)=\boldsymbol{P}(t+1|t)+\boldsymbol{K}(t+1)\boldsymbol{P}_{z_t z_t}\boldsymbol{K}^{\mathrm{T}}(t+1)$;

22. /*修正预测值*/

23. $d_{\mathrm{oc}}=d(\hat{\boldsymbol{C}}_2(t+1|t+1),\bar{\boldsymbol{c}}_t)$;

24. $\hat{\boldsymbol{C}}_3^t=\hat{\boldsymbol{C}}_2(t+1|t+1)+\boldsymbol{N}(0,1)*d_{\mathrm{oc}}$;

25. /*其中d_{oc}是欧氏距离,$\boldsymbol{N}(0,1)$是一维随机正态分布函数*/

值和协方差. 最后计算卡尔曼增益,更新系统的状态及协方差,就得到无迹卡尔曼滤波优化过后的中心点$\hat{\boldsymbol{C}}_2(t+1|t+1)$. 考虑到中心点的修正可能沿着卡尔曼滤波的偏好区域进行而导致陷入局部最优,所以需要对计算出的预测值进行修正. 本算法选取已经求得的近似理想的Pareto前沿面上到原点最短的欧氏距离的一个点$\boldsymbol{s}_{\min}$,把$\boldsymbol{s}_{\min}$对应的解作为理想中心点$\boldsymbol{c}_t$对求得的预测值进行修正.

3.2.3　分级多种群的生成

在3.2.1和3.2.2中获得了两个中心点$\boldsymbol{C}_1^t$和$\boldsymbol{C}_3^t$,使用当前时刻预测的中心点减去前一时刻的中心点,得出预测差分向量,将使用卡尔曼滤波预测得到的中心点计算的差分向量记为$\boldsymbol{D}_1^t$,使用传统方式预测得到的中心点计算的差分向量记为$\boldsymbol{D}_2^t$,沿着差分向量的方向产生新的个体,以此达到种群初始化的目的. 其计算公式如下所示:

$$\boldsymbol{D}_1^t=\boldsymbol{C}_3^t-\boldsymbol{C}^{t-1} \tag{5}$$

$$\boldsymbol{D}_2^t=\boldsymbol{C}_1^t-\boldsymbol{C}^{t-1} \tag{6}$$

上式是一种简单的线性计算过程,通过这种方式所生成的差分向量在面对复杂多变的环境时,会出现预测偏离理想轨迹、无法及时适应环境变化等问题,为了解决这种问题,也为了增加种群多样性,使用无迹卡尔曼滤波对差分向量$\boldsymbol{D}_1^t$进行角度修正.

算法2为UKF优化过程,将卡尔曼滤波预测获得的中心点计算的差分向量作为预测的观测值,使用加权求和的方式计算预测的均值和协方差,然后求出卡尔曼增益,更新系统的状态和协方差.

算法2　UKF优化

输入:观测值$\boldsymbol{D}_1^t$,权值ω,协方差矩阵$\boldsymbol{R}$

输出:$\hat{\boldsymbol{D}}_1(t+1|t+1)$

1. /*采样点的一步预测*/

2. $\boldsymbol{Z}_{\boldsymbol{D}}^{(i)}(t+1|t)=h[\boldsymbol{D}_1^{(i)}(t+1|t)],i=1,2,\cdots,2n+1$;

3. /*加权求和*/

4. $\bar{\boldsymbol{Z}}_{\boldsymbol{D}}(t+1|t)=\sum_{i=0}^{2n}\omega^{(i)}\boldsymbol{Z}_{\boldsymbol{D}}^{(i)}(t+1|t)$;

5. $\boldsymbol{P}_{z_t z_t}=\sum_{i=0}^{2n}\omega^{(i)}[\boldsymbol{Z}_{\boldsymbol{D}}^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{D}}(t+1|t)][\boldsymbol{Z}_{\boldsymbol{D}}^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{D}}(t+1|t)]^{\mathrm{T}}+\boldsymbol{R}$;

6. $\boldsymbol{P}_{d_t z_t}=\sum_{i=0}^{2n}\omega^{(i)}[\boldsymbol{D}_1^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{D}}(t+1|t)][\boldsymbol{Z}_{\boldsymbol{D}}^{(i)}(t+1|t)-\bar{\boldsymbol{Z}}_{\boldsymbol{D}}(t+1|t)]^{\mathrm{T}}$;

7. /*卡尔曼增益*/

8. $\boldsymbol{K}(t+1)=\boldsymbol{P}_{d_t z_t}\boldsymbol{P}_{z_t z_t}^{-1}$;

9. /*更新系统*/

10. $\hat{\boldsymbol{D}}_1(t+1|t+1)=\hat{\boldsymbol{D}}_1(t+1|t)+\boldsymbol{K}(t+1)[\boldsymbol{Z}_{\boldsymbol{D}}(t+1)-\hat{\boldsymbol{Z}}_{\boldsymbol{D}}(t+1|t)]$;

11. $\boldsymbol{P}(t+1|t+1)=\boldsymbol{P}(t+1|t)+\boldsymbol{K}(t+1)\boldsymbol{P}_{z_t z_t}\boldsymbol{K}^{\mathrm{T}}(t+1)$;

3.2.4　多种群初始化

通过上述计算,已获得差分向量$\boldsymbol{D}_1^t$、$\boldsymbol{D}_2^t$、$\boldsymbol{D}_3^t$三个不同的进化方向,将原始种群分为三个子种群,分别对应上述三种进化方向,其中一级主源种群以差分向量$\boldsymbol{D}_1^t$为进化方向;二级原始种群和二级修正种群则分别使用差分向量$\boldsymbol{D}_2^t$和$\boldsymbol{D}_3^t$为进化方向. 三个种群都通过以下方式进行初始化:

$$S_i^t=\mathrm{sign}(\boldsymbol{D}_i^t),i=1,2,3 \tag{7}$$

$$\boldsymbol{y}=\boldsymbol{x}+\boldsymbol{D}_i^t+\boldsymbol{N}(0,d)\times S_i^t,i=1,2,3 \tag{8}$$

其中,$\boldsymbol{x}$是种群中的任意一个个体,$\mathrm{sign}(\boldsymbol{D}_i^t)$表示方向函数,$\boldsymbol{D}_i^t$是各个种群进化所对应的差分向量,$d$是$t$时刻与$t-1$时刻中心点之间的欧氏距离,$\boldsymbol{N}(0,d)$则表示的是一个一维函数,其符合正态分布,且满足均值为0,标准

差为 d. 添加 S 的目的，就是在预测过程中添加噪声，通过这种方式来补偿预测中可能出现的偏差.

