Journal of System Simulation

Volume 32 | Issue 2 Article 7

2-19-2020

MOEA/D Algorithm Based on the Hybrid Framework for Multi-objective Evolutionary Algorithm

Hongjun Tian
1. College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China;;2. Postdoctoral Research Station of Shenwan Hongyuan Secur Co Ltd. and Fudan University, Shanghai 200031, China;

Wang Lei
1. College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China;;

Qidi Wu
1. College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China;;

MOEA/D Algorithm Based on the Hybrid Framework for Multi-objective Evolutionary Algorithm

Abstract

Abstract: Aimto the difficulties of designing the bonding mechanism of global optimization algorithm and local search strategy for hybrid multi-objective evolutionary algorithm, and of improving the performance of multi-objective evolutionary algorithms, *based on the feedback control idea, a systematic and modular hybrid MOEA/D algorithm combining the global optimization and local search is proposed.* In the algorithm, *a diversity measure method based on crowded entropy is designed*; *a local search strategy based on simplified quadratic approximation and population diversity enhancement strategy for MOEA/D is proposed*. The numerical experiments show that the proposed HMOEA/D can achieve a balance between diversity and convergence of algorithm. The proposed hybrid framework can effectively improve the performance of existing multi-objective evolutionary algorithms.

Keywords

multi-objective optimization, evolutionary algorithm, hybrid framework, MOEA/D, feedback control

Recommended Citation

Tian Hongjun, Wang Lei, Wu Qidi. MOEA/D Algorithm Based on the Hybrid Framework for Multi-objective Evolutionary Algorithm[J]. Journal of System Simulation, 2020, 32(2): 201-216.

第 32 卷第 2 期 系统仿真学报© Vol. 32 No. 2
2020 年 2 月 Journal of System Simulation Feb., 2020

基于多目标进化算法混合框架的 MOEA/D 算法

田红军[1,2]，汪镭[1]，吴启迪[1]
(1. 同济大学 电子与信息工程学院，上海 201804；2. 申万宏源证券有限公司-复旦大学博士后科研工作站，上海 200031)

摘要：针对混合多目标进化算法中如何设计全局搜索算法和局部搜索策略结合机制的难点问题以及提高多目标进化算法的求解性能，*基于反馈控制思想，提出了一种系统化、模块化的全局优化与局部搜索相结合的混合 MOEA/D 算法，算法中设计了一种基于拥挤熵的种群多样性度量方法；提出了基于简化二次逼近的局部搜索策略，以及针对 MOEA/D 的种群多样性增强策略。*数值实验表明所提算法具有良好性能，可以兼顾算法求解的多样性和收敛性，所提混合框架可有效提升现有多目标进化算法的求解性能。

关键词：多目标优化；进化算法；混合框架；MOEA/D；反馈控制
中图分类号：TP18 文献标识码：A 文章编号：1004-731X (2020) 02-0201-16
DOI: 10.16182/j.issn1004731x.joss.17-9183

MOEA/D Algorithm Based on the Hybrid Framework for Multi-objective Evolutionary Algorithm

Tian Hongjun[1,2], *Wang Lei*[1], *Wu Qidi*[1]
(1. College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China;
2. Postdoctoral Research Station of Shenwan Hongyuan Secur Co Ltd. and Fudan University, Shanghai 200031, China)

Abstract: Aim to the difficulties of designing the bonding mechanism of global optimization algorithm and local search strategy for hybrid multi-objective evolutionary algorithm, and of improving the performance of multi-objective evolutionary algorithms, *based on the feedback control idea, a systematic and modular hybrid MOEA/D algorithm combining the global optimization and local search is proposed.* In the algorithm, *a diversity measure method based on crowded entropy is designed*; *a local search strategy based on simplified quadratic approximation and population diversity enhancement strategy for MOEA/D is proposed*. The numerical experiments show that the proposed HMOEA/D can achieve a balance between diversity and convergence of algorithm. The proposed hybrid framework can effectively improve the performance of existing multi-objective evolutionary algorithms.

Keywords: multi-objective optimization; evolutionary algorithm; hybrid framework; MOEA/D; feedback control

引言

进化算法是一种模拟自然界生物群体进化过程的随机优化方法，具有良好的通用性、隐并行性、自适应性以及鲁棒性等特点，已经广泛应用于复杂问题和多目标优化设计中[1-3]。在过去数十年来，许多学者提出了不同的进化算法来解决多目标优化问题。然而根据“世界上没有免费的午餐”理论，没有一种算法是万能的，每种算法都有其优点和缺点；另外，这些算法往往存在全局搜索能力与收敛

收稿日期：2017-12-18 修回日期：2018-06-06；
基金项目：国家自然科学基金(61075064，61034004，61005090)；
作者简介：田红军(1986-)，男，山东，博士后，研究方向为智能优化、进化计算等；汪镭(1971-)，男，江苏，博士，教授，博导，研究方向为智能优化与控制、人工智能等。

速度之间的矛盾，仅通过调节控制参数往往无法达到“探索”和“开发”能力之间的平衡。例如基于 Pareto 支配关系的算法能够适应各种形状的 Pareto 前沿，但在处理超多目标优化问题时却显得不尽如人意；基于指标的多目标进化算法通过评价指标来指引算法的搜索方向，指导进化过程中新种群的选择，但该方法在处理较多优化目标时算法复杂度急剧增加、算法性能下降；基于分解的算法通用性较好，但是常规的分解方法却容易导致解集多样性的丢失。其他的优化技术也各有优缺点。大量研究表明，融入一些其他策略如存档策略、局部搜索策略、特殊组合算子等，以及不同算法相互结合可有效改善进化算法的求解精度和求解效率，将全局优化算法混合以及与局部搜索策略进行集成，充分发挥各种算法的优势，构成混合优化算法，已经证明是很有效的策略[4-21]。

到目前为止，许多不同类型的混合多目标进化算法相继被提出。文献[10]提出了一种多种群自适应混合多目标进化算法(HMOEA-AMP)，在 HMOEA-AMP 框架中，粒子群优化算法和差分进化算法相结合分别用于帕累托最优解的开发和探索最优解的分布。文献[12]提出了一种基于分解的混合自适应进化算法(HAEA/D)，该算法采用 4 种不同的交叉算子，其基本思想是在搜索空间中当一种搜索算子表现不佳时算法切换为其他搜索算子。文献[17]提出了多目标遗传局部搜索算法(MOGLS)，首次将局部搜索融入多目标遗传算法求解多目标优化问题。在 MOGLS 算法中，Ishibuchi 和 Murata 采用加权和的方法生成局部搜索的目标函数，以进化算法产生的后代种群作为初始点优化该目标函数，以此提高解的质量。文献[18]提出的混合进化算法中，采用了两种方法将局部搜索融入进化算法中，一种是 MOGLS 算法采用的在线方法，一种是后验方法。在文献[19-20]中，Deb 等在 NSGA-II 算法中，融入了以标量化函数--成绩标量函数(Achievement Scalarizing Function，简记为 ASF)为目标函数的局部搜索策略。文献[21]提出了提出一种基于局部搜索与混合多样性策略的多目标粒子群算法。在上述算法中尽管融入局部搜索的多目标进化算法收敛性能有所提高，然而局部搜索只用于改善多目标进化算法求得的解的质量，并没有考虑所得到的解的分布性，算法并不能很好地兼顾解的分布性和均匀性。另外上述混合算法往往只针对某一种算法进行改进，并没有提出一种系统化通用化的混合框架。

如何在算法设计过程中权衡收敛性和多样性之间的关系，即如何设计合理的结合方式使得算法能够充分利用基于种群的优化算法与局部搜索策略两者的优点是混合算法设计的难点。反馈是控制领域的灵魂，基于这一理念，本文在现有算法基础上，基于“检测-响应”的反馈控制思想，采用系统化和模块化的设计理念，以 MOEA/D 作为启发式算法提出了一种将全局搜索与局部搜索相结合的混合 MOEA/D(HMOEA/D)算法。MOEA/D 及其改进版本已成功解决了一系列复杂的多目标优化问题[22-26]。所提 HMOEA/D 算法包含不同模块，投影聚类模块可以维持和增加进化群体的多样性和分布性，该模块中结合拥挤距离和分布熵的概念，提出了一种基于拥挤熵的种群多样性度量方法；局部搜索模块和种群多样性增强模块可以使算法兼顾解的多样性和收敛性，分别设计了基于简化二次逼近的局部搜索策略和适用于 MOEA/D 算法的种群多样性增强策略。所提 HMOEA/D 算法力图在全局和局部搜索中达到平衡和融合，使之更有效地求解多目标优化问题。最后将该算法用于求解 13 个无约束的 CEC09 标准测试问题来检测算法性能，并将实验结果与当前其他一些优秀的多目标进化算法进行了比较，实验验证了所提算法的有效性。

1 多目标优化问题

不失一般性，以最小化问题为例，多目标优化问题给出以下定义：

定义 1：多目标优化问题

在空间 $\boldsymbol{\Omega}$ 中寻找向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$，使

https://dc-china-simulation.researchcommons.org/journal/vol32/iss2/7
DOI: 10.16182/j.issn1004731x.joss.17-9183

得目标函数 $F(x)$中的 m 个目标 $f_i(x)(i=1,2,...,m)$最小，即：

$$\min_{x\in\Omega} \ y=F(x)=\{f_1(x),f_2(x),\cdots,f_m(x)\} \tag{1}$$

通常，当目标个数 $m>3$ 时，称为高维多目标优化问题，相应地，目标个数为 2 或 3 的问题称为低维多目标优化问题。

定义 2：Pareto 支配

假设 $\boldsymbol{x}_a,\boldsymbol{x}_b\in\Omega$，是式(1)所示多目标优化问题的两个可行解，当且仅当：

$$\forall i\in\{i=1,2,\cdots,m\},f_i(x_a)\leqslant f_i(x_b)$$
$$且\exists j\in\{j=1,2,\cdots,m\},f_j(x_a)\leqslant f_j(x_b) \tag{2}$$

则称与 $\boldsymbol{x}_b$ 相比，$\boldsymbol{x}_a$ 是 Pareto 占优的，记做 $\boldsymbol{x}_a\prec\boldsymbol{x}_b$，也称 $\boldsymbol{x}_a$ 支配 $\boldsymbol{x}_b$。

