计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150806
53(6):1410-1421,2016

基于泛化反向学习的多目标约束差分进化算法

魏文红[1] 王甲海[2] 陶 铭[1] 袁华强[1]
[1](东莞理工学院计算机学院 广东东莞 523808)
[2](中山大学计算机科学系 广州 510006)
(weiwh@dgut.edu.cn)

Multi-Objective Constrained Differential Evolution Using Generalized Opposition-Based Learning

Wei Wenhong[1], Wang Jiahai[2], Tao Ming[1], and Yuan Huaqiang[1]
[1](*Computer Institute, Dongguan University of Technology, Dongguan, Guangdong* 523808)
[2](*Department of Computer Science, Sun Yat-sen University, Guangzhou* 510006)

Abstract Differential evolution is a simple and efficient evolution algorithm to deal with nonlinear and complex optimization problems. Generalized opposition-based learning (GOBL) often guides population to evolve in the evolutionary computing. However, real-world problems are inherently constrained optimization problems often with multiple conflicting objectives. Hence, for resolving multi-objective constrained optimization problems, this paper proposes a constrained differential evolution algorithm using generalized opposition-based learning. In this algorithm, firstly, the transformed population is generated using general opposition-based learning in the population initialization. Secondly, the better individuals that are selected from the initial population and the transformed population using non-dominated sorting, crowding distance sorting and constraint handling techniques compose the new initial population. Lastly, based on a jumping probability, the transformed population is calculated again after generating new populations, and the fittest individuals that are selected from the union of the current population and the transformed population compose new population using the same techniques. The solution can be evolved toward Pareto front slowly according to the generalized opposition-based learning, so that the best solutions set can be found. The proposed algorithm is tested in multi-objective benchmark problems and compared with NSGA-Ⅱ, MOEA/D and other multi-objective evolution algorithms. The experimental results show that our algorithm is able to improve convergence speed and generate solutions which approximate to the best optimal Pareto front.

Key words differential evolution; generalized opposition-based learning (GOBL); multi-objective optimization; constrained optimization; non-dominated sorting

摘 要 差分进化算法是一种简单有效的进化算法,基于泛化反向学习的机制在进化算法中经常可以引导种群的进化.针对多目标的约束优化问题,提出了一种基于泛化反向学习的多目标约束差分进化算法.该算法采用基于泛化反向学习的机制(generalized opposition-based learning, GOBL)产生变换种群,然后在种群初始化和代跳跃阶段,利用非支配排序、拥挤距离和约束处理技术从原始种群和其变换

收稿日期:2015-09-08;**修回日期**:2015-12-30
基金项目:国家自然科学基金项目(61170216,61300198,61572131);广东高校科技创新项目(2013KJCX0178)
This work was supported by the National Natural Science Foundation of China (61170216, 61300198, 61572131) and the Guangdong Science and Technology Innovation Project of Higher Education (2013KJCX0178).
通信作者:袁华强(hyuan66@163.com)

种群中选择更优的种群个体作为新的种群继续迭代进化;该算法通过采用基于泛化反向学习的机制,可以引导种群个体慢慢向最优的 Pareto 前沿逼近,以求得最优解集. 最后采用多目标 Benchmark 问题对该算法进行了实验评估,实验结果表明:与 NSGA-Ⅱ,MOEA/D 及其他的多目标进化算法相比,提出的算法具有更好的收敛性,并且产生的解能够逼近最优的 Pareto 前沿.

关键词 差分进化;泛化反向学习;多目标优化;约束优化;非支配排序

中图法分类号 TP18

多目标优化问题(multi-objective optimization problems, MOPs)在许多领域都有广泛的应用,特别是科学、工程和经济领域[1]. 与单目标优化问题不同,多目标优化问题必须同时优化多个目标函数. 由于多个目标函数相互冲突,所以不能像求解单目标优化问题一样,总能找到一个最优或近优解,而是求解到一个满足所有目标函数折中的解集,这个解集称之为 Pareto 最优解[2]. 由于现实中的科学和工程问题都存在着各种各样的约束条件,因此多目标约束优化问题(multi-objective constrained optimization problems, MCOPs)引起了研究者们的关注[3-5]. 因为约束条件的出现,会导致可行解区域减小并使得搜索解的过程变得更为复杂,因此如何采用约束处理机制来处理 MCOPs,将是一个值得研究的课题.

进化算法(evolution algorithm, EA)是一种随机的、基于种群的优化算法,它在每一次运行中可以找到多个最优解,天生具有解决 MOPs 的特性. 在过去的几十年中,提出了大量的多目标进化算法[3-8],典型的代表主要有:Deb 等人[3]提出的快速和精英选择的多目标非支配排序遗传算法(nondominated sorting genetic algorithm, NSGA-Ⅱ)和 Zhang 等人[8]提出的基于分解的多目标进化算法(multi-objective evolutionary algorithm based on decomposition, MOEA/D).

差分进化算法(differential evolution, DE)是由 Storn 和 Price[9]首次提出的,它是一种功能强大、结构简单、易用且鲁棒性强的全局优化算法,该算法在解决单目标约束优化问题(COPs)时已经取得了显著的成就. Abbass 等人[10]首次将 DE 算法用来解决 MOPs,称为 Pareto 前沿差分进化算法(Pareto-frontier differ-ential evolution, PDE). 在 PDE 算法中,通过 DE 产生子代,然后子代与父代通过支配关系进行比较,丢弃支配解,只保留非支配解到下一代中. Madavan[11]提出了 Pareto 差分进化方法(Pareto differential evolution approach, PDEA),该算法与 PDE 类似,通过 DE 产生子代,然后合并父代与子代个体,再计算它们的 Pareto 非支配排序和多样性排序,最后根据排序选择保留最优个体. Xue 等人[12]提出了多目标差分进化算法(multi-objective differential evolution, MODE),该算法与 NSGA-Ⅱ类似,也采用了 Pareto 非支配排序和拥挤距离排序技术,不同的是,MODE 的适应值首先通过 Pareto 非支配排序计算,然后根据拥挤距离值改变. Robic 等人[13]提出了多目标优化差分进化算法(differential evolution for multi-objective optimization, DEMO),该算法与 MODE 有点相似,仍然采用了 Pareto 非支配排序和拥挤距离排序技术,但是它又像 PDEA 一样,把父代和子代合并,再计算它们的 Pareto 非支配排序和拥挤距离排序,以提升候选解的均匀分布.