3.3 ACHMP算法描述

ACHMP算法的基本思想是在环境未改变时，对种群的历史信息进行采集，当发生变化时，根据历史信息，使用无迹卡尔曼滤波模型预测种群的中心点，使用预测的中心点指导种群的进化方向；其次使用预测后中心点减去前一时刻中心点的方式产生差分向量，然后再使用无迹卡尔曼滤波对此差分向量进行角度修正，得到修正后的差分向量；将种群一分为三，分别沿着未经卡尔曼滤波的修正的差分向量、经过卡尔曼滤波修正后的差分向量以及使用传统方式中心点预测方式生成的差分向量的方向产生新的种群个体. 这种方式使得种群收敛性较高，且具有较好的种群多样性. ACHMP算法的伪代码如算法3所示.

算法3 ACHMP算法

输入:种群规模 N,当前种群 P^t

输出:新种群 P^t

1. 对种群进行初始化,形成大小为 N 的种群(P^t 表示 t 时刻的种群);
2. 对当前环境进行检测,当环境发生变化时,进行后续操作,否则直接跳至步骤5;
3. 使用两种方式分别计算 t 时刻种群中心点,记为 $\boldsymbol{C}_1^t$、$\boldsymbol{C}_2^t$;
4. 使用UKF预测模型预测 $t+1$ 时刻的中心点,并对结果进行修正,得到修正后的中心点 $\boldsymbol{C}_3^t$;
5. 分别使用 $\boldsymbol{C}_3^t$、$\boldsymbol{C}_1^t$ 通过式(5)和式(6)计算得到差分向量 $\boldsymbol{D}_2^t$、$\boldsymbol{D}_1^t$;
6. 使用UKF对 $\boldsymbol{D}_2^t$ 进行修正,得到 $\boldsymbol{D}_3^t$;
7. FOR i=1,$\cdots$,r_1*N do (r_1=0.5)
8. 从 P^t 中随机选取一个个体 $\boldsymbol{x}$;
9. 使用式(8)产生新的个体 $\boldsymbol{y}$;
10. 边界检查;
11. END

ACHMP是以NSGA-Ⅱ算法为框架，使用DSS2(Directed Search Strategy，DSS)搜索算法[18]加速种群收敛，该算法的伪代码如算法4所示.

算法4 DSS2算法

输入:种群规模 N,当前种群 P^t,前一时刻种群中心点 $\boldsymbol{C}^{t-1}$

输出:新的点

1. 计算 P^t 的中心点 $\boldsymbol{C}^t$ 及其移动方向;
2. FOR i=1:r_2*N (r_2=0.05)
3. 随机从当前种群中挑出一个个体,并使用式(8)生成一个新点;
4. 进行边界检查和修复;
5. END
6. 使用新点替换原来相对应的点

4 实验

使用了DMOEA中的25个测试函数来进行测试，分别为FDA系列[4]、DMOP系列[19]、DF系列[20]以及 F 系列[5]中的 $F5$~$F7$ 测试函数. 前三组测试函数属于线性系统，他们的决策变量呈线性相关状态，最后一组测试函数属于复杂的非线性系统. 其中三目标测试函数有DF10、DF11、DF12、DF13、DF14以及FDA4和FDA5共7个测试函数，剩余的18个测试函数是双目标测试函数.

4.1 评价指标

为了评估ACHMP算法的多样性和收敛性，本文选择以下指标从不同角度评价算法性能.

(1)逆世代距离(Inverted Generational Distance，IGD)[21]和改进的逆世代距离(Modified Inverted Generational Distance，MIGD)，计算公式如下所示：

$$\mathrm{IGD}(P^{t*},P^t)=\frac{\sum_{\boldsymbol{x}\in P^{t*}} d(\boldsymbol{x},P^t)}{|P^{t*}|} \tag{9}$$

$$\mathrm{MIGD}=\frac{1}{|T|}\sum_{t\in T}\mathrm{IGD}(P^{t*},P^t) \tag{10}$$

其中，P^{t*} 为真实PF，P^t 为近似PF，$d(\boldsymbol{x},P^t)$ 为 P^{t*} 上的个体 $\boldsymbol{x}$ 与 P^t 中的个体之间的最小欧式距离，$|P^{t*}|$ 表示 P^{t*} 的大小. T 表示一组离散的时间点. IGD利用真实PS中每个解的目标向量与Pareto非支配解集的前沿之间的距离，对算法的综合性能进行评估. IGD在大多数情况下是对静态MOEA的算法性能进行评估，所以采用改进的IGD(MIGD)进一步评估DMOEA. IGD和MIGD的值越小，说明测试算法的性能越好.

(2)超体积(Hyper-volume，HV)[22]：HV用于评估近似Pareto前沿的多样性和分布. HV表示非支配解集覆盖的目标空间区域的大小，HV值越大，算法性能越好. 其计算方法如下：

$$\mathrm{HV}_t=\mathrm{HV}(P^t) \tag{11}$$

其中，$\mathrm{HV}(P^t)$ 表示 P^t 的超体积. MHV是在给定运行中在某些时间步长上的HV值的平均值，通常用于衡量算法在动态优化问题上的表现. MHV值越大，算法性能越好. 其计算方法如下：

$$\mathrm{MHV}=\frac{\sum_{t\in T}\mathrm{HV}_t}{|T|} \tag{12}$$

4.2 参数设置与参数分析

4.2.1 实验参数设置

公共参数设置：决策变量维数为10，种群大小为100，环境变化强度 n_τ=10，最大迭代次数 τ_T=500，变化频率 τ_t=10，环境变化50次. 通过Wilcoxon秩和检验方法比较ACHMP算法和其他对比算法的实验结果，显著性水平取5%，“+”、“−”、“=”分别表示ACHMP算法所得结果优于、劣于以及相似于对比算法.

4.2.2 参数分析

为研究各种群比例对算法性能的影响,找出合适的种群分配比例,现将一级主源种群、二级原始种群和二级修正种群依次设置不同比例,进行多次实验,现将一些典型比例分配得到的MIGD值汇总在表1中,其中加粗的数据为表现最优异的数据.