定义 3：Pareto 最优解

可行解 $\boldsymbol{x}^*\in\boldsymbol{X}$，$\boldsymbol{x}^*$ 为 Pareto 最优解(或非支配解)，当且仅当 $\boldsymbol{x}^*$ 满足：

$$\neg\exists\boldsymbol{x}\in\Omega:\boldsymbol{x}\prec\boldsymbol{x}^* \tag{3}$$

定义 4：Pareto 最优解集(***PS***)

Pareto 最优解集是决策空间 $\boldsymbol{X}$ 中的所有 Pareto 最优解构成的集合，其定义如下：

$$\boldsymbol{PS}=\{\boldsymbol{x}^*|\neg\exists\boldsymbol{x}\in\Omega:\boldsymbol{x}\prec\boldsymbol{x}^*\} \tag{4}$$

定义 5：Pareto 前沿(***PF***)

Pareto 前沿是由 Pareto 最优解集 ***PS*** 中的所有 Pareto 最优解在目标空间中对应的目标向量组成的集合，其定义如下：

$$F=\{x^*\in PS\,|\,F(x^*)=(f_1(x^*),f_2(x^*),\cdots,f_m(x^*)\} \tag{5}$$

多目标优化问题就是获取一组在目标空间中尽量靠近 Pareto 前沿且均匀分布的解。

2 HMOEA/D 算法

2.1 HMOEA/D 算法框架

所提 HMOEA/D 算法框架如图 1 所示。

2.2 HMOEA/D 算法设计

在 HMOEA/D 算法中，聚类算法、种群多样性度量方法、局部搜索策略、应用局部搜索的概率、局部搜索的个体选择以及引入局部搜索后如何平衡“探索”和“开发”能力等都是影响算法性能的关键。本节针对以上问题及算法各子模块进行详细描述。

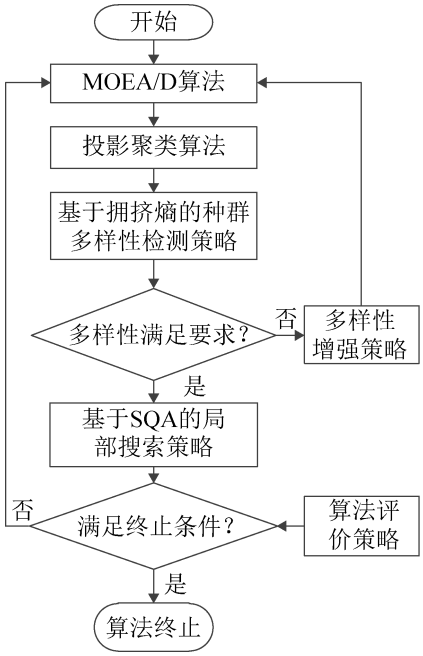

图 1 HMOEA/D 算法框架
Fig. 1 Framework for HMOEA/D

2.2.1 投影聚类模块

投影聚类的目的是维持和增加进化群体的多样性和分布性，它将进化群体中的个体划分为若干个由相似个体组成的子种群，通过计算个体或类之间的相似度来实现。聚类的质量指标作为种群多样性的近似度量，保持种群多样性去搜索更广的目标空间以及获得均匀分布的 Pareto 前沿是非常必要的。该模块算法设计如算法 1 所示。

算法 1 投影聚类算法

模块输入：种群规模 N；聚类数 K；第 t 代种群 P_t

模块输出：第 t 代聚类质量指标 $Q_{\text{-current}}$ 第 t 代聚类 RP_i^t，$i=1,2,\cdots,K$

step 1: 考虑包含目标向量的解集 $\boldsymbol{\Psi}$：

$F_i=(0,\cdots,0,f_i^{\max},0,\cdots,0)^{\mathrm{T}}\in R^k$，这里 $f_i^{\max}$ 是种群 P_t 中目标函数 f_i 的最大值，定义超平面 H：$H=\{f\in R^k:\langle w^S,f\rangle+b^S=0\}$，这里，$H$ 是 k 维欧式空间里的一个 k–1 维线性子空间。$\langle\cdot,\cdot\rangle$ 为欧氏点积，$\boldsymbol{\Psi}\subset H$，$w^S=\left(1/f_1^{\max},\cdots 1/f_m^{\max}\right)$，$b^S=-1$，目

标向量 $\boldsymbol{f}$ 在超平面 H 上的正交投影 PI 定义如下：

$$PI = \frac{1 - \langle w^S, f \rangle}{\|w^S\|^2} w^S + f \tag{6}$$

利用公式(6)，种群 P_t 中所有个体在超平面 H 上的正交投影得到种群 P_t'，在公式(6)中，若 $f_i^{\max} = 0$，则令 $f_i^{\max} = 10^{-6}$。

step 2：种群 P_t' 生成 K 个聚类，每个聚类子群记为 $RP_i^t, i = 1,2,\cdots,K$。

step 3：评价种群 P_t' 的多样性。

step 4：用种群 P_t 中的相应个体更新每个聚类中的子群。

在算法 1 中，种群的多样性指标非常重要，它决定着算法下一步所要采取的进化策略。为了准确测量进化种群的种群多样性，一种有效评估种群拥挤程度的测量方法是必要的。在已有 MOEAs 中提出了众多的拥挤度估计方法[26]。在 SPEA 和 SPEA2 算法中，基于密度估计的方法被用来测量个体间的拥挤度，然而这种方法具有较大的计算复杂度。Deb 等在 NSGA-II 中通过计算其相邻解在每个目标上的平均距离来获得一个解与相邻个体之间的拥挤估计，然而该方法没有考虑相邻解之间的分布，单一的平均距离不能准确地反映解之间的拥挤程度[20]。如图 2 所示，解 A 比解 B 更加拥挤，然而采用基于拥挤距离的方法计算，解 A 的拥挤距离为 13l 却大于解 B 的拥挤距离 10l。

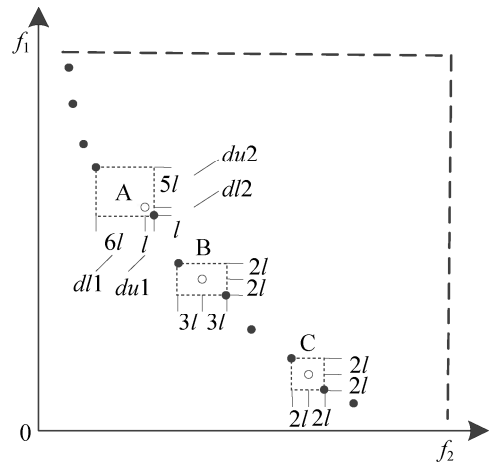

图 2 基于拥挤距离的拥挤程度估计

Fig. 2 Congestion estimation based on crowding distance

熵是一种度量微观分布均匀性的方法。在热力学中，熵表示系统混乱状态，而在生态学中熵表示物种的多样性。我们采用熵的概念来描述解在目标空间中的分布[26]。第 i 个解沿着第 j 个目标在目标空间中的分布熵定义如下：

$$\begin{cases} E_{ij} = 1\left[pl_{ij} \log_2(pl_{ij}) + pu_{ij} \log_2(pu_{ij}) \right] \\ pl_{ij} = \dfrac{dl_{ij}}{c_{ij}} \\ pu_{ij} = \dfrac{du_{ij}}{c_{ij}} \\ c_{ij} = dl_{ij} + du_{ij} \end{cases} \tag{7}$$

式中：dl_{ij} 和 du_{ij} 为第 i 个解分别与其上下相邻解在第 j 个目标上的距离，当一个点落在相邻 2 个点的正中间时，该点具有最好的分布。根据式(7)，设 $l = 1$，分别计算解 A 和解 B 沿着目标函数 f_1 的分布熵为 $EA_1 = 0.59$ 和 $EB_1 = 1$，显然，分布熵能够正确地描述目标空间中解的分布情况。然而，分布熵也不能准确地反映一个解与相邻解之间的拥挤程度。例如在图 2 中，计算解 C 得分布熵为 $EC_1 = 1$，解 C 具有比解 A 更好的分布熵，而实际上解 C 的拥挤度大于解 A，这是由于分布在解 C 两边相邻解之间的平均距离远小于分布在解 A 两边相邻解之间的平均距离。因此，当两个解具有相同的分布熵值时，我们还要考虑解的拥挤距离。为了准确估计目标空间中解的分布情况，必须同时考虑解的拥挤距离和分布熵。基于以上分析，结合分布熵和拥挤距离，我们采用拥挤熵的新的拥挤程度估计方法来准确估计目标空间中解的拥挤程度，在计算拥挤熵之前，对每一目标进行归一化。

$$CE_i = \sum_{j=1}^{k} (c_{ij} E_{ij}) / (f_j^{\max} - f_j^{\min}) = - \frac{\sum_{j=1}^{k} [dl_{ij} \log_2(pl_{ij}) + du_{ij} \log_2(pu_{ij})]}{(f_j^{\max} - f_j^{\min})} \tag{8}$$

式中：$f_j^{\max}$ 和 $f_j^{\min}$ 分别为第 j 个目标函数的最大最小值；k 为优化目标个数。根据式(8)，可得解 A，B 和 C 的拥挤熵分别 $CE_A = 8.04$，$CE_B = 10$ 和

$CE_C=3$。由此可见，基于拥挤熵的评估方法能够准确地评估一个解与周围其他解在目标空间中的拥挤程度。

为计算拥挤熵，需要对每个解的每一目标函数值进行升序排序，对于每一目标函数的边界解(具有最大和最小目标函数值的解)，通过赋予其一个无穷大的拥挤熵值使其始终被选择，而所有中间解按式(8)赋予一个拥挤熵值。将每一目标下的拥挤熵值相加就得到了每个解总的熵值。

根据上面所提的拥挤熵概念，我们提出了一个新的种群多样性估计方法，定义如下：

$$Q=\sum_{i=1}^{K}\frac{1}{\left|RP_i^t\right|}\sum_{j=1}^{\left|RP_i^t\right|}CE_j \tag{9}$$