基于反向学习(opposition-based learning, OBL)的机制是由 Al-Qunaieer 等人[14]首次提出的,该方法为了获得更优的解来进行下一次迭代,在每次迭代过程中,不仅要评价本次搜索到的最优解,而且还要评价与该最优解处于相反方向的解,然后得出最终的最优解,用来进行下一次迭代. 最近基于反向学习的机器学习方法被广泛用于一些启发式进化算法如差分进化、粒子群、人工神经网络、蚁群和人工蜜蜂算法等[15-16]. 根据基于反向学习的原理,Wang 等人[17]提出了一种基于泛化反向学习(generalized opposition-based learning, GOBL)的机制,GOBL 采用转换搜索空间技术把当前空间的解转换到一个新的空间去,即在求解过程中,不但要考虑当前空间的候选解,而且还要考虑转换空间的候选解. 由于同时搜索当前空间和转换空间的解,GOBL 能够很快地发现最优解.

Rahnamayan 等人[18]首次提出了基于反向学习的差分进化算法,在该算法中,采用基于反向学习的机制来初始化种群. 之后的几年里,包括 Rahnamayan 在内的许多专家学者,分别提出了改进的基于反向学习的差分进化算法[19-23]. 后来 OBL 机制也被用于解决多目标优化问题,Peng 等人[24]提出了一种解决多目标优化问题的基于反向学习的多目标差分进化

算法(opposition-based multi-objective differential evolution algorithm, OMODE),该算法采用 OBL 机制产生初始种群的反向种群,然后通过 Pareto 非支配排序和拥挤距离排序从原初始种群和其反向种群中选最优个体组成新的初始种群继续迭代. Dong 等人[25]提出了一种基于反向操作的多目标差分进化算法(multi-objective differential evolution based on opposite operation, MDEOO),与 OMODE 不同的是,该算法不但在种群初始化阶段产生反向种群,而且在子代的迭代过程中也产生子代的反向种群,然后仍然采用 Pareto 非支配排序和拥挤距离排序技术从原始种群和其反向种群中选择最优的个体继续迭代.

最近 Wang 等人[22]采用 GOBL 机制,提出了一种基于泛化反向学习的差分进化算法(generalized opposition-based differential evolution, GODE),该算法被证明具有较快的收敛速度和求解精度. 然而该算法却不能用于解决多目标优化问题,更不能解决多目标约束优化问题. 另外包括 OMODE 和 MDEOO 在内的基于反向学习的多目标差分进化算法也不能解决多目标约束优化问题. 在这种背景下,本文提出了一种基于泛化反向学习的多目标约束差分进化算法(generalized opposition-based multi-objective constrained differential evolution, GOMCDE),并且针对多目标测试函数集进行了实验测试. 测试结果显示,与 NSGA-Ⅱ和 MOEA/D 及其他相关算法相比,GOMCDE 算法具有更强的全局搜索性能、更快的收敛速度和更好的 Pareto 前沿.

1 相关背景知识

1.1 多目标约束优化问题

在多目标约束优化问题中,既存在等式约束条件和不等式约束条件,还包括向量 $\boldsymbol{x}$ 的上界和下界,具体如下:

$$\min\ f(\boldsymbol{x})=[f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_k(\boldsymbol{x})], \tag{1}$$

$$\text{s.t.}\ \ g_l(\boldsymbol{x})\leqslant 0,\ l=1,2,\cdots,q, \tag{2}$$

$$h_l(\boldsymbol{x})=0,\ l=q+1,q+2,\cdots,m, \tag{3}$$

$$low_j\leqslant x_j\leqslant up_j,\ j=1,2,\cdots,n, \tag{4}$$

其中,$f(\boldsymbol{x})$ 为 k 个目标函数,需要同时优化;$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ 为 n 维决策向量;$g_j(\boldsymbol{x})\leqslant 0$ 和 $h_j(\boldsymbol{x})=0$ 分别表示 q 个不等式约束和 $m-q$ 个等式约束;函数 f_k,g_l 和 h_l 为线性或非线性实数函数;up_j 和 low_j 分别为变量 x_j 的上界和下界. 另外,假定可行解空间中满足所有约束的点集合用 U 表示,搜索空间中满足上界和下界约束的点集合用 S 表示,其中 $S\subset U$.

为了考虑多个目标之间的平衡,引入了解之间的支配概念. 假定解向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$,$\boldsymbol{y}=(y_1,y_2,\cdots,y_n)$,如果 $x_i\leqslant y_i$,$i=1,2,\cdots,n$ 且 $\boldsymbol{x}\neq\boldsymbol{y}$,则 $\boldsymbol{x}\prec\boldsymbol{y}$,即称为 $\boldsymbol{x}$ 支配 $\boldsymbol{y}$. 如果对于一个解向量 $\boldsymbol{x}^*=(x_1^*,x_2^*,\cdots,x_n^*)$ 满足 $\neg\exists\boldsymbol{x}\in U,\boldsymbol{x}\prec\boldsymbol{x}^*$,则称 $\neg\exists\boldsymbol{x}\in U,\boldsymbol{x}\prec\boldsymbol{x}^*$ 为 Pareto 最优解. 由所有 Pareto 最优解组成的集合,称为 Pareto 最优解集(Pareto set, PS). 在多目标优化问题中,Pareto 最优解集在目标空间对应的目标向量称为 Pareto 前沿(Pareto front, PF),具体表示如下:

$$PF=\{f(\boldsymbol{x}^*)\mid\boldsymbol{x}^*\in PS\}. \tag{5}$$

求解多目标约束优化问题的目标,就是要找到一个满足各种约束条件的 Pareto 前沿.

1.2 差分进化算法

差分进化算法是一类比较流行的进化算法,并且具有良好的性能,广泛地应用于各种应用领域. 在差分进化算法中,一般包括 NP 个种群,每个个体向量的维度为 n. 一般地,个体表示为:向量 $\boldsymbol{x}_{i,t}=(x_{1i,t},x_{2i,t},\cdots,x_{mi,t})$,其中 $i=1,2,\cdots,NP$,NP 为种群大小,n 为个体向量的维度,t 为当前种群的代数. 差分进化算法包括 3 个主要的操作:变异、交叉和选择[13].