表1　不同比例多种群性能对比

测试函数	8:1:1	7:2:1	7:1:2	6:1:3	6:3:1	6:2:2	5:3:2	4:3:3	3:4:3	2:3:5	1:5:4
*F*5	**0.164 2**	0.177 9	0.171 4	0.167 2	0.172 8	0.178 4	0.177 1	0.179 2	0.755 3	0.759 4	0.756 4
*F*6	**0.077 5**	0.096 5	0.081 2	0.087 4	0.095 0	0.092 2	0.099 3	0.102 5	0.759 0	0.755 3	0.754 1
*F*7	**0.067 2**	0.071 5	0.072 2	0.072 7	0.079 8	0.075 7	0.078 4	0.084 8	0.750 8	0.752 7	0.745 2
DF1	**0.013 4**	0.015 8	0.015 5	0.016 5	0.014 6	0.014 2	0.015 6	0.015 9	0.831 0	0.831 7	0.831 9
DF2	0.034 1	0.033 9	0.032 1	**0.030 4**	0.036 2	0.033 3	0.032 8	0.035 6	0.681 6	0.681 8	0.681 8
DF3	**0.026 6**	0.028 9	0.027 0	0.027 6	0.028 2	0.029 1	0.027 9	0.030 4	0.864 8	0.865 8	0.864 8
DF4	0.144 9	0.143 1	**0.142 4**	0.146 7	0.147 5	0.145 3	0.143 6	0.146 6	1.506 7	1.506 2	1.506 0
DF5	**0.163 8**	0.184 5	0.183 5	0.191 7	0.188 0	0.184 5	0.175 9	0.202 8	0.779 4	0.775 7	0.756 5
DF6	**0.272 8**	0.441 6	0.386 6	0.443 7	0.395 1	0.370 1	0.546 7	0.552 0	0.682 2	0.683 1	0.680 3
DF7	**0.463 7**	0.468 7	0.466 6	0.481 3	0.468 2	0.479 3	0.480 2	0.482 7	—	—	—
DF8	**0.111 4**	0.114 0	0.113 4	0.114 4	0.116 2	0.115 7	0.118 2	0.120 3	0.660 8	0.606 3	0.624 2
DF9	**0.070 0**	0.073 2	0.072 2	0.070 4	0.071 3	0.073 7	0.072 9	0.075 2	0.801 6	0.802 0	0.801 3
DMOP1	**0.022 3**	0.031 2	0.030 5	0.025 3	0.035 3	0.030 3	0.028 6	0.025 9	0.845 1	0.843 3	0.844 4
DMOP2	**0.013 8**	0.014 1	0.014 5	0.014 3	0.014 8	0.014 9	0.015 9	0.016 2	0.831 9	0.832 0	0.833 2
DMOP3	**0.007 4**	0.008 7	0.008 9	0.009 7	0.008 3	0.008 0	0.010 0	0.009 0	0.683 6	0.681 7	0.682 0
FDA1	**0.010 4**	0.011 1	0.012 1	0.011 5	0.011 8	0.011 6	0.011 9	0.011 6	0.682 6	0.682 4	0.682 7
FDA2	**0.113 5**	0.114 4	0.115 5	0.115 7	0.114 5	0.114 4	0.114 9	0.114 9	0.611 4	0.601 0	0.640 4
FDA3	**0.047 3**	0.051 6	0.050 2	0.053 5	0.052 2	0.051 2	0.052 4	0.054 1	1.301 4	1.301 4	1.301 9

在表1中,当一级主源种群的占比逐步下降时,算法的总体性能也随之下降. 当一级主源种群的占比小于任意的二级种群时,由于一级主源种群占比过小,种群分布主要由二级种群决定,算法性能出现明显的断崖式下降. 其中二级原始种群占大比例时,其指引种群生成方向的差分向量误差较大,导致最终结果较差;当二级修正种群占大比例时,其依据角度修正思想产生的新差分向量,在面对环境的突然改变时,因为修正算法的偏好性,会在拐点处产生较大的误差偏移,对最终的算法性能也会产生一定影响.

对表1中4~6列的数据进行横向对比,在一级主源种群占比相同的情况下,当二级原始种群占比高于二级修正种群时,算法在DF1、DF5、DF6、DF7、DMOP3、FDA2、FDA3上的性能更优异,这些测试函数的PS和PF大多都是随时间变化的,且PS变化简单、PF多为凸到非凸形的,而因为二级原始种群的生成方向是由传统方式生成的差分向量所决定的,受影响更小,所以表现更佳;当二级修正种群占比高于二级原始种群时,算法在则在其他测试函数上的性能更优异,这是因为二级修正种群的生成方向是依据角度修正原理得来,在环境变化复杂的情况下,二级修正种群可以持续发挥其对进化的修正作用,且由于占比小,所带来的修正偏好性弊端不明显,因此在这些测试函数上有更好的表现;当二级修正种群占比与二级原始种群相平衡时,由于两个二级种群对种群进化方向影响的权重相同,可以最大化的发挥两个二级种群自身应有的作用,算法整体性能要优于其他任何一种情况. 进一步研究表1的2、3列结果,在一级主源种群占比提高后,二级修正种群所带来的不利影响有所减小,在绝大多数测试函数中的效果都要好于同占比下的二级原始种群,充分发挥出了角度修正的优势,而又极大的避免了其带来的弊端.

综上所述,ACHMP算法的一级主源种群占比要足够大,二级原始种群和二级修正种群也要有相平衡的占比,所以ACHMP算法选择的一级主源种群、二级原始种群和二级修正种群的比例为8:1:1.

4.3 实验结果与分析

4.3.1 多算法的结果对比分析

本文所提出的算法(ACHMP)与几种流行的算法进行了比较,其中包括基于支持向量回归(Support Vector Regression, SVR)的算法SVR-MOEA/D(Multi-Objective Evolutionary Algorithm based on Decomposition, MOEA/D)[23]、基于迁移学习(Transfer learning, Tr)的算法Tr-DMOEA/D[24]、基于卡尔曼滤波器的算法KF-MOEA/D[15]、基于种群(Population Prediction Strategy, PPS)的算法PPS-MOEA/D[5]、基于膝关节点的迁移学习(Knee point-

based Transfer learning,KT)的算法KT-MOEA/D[12]和RI-MOEA/D[12,25]. 另外两种经典算法DSS[18]和DNSGA-Ⅱ-A[26]也参与了比较,测试函数为DF1~DF14、*F*5~*F*7、DMOP1~DMOP3以及FDA1~FDA5,各个算法的特有参数设置基于文献[12]与原始文献设置,其中双目标测试函数的种群大小为100,三目标测试函数的种群大小为150. 表2和表3为MIGD的统计数据,并在表4中进行了非参数检验,表5和表6为MHV的统计数据,表中KT-MOEA/D、PPS-MOEA/D、SVR-MOEA/D、Tr-MOEA/D、KF-MOEA/D和RI-MOEA/D的实验数据均来自文献[12],表中粗体表示同一测试函数中表现最优异的数据.

从表2和表3中可以看到,除了DF2、DF10、DF11和FDA3之外,ACHMP在剩下21个测试函数的表现上均优于其他几种对比算法,表明了ACHMP算法的有效性,在解决大部分问题时具有良好的普适性,反映了ACHMP算法具有良好的收敛性以及鲁棒性. 在DF10测试函数中,ACHMP算法的效果不如DSS算法,主要因为DF10测试函数的中心点基本没有移动,ACHMP算法采用基于中心点的预测策略,导致效果不好. 综合对DF2、DF10、DF11和FDA3这四个测试函数的结果进行分析,可以发现,当测试函数对于多样性的要求较高或中心点基本未发生移动时,ACHMP算法的结果略逊于最好结果. 这是因为ACHMP算法主要基于中心点的移动进行预测,当中心点移动不明显时,其优势不再明显.

9种算法在所有测试问题上的MIGD均值的非参数检验结果如表4所示. 其中,R^+和R^-分别为相应算法的正秩和负秩,P value值为显著性水平,Rank是每个算法的平均秩. 通过Wilcoxon's检验可以发现ACHMP以$\alpha=0.05$的显著性水平明显优于其他几种算法,在Friedman检验中,ACHMP在多个测试函数中排名第一. 在非参数检验结果的总排名中,ACHMP也是最优异的.