式中：$\left|RP_i^t\right|$为某个聚类的个体数，算法中，$Q_{\text{-current}}$为第 t 代聚类质量指标，利用 $Q_{\text{-current}}$ 为后面 t_c 次迭代设置下限阈值 $Q_{\text{-bound}}=0.25\cdot Q_{\text{-current}}$，从($t$+1)代起，每一代 $Q_{\text{-current}}$ 都与 $Q_{\text{-bound}}$ 相比较，直到第 $(t+t_c)$ 次迭代完成。在这 t_c 次迭代中，如果 $Q_{\text{-current}}$ 小于 $Q_{\text{-bound}}$，表示种群具有较差的种群多样性，此时，通过激活种群多样性增强策略增加种群多样性。反之种群具有较好的种群多样性，种群多样性增强策略不会被激活。在该模块中，根据相关文献的研究表明，用 $Q_{\text{-bound}}$ 代替第 t 代的 $Q_{\text{-current}}$ 作为后续 t_c 次迭代的下限阈值，即可以防止陷入局部最优又可以防止频繁激活种群多样性增强模块[27]。

2.2.2 基于 SQA 的局部搜索策略

在 HMOEA/D 算法中，局部搜索以概率 p_{local} 选择改进个体，加快算法收敛和提高算法求解精度。只有当投影聚类模块的聚类质量指标表示种群具有良好的种群多样性时该模块才会被激活，否则算法转入种群多样性增强模块增加种群多样性。

局部搜索策略采用文献[28]中的基于简化二次逼近(SQA)的局部搜索策略，改善算法的局部搜索能力，使算法在计算量增加不多的情况下，能提高算法的整体性能。SQA 不需要目标函数的导数信息，计算简捷，是一种简单有效的直接搜索方法；在单目标优化问题中，常用来作为启发式搜索算子加快算法的收敛速度、提高解的精度。在 HMOEA/D 算法中将作为局部搜索算子融入到算法中，以提高算法的搜索性能，来求解多目标优化问题[28]。算法描述如算法 2 所示。

算法 2　局部搜索算法

模块输入：第 t 代种群 P_t，第 t 代聚类 RP_i^t，$i=1,2,\cdots,K$，局部搜索概率 p_{local}

模块输出：改进的 t 代种群 p_t'

step 1：随机选择一个聚类 i，以概率 p_{local} 从 RP_i^t 中选择个体 A，如果没有个体被选中，则终止算法。

step 2：采用基于 SQA 的局部搜索算法进行局部搜索。

step 3：利用局部搜索获得的解取代 P_t 中与 A 相对应的个体，更新后的种群记为 P_t'。

step 4：根据所选择的全局进化算法，重置算法参数。

以最小化问题为例，设 $f(x^a)$，$f(x^b)$，$f(x^c)$ 为优化目标中 3 个最小的函数值，$f(x^a)<f(x^b)<f(x^c)$，$\boldsymbol{x^a}$，$\boldsymbol{x^b}$，$\boldsymbol{x^c}$ 为对应向量，其中，$\boldsymbol{x^a}=(\boldsymbol{x_1^a},\boldsymbol{x_2^a},\cdots,\boldsymbol{x_n^a})^{\mathrm{T}}$，$\boldsymbol{x^b}=(\boldsymbol{x_1^b},\boldsymbol{x_2^b},\cdots,\boldsymbol{x_n^b})^{\mathrm{T}}$，$\boldsymbol{x^c}=(\boldsymbol{x_1^c},\boldsymbol{x_2^c},\cdots,\boldsymbol{x_n^c})^{\mathrm{T}}$ 则可由式(10)得到逼近点 $\boldsymbol{x^{i'}}=(\boldsymbol{x_1^{i'}},\boldsymbol{x_2^{i'}},\cdots,\boldsymbol{x_n^{i'}})$。

$$\begin{cases} x_k^{i'}=\dfrac{1}{2}\dfrac{A_k}{B_k},k=1,2,\cdots,n \\ A_k=[(x_k^a)^2-(x_k^c)^2]f(x^b)+[(x_k^a)^2-(x_k^b)^2] \\ \qquad f(x^c)+\left[(x_k^b)^2-(x_k^c)^2\right]f(x^a) \\ B_k=(x_k^c-x_k^b)f(x^a)+(x_k^a-x_k^c) \\ \qquad f(x^b)+(x_k^a-x_k^b)f(x^c) \end{cases} \tag{10}$$

对于分解的每个子问题，构建 SQA 模型时，3 个点 $\boldsymbol{x^a}$，$\boldsymbol{x^b}$ 和 $\boldsymbol{x^c}$ 的选取对两次逼近的效果非常关键，当公式(10)分母为零时公式失效，为避免这种情况且能得到较好的逼近点，我们采用文献[29]所采用的启发式选择方法：对于子问题 i，以子问题 i 的当前最优解 $f(x^{ia})$ 为一个点，另两点 $f(x^{ib})$ 和 $f(x^{ic})$ 在它的邻居中随机选择，或选择与子问题 i 关系最近的两个邻居，选择的 3 点互不相同。

在进化过程中，当碰巧选择的 3 个点 $\boldsymbol{x^a}$，$\boldsymbol{x^b}$ 和

$\boldsymbol{x}^c$ 的某一分量相等或 3 项和为零，则可能出现 B_k 为零的情况。在进化后期，随着种群的多样性降低，这种情况容易出现。对于 $k \in \{1,2,\cdots,n\}$，若 $B_k \neq 0$，则由式(10)计算逼近点；若 $B_k=0$，则不做二次逼近，令 $\boldsymbol{x}_k^{i'}=\boldsymbol{x}_k^a$，然后对新生成的逼近点进行更新。

2.2.3 种群多样性增强策略

只有当投影和聚类模块检测到种群具有较差的种群多样性时，该模块才会被激活来增加种群多样性。算法描述如算法 3 所示。

算法 3　HMOEA/D 多样性增强策略

模块输入：第 t 次迭代混合种群：$R_t = P_t \cup Q_t$，P_t 和 Q_t 分别为父代和 MOEA/D 算法更新解的过程中删除个体组成的种群；种群规模：N

模块输出：第 t+1 次迭代父代种群：P_{t+1}

step 1：将种群 R_t 作为投影聚类模块输入，得到 k 个聚类 RP_j^t，$j=1,2,\cdots,k$。

step 2：根据适应度值对每个子种群 RP_j^t 中的个体进行排序，利用排序依次抽取 RP_j^t 个体，组成新的子群 $RP1_j^t$，$j=1,2,\cdots,k$，例如，$RP1_1^t$ 是包含种群 R_t 中最好适应度值个体的子种群，$RP1_k^t$ 是包含种群 R_t 中最差适应度值个体的子种群。

step 3：利用 $RP1_j^t$ 中的个体 R_j，$j=1,2,\cdots,k$ 重建 t+1 代父代种群 P_{t+1}，计算 P_{t+1} 中来自各个子群的个体数：$\left|R_j\right| = N\frac{1-r}{1-r^{k+1}}r^{j-1}$，若对于任意 j，$\left|R_j\right| > \left|RP1_j^t\right|$，则重置其他子种群解的数量 $\sum_{j=1}^{k}\left|R_j\right| = N$。

step 4：根据适应度值对子种群 $RP1_j^t$，$j=1,2,\cdots,k$ 中的个体进行升序排序，并根据 step3 确定的个体数选择 R_j 最佳个体。

step 5：初始化 $P_{t+1}=\varnothing$，$i=1$，$j=1$；令 $P_{t+1}=P_{t+1}\cup\alpha_{i,j}$，$\alpha_{i,j}\in RP1_j^t$，$i=i+1$，$i \leqslant \left|R_j\right|$；如果 $j>K$，则 step6，否则 $j=j+1$。

step 6：stop。

2.2.4 外部精英存档策略

在 HMOEA/D 算法中，为保证最终求得的解是非占优的，算法采用精英保留策略，设置一个外部档案集来保存整个进化过程中所发现的非劣解。进化开始时，令外部存档为空。随着进化的进行，通过将新产生的个体与外部存档中的个体逐一进行比较，选择优良个体进入外部存档。非占优个体与外部存档中个体进行比较采用以下精英存档更新规则：如果新个体支配外部存档中的任何个体，则从外部存档中删除被支配的所有个体；如果新个体被外部存档中的任何个体所支配，则拒绝新个体进入外部存档；如果新个体与外部存档中的个体互不占优，则新个体属于最优解而进入外部存档。该精英存档更新规则决定了非占优新个体进入外部存档与否。在本文实验中，当外部存档达到设定的最大容量时，采用上文提到的拥挤熵方法来删除最拥挤的个体以保持外部存档的大小。

2.2.5　HMOEA/D 算法描述

在本文所提的 HMOEA/D 算法中，我们采用切比雪夫法(Tchebycheff Approach)将求解的问题分解为几个单目标子问题[30-31]。算法描述如算法 4 所示。

算法 **4**　HMOEA/D

算法输入：

① MOP(1-1)；

② 停止准则；

③ N：MOEA/D 所分解子问题个数；

④ $\boldsymbol{\lambda}^1,\boldsymbol{\lambda}^2,\cdots,\boldsymbol{\lambda}^N$：分布均匀的 N 个权向量，$\boldsymbol{\lambda}^N=\boldsymbol{\lambda}_1^N,\boldsymbol{\lambda}_2^N,\cdots,\boldsymbol{\lambda}_m^N$；

⑤ T：权向量的邻居个数；

⑥ $A_{\max}$：外部档案 A 存档规模。

算法输出：

① $\boldsymbol{PS}$：$\{\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^N\}$

② $\boldsymbol{PF}$：$\{\boldsymbol{F}(\boldsymbol{x}^1),\boldsymbol{F}(\boldsymbol{x}^2),\cdots,\boldsymbol{F}(\boldsymbol{x}^N)\}$

step 1：算法初始化：

step 1.1：计算任意两个权向量之间的欧氏距离，为每个权向量选出最近的 T 个向量作为它的邻居，设 $B(i)=\{i_1,i_2,\cdots,i_T\},i=1,2,\cdots,N$，其中（$\boldsymbol{\lambda}^{i1},\boldsymbol{\lambda}^{i2},\cdots,\boldsymbol{\lambda}^{iT}$）为与 $\boldsymbol{\lambda}^i$ 距离最近的 T 个权向量。

step 1.2：采用随机方法初始化规模为 N 的种群 P_0: $\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^N$，设 $FV^i=F(x^i),i=1,2,\cdots,N$。

step 1.3：初始化 $\mathrm{z}=(\mathrm{z}_1,\mathrm{z}_2,\cdots,\mathrm{z}_m)^{\mathrm{T}}$，