1) 变异. 在变异操作中,对于每个种群产生目标向量 $\boldsymbol{v}_{i,t}$,变异策略主要如下:

① rand/1:

$$\boldsymbol{v}_{i,t}=\boldsymbol{x}_{r_1,t}+F\times(\boldsymbol{x}_{r_2,t}-\boldsymbol{x}_{r_3,t}). \tag{6}$$

② best/1:

$$\boldsymbol{v}_{i,t}=\boldsymbol{x}_{\text{best},t}+F\times(\boldsymbol{x}_{r_1,t}-\boldsymbol{x}_{r_2,t}). \tag{7}$$

③ current-to-best/1:

$$\boldsymbol{v}_{i,t}=\boldsymbol{x}_{i,t}+F\times(\boldsymbol{x}_{\text{best},t}-\boldsymbol{x}_{i,t})+F\times(\boldsymbol{x}_{r_1,t}-\boldsymbol{x}_{r_2,t}). \tag{8}$$

④ best/2:

$$\boldsymbol{v}_{i,t}=\boldsymbol{x}_{\text{best},t}+F\times(\boldsymbol{x}_{r_1,t}-\boldsymbol{x}_{r_2,t})+F\times(\boldsymbol{x}_{r_3,t}-\boldsymbol{x}_{r_4,t}). \tag{9}$$

⑤ rand/2:

$$\boldsymbol{v}_{i,t}=\boldsymbol{x}_{r_1,t}+F\times(\boldsymbol{x}_{r_2,t}-\boldsymbol{x}_{r_3,t})+F\times(\boldsymbol{x}_{r_4,t}-\boldsymbol{x}_{r_5,t}). \tag{10}$$

其中,下标 r_1,r_2,r_3,r_4,r_5 都是随机从集合 $\{1,2,\cdots,NP\}\backslash\{i\}$ 中选取的;$\boldsymbol{x}_{\text{best},t}$ 为种群在第 t 代最好的个体;缩放因子 F 为实数,$F\in[0,1]$.

2) 交叉. 交叉操作主要产生试验变量 $\boldsymbol{u}_{i,t}=(u_{1i,t},u_{2i,t},\cdots,u_{mi,t})$,具体如下:

$$u_{ji,t}=\begin{cases}v_{ji,t}, & \text{if } random_j(0,1)\leqslant CR \text{ 或 } j=j_{random},\\ x_{ji,t}, & \text{otherwise},\end{cases} \tag{11}$$

其中,$i=1,2,\cdots,NP$, $j=1,2,\cdots,n$;j_{random} 是属于 1～n 之间的一个随机整数;$random_j(0,1)$为对于每个 j 产生[0,1]均匀分布的随机数;使用参数 j_{random} 是为了保证试验向量 $\boldsymbol{u}_{i,t}$不同于目标向量 $\boldsymbol{x}_{i,t}$;交叉概率因子 CR 的取值通常为 0～1.

3) 选择. 在选择操作中,目标向量 $\boldsymbol{x}_{i,t}$和试验向量 $\boldsymbol{u}_{i,t}$根据它们的适应值进行比较,选择适应值更优的进入下一代种群:

$$\boldsymbol{x}_{i,t+1}=\begin{cases}\boldsymbol{u}_{i,t}, & \text{if } f(\boldsymbol{u}_{i,t})\leqslant f(\boldsymbol{x}_{i,t}).\\ \boldsymbol{x}_{i,t}, & \text{otherwise.}\end{cases} \tag{12}$$

1.3 反向学习

为了更形象地、更清楚地解释反向学习的概念,必须先理解反向点的概念. 图 1 显示了 1 维区间[a, b]之间 z 的反向点 $\bar{z}$ 的例子.

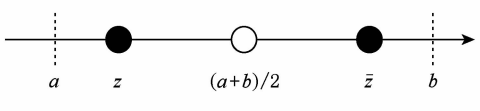


Fig. 1 An example for opposite point.

图 1 反向点的例子

定义 1. 假设 $z\in[a,b]$,z 为实数,z 的 1 维反向点 $\bar{z}$ 表示为

$$\bar{z}=a+b-z,\bar{z}\in[a,b]. \tag{13}$$

如果对于一个 n 维向量的反向点,则用如下定义表示.

定义 2. 假设 $\boldsymbol{P}=(z_1,z_2,\cdots,z_n)$为一个 n 维空间的点,其中 $z_1,z_2,\cdots,z_n\in\mathbb{R}$且 $z_i\in[a_i,b_i]$, $\forall i\in\{1,2,\cdots,n\}$,则 $\boldsymbol{P}$ 的反向点 $\bar{\boldsymbol{P}}=(\overline{z_1},\overline{z_2},\cdots,\overline{z_n})$由式(14)决定.

$$\overline{z_i}=a_i+b_i-z_i. \tag{14}$$

有了反向点的定义之后,那么基于反向学习的优化可以定义如下:

定义 3. 假设 $\boldsymbol{P}=(z_1,z_2,\cdots,z_n)$为一个 n 维空间的点(比如 $\boldsymbol{P}$ 是候选解),$f(\cdot)$是候选解的目标函数适应值,另外根据反向点的定义可知 $\boldsymbol{P}$ 的反向点为 $\bar{\boldsymbol{P}}=(\overline{z_1},\overline{z_2},\cdots,\overline{z_n})$. 如果 $f(\bar{\boldsymbol{P}})<f(\boldsymbol{P})$,则表示 $\bar{\boldsymbol{P}}$ 比 $\boldsymbol{P}$ 具有更好的适应值,此时选择 $\bar{\boldsymbol{P}}$ 代替 $\boldsymbol{P}$,否则保持 $\boldsymbol{P}$ 不变.

1.4 泛化反向学习

OBL 仅仅是求反向点,而 GOBL 则是把当前空间的点或候选解变换到一个新的空间,然后和 OBL 原理一样,同时评价当前空间和变换空间的候选解,选出最优的候选解. 在文献[17]中,GOBL 被证明具有更强的能力发现全局最优解.

定义 4. 假设 $z\in[a,b]$,z 为当前 1 维空间 S 中的点,转换空间 $\bar{S}$ 中的点 $\bar{z}$ 表示为

$$\bar{z}=k(a+b)-z, \tag{15}$$

其中,k 是一个属于[0,1]的随机数,并且 $\bar{z}\in[k(a+b)-b,k(a+b)-a]$.