表2 DNSGA-Ⅱ-A、DSS算法的MIGD值

测试函数	ACHMP	DNSGA-II-A	DSS	测试函数	ACHMP	DNSGA-II-A	DSS
DF1	**0.013 4**	0.045 9+	0.016 7+	DF14	**0.381 1**	0.634 9+	0.457 7+
DF2	0.034 1	0.039 6+	**0.020 7-**	*F*5	**0.164 2**	1.069 7+	0.183 1+
DF3	**0.026 6**	0.046 7+	0.037 1+	*F*6	**0.077 5**	1.028 6+	0.104 2+
DF4	**0.144 9**	0.147 3+	0.145 4+	*F*7	**0.067 2**	0.886 6+	0.087 9+
DF5	**0.163 8**	0.241 6+	0.200 5+	DMOP1	**0.022 3**	0.086 7+	0.034 3+
DF6	**0.272 8**	5.594 8+	1.550 0+	DMOP2	**0.013 8**	0.050 0+	0.016 7+
DF7	**0.463 7**	0.479 1+	0.515 7+	DMOP3	**0.007 4**	0.032 7+	0.010 5+
DF8	**0.111 4**	0.112 0+	0.117 8+	FDA1	**0.010 4**	0.050 6+	0.013 1+
DF9	**0.070 0**	0.138 1+	0.074 5+	FDA2	**0.113 5**	0.114 3+	0.115 5+
DF10	0.127 5	0.810 8+	**0.107 1−**	FDA3	0.047 3	**0.040 4−**	0.043 0−
DF11	0.653 3	0.986 1+	0.670 1+	FDA4	**0.064 4**	0.986 8+	0.068 2+
DF12	**0.313 2**	0.990 9+	0.357 0+	FDA5	**0.050 1**	0.986 5+	0.051 1+
DF13	**0.440 1**	1.847 1+	0.454 6+	+/−/=	-	24/1/0	22/3/0

注:+/−/=分别表示ACHMP算法与其他算法相比更好、更差和相似.

由表5和表6中ACHMP、DNSGA-II-A和DSS的实验数据可以得出,ACHMP在非线性测试问题(*F*5、*F*6、*F*7)上效果良好,在解决PS变化的测试问题方面也有很好的表现,如FDA1、FDA3、DMOP3、DF1、DF3、DF5和DF6. ACHMP在PS不变或中心点不变时表现不是很理想(如DF4). 虽然DNSGA-II-A在所有三目标函数下都得到了最佳的MHV值,但其结果在任何两个目标测试函数下都没有竞争力. 相比之下,DSS和ACHMP在三目标测试函数上的性能没有显著差异,并且ACHMP算法整体上略好于DSS算法. ACHMP的数据与其余6组对比算法相比,所有算法的MHV值都没有竞争性,但ACHMP、RI-MOEA/D和KT-MOEA/D的最优值数最高. 说明ACHMP算法能够快速响应环境变化,该算法具有良好的多样性和收敛性.

4.3.2 与其他先进算法的比较

为了测试在不同环境设置下算法的性能,与KF[15]、PPS[5]、SGEA(Steady-state and Generational Evolutionary Algorithm)[27]、MCPDMO[11]四种算法又进行了对比. 将公共参数设置为:种群大小为100,决策变量维数为10,环境变化强度$n_\tau=10$,变化频率为$\tau_t=5,20$,环境变化100次,各算法独立运行20次. 其他算法设置均按照文献[11]设定,结果如表7所示,表中加粗的值是同一测试函数中五种算法MIGD值中最优的值. KF、PPS、SGEA、MCPDMO四种算法测试数据均引用自文献[11].

由表7可以看出,ACHMP算法在DF2、FDA1和FDA3三个测试函数上,不论是环境变化频率为5还是20时,均显著优于其他算法. 在DMOP1和FDA4上SGEA表现较好,SGEA在面对PF不变类型的函数时,

表3　KT-MOEA/D、PPS-MOEA/D、SVR-MOEA/D、Tr-MOEA/D、KF-MOEA/D、RI-MOEA/D和ACHMP算法的MIGD值

测试函数	ACHMP	KT-MOEA/D	PPS-MOEA/D	SVR-MOEA/D	Tr-MOEA/D	KF-MOEA/D	RI-MOEA/D
DF1	**0.013 4**	0.085 5+	0.100 2+	0.092 0+	0.115 2+	0.159 4+	0.122 2+
DF2	**0.034 1**	0.073 3+	0.075 8+	0.084 6+	0.079 6+	0.105 2+	0.089 8+
DF3	**0.026 6**	0.351 0+	0.423 3+	0.393 4+	0.323 9+	0.366 3+	0.420 7+
DF4	**0.144 9**	0.964 3+	0.956 7+	1.087 1+	1.167 8+	1.299 5+	1.111 2+
DF5	**0.163 8**	1.274 4+	1.305 9+	1615.366 0+	1.588 6+	1.303 2+	1.544 0+
DF6	**0.272 8**	2.348 0+	4.057 9+	2.938 5+	1.815 7+	3.148 8+	2.737 8+
DF7	**0.463 7**	2.234 5+	4.174 4+	2.752 9+	2.029 1+	4.005 4+	2.984 9+
DF8	**0.111 4**	0.827 4+	1.094 3+	0.977 4+	0.979 2+	1.148 6+	0.951 7+
DF9	**0.070 0**	1.657 9+	1.754 6+	551.812 2+	1.630 6+	1.684 6+	1.658 7+
DF10	**0.127 5**	0.188 0+	0.189 1+	64.903 7+	0.188 3+	0.214 4+	0.203 6+
DF11	0.653 3	**0.138 7**–	0.194 8–	237.626 9+	0.195 3–	0.185 1–	0.191 6–
DF12	**0.313 2**	1.008 2+	1.177 1+	252.611 5+	1.008 4+	0.989 0+	0.939 0+
DF13	**0.440 1**	1.347 4+	1.395 8+	1.378 5+	1.506 8+	1.441 3+	1.475 1+
DF14	**0.381 1**	0.838 1+	0.865 7+	4.188 3+	0.931 6+	0.906 5+	0.923 9+
+/−/=	—	13/1/0	13/1/0	14/0/0	13/1/0	13/1/0	13/1/0

注:+/−/=分别表示ACHMP算法与其他算法相比更好、更差和相似

表4　非参数检验的结果

算法	Wilcoxon's检验						Friedman检验	
	+/−/=	R^+	R^-	P value	$\alpha = 0.05$	$\alpha = 0.1$	Rank	P value
DNSGA-Ⅱ-A	24/1/0	320	5	0.000 023	Yes	Yes	4.57	—
DSS	22/3/0	284	41	0.001 079	Yes	Yes	2.36	—
KT-MOEA/D	13/1/0	99	6	0.003 510	Yes	Yes	4.07	—
PPS-MOEA/D	13/1/0	100	5	0.002 865	Yes	Yes	6.43	—
SVR-MOEA/D	14/0/0	105	0	0.000 982	Yes	Yes	7.21	—
Tr-MOEA/D	13/1/0	100	5	0.002 865	Yes	Yes	5.93	—
KF-MOEA/D	13/1/0	100	5	0.002 865	Yes	Yes	6.71	—
RI-MOEA/D	13/1/0	100	5	0.002 865	Yes	Yes	6.21	—
ACHMP	—	—	—	—	—	—	1.50	3.448×10^{-10}

注:+/−/=分别表示ACHMP算法与其他算法相比更好、更差和相似的个数.