$z_j = \min_{1\leqslant i\leqslant m} f_j(x^i)$， $j=1,2,\cdots,m$。

step 1.4：根据公式(9)，计算个体拥挤熵，更新外部精英存档 A。

step 1.5：初始化 gen=1，$\pi^i=1$，$i=1,2,\cdots,N$。

step 2：选择参与进化的子问题：

采用锦标赛方法，通过计算子问题的效用值 π^i，选出 $\lfloor N/5 \rfloor$ 个子问题进行进化操作。

step 3：对种群 P_t 进行投影聚类操作，采用 2.2.1 节算法 1 中的投影聚类方法，种群多样性判别采用 2.2.1 节基于拥挤熵的多样性判别方法。

step 4：设置 k=1。

step 5：更新：

For $i\in I$，do

step 5.1 选择交叉：采用随机方法生成随机数 rand$\in[0,1]$，则：

$$\xi_{\mathrm{t}}=\begin{cases}B(i) & \text{if rand}<\delta\\ \{1,2,\cdots,N\} & \text{otherwise}\end{cases}$$

δ 为交叉的父代个体来自邻居的概率。

step 5.2 繁殖：设 $r_1=i$，从 B(i) 中随机选出两个邻居 r_2,r_3，由 $\boldsymbol{x}^{r_1},\boldsymbol{x}^{r_2},\boldsymbol{x}^{r_3}\in P_t$ 用 DE 交叉生成新解 y。

step 5.3 修正：采用一定策略作用于 y 生成 y^*。

step 5.4 更新：若 $z_j>f_j(y^*)$，则 $z_j=f_j(y^*)$，$j=1,2,\cdots,m$。

step 6：更新当前解：设置 $c=0$

(1) 若 $c=\eta_r$（η_r 为控制参数)或 $\xi_{\mathrm{t}}=\varnothing$，转向 Step 7。否则，从 ξ_{t} 中随机选择一个个体 j。

(2) $j\in B$(i)，若 $\mathrm{g}^{te}(\mathrm{y}^*\mid\lambda^j,z)\leqslant\mathrm{g}^{te}(x^j\mid\lambda^j,z)$，$x^j\cap Q_t=\varnothing$，则 $Q_t=Q_t\cup x^j$，$\mathrm{x}^j=\mathrm{y}^*$，否则，$Q_t=Q_t\cup x^j$，$\mathrm{x}^j=\mathrm{y}^*$，且 $c=c+1$。

(3) 从 ξ_{t} 中删除 j，返回(1)。

step 7：若 $j<N, j=j+1$, 则转向 step 5。

step 8：种群 P_t 输入投影聚类模块，若种群 P_t 具有良好种群多样性，则对种群 P_t 进行局部搜索，转向 step 9；否则转向种群增强模块 step10。

step 9：SQA 局部搜索，局部搜索概率 $p_{\mathrm{local}}=1/(\eta\cdot N_K)$，$N_K$ 为每个聚类的种群规模：计算 $\mathrm{g}^{te}(x^{i_k}\mid\lambda^i,z),i_k\in B$(i)，设 G=$\left\{\mathrm{g}^{te}(x^{i_k}\mid\lambda^i,z),k=1,2,\cdots,T\right\}$，从 G 中选出对应最小函数值的 3 个点：$x^{i_a},x^{i_b}$和$x^{i_c}$，$x^{i_a}=(x_1^{i_a},x_2^{i_a},\cdots,x_n^{i_a})^{\mathrm{T}}$，$x^{i_b}=(x_1^{i_b},x_2^{i_b},\cdots,x_n^{i_b})^{\mathrm{T}}$，$x^{i_c}=(x_1^{i_c},x_2^{i_c},\cdots,x_n^{i_c})^{\mathrm{T}}$，$n$ 为自变量的个数，且 $\mathrm{g}^{te}(x^{i_a}\mid\lambda^i,z)<\mathrm{g}^{te}(x^{i_b}\mid\lambda^i,z)<\mathrm{g}^{te}(x^{i_c}\mid\lambda^i,z)$，$\mathrm{g}^{te}(x^{i_k}\mid\lambda^i,z)$ 简记为 $\mathrm{g}_{i_k}^{te}$，由公式 $x^{i'}=(x_1^{i'},x_2^{i'},\cdots,x_n^{i'})$ 计算逼近点，若 $\left(x_k^{i_c}-x_k^{i_b}\right)\mathrm{g}_{i_a}^{te}+\left(x_k^{i_a}-x_k^{i_c}\right)\mathrm{g}_{i_b}^{te}+\left(x_k^{i_a}-x_k^{i_b}\right)\mathrm{g}_{i_c}^{te}<\varepsilon$，($\varepsilon$ 为较小正实数)，则 $x_k^{i'}=x_k^i$，

否则

$$x_k^{j'}=\frac{\frac{1}{2}\left[\left(x_k^{i_c}\right)^2-\left(x_k^{i_b}\right)^2\right]\mathrm{g}_{i_a}^{te}+\left[\left(x_k^{i_a}\right)^2-\left(x_k^{i_c}\right)^2\right]\mathrm{g}_{i_b}^{te}+\left[\left(x_k^{i_a}\right)^2-\left(x_k^{i_c}\right)^2\right]\mathrm{g}_{i_c}^{te}}{\left(x_k^c-x_k^b\right)\mathrm{g}_{i_a}^{te}+\left(x_k^a-x_k^c\right)\mathrm{g}_{i_b}^{te}+\left(x_k^a-x_k^b\right)\mathrm{g}_{i_c}^{te}}$$

若 $\mathrm{g}^{te}(x^{i'}\mid\lambda^i,z)<\mathrm{g}^{te}(x^i\mid\lambda^i,z)$，则 $x^i=x^{i'}$。

step 10：产生混合种群 $R_t=P_t\cup Q_t$，并将 R_t 作为种群多样性模块的输入增加种群多样性，采用 2.2.3 节算法 3。

step 11：更新外部精英存档 A，若 A 的规模大于 $A_{\max}$，则计算个体的拥挤熵并删除最拥挤个体。否则，转向 step 12。

step 12：检查是否满足停止条件，若是，则算法终止，反之则 gen=gen+1，若 gen 为 50 的整数倍，则计算该 50 代内每个子问题的函数相对变化值 Δ^i：

$$\Delta^i=\frac{\mathrm{g}^{te}(x_{old}^i\mid\lambda^i,z)-\mathrm{g}^{te}(x_{new}^i\mid\lambda^i,z)}{\mathrm{g}^{te}(x_{old}^i\mid\lambda^i,z)}$$

更新子问题效用值并返回 step2：

$$\pi^i=\begin{cases}1 & \text{if } \Delta^i>0.001\\ \left(0.95+0.05\dfrac{\Delta^i}{0.001}\right)\pi^i & \text{otherwise}\end{cases}。$$

step 13：算法终止，输出 $\left\{\boldsymbol{x}^1,\boldsymbol{x}^2,\cdots,\boldsymbol{x}^N\right\}$ 和 $\{\boldsymbol{F}(\boldsymbol{x}^1),\boldsymbol{F}(\boldsymbol{x}^2),\cdots,\boldsymbol{F}(\boldsymbol{x}^N)\}$。

在上述算法中使用 DE 策略为 DE/rand/1/bin 方法[32]：

$$\mathrm{DE/best/1/bin}: v_k=x_k^{r_1}+F\times(x_k^{r_2}-x_k^{r_3})$$

由上述方法生成的新解为 $y=(y_1,y_2,\cdots,y_n)$，

其中分量 y_k 由下式生成：

$$y_k = \begin{cases} v_k & if\ \text{rand1} < CR \text{ 或 } k = \text{rand2} \\ x_k^{r_1} & \text{otherwise} \end{cases}$$

式中：rand1 和 rand2 为两个随机数，$\text{rand1} \in [0,1]$，$\text{rand2} \in \{1,2,\cdots,n\}$，$k = 1,2,\cdots,n$，$CR$ 和 F 为控制参数。

3 数值仿真实验

3.1 性能评价标准

我们选用多目标优化问题 CEC09 测试集 UF1-UF13[33] 和 ZDT4 测试函数来检测所提 HMOEA/D 算法的性能，并将仿真实验结果与一些经典的多目标进化算法 MOEA/D[34]，GDE3[35]，MTS[36]，DMOEADD[37]，NSGAIILS[38]，Clustering MOEA[39]，MOEP[40]和 Gong[41]进行比较。CEC09 测试集包含有复杂 Pareto 解、可微及不可微的多目标测试问题。在所选的测试问题中，ZDT4 有多个局部最优，UF1-UF7 有 2 个优化目标，UF8-UF10 有 3 个优化目标，UF11-UF13 有 5 个优化目标。仿真采用 Matlab2013a，实验平台：win7 企业版，Pentium (R)Dual-Core CPU E5300 @2.60GHZ，32 位操作系统，4.00G RAM。仿真实验采用的评价指标为 Inverted Generational Distance(IGD)指标[42]、Generational Distance(GD)指标[43]、Space(S)指标[42]和 Hypervolume (HV)指标[44]。

3.2 参数设置

在本文实验中所涉及到的算法参数设置如下：

① 种群规模 N：UF1-UF7(2-obj)，设为 600；UF8-UF10(3-obj)设为 1000，UF11-UF13(5-obj)，设为 1 500。

② 向量邻居个数：$T = 0.1N$，$\eta_r = 0.01N$。

③ 选择交叉/更新区域概率：$\delta = 0.9$。

④ DE 操作算子中：DE 交叉概率 $CR = 1.0$；DE 比例系数 $F = 0.5$；变异分布指数 $\eta_m = 20$；变异概率 $p_m = 1/n$。

⑤ 在局部搜索中：$\varepsilon = 10^{-6}$，$\eta = 10$。

⑥ 停止准则：当函数评价次数达到 300 000 时算法终止。

⑦ 算法对每个测试问题均独立运行 30 次。

另外，聚类算法采用基于距离的聚类方法，聚类数 $K = 2 \cdot M$，M 为优化目标数。用于性能评价的 Pareto 解的个数，UF1-UF7(2-obj)设为 100；UF8-UF10(3-obj)设为 150；UF11-UF13(5-obj)设为 400。