在实际问题中,$\bar{z}$ 可能会超出区间 $[a,b]$,此时会因为转换空间的候选解不可行而使得 GOBL 无效,当出现此情况时,可以让 $\bar{z}$ 在[a,b]中随机选择一个值. 具体如下:

$$\bar{z}=random(a,b), \text{ if } \bar{z}<a \text{ 或 } \bar{z}>b, \tag{16}$$

其中,$random(a,b)$为产生[a,b]中的随机数.

定义 5. 假设 $\boldsymbol{P}=(z_1,z_2,\cdots,z_n)$为一个 n 维空间的点,其中 $z_1,z_2,\cdots,z_n\in\mathbb{R}$且 $z_i\in[a_i,b_i]$, $\forall i\in\{1,2,\cdots,n\}$. 则 $\boldsymbol{P}$ 的转换空间点 $\bar{\boldsymbol{P}}=(\overline{z_1},\overline{z_2},\cdots,\overline{z_n})$的元素可由式(17)计算.

$$\overline{z_i}=k(a_i+b_i)-z_i, \tag{17}$$

其中,$k=random(0,1)$.

$$\overline{z_i}=random(a_i,b_i), \text{ if } \overline{z_i}<a_i \text{ 或} \overline{z_i}>b_i. \tag{18}$$

对于 GOBL 机制的优化,采用定义 3 中的方法从 $\boldsymbol{P}$ 和 $\bar{\boldsymbol{P}}$ 中选择更优的个体进行迭代.

2 GOMCDE 算法

根据 GOBL 定义可知,基于泛化反向学习的机制需要比较种群与转换种群的适应值. 而对于多目标约束优化问题的适应值比较不能像单目标无约束优化问题一样简单处理. 本文采用适应值变换、Pareto 非支配排序和拥挤距离排序技术来处理 GOBL 中的适应值比较问题.

2.1 适应值变换约束处理技术

为了解决多目标约束优化问题,我们必须在 GOMCDE 算法中加入约束处理技术. 我们知道,在无约束的差分进化算法中,适应值都等于目标函数值,但是在约束差分进化算法中,由于约束的存在,不能简单地考虑适应值等于目标函数值,必须要考虑约束条件的存在. 比如,对于最小值优化问题,解向量 $\boldsymbol{x}$ 的目标函数值比 $\boldsymbol{y}$ 小,但 $\boldsymbol{x}$ 却违反了约束条件而 $\boldsymbol{y}$ 没有违反约束条件. 此时,应该考虑可能解向量 $\boldsymbol{y}$ 比 $\boldsymbol{x}$ 更优. 基于这种原理,我们采用了适应值变换技术来处理约束优化问题.

一般地,对于约束优化问题,经常会把等式约束转化成不等式约束,具体如下:

$$|h_l(\boldsymbol{x})|-\delta\leqslant 0, \tag{19}$$

其中,$l\in\{q+1,q+2,\cdots,m\}$;δ 为等式约束违反的容忍因子,一般取正整数.解 $\boldsymbol{x}$ 到第 l 个约束的距离可以表示为

$$G_l(\boldsymbol{x})=\begin{cases}\max\{0,g_l(\boldsymbol{x})\}, & 1\leqslant l\leqslant q,\\ \max\{0,|h_l(\boldsymbol{x})|-\delta\}, & q+1\leqslant l\leqslant m,\end{cases}\tag{20}$$

那么解 $\boldsymbol{x}$ 到可行解区域的边界距离,即约束违反程度可以表示为

$$G(\boldsymbol{x})=\sum_{l=1}^{m}G_l(\boldsymbol{x}).\tag{21}$$

在适应值变换约束处理技术中,为了更好地处理约束差分进化算法中的适应值,采用应值变换方法把种群分成 3 种状态:不可行状态、半可行状态和可行状态.

1) 不可行状态.在不可行状态下,种群只包含了不可行解,因此不用考虑目标函数值,只需要考虑约束违反程度.约束违反程度可以通过式(21)计算,此时适应值计算式为

$$f_{\text{fitness}}(\boldsymbol{x}_{i,t})=G(\boldsymbol{x}_{i,t}).\tag{22}$$

2) 半可行状态.在半可行状态下,种群既包含了部分可行解又包含了部分不可行解,因此就必须在目标函数值和约束违反程度之间找到一个平衡点.从解的层面分析,此时把种群细分为可行解组(Z_1)和不可行解组(Z_2).因此解 $\boldsymbol{x}_{i,t}$ 的目标函数值 $f'(\boldsymbol{x}_{i,t})$ 就可以转换成:

$$f'(\boldsymbol{x}_{i,t})=\begin{cases}f(\boldsymbol{x}_{i,t}), & i\in Z_1,\\ \max\{\varphi\times f(\boldsymbol{x}_{\text{best},t})+(1-\varphi)\times f(\boldsymbol{x}_{\text{worst},t}),f(\boldsymbol{x}_{i,t})\}, & i\in Z_2,\end{cases}\tag{23}$$

其中,φ 是上一代种群的可行解比率;$\boldsymbol{x}_{\text{best},t}$ 和 $\boldsymbol{x}_{\text{worst},t}$ 分别是可行解组 Z_1 最优和最差的解.进一步把式(23)归一化为

$$f_{\text{nor}}(\boldsymbol{x}_{i,t})=\frac{f'(\boldsymbol{x}_{i,t})-\min\limits_{j\in Z_1\cup Z_2}f'(\boldsymbol{x}_{j,t})}{\max\limits_{j\in Z_1\cup Z_2}f'(\boldsymbol{x}_{j,t})-\min\limits_{j\in Z_1\cup Z_2}f'(\boldsymbol{x}_{j,t})}.\tag{24}$$

式(20)可以计算约束违反程度,在此状态下,式(21)归一化为

$$G_{\text{nor}}(\boldsymbol{x}_{i,t})=\begin{cases}0, & i\in Z_1,\\ \dfrac{G(\boldsymbol{x}_{i,t})-\min\limits_{j\in Z_2}G(\boldsymbol{x}_{j,t})}{\max\limits_{j\in Z_2}G(\boldsymbol{x}_{j,t})-\min\limits_{j\in Z_2}G(\boldsymbol{x}_{j,t})}, & i\in Z_2,\end{cases}\tag{25}$$