表5　DNSGA-Ⅱ-A、DSS算法的MHV值

测试函数	ACHMP	DNSGA-II-A	DSS	测试函数	ACHMP	DNSGA-II-A	DSS
DF1	**0.535 6**	0.477 6+	0.529 9+	DF14	0.219 5	**0.980 4**–	0.156 9+
DF2	0.670 2	0.659 4+	**0.692 5**–	*F*5	**0.741 8**	0.108 8+	0.713 1+
DF3	**0.488 6**	0.452 4+	0.478 6+	*F*6	**0.601 5**	0.129 3+	0.566 9+
DF4	0.680 8	**0.691 0**–	0.687 9–	*F*7	**0.611 3**	0.199 7+	0.597 0+
DF5	**0.394 6**	0.298 3+	0.356 9+	DMOP1	**0.532 9**	0.518 2+	0.530 4+
DF6	**0.657 5**	0.073 1+	0.484 1+	DMOP2	**0.535 1**	0.478 7+	0.529 5+
DF7	**0.305 5**	0.293 2+	0.290 1+	DMOP3	**0.713 9**	0.676 8+	0.709 6+
DF8	0.560 5	**0.577 5**–	0.564 6–	FDA1	**0.709 4**	0.648 0+	0.705 7+
DF9	**0.485 3**	0.400 2+	0.482 9+	FDA2	0.935 8	0.938 6–	**0.939 3**–
DF10	0.666 3	**0.996 9**–	0.692 1–	FDA3	**0.508 3**	0.483 2+	0.508 3=
DF11	0.075 0	**0.982 5**–	0.069 0+	FDA4	0.519 1	**0.983 1**–	0.513 3+
DF12	0.602 0	**0.990 9**–	0.596 3+	FDA5	0.517 4	**0.983 2**–	0.512 1+
DF13	0.449 1	**0.984 8**–	0.421 2+	+/−/=	—	15/10/0	19/5/1

注:+/−/=分别表示ACHMP算法与其他算法相比更好、更差和相似.

表 6 KT-MOEA/D、PPS-MOEA/D、SVR-MOEA/D、Tr-MOEA/D、KF-MOEA/D、RI-MOEA/D 和 ACHMP 算法的 MHV 值

测试函数	ACHMP	KT-MOEA/D	PPS-MOEA/D	SVR-MOEA/D	Tr-MOEA/D	KF-MOEA/D	RI-MOEA/D
DF1	**0.535 6**	0.518 7+	0.343 3+	0.517 5+	0.520 5+	0.464 5+	0.529 2+
DF2	**0.670 2**	0.647 7+	0.478 2+	0.628 7+	0.632 8+	0.575 0+	0.640 1+
DF3	**0.488 6**	0.398 0+	0.216 4+	0.243 4+	0.482 0+	0.440 7+	0.430 1+
DF4	**0.680 8**	0.435 0+	0.221 2+	0.351 3+	0.450 7+	0.291 9+	0.445 0+
DF5	0.394 6	0.664 6−	0.455 1−	0.049 5+	0.678 8−	**0.695 5**−	0.669 3−
DF6	0.657 5	0.842 6−	0.450 2+	0.193 6+	**0.920 8**−	0.891 3−	0.914 5−
DF7	0.305 5	0.868 1−	0.383 9−	**0.932 2**−	0.930 2−	0.877 2−	0.904 5−
DF8	0.560 5	0.682 7−	0.322 2+	0.834 7−	**0.837 8**−	0.739 9−	0.783 7−
DF9	0.485 3	0.452 7+	0.297 9+	0.282 6+	0.534 7−	0.513 4−	**0.546 9**−
DF10	0.666 3	**0.911 8**−	0.706 8−	0.815 9−	0.875 0−	0.841 7−	0.891 7−
DF11	0.075 0	**0.825 5**−	0.175 5−	0.725 2−	0.266 5−	0.255 4−	0.276 8−
DF12	0.602 0	0.172 7+	0.217 5+	0.157 6+	0.394 9+	0.471 5+	**0.645 1**−
DF13	0.449 1	**0.672 7**−	0.383 6+	0.115 8+	0.668 0−	0.671 0−	0.670 3−
DF14	0.219 5	**0.635 2**−	0.456 3−	0.056 5+	0.628 4−	0.587 6−	0.610 8−
+/−/=	—	6/8/0	9/5/0	10/4/0	5/9/0	5/9/0	4/10/0

注：+/−/=分别表示 ACHMP 算法与其他算法相比更好、更差和相似.

表 7 KF、PPS、SGEA、MCPDMO 和 ACHMP 算法的性能评估结果对比

测试函数	(τ_t,n_τ)	ACHMP	KF	PPS	SGEA	MCPDMO
DF2	(5,10)	**0.110 166**	0.218 180+	0.599 520+	0.122 230+	0.146 530+
	(20,10)	**0.011 178**	0.029 080+	0.123 120+	0.102 470+	0.026 331+
DMOP1	(5,10)	0.033 123	0.048 752+	0.167 230+	**0.010 440**−	0.042 246+
	(20,10)	0.014 703	**0.006 173**−	0.029 608+	0.010 254−	0.011 198−
FDA1	(5,10)	**0.017 796**	0.282 400+	1.750 100+	0.048 000+	0.176 810+
	(20,10)	**0.006 614**	0.011 341+	0.084 774+	0.021 999+	0.017 046+
FDA2	(5,10)	0.106 823	0.189 820+	0.420 530+	0.026 520−	**0.011 133**−
	(20,10)	0.116 822	0.183 590+	0.188 970+	0.016 478−	**0.005 633**−
FDA3	(5,10)	**0.048 933**	0.456 620+	1.642 100+	0.085 423+	0.112 590+
	(20,10)	**0.047 336**	0.292 580+	0.380 740+	0.053 304+	0.069 861+
FDA4	(5,10)	0.117 161	0.150 950+	0.402 790+	**0.084 430**−	2.810 200+
	(20,10)	0.076 424	0.054 383−	0.089 170+	**0.051 876**−	2.454 500+
+/−/=	—	—	10/2/0	12/0/0	6/6/0	9/3/0