3.3 实验分析

为检测所提 HMOEA/D 算法的有效性，将 HMOEA/D 与 MOEA/D 进行对比，两种算法采用相同的参数设置，对于采用的 UF1-UF13 测试问题，两种算法都独立运行 30 次并统计所求得的最优 Pareto 前沿的性能评价指标，包括指标的最好值、最差值、方差和平均值。表 1 为 30 次运行的统计结果，最好的指标值用加粗表示，并且以指标的平均值为依据在表格的最后一列列出了 HMOEA/D 与 MOEA/D 算法相比较的结果，符号“+”，“–”和“=”分别表示 HMOEA/D 算法对相应的测试问题表现优于、劣于和等于 MOEA/D 算法。

从表 1 和表 2 可以看出，对于 CEC09 测试集，HMOEA/D 算法求得的近似 PF 的多样性和收敛性优于 MOEA/D 算法。对于测试问题 UF7 和 UF9，MOEA/D 算法虽优于 HMOEA/D 算法，但两者相差不大。从表 3 可以看出，除 UF12 测试问题之外，两种算法求得的 S 指标均很小，以 30 次运行中 S 指标的平均值为依据，除了 UF9 和 UF13 两个测试问题外，HMOEA/D 算法对其他 11 个测试问题，求得的 PF 的均匀性要优于 MOEA/D 算法。从表 4 可以看出，总体上，HMOEA/D 算法求得的近似 PF 的多样性和收敛性优于 MOEA/D 算法。

从表 1~4 还可以看出，除了 UF12 问题外，对于其他测试问题，HMOEA/D 算法求得的性能指标值的统计方差都很小，并且都小于 MOEA/D 算法求得的方差值，这表明 HMOEA/D 算法在解决此类问题时表现稳定，具有很好的鲁棒性。

DOI: 10.16182/j.issn1004731x.joss.17-9183

HMOEA/D 算法通过局部搜索模块采用 SQA 局部搜索策略加速算法收敛、利用多样性增强模块增加种群多样性，理论上会增加算法求解测试问题所需的运行时间。为此，我们对两种算法求解效率进行比较。两种算法采用相同的进化代数并统计两种算法求解测试问题所需的运行时间。从表 5 可以看出，HMOEA/D 算法的运行时间均值要大于 MOEA/D 算法，这是因为当种群经投影聚类后，若种群具有良好的种群多样性，则进行一次局部搜索，否则进行种群多样性增强。这虽然会增加算法的运行时间，但同时基于 SQA 的局部搜索加速了算法收敛，另外 SQA 计算量小，方便使用，所以两种算法计算效率相差不大。从表 1~4 可以看出，在相同的函数评价次数下，HMOEA/D 算法的性能要优于 MOEA/D 算法。

表 1 HMOEA/D 和 MOEA/D 算法 30 次独立运行求得的 IGD 指标统计数据比较
Tab. 1 Comparisons of IGD obtained by 30 independent runs between proposed HMOEA/D and MOEA/D

Problems	HMOEA/D				MOEA/D				SS
	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
UF1	0.004 15	0.004 62	**0.004 29**	0.000 09	0.003 92	0.005 16	0.004 36	0.000 29	+
UF2	0.004 41	0.010 13	**0.005 59**	0.001 32	0.004 78	0.010 84	0.006 81	0.001 83	+
UF3	0.004 09	0.007 81	**0.004 59**	0.000 95	0.003 92	0.024 36	0.007 43	0.005 18	+
UF4	0.040 47	0.047 15	**0.043 77**	0.001 72	0.056 87	0.081 38	0.063 86	0.005 30	+
UF5	0.043 10	0.553 18	**0.144 95**	0.121 40	0.082 26	0.306 26	0.180 72	0.068 08	+
UF6	0.002 11	0.004 83	**0.003 12**	0.000 76	0.003 46	0.010 01	0.005 88	0.001 75	+
UF7	0.004 08	0.315 72	0.014 5	0.056 80	0.003 27	0.010 62	**0.004 46**	0.001 16	-
UF8	0.048 45	0.086 46	**0.057 59**	0.008 55	0.050 70	0.065 61	0.058 42	0.003 29	+
UF9	0.030 36	0.145 82	**0.085 26**	0.057 01	0.035 05	0.149 78	0.088 96	0.058 12	+
UF10	0.107 37	0.543 11	**0.288 97**	0.109 02	0.364 01	0.649 52	0.474 16	0.007 352	+
UF11	0.100 93	0.187 87	**0.106 38**	0.015 47	0.106 88	0.115 16	0.110 22	0.002 28	+
UF12	73.176 40	206.056 1	**119.715 4**	33.596 61	66.168 2	214.226 0	145.788 4	41.827 5	+
UF13	1.184 15	1.855 02	**1.837 25**	0.003 80	1.834 62	1.899 21	1.856 64	0.019 76	+

表 2 HMOEA/D 和 MOEA/D 算法 30 次独立运行求得的 GD 指标统计数据比较
Tab. 2 Comparisons of GD obtained by 30 independent runs between proposed HMOEA/D and MOEA/D

Problems	HMOEA/D				MOEA/D				SS
	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
UF1	2.62E-04	4.58E-04	**3.12E-04**	3.58E-05	2.70E-04	4.93E-04	3.44E-04	5.64E-05	+
UF2	3.22E-04	9.66E-04	**3.18E-04**	1.68E-04	2.41E-04	1.14E-03	5.85E-04	2.32E-04	+
UF3	2.68E-04	1.11E-03	**2.68E-04**	2.59E-04	1.61E-04	7.32E-03	9.56E-04	0.001 422	+
UF4	0.004 758	0.005 268	**0.004 758**	2.44E-04	0.005 817	0.007 936	0.006 866	5.62E-04	+
UF5	0.011 788	0.063 168	**0.011 801**	0.014 602	0.009 166	0.094 116	0.022 778	0.018 920	+
UF6	5.08E-04	1.24E-03	**5.10E-04**	2.38E-04	0.001 306	0.006 268	0.003 321	0.001158	+
UF7	3.18E-04	6.33E-03	4.19E-04	0.001 125	1.91E-04	7.22E-04	**3.11E-04**	9.88E-05	-
UF8	0.002 768	0.005 28	**0.002 798**	9.62E-04	0.002 212	0.012 576	0.004 322	0.002 08	+
UF9	0.018 053	0.042 918	0.018 060	0.012 464	0.004 118	0.029 124	**0.011 236**	0.007 032	-
UF10	0.058 168	0.195 678	**0.058 172**	0.506 48	0.054 322	0.194 872	0.119 662	0.032 246	+
UF11	0.003 876	0.004 086	**0.003 878**	1.06E-04	0.004 126	0.004 554	0.004 202	9.38E-05	+
UF12	10.791 62	15.104 68	**10.791 52**	2.528512	8.032 658	17.861 58	11.541 26	2.072 642	+
UF13	0.058 762	0.069 264	**0.064 768**	1.58E-04	0.063 682	0.074 622	0.066 232	1.92E-04	+

表 3 HMOEA/D 和 MOEA/D 算法 30 次独立运行求得的 S 指标统计数据比较
Tab. 3 Comparisons of S obtained by 30 independent runs between proposed HMOEA/D and MOEA/D

Problems	HMOEA/D				MOEA/D				SS
	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
UF1	0.001 314	0.004 762	**0.001 78**	0.000 38	8.91E-04	0.005 262	0.001 85	0.000 40	+
UF2	0.001 41	0.002 134	**0.001 62**	0.000 18	0.001190	0.002 092	0.001 62	0.000 24	+
UF3	9.22E-04	0.002 116	**0.001 41**	0.000 32	9.32E-04	0.011 056	0.001 78	0.001 93	+
UF4	0.001 346	0.002 842	**0.001 58**	0.000 26	0.001468	0.003 758	0.001 76	0.000 44	+
UF5	1.96E-05	0.006 468	**0.001 66**	0.001 88	0.001198	0.014 26	0.004 66	0.003 58	+
UF6	0.001 468	0.003 602	**0.001 82**	0.000 55	0.001446	0.006 428	0.002 34	0.001 36	+
UF7	6.30E-04	0.003 022	**0.001 02**	0.000 46	8.22E-04	0.001 459	0.001 16	0.000 17	+
UF8	0.022 056	0.031 438	**0.024 88**	0.002 02	0.021328	0.046 252	0.025 18	0.004 62	+
UF9	0.014 628	0.060 562	0.026 46	0.010 22	0.015016	0.050 42	**0.021 24**	0.007 24	-
UF10	0.006 024	0.121 024	**0.050 24**	0.028 6	0.05036	0.250 125	0.086 52	0.036 56	+
UF11	0.051 22	0.091 862	**0.087 22**	0.006 98	0.09572	0.100 136	0.098 56	0.001 26	+
UF12	4.748 624	53.886 54	21.881 24	12.441 86	5.582912	56.571 26	22.195 22	13.067 36	+
UF13	0.134 28	0.148 264	0.141 18	0.003 02	0.126058	0.142 88	**0.132 16**	0.004 28	-

表 4 HMOEA/D 和 MOEA/D 算法 30 次独立运行求得的 HV 指标统计数据比较
Tab. 4 Comparisons of HV obtained by 30 independent runs between proposed HMOEA/D and MOEA/D

Problems	HMOEA/D				MOEA/D				SS
	Best	Worst	Mean	Std	Best	Worst	Mean	Std	
UF1	0.872 436	0.866 808	**0.869 891**	0.000 262	0.865 642	0.776 856	0.864 712	0.016 318	+
UF2	0.869 832	0.861 587	**0.868 822**	0.001 876	0.866 523	0.858 662	0.856 624	0.002 342	+
UF3	0.881 682	0.854 246	**0.878 392**	0.003 886	0.868 762	0.762 5	0.855 168	0.027 162	+
UF4	0.481 288	0.457 856	**0.458 892**	0.003 252	0.448 256	0.409 982	0.434 262	0.008 462	+
UF5	0.856 478	0.395 402	**0.714 982**	0.126 402	0.771 762	0.175 662	0.567 168	0.149 538	+
UF6	0.645 236	0.642 738	**0.642 682**	0.000 308	0.644 862	0.612 123	0.636 518	0.008 102	+
UF7	0.703 998	0.388 564	0.692 278	0.055 467	0.711 264	0.688 018	**0.701 656**	0.000 832	-
UF8	1.942 841	1.832 264	**1.926 476**	0.020 268	1.914 875	1.828 516	1.898 368	0.014 947	+
UF9	2.286 812	2.081 921	2.182 924	0.098 232	2.275 252	2.059 468	**2.221 562**	0.080 264	-
UF10	20.862 58	13.101 02	**16.758 62**	2.823 026	18.283 26	11.828 54	14.412 54	1.384 258	+
UF11	73.563 22	73.122 16	**73.636 24**	0.090 288	73.644 8	73.683 21	73.616 92	0.008 269	+
UF12	2.22E+12	1.22E+11	1.55E+12	7.02E+11	2.12E+12	9.85E+11	**1.71E+12**	3.22E+11	-
UF13	1 412.216	1 402.118	1.42E+03	2.220 908	1 423.825	1 412.826	**1.43E+03**	2.916 824	-