因此最终的适应值可以表示为

$$f_{\text{fitness}}(\boldsymbol{x}_{i,t})=f_{\text{nor}}(\boldsymbol{x}_{i,t})+G_{\text{nor}}(\boldsymbol{x}_{i,t}).\tag{26}$$

3) 可行状态.在可行状态下,种群中所有个体都是可行解,因此就可以看作是无约束优化问题来处理.此时适应值就等于目标函数值 $f(\boldsymbol{x}_{i,t})$,具体为

$$f_{\text{fitness}}(\boldsymbol{x}_{i,t})=f(\boldsymbol{x}_{i,t}).\tag{27}$$

通过适应值变换技术,所有的个体都基于变换后的适应值进行比较,选择适应值更优的个体继续进化.对于多目标约束优化问题,种群中的个体采用 Pareto 非支配排序技术进行比较.从适应值变换的原理可以看出,对于多目标无约束优化问题,适应值变换技术同样适用,即无约束优化问题中的种群个体属于适应值变换中的第 3 种状态:可行状态.

2.2 Pareto 非支配排序

在 Pareto 非支配排序之前,必须先计算:1)支配数 n_p,它是解 p 支配其他解的个数;2)支配集 S_p.在 Pareto 非支配排序过程中,首先从支配集 S_p 中选取所有 $n_p=0$ 的成员 q 加入链表 Q 中,然后设置其他成员的支配数减 1,重复这 2 步直到所有的成员都加入到链表 Q 中为止.

2.3 拥挤距离

拥挤距离用来衡量解分布的密度,在非支配集 I 中,解 i 的拥挤距离等于解 i 的邻居解 $i-1$ 和解 $i+1$ 之间沿着每个目标的平均距离,具体可根据式(28)计算.在计算拥挤距离时,首先要根据目标函数值排序种群.在本文中,由于约束的存在,我们根据变换后的适应值来排序种群,然后再根据式(28)计算每个解的拥挤距离.由于式(28)并不能应用于边界解的拥挤距离的计算,所以边界解的拥挤距离通常被设置为无穷大.

$$I[i]_{\text{distance}}=(I[i+1].m-I[i-1].m)/(f_m^{\max}-f_m^{\min}),\tag{28}$$

其中,$I[i].m$ 表示非支配集 I 中第 i 个个体的第 m 个目标函数值;$f_m^{\max}$ 和 $f_m^{\min}$ 分别表示第 m 个目标函数的最大值和最小值.

2.4 算法描述

绝大多数的进化算法都是在种群初始化阶段和代跳跃阶段使用 OBL 机制求其反向种群.在本文中,我们也在种群初始化阶段和代跳跃阶段使用 GOBL 机制求其变换种群.对于随机生成的初始种群 $\boldsymbol{P}_0$(其中种群大小为 NP,个体向量的维度为 n),通过式 $GOP_{ji,0}=k(a_j+b_j)-P_{ji,0}$ 计算出其变换种群 $\boldsymbol{GOP}_0$.其中,$i=1,2,\cdots,NP$;$j=1,2,\cdots,n$;k 为[0,1]之间的随时数.然后采用 Pareto 非支配排序和拥挤距离排序技术根据变换后的适应值,从种群 $\boldsymbol{P}_0$ 和 $\boldsymbol{GOP}_0$ 中选出 NP 个适应值更优的个体组成新的初始种群 $\boldsymbol{P}_0$.在种群代跳跃阶段,与单目标优化问题相同的是,该阶段的执行也必须满足一个代跳跃概率 J,在绝大部分情况下 $J=0.3$[18].与单目标优化问题不同的是,变换种群的计算式与种群初始化阶段类似,仍然采用[a,b]作为决策空间.即

$GOP_{ji,t}=k(a_j+b_j)-P_{ji,t}$,其中 t 为种群的代数.最后在个体的生成选择阶段,与其他多目标差分进化算法的流程一样,采用 Pareto 非支配排序和拥挤距离排序技术根据变换后的适应值,从父代群 $\boldsymbol{P}_{t-1}$、子代种群 $\boldsymbol{P}_t$ 和变换种群 $\boldsymbol{GOP}$ 中选出 NP 个适应值更优的个体继续迭代.GOMCDE 算法伪代码如下:

算法 1. GOMCDE 伪代码.

输入:初始种群 $\boldsymbol{P}_0=(\boldsymbol{x}_{1,0},\boldsymbol{x}_{2,0},\cdots,\boldsymbol{x}_{NP,0})$;

输出:外部非支配集 E 中的个体.

① 评价种群 $\boldsymbol{P}_0$,求出变换后的适应值和约束违反程度;
② for ($i=0;i<NP;i++$)
③ for ($j=0;j<n;j++$)
④ $GOP_{ji,0}=k\times(a_j+b_j)-P_{ji,0}$;
/ * $k=random(0,1)$ * /
⑤ if $GOP_{ji,0}<a_j \parallel GOP_{ji,0}>b_j$
⑥ $GOP_{ji,0}=random(a_j,b_j)$;
⑦ end if
⑧ end for
⑨ end for
⑩ 评价变换种群 $\boldsymbol{GOP}_0$,求出变换后的适应值和约束违反程度;
⑪ 根据变换后的适应值,采用 Pareto 非支配排序和拥挤距离排序技术从种群 $\boldsymbol{P}_0$ 和变换种群 $\boldsymbol{GOP}_0$ 中选取 NP 个最优的个体组成新种群 $\boldsymbol{P}_0$,并把非支配个体存入外部非支配集 E 中;
⑫ 评价新种群 $\boldsymbol{P}_0$,求出变换后的适应值和约束违反程度;
⑬ $t=0$; / * t 表示种群代数 * /
⑭ while 停止条件不满足
⑮ $t=t+1$;
⑯ for 种群 $\boldsymbol{P}_{t-1}$ 中的每个个体 do
⑰ 执行变异、交叉和选择策略获得新种群 $\boldsymbol{P}'_t$;
⑱ end for
⑲ $\boldsymbol{P}_t=\boldsymbol{P}'_t$;
⑳ 评价种群 $\boldsymbol{P}_t$,求出变换后的适应值和约束违反程度;
㉑ if ($random(0,1)<J$) / * J 是代跳跃概率 * /
㉒ for ($i=0;i<NP;i++$)
㉓ for ($j=0;j<n;j++$)
㉔ $GOP_{ji,t}=k\times(a_j+b_j)-P_{ji,t}$;
㉕ if $GOP_{ji,t}<a_j \parallel GOP_{ji,t}>b_j$
㉖ $GOP_{ji,t}=random(a_j,b_j)$;
㉗ end if
㉘ end for
㉙ end for
㉚ end if
㉛ 评价种群 $\boldsymbol{GOP}_t$,求出变换后的适应值和约束违反程度;
㉜ 根据变换后的适应值,采用 Pareto 非支配排序和拥挤距离排序技术从种群 $\boldsymbol{P}_{t-1}$,$\boldsymbol{P}_t$ 和变换种群 $\boldsymbol{GOP}_t$ 中选取 NP 个最优的个体,并更新外部非支配集 E;
㉝ end while