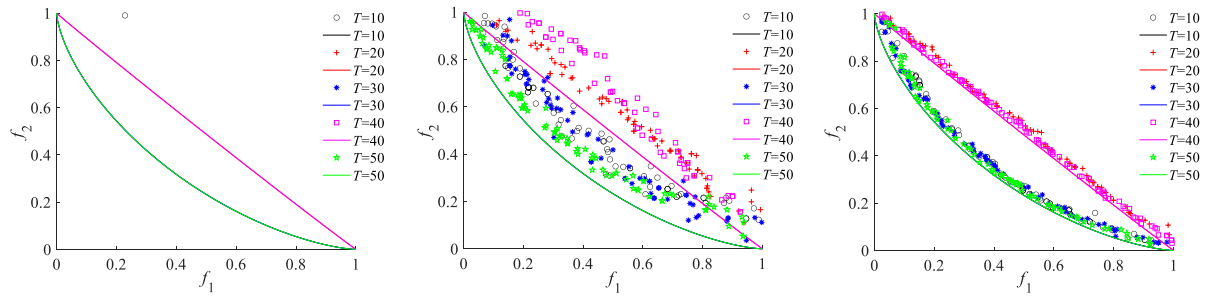
注：+/−/=分别表示 ACHMP 算法与其他算法相比更好、更差和相似.

可以较好的利用以往优异解的历史信息，使算法收敛性更好. 在 FDA2 上 MCPDMO 表现较为优秀，展示了其多中心点预测策略的优异性. 在上述的 DMOP1、FDA2、FDA4 三个测试函数中，ACHMP 算法虽然不是最优的，但也是次优，说明了 ACHMP 具有较强的竞争性.

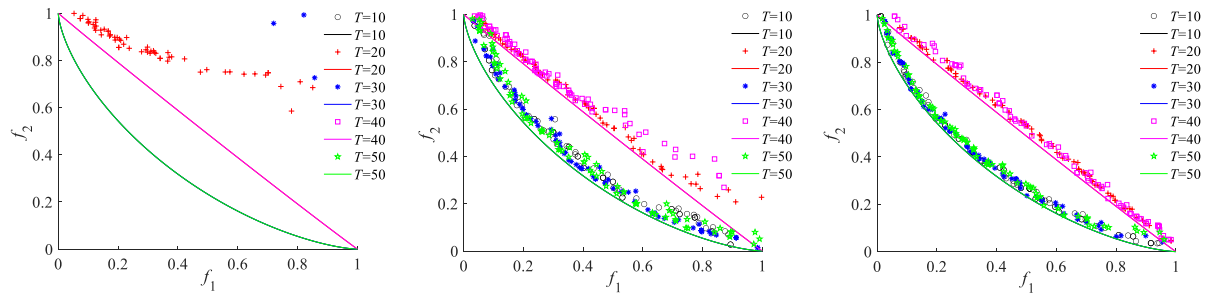
为了分析环境变化的严重程度对于算法性能的影响，我们在不同测试函数上进行了实验. 其中，种群大小为 100，τ_t 为 10，n_τ 分别设置为 5、10 和 20. 例如在强度为 5、10 和 20 时，DMOP1 的 MIGD 值分别为 0.039 8、0.022 3 和 0.019 1；DF2 的 MIGD 值分别为 0.500、0.034 1 和 0.024 9；FDA2 的 MIGD 值分别为 0.115 3、0.113 5 和 0.046 7；FDA3 的 MIGD 值分别为 0.110 5、0.047 3 和 0.024 2. 可以看出，ACHMP 算法对 n_τ 很敏感，n_τ 越大，算法的性能越好. 在测试的 13 个测试函数中，ACHMP 算法在 10 个上领先，展现了 ACHMP 在环境参数变化的时候整体效果优于 DSS 和 DNSGA-Ⅱ-A.

4.3.3 获得的前沿面分布图对比

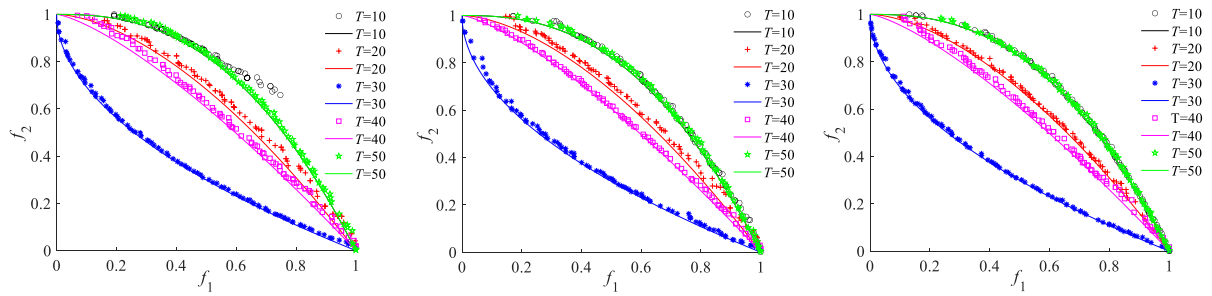
选取环境变化的第 10、20、30、40、50 次这五个不同时间点的 Pareto 最优解集，并将其绘制在目标空间的 Pareto 前沿图上，对 DNSGA-Ⅱ-A、DSS 和 ACHMP 三种算法的性能进行具体直观的比较. 三种算法在各个测试函数上的结果如图 2 所示.



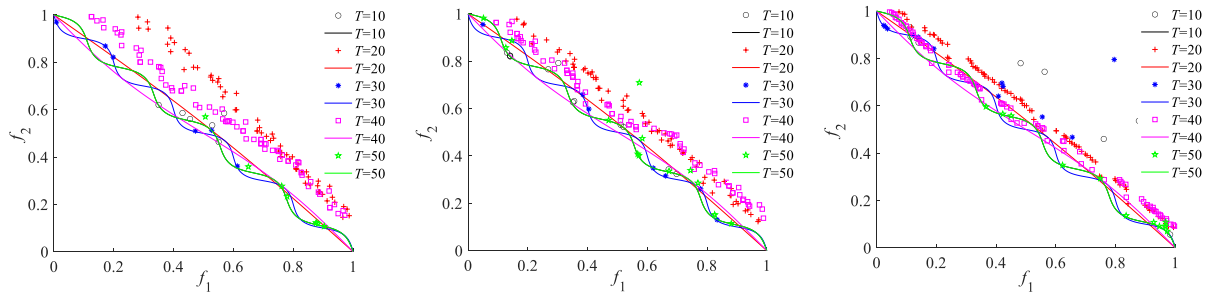
(a) DNSGA-Ⅱ-A($F6$)　(b) DSS($F6$)　(c) ACHMP($F6$)