为比较 HMOEA/D 算法的性能，在相同的停止准则和评价条件下还与当前其他一些优秀的多目标进化算法进行了比较，这些算法是几年来求解性能比较突出的新型算法。以 IGD 指标为对比指标，性能最好的 IGD 指标以加粗表示，对比结果如表 6 所示。

HMOEA/D 算法与其他 8 种多目标进化算法相比，HMOEA/D 算法在大多数测试问题上均优于其他算法，即使在 UF4，UF5，UF7 和 UF9 测试函数上算法表现分别不如 MTS，MOEA/D 和 NSGAIILS 算法，但性能相差不大。根据“世界上没有免费的午餐”定理，我们不期望所提算法在所有测试问题上都到达最优。

DOI: 10.16182/j.issn1004731x.joss.17-9183

表 5 两种算法 30 次独立运行所需时间(秒)的统计数据比较

Tab. 5 Comparisons of times used by30 independent runs between proposed algorithm and other MOEAs

Problems	Iteration	HMOEA/D				MOEA/D			
		Max	Min	Mean	Std	Max	Min	Mean	Std
UF1	2 500	100.458	96.864	98.453 8	0.864 584	90.118	84.268	86.516 2	1.103 708
UF2	2 500	112.358	98.036 2	99.845 9	1.107 826	90.526	87.252	89.057 6	0.583 733
UF3	2 500	110.102 4	91.415 2	95.021 8	1.845 832	86.312	81.534	84.603 1	1.194 172
UF4	2 500	127.259	98.415 8	104.352 6	1.524 246	89.765	86.586	88.725 4	0.834 201
UF5	2 500	83.243	75.167	78.781 4	2.416 524	72.356	64.532	69.826 2	1.836 312
UF6	2 500	106.146	94.448	98.652 4	1.754 576	83.406	78.155	80.576	1.294 165
UF7	2 500	121.558	92.671	111.379 6	1.935 264	88.004	86.158	87.089	0.474 431
UF8	1 500	363.578	354.354 2	357.880 2	2.487 516	308.536	304.121	304.515 3	1.142 867
UF9	1 500	411.328	350.128	360.422 6	12.682 14	381.709	309.442	326.872 4	22.090 12
UF10	1 500	376.455 6	313.45	330.321 8	12.825 32	339.622	268.558	289.326 8	23.240 61
UF11	1 000	1 248.258	1 031.135	1 072.264	51.652 21	1 442.01	1 000.26	1 076.546	120.213 7
UF12	1 000	919.3542	812.516	853.657 2	22.089 26	827.018	732.112	772.546 3	26.112 3
UF13	1 000	1 374.646	1 017.138	1 091.064	90.452 86	1 268.25	1 029.35	1 052.402	42.653 82

表 6 HMOEA/D 算法与其他算法对比研究

Tab. 6 Comparisons between proposed algorithm and other algorithms

Problems	HMOEA/D	MOEA/D	GDE3	MTS	DMOEADD	NSGAIILS	ClusteringMOEA	MOEP	Gong
UF1	**0.004 29**	0.004 36	0.005 34	0.006 46	0.010 38	0.011 53	0.029 9	0.059 6	0.005 26
UF2	**0.005 59**	0.006 81	0.011 95	0.006 15	0.006 79	0.012 37	0.022 8	0.018 9	0.005 93
UF3	**0.004 59**	0.007 43	0.106 39	0.053 1	0.033 37	0.106 03	0.054 9	0.099	0.051 1
UF4	0.043 77	0.063 86	0.026 5	**0.0235 6**	0.042 68	0.058 4	0.058 5	0.042 7	0.038 11
UF5	0.014 95	0.180 72	0.0392 8	**0.014 89**	0.314 54	0.565 7	0.247 3	0.224 5	0.022 89
UF6	**0.003 12**	0.005 88	0.250 91	0.059 17	0.066 73	0.310 32	0.087 1	0.103 1	0.069 21
UF7	0.014 5	**0.004 46**	0.025 22	0.040 79	0.010 32	0.021 32	0.022 3	0.019 7	0.046 72
UF8	**0.057 59**	0.058 42	0.248 55	0.112 51	0.068 41	0.086 3	0.238 3	0.423	0.113 58
UF9	0.085 26	0.088 96	0.082 48	0.114 42	0.048 96	**0.071 9**	0.293 4	0.342	0.094 31
UF10	**0.288 97**	0.474 16	0.433 26	0.153 06	0.322 11	0.844 68	0.411 1	0.362 1	0.253 7
UF11	**0.106 38**	0.110 22	0.234 25	0.455 05	1.203 28	0.175 2	1.240 1	0.433 7	0.475 8
UF12	**119.715 4**	145.788 4	202.12	305.2	477.65	158.05	1 039.36	885.89	135.32
UF13	**1.837 25**	1.856 64	3.205 7	1.907 9	1.997 1	3.232 3	3.404 3	2.014 5	1.882 6

3.4 模块作用验证和参数敏感性分析

3.4.1 模块作用验证

为验证投影聚类模块和种群多样性增强模块的作用，我们提出一种对比算法，该算法只采用上文提出的 SQA 局部搜索策略，称之为 MOEA/D-SQA，3 种算法具有相同的参数设置，以 IGD 为指标，测试结果如表 7 所示。

针对上述测试问题，从表 7 中可以看出，HMOEA/D 算法表现优于 MOEA/D-SQA 和 MOE/D 算法，这说明从不同的聚类中选择个体来进行局部搜索和采用种群多样性增强策略可使得算法求得的 PF 前沿更逼近理想 PF。

在所提混合多目标优化算法框架中，投影聚类模块采用聚类质量指标 Q 来判别种群多样性的好坏。如图 3 所示，以测试函数 ZDT4 为例演示聚类质量指标 Q 的进化过程，ZDT4 拥有多个局部最优前沿。第 t 代的聚类质量指标 $Q_{\text{-current}}$ 用圆圈表示，

质量指标下限阈值用 $Q_{\text{-bound}}$ 加粗实线表示，蓝色实心圆点表示种群多样性模块被激活。

表 7 模块作用验证
Tab. 7 Module action verification

Test Problem	MOEA/D	MOEA/D-SQA	HMOEA/D
UF1	0.004 36	0.004 32	**0.004 29**
UF2	0.006 81	0.005 65	**0.005 59**
UF3	0.007 43	0.004 68	**0.004 59**
UF4	0.063 86	0.043 84	**0.043 77**
UF5	0.180 72	0.145 06	**0.144 95**
UF6	0.005 88	0.003 17	**0.003 12**
UF7	**0.004 46**	0.015 0	0.014 5
UF8	0.058 42	0.057 68	**0.057 59**
UF9	0.088 96	0.095 76	**0.085 26**
UF10	0.474 16	0.289 08	**0.288 97**
UF11	0.110 22	0.106 42	**0.106 38**
UF12	145.788 4	119.816 65	**119.715 4**
UF13	1.856 64	1.847 65	**1.837 25**

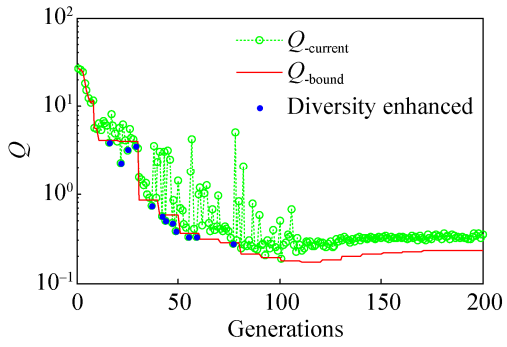

图 3 ZDT4 聚类质量指标 Q 进化曲线
Fig. 3 Q evolution over generations for ZDT4

从图 3 可以看出，初始阶段 $Q_{\text{-current}}$ 有较高的取值，表示初始种群具有较好的多样性，随着种群逐渐逼近 PF，$Q_{\text{-current}}$ 逐渐减小并稳定，其进化曲线类似“之”字型，中间有很多峰值突起，这归因于种群增强模块的多样性增强作用，ZDT4 为多峰函数，具有多个局部最优，局部搜索和局部最优前沿的“欺诈”均可导致种群多样性丧失和最优解的丢失，在混合算法中，多样性增强模块通过交叉和变异操作算子增加种群多样性。研究表明混合算法能有效防止种群多样性的丢失。

3.4.2 参数敏感性分析

在 HMOEA/D 算法中，参数设置如局部搜索概率 P_{local}、权向量的邻居个数 T 等对算法性能具有一定的影响，为此，实验分析了 HMOEA/D 算法对参数 P_{local}、种群规模 N、权向量的邻居个数 T 以及杂交概率 CR 和尺度因子 F 的敏感性。

(1) 参数 P_{local} 研究

P_{local} 取值不同会影响算法的种群多样性和收敛性，较大 P_{local} 取值意味着更多个体参与局部搜索，这样会增大生成超级个体的概率，而较小的 P_{local} 取值会导致局部搜索加速收敛的能力不足。因此我们采用启发式方法研究 P_{local} 应如何取值。采用上述的参数设置，根据下式设置 P_{local} 不同的取值。

$$P_{\text{local}} = 1/(\eta \cdot N_K), \quad \eta = 5,10,15,20,25 \tag{11}$$

式中：N_K 为每个聚类的种群规模，选取优化目标数分别为 2，3 和 5 的 UF1，UF8 和 UF11 为测试函数，通过改变 P_{local} 的取值测试 HMOEA/D 算法对不同测试问题对 P_{local} 的敏感性，对所选测试函数，运行结果如表 8 所示。