从算法 1 的描述中可以看出,GOMCDE 算法简单且容易操作,它采用适应值变换技术不但可以解决多目标约束优化问题,同时也适用于多目标无约束优化问题.由于 GOBL 的时间复杂度为 $O(NP\times n)$,Pareto 非支配排序的时间复杂度为 $O(k\times NP\times NP)$,拥挤距离排序的时间复杂度为 $O(k\times NP\times \log NP)$,所以 GOMCDE 算法的时间复杂度为 $O(NP\times(n+k\times NP))$.

3 实验测试

为了进一步验证 GOMCDE 算法的有效性,我们对 GOMCDE 算法和 NSGA-Ⅱ[3],EAMO[26],CHMEO[5],CMODE[6],CMOPSO[27]等算法针对多目标约束优化问题进行了对比实验测试.用于测试的约束优化问题主要包括 BNN[28],SRN[29],TNK[30],CONSTR[31],CTP1-CTP7[31],Welded Beam Problem[26],其中 Welded Beam Problem 是一个现实生活中的测试问题.在这些测试问题中,有些具有连续的 Pareto 前沿,有些则是不连续的.为了证明 GOMCDE 算法对于多目标无约束优化问题同样有效,我们对 GOMCDE 算法和 NSGA-Ⅱ[3],MOEA/D[8],MODE-RMO[32],OMODE[24],MDEOO[25] 等算法针对 ZDT1-ZDT4[33]和 ZDT6[33]等无约束优化问题进行了对比实验测试.在所有的实验过程中,实验环境为 64 位的 Windows 7 系统,其中 CPU 为 Intel Core TM 2.83 GHz,内存为 4 GB,编程语言为 Maltab 2013 b.

3.1 实验参数设置

1) 种群大小.$NP=100$.
2) 交叉概率.$CR=0.8$.
3) 缩放因子.$F=0.2$.
4) 代跳跃概率.$J=0.3$.
5) 最大代数.$G_{\max}=100$.

3.2 性能评价指标

1) Pareto 前沿 PF[3]. Pareto 非支配集中的最优解映射在目标空间中的目标向量.

2) 收敛性评价指标 r[3]. 表示近似解到 Pareto 前沿最优解的欧氏距离,r 值越小,表示近似解越接近 Pareto 前沿,收敛性越好. r 的计算公式为

$$r=\frac{\sum_{i=1}^{N}d_i}{N}, \tag{29}$$

其中,N 为非支配解的个数,d_i 为相邻非支配解之间的欧氏距离.

3) 多样性评价指标 Δ[3]. 表示近似解集的分散程度,Δ 值越小,表示解集的分布性越好,即多样性就越好. 当 $\Delta=0$ 时,说明所有的解沿着 Pareto 前沿均匀分布. Δ 的计算公式为

$$\Delta=\frac{d_f+d_l+\sum_{i=1}^{N-1}|d_i-\bar{d}|}{d_f+d_l+(N-1)\bar{d}}, \tag{30}$$

其中,d_i 表示非支配集中 2 个相邻解之间的欧氏距离,$\bar{d}$ 表示所有 d_i 的平均值,d_f 和 d_l 是非支配集中处于极限位置解和边界位置解之间的欧氏距离.

4) 超体积评价指标 I_{H} (hypervolume indicator)[34]. 表示近似解集 A 在目标空间相对于参考点所支配的空间大小,I_{H} 值越大,解集 A 在目标空间所支配的区域就越大,即该解集 A 就越接近最优解,分布性越好,同时也可以表示收敛性越好.

5) 基于参考集的超体积评价指标 I_{H} (hypervolume difference to a reference set)[34]. 在 I_{H} 的基础上,考虑参考集 R,基于参考集的超体积评价指标为 $I_{\mathrm{H}}=I_{\mathrm{H}}(R)-I_{\mathrm{H}}(A)$,与 I_{H} 相反,I_{H} 的值越小,表示解集 A 就越接近最优解,分布性和收敛性越好.

6) 基于 ε 支配的评价指标 I_ε[34]. 对于给定的 $\varepsilon>0$ 和参考集 R,I_ε 可以定义为

$$I_\varepsilon(A,B)=\inf_{\varepsilon\in R}\{\forall z_2\in B,\exists z_1\in A:z_1\succeq_\varepsilon z_2\}, \tag{31}$$

其中,A 和 B 为 2 个近似解集. I_ε 值越小,表示算法的收敛性越好.

为了比较的公平性,在实验中,针对无约束优化问题,我们采用评价指标 r 和 Δ 对算法进行性能评价. 针对约束优化问题,我们采用评价指标 I_{H} 和 I_ε 对算法进行性能评价. 这是因为对于多目标无约束优化问题,绝对大部分的算法都是采用 r 和 Δ 指标进行评价,而对于多目标约束优化问题,则采用 I_{H} 和 I_ε 指标进行评价.

7) Mann-Whitney 检验分析[35]. 该检验分析是由 Mann 和 Whitney 提出的针对 2 个独立样本进行分析的非参数检测方法,它在这 2 独立样本的总体分布未知的情况下,通过 2 组样本的分析来推断总体的分布是否存在显著差异.

3.3 性能评价指标

实验中所有算法对于每一个测试函数都运行 50 次,在每一次运行过程中设置 $Max\_FEs=2.5\times10^4$.