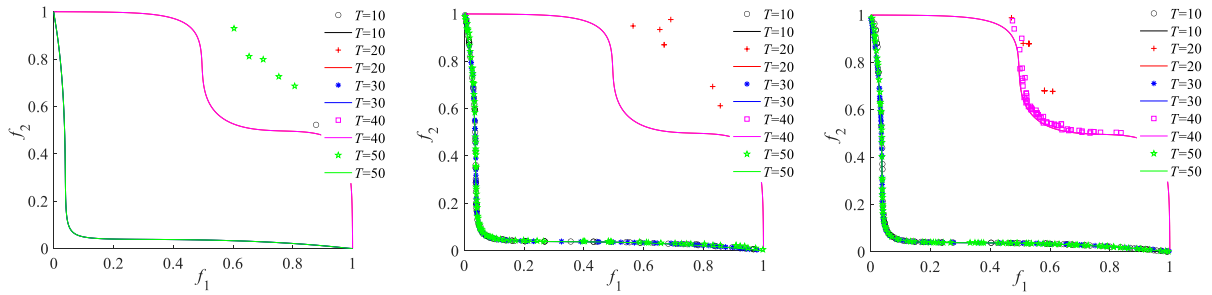
(d) DNSGA-Ⅱ-A($F7$)　(e) DSS($F7$)　(f) ACHMP($F7$)



(g) DNSGA-Ⅱ-A(DF3)　(h) DSS(DF3)　(i) ACHMP(DF3)



(j) DNSGA-Ⅱ-A(DF5)　(k) DSS(DF5)　(l) ACHMP(DF5)



(m) DNSGA-Ⅱ-A(DF6)　(n) DSS(DF6)　(o) ACHMP(DF6)

图2　DNSGA-Ⅱ-A、DSS、ACHMP三种算法在各个测试函数上所得Pareto前沿

由图2可以发现，DNSGA-Ⅱ-A在大多数测试函数上的收敛性都比较差，尤其是在进化的后期，这种情况更为明显，DSS的收敛性要优于DNSGA-Ⅱ-A，但是个别测试函数的后期仍难以收敛，这种情况在*F*6、*F*7等函数测试上尤为明显.

与这两种函数产生鲜明对比的是，在*F*6、*F*7、DF3等绝大多数的测试函数中，ACHMP收敛性都要更强，在前中后期的分布相对都更加均匀，要显著优于DNSGA-Ⅱ-A，相对于DSS也要更贴近测试函数的理想Pareto前沿. 如在DF6测试函数上，DNSGA-Ⅱ-A基本无法收敛至理想Pareto前沿，DSS在第20次环境变化和第40次环境变化时也基本无法收敛，而ACHMP则在环境变化至第40次时基本收敛至理想Pareto前沿.

4.3.4 消融实验

为了验证ACHMP算法的无迹卡尔曼中心点预测、差分向量角度修正、多种群方法的有效性，设计了消融实验，依次去除上述三种方法，得到的三种算法分别记为ACHMPT1、ACHMPT2和ACHMPT3，并在*F*5、*F*6等测试函数上进行了测试，结果如表8所示.

表8 消融实验下三种算法的MIGD值

测试函数	ACHMP	ACHMPT1	ACHMPT2	ACHMPT3
*F*5	**0.164 2**	1.287 8	0.179 7	0.801 2
*F*6	**0.077 5**	1.081 3	0.090 6	0.459 4
*F*7	**0.067 2**	0.398 6	0.077 7	0.248 5
DF3	**0.026 6**	0.048 2	0.030 1	0.036 2
DF5	**0.163 8**	0.228 5	0.182 3	0.238 6
DF7	**0.463 7**	0.535 3	0.476 8	0.547 1
DMOP1	**0.022 3**	0.034 4	0.039 9	0.037 4
DMOP2	**0.013 8**	0.019 4	0.015 5	0.024 1
DMOP3	**0.007 4**	0.014 2	0.008 9	0.018 4

对比表8中的数据，可以发现，在ACHMPT1上所有测试函数函数的MIGD值均偏大. 在中心点变化较为复杂的*F*5、*F*6、*F*7三个测试函数上表现明显，是因为传统中心点预测方式误差较大，差分向量角度修正是基于中心点得到的，中心点误差过大，角度修正便失去了其原有的意义. 相比之下，在中心点变化较简单的DMOP1、DMOP2等测试函数中，传统中心点预测的误差小，差分向量的角度修正便可以发挥其积极的作用，使算法性能更优秀.

在ACHMPT3上，中心点的预测策略被替换为无迹卡尔曼滤波预测，*F*5、*F*6和*F*7三个测试函数的MIGD值发生了明显的减小，充分表明了无迹卡尔曼预测中心点相较于传统的预测方式更为精准，其预测的中心点更贴近于真实的Pareto前沿，更能发挥角度修正的作用. 而在一些中心点变化相对较简单的测试函数上，使用无迹卡尔曼预测中心点与角度修正预测的效果并没有显著提升，甚至在一些测试函数中效果还略差. 对比ACHMP算法，分析其原因是种群多样性降低，中心点的预测陷入了无迹卡尔曼的偏好区域，角度修正也就无法发挥积极作用. 而当引入多种群协同进化后，种群多样性增加，算法性能有所提升.

对比表8中ACHMP和ACHMPT2的数据，可以看出使用了角度修正的ACHMP算法性能更优异，但是提升幅度不是很大，尤其在一些变化相对简单的函数上，增幅更不明显. 这种情况是由于在预测相对较为精准的情况下，角度修正的误差就变得尤为突出. 由于文章提出的角度修正是对十个维度的共同修正，而一些简单的测试函数往往只有几个维度的变化较为明显，所以对十个维度同时修正时，就会出现误差. 这种误差在中心点预测较为准确时，就变成了算法整体的主要误差来源，造成了算法性能幅度提升较小.

4.3.5 时间复杂度

设目标函数个数为M，种群规模为N，决策变量维度为D，根据算法3和算法4可计算ACHMP时间复杂度：初始化种群的时间复杂度为$O(ND)$；环境检测的时间复杂度为$O(0.05MN)$；计算及修正中心点均为一步操作，时间复杂度为为$O(1)$；多种群协同进化是顺序进行的，因此一级主源种群、二级原始种群、二级修正种群进化时的时间复杂度依次为$O(0.8MND)$、$O(0.1MND)$、$O(0.1MND)$；使用NSGA-II进行静态优化的时间复杂度为$O(MND)$. 因此，所提算法的时间复杂度为$O(ND)+O(1)+O(0.05MN)+O(0.8MND)+O(0.1MND)+O(0.1MND)+O(MND)=O(ND)$，时间复杂度没有明显增加.

5 结论

对于动态多目标问题，如何能在保持多样性与快速收敛之间实现良好的平衡是非常重要的. 本文提出了一种基于角度修正与分级多种群的动态多目标进化算法(ACHMP)，使用UKF模型来预测下一时刻的种群中心点，通过使用不同的预测方式，产生不同的中心点和差分向量，并通过角度修正和分级多种群协同进化的方式，在快速响应环境变化的同时，保证了种群的多样性，避免陷入卡尔曼滤波的偏好区域. 通过与RI-MOEA/D、KT-MOEA/D等6种优异的算法在DF系列测试函数上的比较，可以看出ACHMP算法的结果明显优于其他算法. 在与DSS和DNSGA-Ⅱ-A这两种经典算法多达25个测试函数的对比中，更加体现出ACHMP算法的优势，说明ACHMP算法具有一定的普适性. 在与较为新颖的SGEA、MCPDMO等算法的比较中，ACHMP在处理大多数问题上也具有良好的竞争优势.