表 8 P_{local} 取值研究
Tab. 8 P_{local}value study

Test Problem	$P_{\text{local}} = 1/(\eta \cdot N_K)$				
	Medium IGD Value (取值越小越好)				
	η=5	η=10	η=15	η=20	η=25
UF1	0.004 35	**0.004 29**	*0.004 31*	0.004 43	0.004 51
UF2	0.005 63	**0.005 59**	*0.005 65*	0.005 71	0.005 79
UF3	0.004 66	*0.004 59*	**0.004 57**	0.004 76	0.004 80
UF4	0.043 89	**0.043 77**	*0.043 79*	0.043 91	0.044 06
UF5	**0.144 93**	*0.144 95*	0.145 01	0.145 18	0.145 36
UF6	0.003 16	*0.003 12*	**0.003 09**	0.003 26	0.003 39
UF7	*0.014 7*	**0.014 5**	0.015 6	0.016 9	0.017 2
UF8	*0.057 63*	**0.057 59**	0.057 66	0.057 73	0.0579 1
UF9	0.085 31	**0.085 26**	*0.085 29*	0.085 47	0.085 55
UF10	0.289 06	**0.288 97**	*0.289 03*	0.289 26	0.289 31
UF11	0.106 41	*0.106 38*	**0.106 33**	0.106 51	0.106 58
UF12	*119.728 9*	**119.715 4**	119.821 7	119.928 5	120.117 4
UF13	1.837 91	**1.837 25**	*1.838 67*	1.839 55	1.841 32

通过测试获取了 P_{local} 的不同取值下每一个测试问题 IGD 的中间值。对于每一个测试问题，最

好的 IGD 值和第 2 好的 IGD 取值分别以加粗和斜体标示。从表 8 我们可以看出，HMOEA/D 算法对于 P_{local} 取值不是很敏感。较小的 P_{local} 取值有利于 HMOEA/D 算法向最优 Pareto 前沿收敛，较大的 P_{local} 取值易产生上文描述的超级个体而使种群丧失多样性，较小的 P_{local} 取值又使得算法加速收敛能力不足，为了平衡 P_{local} 对种群多样性及算法收敛性的影响，我们选择 $P_{\text{local}}=1/(10\cdot N_K)$。

(2) 种群规模 N 对算法性能影响

种群规模 N 是多目标进化算法中一个比较重要的参数。在运用多目标进化算法求解过程中，决策者往往希望通过较小的计算代价求得一组分布均匀的 Pareto 最优解集。为检测不同的种群规模 N 对 HMOEA/D 算法性能的影响，在实验中选取 UF1(2-obj)，UF10(3-obj)的和 UF13(5-obj)为测试函数，测试在 HMOEA/D 解决这些问题时对 N 的敏感性。3 个测试函数的最大进化代数分别设为 2 500，1 500 和 1 000，除 N 外其他参数的设置与 3.2 节中相同。以 30 次运行中 IGD 指标的平均值为依据，图 4 显示了算法求得的 IGD 指标值对于不同的种群规模 N 的变化情况。从图 4 可以看出，HMOEA/D 算法对种群规模并不十分敏感，这也说明 HMOEA/D 具有较好的鲁棒性，表现较为稳定。

(3) T 对 HMOEA/D 性能影响

权向量的邻居个数 T 是 HMOEA/D 算法中的重要参数。在实验中选取测试函数 UF1(2-obj)，UF10(3-obj)的和 UF11(5-obj)，测试 HMOEA/D 解决这些问题时对 T 的敏感性，除 T 外所有其他参数的设置与 3.2 节中相同。以 30 次运行中 IGD 指标的平均值为依据，由图 5 可知，对于 3 个测试函数，随着 T 的变化，IGD 曲线变化比较平缓，这表明求解这些测试问题时对参数 T 的设置并不敏感，这种情况在 UF1 测试函数上表现得尤为明显。对于 3 个测试函数，当权向量的邻居个数 T 取值在 0.3 T 和 0.6 T 之间时，3 个测试函数表现都很好。当 T 取值较小时，算法表现较差，其原因是较小的取值使得 HMOEA/D 算法的搜索能力变差。而 T 取值较大时，相邻个体的解并不一定近似，算法的更新操作效率降低，也会使得 HMOEA/D 算法的表现变差。

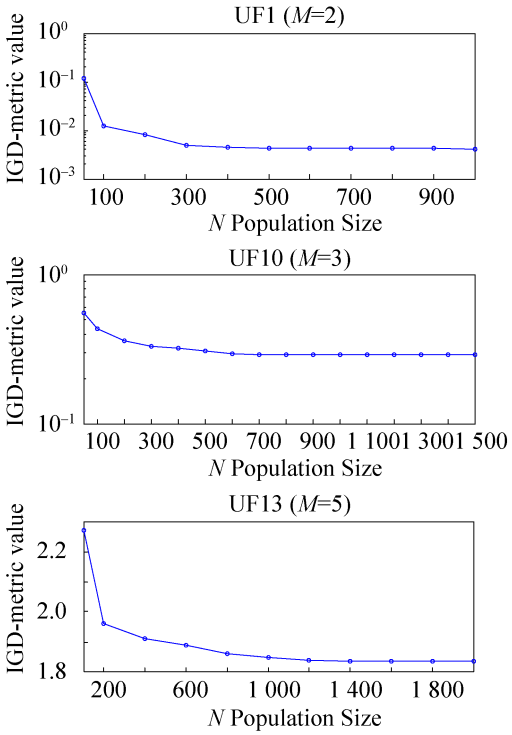

图 4 种群规模 N 对 HMOEA/D 算法性能影响

Fig. 4 Effect of population size N on HMOEA/D algorithm performance

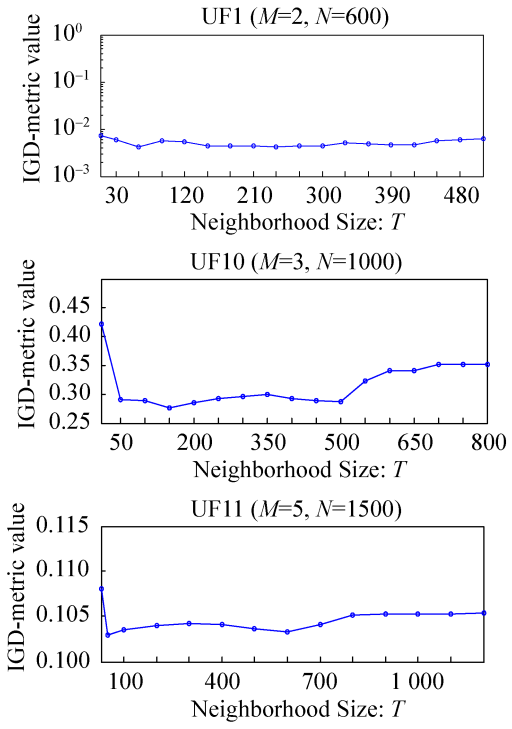

图 5 参数 T 对 HMOEA/D 算法性能影响

Fig. 5 Effect of parameter T on HMOEA/D algorithm performance

(4) *CR* 和 *F* 对 HMOEA/D 性能影响

交叉因子 *CR* 和缩放因子 *F* 是差分进化算子中的两个重要参数。交叉因子 $CR \in [0,1]$ 是一个实值杂交概率常数，它控制一个试验个体的分量是取自 DE 交叉个体的分量，还是取自父代个体的分量。缩放因子 *F* 是一个正实数，它通过控制差分向量的收缩调节种群的多样性，*F* 值较大时，可使种群有较好的多样性，*F* 较小时，扰动较小，缩放因子能起到局部精细化搜索的作用可加快算法收敛。为检测 *CR* 和 *F* 的不同设置对 HMOEA/D 算法性能的影响，我们选取 UF8 为测试函数，测试 HMOEA/D 算法对两个参数取值在[0,1]上的敏感性，除 *CR* 和 *F* 外其他所有参数的设置与 3.2 节相同。以 30 次运行中 IGD 指标的平均值为依据，对于不同的 *CR* 和 *F* 值，求得的 *PF* 的 IGD 指标值的变化情况如图 6 所示。从图 6 可以看出，当 $CR \in [0.5,1]$，$F \in [0.1,0.5]$ 时，IGD 取值较小，即较大的 *CR* 值和较小的 *F* 值的组合会使 HMOEA/D 算法的表现相对较好。

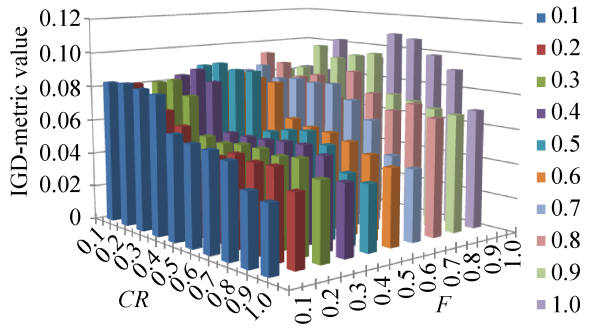

图 6 参数 *CR* 和 *F* 对 HMOEA/D 算法性能影响
Fig. 6 Effect of parameter *CR* and *F* on HMOEA/D algorithm performance

4 结论

本文研究了多种混合算法并针对混合多目标进化算法存在的一些缺点，基于反馈控制思想和系统化设计理念，提出了一种以 MOEA/D 算法作为启发式算法的全局搜索与局部搜索相结合的混合多目标进化算法，算法中，提出了一种基于拥挤熵的种群多样性评价方法来准确估计进化种群的多样性状态，设计了一种针对 MOEA/D 算法的多样性增强策略，采用基于 SQA 的局部搜索策略，使混合算法在保证种群多样性的前提下加快收敛，在全局和局部搜索中达到平衡，使之更有效地解决多目标优化问题。为检测算法的有效性，将该算法用于解决多个无约束的标准测试问题，并将实验结果和当前其他一些优秀的多目标进化算法所求得的结果进行了比较，验证了所提 HMOEA/D 算法的有效性。最后分析了参数设置对算法性能的影响，HMOEA/D 算法对局部搜索概率 P_{local}、种群规模 *N* 和权向量邻居个数 *T* 并不十分敏感，HMOEA/D 算法具有较好的鲁棒性；较大的 *CR* 值和较小的 *F* 值的组合会使 HMOEA/D 算法的表现相对较好。

在未来研究中，所提算法框架可扩展到其他多目标进化算法，验证和设计多种算法融合以及不同算法之间的切换机制以及框架细化等方面，提出更有效的局部搜索策略和多样性增强策略、种群多样性检测策略、研究更高效的自适应聚类方法以及局部搜索概率的自适应取值方法，可实现算法快速设计、减少人为干扰对算法性能的影响。

参考文献：

[1] Coello C A, Lamont G B. Applications of Multi-Objective Evolutionary Algorithms[M]. Singapore: World Scientific, 2004: 3-11.