1) 针对约束优化问题的比较

我们对 GOMCDE 算法和 NSGA-Ⅱ, EAMO, CHMEO, CMODE, CMOPSO 等算法进行了对比实验测试,表 1 显示了对 I_{H} 和 I_ε 的 Mann-Whitney 显著性测试结果,其中粗体表示该算法的值更优. 从表 1 看出,当置信水平为 0.05 时,GOMCDE 算法明显优于其他 5 种算法.

Table 1 Mann-Whitney Analysis on I_{H} and I_ε Between GOMCDE and Other Algorithms

表 1 GOMCDE 算法与其他算法关于 I_{H} 和 I_ε 值的 Mann-Whitney 检验分析

Problems	GOMCDE vs NSGA-Ⅱ		GOMCDE vs EAMO		GOMCDE vs CHMEO		GOMCDE vs CMODE		GOMCDE vs CMOPSO	
	I_{H}	I_ε	I_{H}	I_ε	I_{H}	I_ε	I_{H}	I_ε	I_{H}	I_ε
BNH	**3.21E−04**	**4.41E−05**	**4.12E−05**	**8.32E−05**	**8.29E−03**	**2.28E−04**	**3.42E−05**	**3.81E−05**	**5.45E−03**	**4.11E−03**
SRN	**2.41E−04**	**4.32E−05**	**4.13E−05**	**4.13E−05**	**7.17E−03**	**3.41E−04**	**1.43E−03**	**4.89E−04**	**2.21E−03**	**5.09E−04**
TNK	**3.22E−02**	**2.31E−03**	**1.78E−04**	**1.22E−02**	**2.41E−02**	**6.37E−04**	**4.21E−05**	**1.67E−06**	**3.11E−02**	**2.17E−04**
CONSTR	**5.51E−03**	**4.56E−03**	**4.13E−05**	**3.49E−04**	**3.38E−03**	**4.52E−03**	**3.21E−05**	**3.76E−05**	**1.31E−04**	**7.76E−03**
CTP1	**7.42E−03**	**4.21E−05**	**4.57E−03**	**4.14E−05**	**2.38E−03**	**2.04E−05**	**3.32E−05**	**2.89E−05**	**2.12E−03**	**2.22E−04**
CTP2	**4.22E−02**	1.34E−01	3.29E−01	1.38E−01	**6.71E−03**	**1.03E−03**	**1.25E−04**	3.51E−01	5.57E−01	4.08E−01
CTP3	2.25E−01	**4.13E−05**	**4.12E−05**	**1.66E−04**	**6.47E−02**	**1.59E−04**	7.42E−01	**6.87E−03**	**1.72E−03**	**1.51E−03**
CTP4	**5.22E−03**	**5.61E−03**	**4.12E−05**	**4.33E−04**	**5.35E−03**	**1.42E−04**	**2.43E−03**	**8.97E−03**	**4.03E−03**	**1.12E−04**
CTP5	**3.21E−02**	**3.41E−02**	**2.73E−03**	**3.72E−04**	5.34E−02	**1.28E−03**	**3.47E−06**	**5.96E−06**	**1.57E−04**	**2.28E−05**
CTP6	8.32E−02	7.32E−02	**2.38E−02**	**3.24E−04**	**2.81E−02**	5.73E−02	1.31E−01	**4.52E−03**	**5.81E−03**	6.12E−02
CTP7	**4.13E−02**	**4.97E−02**	**1.79E−02**	**2.12E−04**	**2.78E−02**	**1.16E−03**	**4.33E−06**	**7.34E−05**	**9.03E−03**	**8.25E−03**
Welded Beam	**7.42E−03**	**6.78E−03**	**5.43E−04**	**4.98E−03**	**7.45E−03**	**4.14E−03**	**4.38E−06**	**3.12E−06**	**3.38E−02**	**6.84E−03**

我们随机地选取了 CTP1 测试问题做了 Pareto 前沿的测试,图 2 显示了这 6 种算法的 Pareto 前沿测试结果,NSGA-Ⅱ算法虽然能够逼近最优 Pareto 前沿,但其解的分布性较差;EAMO 算法的分布性较优于 NSGA-Ⅱ算法,但是它的收敛性稍差;CMODE 算法与 EAMO 算法类似;CHMEO,CMOPSO 算法在收敛性和分布性方面都明显优于前面 3 种算法,但是当它们与 GOMCDE 算法比较时,却又不及 GOMCDE 算法. 所以 GOMCDE 算法不但具有较好的收敛性,而且还具有比较均匀的分布性,明显优于其他 5 种算法.

另外,我们又随机地选取了 CTP4 测试问题做了统计盒图的测试,图 3 显示了这 6 种算法对于 I_H 和 I_ε 指标的盒图测试结果. 盒图可以直观地观察到测试数据的异常情况,盒图一般由 5 条线构成,从上到下分别表示最大值、上四分位数、中位数、下四分位数和最小值.