虽然ACHMP算法在处理动态多目标问题上表现

的非常有竞争性,但在面对剧烈变化的环境时,如何能快速摆脱以往数据权重的干扰,快速响应环境变化仍是未来需要解决的问题.

参考文献

[1] 钟沛龙, 黎明, 何超, 等. 基于SOM聚类和自适应算子选择的高维多目标进化算法[J]. 电子学报, 2022, 50(8): 1959-1974.
ZHONG P L, LI M, HE C, et al. Many-objective evolutionary algorithm based on som clustering and adaptive operator selection[J]. Acta Electronica Sinica, 2022, 50(8): 1959-1974. (in Chinese)

[2] 马永杰, 陈敏, 龚影, 等. 动态多目标优化进化算法研究进展[J]. 自动化学报, 2020, 46(11): 2302-2318.
MA Y J, CHEN M, GONG Y, et al. Research progress of dynamic multi-objective optimization evolutionary algorithm[J]. Acta Automatica Sinica, 2020, 46(11): 2302-2318. (in Chinese)

[3] 刘淳安, 王宇平. 动态多目标优化的进化算法及其收敛性分析[J]. 电子学报, 2007, 35(6): 1118-1121.
LIU C A, WANG Y P. Evolutionary algorithm for dynamic multi-objective optimization problems and its convergence[J]. Acta Electronica Sinica, 2007, 35(6): 1118-1121. (in Chinese)

[4] FARINA M, DEB K, AMATO P. Dynamic multiobjective optimization problems: Test cases, approximations, and applications[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(5): 425-442.

[5] 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.

[6] AZZOUZ R, BECHIKH S, SAID L BEN. A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy[J]. Soft Computing, 2017, 21(4): 885-906.

[7] YANG Z, JIN Y C, HAO K R. A bio-inspired self-learning coevolutionary dynamic multiobjective optimization algorithm for Internet of Things services[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(4): 675-688.

[8] LI K, FIALHO A, KWONG S, et al. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(1): 114-130.

[9] RUAN G, YU G, ZHENG J H, et al. The effect of diversity maintenance on prediction in dynamic multi-objective optimization[J]. Applied Soft Computing, 2017, 58: 631-647.

[10] 郑金华, 彭舟, 邹娟, 等. 基于引导个体的预测策略求解动态多目标优化问题[J]. 电子学报, 2015, 43(9): 1816-1825.
ZHENG J H, PENG Z, ZOU J, et al. A prediction strategy based on guide-individual for dynamic multi-objective optimization[J]. Acta Electronica Sinica, 2015, 43(9): 1816-1825. (in Chinese)

[11] 马学敏, 杨景明, 孙浩, 等. 基于多区域中心点预测的动态多目标优化算法[J]. 控制与决策, 2022, 37(10): 2477-2486.
MA X M, YANG J M, SUN H, et al. Dynamic multi-objective optimization algorithm based on multi-regional center point prediction[J]. Control and Decision, 2022, 37(10): 2477-2486. (in Chinese)

[12] JIANG M, WANG Z Z, HONG H K, et al. Knee point-based imbalanced transfer learning for dynamic multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2021, 25(1): 117-129.

[13] 李二超, 周扬. 基于分类的多策略预测方法求解动态多目标优化问题[J]. 控制与决策, 2021, 36(7): 1569-1580.
LI E C, ZHOU Y. Classification-based multi-strategy prediction method for dynamic multiobjective optimization problems[J]. Control and Decision, 2021, 36(7): 1569-1580. (in Chinese)

[14] CHEN Y, ZOU J, LIU Y, et al. Combining a hybrid prediction strategy and a mutation strategy for dynamic multiobjective optimization[J]. Swarm and Evolutionary Computation, 2022, 70: 101041.

[15] MURUGANANTHAM A, TAN K C, VADAKKEPAT P. Evolutionary dynamic multiobjective optimization via Kalman filter prediction[J]. IEEE Transactions on Cybernetics, 2016, 46(12): 2862-2873.

[16] CHEN M, MA Y J. Dynamic multi-objective evolutionary algorithm with center point prediction strategy using ensemble Kalman filter[J]. Soft Computing, 2021, 25(7): 5003-5019.

[17] 胡振涛, 杨诗博, 胡玉梅, 等. 基于变分贝叶斯的分布式融合目标跟踪[J]. 电子学报, 2022, 50(5): 1058-1065.
HU Z T, YANG S B, HU Y M, et al. Distributed fusion target tracking based on variational Bayes[J]. Acta Electronica Sinica, 2022, 50(5): 1058-1065. (in Chinese)

[18] WU Y, JIN Y C, LIU X X. A directed search strategy for evolutionary dynamic multiobjective optimization[J]. Soft Computing, 2015, 19(11): 3221-3235.

[19] GOH C K, TAN K C. A competitive-cooperative coevolu-

tionary paradigm for dynamic multiobjective optimization [J]. IEEE Transactions on Evolutionary Computation, 2009, 13(1): 103-127.

[20] JIANG S Y, YANG S X, YAO X, et al. Benchmark functions for the CEC' 2018 competition on dynamic multiobjective optimization[R]. Newcastle: Newcastle University, 2018.

[21] SIERRA M R, COELLO COELLO C A. Improving PSO-based multi-objective optimization using crowding, mutation and ∈-dominance[C]//Evolutionary Multi-Criterion Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005: 505-519.

[22] WHILE L, HINGSTON P, BARONE L, et al. A faster algorithm for calculating hypervolume[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(1): 29-38.

[23] CAO L L, XU L H, GOODMAN E D, et al. Evolutionary dynamic multiobjective optimization assisted by a support vector regression predictor[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(2): 305-319.

[24] 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.

[25] ZHANG Q F, LI H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712-731.

[26] 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. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007: 803-817.

[27] JIANG S Y, YANG S X. A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(1): 65-82.

作者简介

杨 乐 女,1999年出生于陕西省西安市. 现为西北师范大学硕士研究生,主要研究方向为进化计算.
E-mail: ylyanglele@163.com

马永杰 男,1967年出生于甘肃省平凉市. 西北师范大学教授、博士生导师. 主要研究方向为智能计算、图像处理、测控技术.
E-mail: myjmyj@nwnu.edu.cn

平镐羽 男,1998年出生于安徽省合肥市. 现为西北师范大学硕士研究生,主要研究方向为图像处理、进化计算.
E-mail: 2021222472@nwnu.edu.cn

杨 岳 男,1997年出生于甘肃省白银市. 现为西北师范大学硕士研究生,主要研究方向为进化计算.
E-mail: 205752504@qq.com