[2] Medaglia A L, Villegas J G, Rodriguez-Coca D M. Hybrid Bi-objective Evolutionary Algorithms for The Design of A Hospital Waste Management Network[J]. Heuristics (S1381-1231), 2009, 15(2): 153-176.

[3] Fei Z S, Li B, Yang S S, et al. A Survey of Multi-Objective Optimization in Wireless Sensor Networks: Metrics, Algorithms, and Open Problems [J]. IEEE Communications Surveys & Tutorials (S1553-877X), 2017, 19(1): 550-586.

[4] Qi Y T, Hou Z T, Li H H, et al. A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows[J]. Computers & Operations Research (S0305-0548), 2015, 62: 61-67.

[5] Lin Q Z, Zhu Q L, Huang P Z, et al. A Novel Hybrid Multi-objective Immune Algorithm with Adaptive Differential Evolution[J]. Computers & Operations Research (S0305-0548), 2015, 62: 95-111.

[6] Liu C A, Jia H M. Dynamic Multi-objective Evolutionary Algorithm with Two Stages Evolution Operation[J]. Intelligent Automation and Soft Computing (S1079-8587), 2015, 4(21): 575-588.

[7] Aimin Z, Jin Y C, Zhang Q F. A Population Prediction Strategy for Evolutionary Dynamic Multi-objective Optimization[J]. IEEE Transactions on Cybernetics (S1083-4419), 2014, 44(1): 40-53.

[8] Deb K, Lele S, Datta R. A hybrid Evolutionary Multi-objective and SQP Based Procedure for Constrained Optimization[C]. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2007: 36-45.

[9] Rafiul H M, Nath B, Kirley M, et al. A hybrid of Multi-objective Evolutionary Algorithm and HMM-Fuzzy Model for Ttime Series Prediction[J]. Neuro Computing (S0925-2312), 2012, 81: 1-11.

[10] Wang H F, Fu Y P, Huang M, et al. A hybrid Evolutionary Algorithm with Adaptive Multi-population Strategy for Multi-objective Optimization Problems[J]. Soft Computing (S2588-2872), 2017, 21: 5975-5987.

[11] Liu R C, Li J X, F J, et al. A Co-evolutionary technique Based on Multi-swarm Particle Swarm Optimization for Dynamic Multi-objective Optimization [J]. European Journal of Operational Research (S0377-2217), 2017: 1028-1051.

[12] Wali K M, Abdellah S, Ozgur Y, et al. Hybrid Adaptive Evolutionary Algorithm Based on Decomposition[J]. Applied Soft Computing (S1568-4946), 2017, 57: 363-378.

[13] Yuan Y, Xu H, Wang B, et al. Balancing Convergence and Diversity in Decomposition-based Many-objective Optimizers[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2016, 20(2): 180-198.

[14] Wang R, Zhang Q, Zhang T. Decomposition-based Algorithms Using Pareto Adaptive Scalarizing Methods[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2016, 20(6): 821-837.

[15] Li K, Zhang Q F, Kwong S, et al. Stable Matching-based Selection in Evolutionary Multi-objective Optimization[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2014, 18(6): 909-923.

[16] Wang W L, Ying S L, Li L, et al. An improved Decomposition-based Multi-objective Evolutionary Algorithm with A Better Balance of Convergence and Diversity[J]. Applied Soft Computing (S1568-4946), 2017, 57: 627-641.

[17] Ishibuchi H, Murata T. A multi-objective Genetic Local Search Algorithm and Its Application to Flow Shop Scheduling[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 1998, 28(3): 392-403.

[18] Goel T, Deb K. Hybrid Methods for Multi-objective Evolutionary Algorithms[C]. Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning. Singapore: (SEAL 2002), 2002: 188-192.

[19] Sindhya K, Sinha A, Deb K, et al. Local Search Based Evolutionary Multi-objective Optimization Algorithm for Constrained and Unconstrained Problems[C]. Proceedings of Congress on Evolutionary Computation. Norway: IEEE, 2009: 2919-2926.

[20] Sindhya K, Deb K, Miettinen K. A Local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence[C]. Parallel Problem Solving from Nature (PPSN X). Berlin: Springer Berlin Heidelberg, 2008: 815-824.

[21] 贾树晋, 杜斌, 岳恒. 基于局部搜索与混合多样性策略的多目标粒子群算法[J]. 控制与决策, 2012, 27(6): 813-826.
Jia Shujin, Du Bin, Yue Heng. Local Search and Hybrid Diversity Strategy Based Multi-objective Particle Swarm Optimization Algorithm[J]. Control and Decision, 2012, 27(6): 813-826.

[22] Zhang Q F, Li H. MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2007, 11(6): 712-731.

[23] Li H, Zhang Q F. Multi-objective Optimization Problems with Complicated Pareto Sets MOEA/D and NSGA-II[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2009, 12(2): 284-302.

[24] Zhang Q F, Liu W, Li H. The Performance of A New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances[C]. IEEE Congress on Evolutionary Computation (CEC'09). Piscataway, 2009: 203-208.

[25] Ishibuchi H, Sakane Y, Tsukamoto N, et al. Adaptation of sScalarizing Functions in MOEA/D: An Adaptive Scalarizing Function Based Multi-objective Evolutionary Algorithm[C]. Proceedings of the 5th international conference devoted to evolutionary multi-criterion optimization (EMO'09). France: Nantes, 2009: 438-452

[26] 吴亮红, 王耀南. 动态差分进化算法及其应用[M].

北京: 科学出版社, 2014: 66-74.
Wu Lianghong, Wang Yaonan. Dynamic Differential Evolutionary Algorithm and Application[M]. Beijing: Science Press, 2014: 66-74.

[27] Sindhya K, Miettinen K, Deb K. An Improved Concurrent-hybrid Algorithm for Enhanced Diversity and Accuracy in Evolutionary Multi-objective Optimization in Evolutionary and Deterministic Methods for Design, Optimization and Control, Applications to Industrial and Societal Problems[M]. Burczynski T, Periaux J, eds. Barcelona, Spain: CIMNE, 2011: 182-187.

[28] 谭艳艳. 几种改进的分解类多目标进化算法及其应用[D]. 西安: 西安电子科技大学, 2013: 64-65.
Tan Yanyan. Several Modified Decomposition-Based Multi-objective Evolutionary Algorithms and Their Applications[D]. Xi'an: Xidian University, 2013: 64-65.

[29] Ali M M, Torn A, Vitanen S. A numerical Comparison of Some Modified Controlled Random Search Algorithms[J]. Journal of Global Optimization (S0925-5001), 1997, 11(4): 377-385.

[30] Fabre M G, Pulido G T, Coello C A C. Two Novel Approaches for Many-objective Optimization[C]. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2010). Barcelona: Springer-Verlag, 2010: 1-8.

[31] Miettinen K. Nonlinear Multi-objective Optimization. Norwell[M]. MA: Kluwer, 1999: 171-179.

[32] Sindhya K, Deb K, Miettinen K. A local Search Based Evolutionary Multi-objective Optimization Approach for Fast and Accurate Convergence[C]. Parallel Problem Solving from Nature (PPSN X). Berlin: Springer Berlin Heidelberg, 2008: 815-824.

[33] Zhang Q F, Zhou A, Zhao S, et al. Multi-objective optimization test instances for the CEC 2009 special session and competition[R]. Technical Report CES-487, School of Computer Science and Electronic Engineering, University of Essex, UK, 2008.

[34] Zhang Q F, Li H. MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2007, 11(6): 712-731.

[35] Kukkonen S, Lampinen J. Performance Assessment of Aeneralized Differential Evolution with A Given Set of Constrained Multi-objective Test Problems[C]. IEEE Congress on Evolutionary Computation (CEC'09). NJ: IEEE Press, 2009: 3593-3600.

[36] Tseng L Y, Chen C. Multiple Trajectory Search for Unconstrained/Constrained Multi-objective Optimization [C]. IEEE Congress on Evolutionary Computation (CEC'09). NJ: IEEE Press, 2009: 1951-1958.

[37] Liu M, Zou X, Chen Y, et al. Performance Assessment of DMOEA-DD with CEC 2009 MOEA Competition Test Instances[C]. IEEE Congress on Evolutionary Computation (CEC'09). NJ: IEEE Press, 2009: 2913-2918.

[38] Sindhya K, Sinha A, Deb K, et al. Local Search Based Evolutionary Multi-objective Optimization Algorithm for Constrained and Unconstrained Problems[C]. IEEE Congress on Evolutionary Computation (CEC'09). NJ: IEEE Press, 2009: 2919-2926.

[39] Wang Y, Dang C, Li H, et al. A Clustering Multi-objective Evolutionary Algorithm Based on Orthogonal and Uniform Design[C]. IEEE Congress on Evolutionary Computation (CEC'09). NJ: IEEE Press, 2009: 2927-2933.

[40] Qu B Y, Suganthan P N. Multi-objective Evolutionary Programming Without Non-domination Sorting Is Up to Twenty Times Faster[C]. IEEE Congress on Evolutionary Computation (CEC'09). NJ: IEEE Press, 2009: 2934-2939.

[41] Gong D, Liu Y, Sun X, et al. Parallel Many-objective Evolutionary Optimization Uusing Oobjectives Decomposition[J]. Automatica (S0005-1098), 2015, 41(8): 1438-1451.

[42] Zitzler E, Thiele L, Laumanns M, et al. Performance Assessment of Multi-objective Ooptimizers: An Analysis and Review[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 2003, 7(2): 117-132.

[43] Van Veldhuizen D A, Lamont G B. Multi-objective Evolutionary Algorithm Test Suites[C]. Proceedings of the 1999 ACM symposium on applied computing. New York: ACM Press, 1999: 351-357.

[44] Zitzler E, Thiele L. Multi-Objective Evolutionary Algorithms: A comparative Case Study and The Strength Pareto Approach[J]. IEEE Transaction on Evolutionary Computation (S1089-778X), 1999, 3(4): 257-271.

https://dc-china-simulation.researchcommons.org/journal/vol32/iss2/7
DOI: 10.16182/j.issn1004731x.joss.17-9183