从图 3 可以看出,GOMCDE 算法比其他 5 种

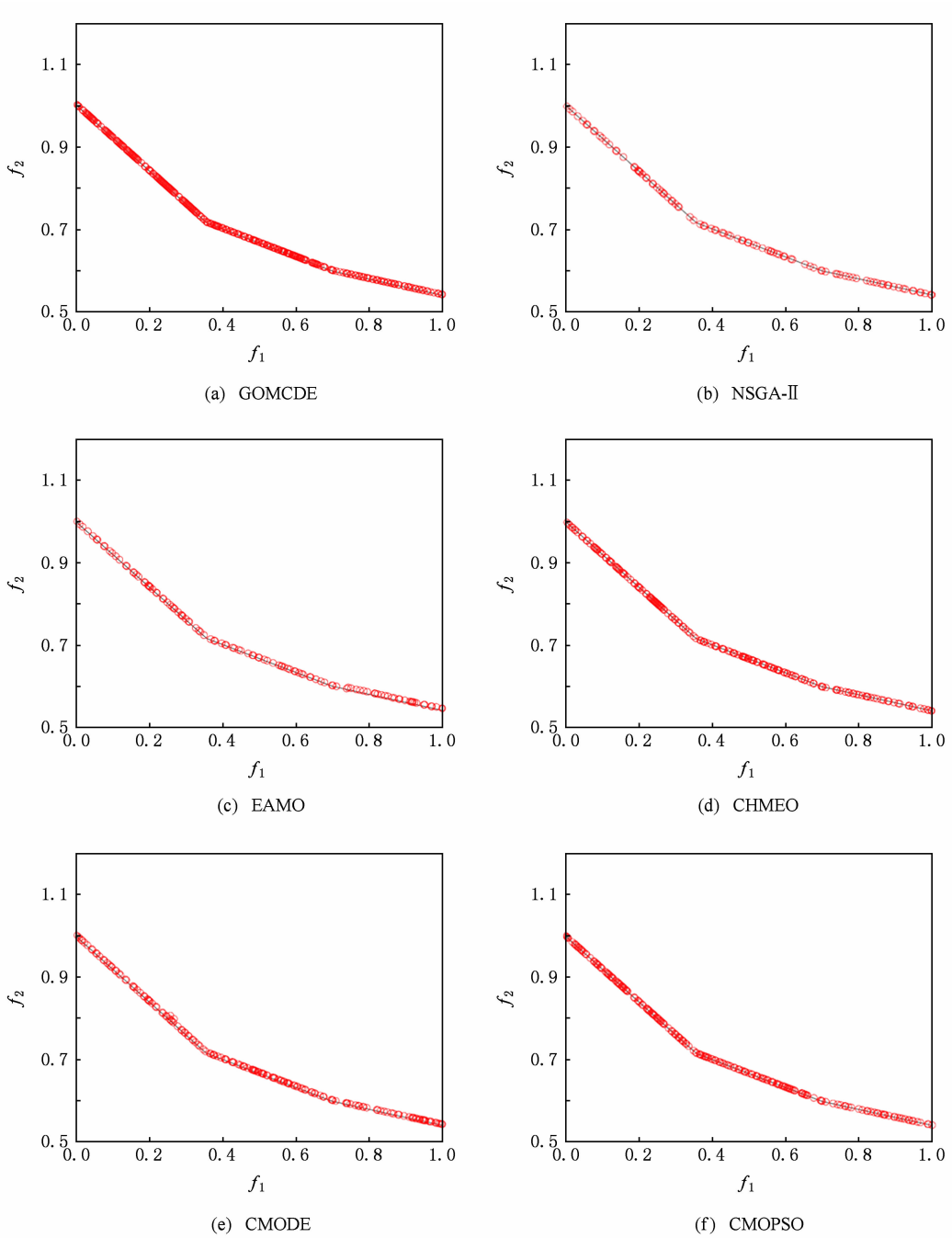
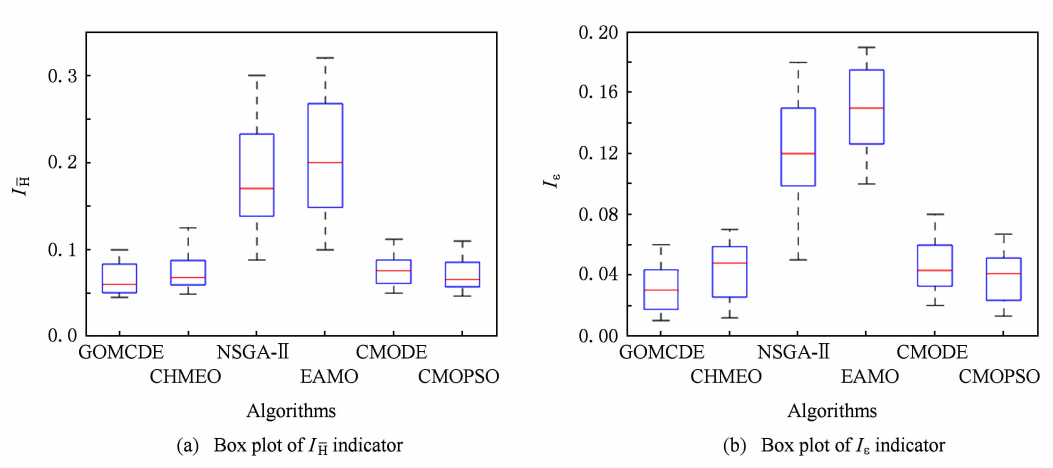


Fig. 2 Pareto front on CTP1 for six algorithms.

图 2 6 种算法针对 CTP1 测试问题的 Pareto 前沿

算法无论是对于 I_H 还是对于 I_ε 指标，都具有较好的平均值和最小值；另外，GOMCDE 算法得到的统计数据并没有出现异常情况，这说明 GOMCDE 算法的鲁棒性较强.



(a) Box plot of I_H indicator (b) Box plot of I_ε indicator

Fig. 3 Box plots of I_H and I_ε on CTP4 for six algorithms.

图 3 6 种算法针对 CTP4 测试问题关于 I_H 和 I_ε 的统计盒图

2) 针对无约束优化问题比较

因为 GOMCDE 算法也可以适用于无约束优化问题，所以我们也对 GOMCDE 算法与 NSGA-Ⅱ，MOEA/D，MODE-RMO 以及其他 2 类基于 OBL 机制的多目标算法 OMODE，MDEOO 针对无约束的 Benchmark 问题进行了实验测试，表 2 显示了比较结果，其中粗体表示该算法的值最优，旁边括号表示这 6 种算法的排名.

Table 2 Convergence and Diversity Comparison Between GOMCDE and Other Algorithms

表 2 GOMCDE 算法与其他算法关于收敛和性和多样性的比较

Algorithms	Criterion	ZDT1		ZDT2		ZDT3		ZDT4		ZDT6	
		Mean(Ranking)	Std	Mean(Ranking)	Std	Mean(Ranking)	Std	Mean(Ranking)	Std	Mean(Ranking)	Std
GOMCDE	r	**2.51E−04(1)**	5.12E−05	**1.04E−05(1)**	5.84E−04	**1.02E−03(1)**	8.94E−06	**3.94E−03(1)**	5.89E−01	3.94E−03(2)	1.87E−04
	Δ	**2.24E−01(1)**	2.13E−02	**1.39E−01(1)**	1.37E−02	**2.84E−01(1)**	3.38E−03	**1.98E−01(1)**	3.64E−01	**2.24E−01(1)**	1.23E−01
NSGA-Ⅱ	r	8.94E−04(4)	0.00E+00	8.24E−04(6)	0.00E+00	4.34E−02(5)	4.20E−05	3.23E+00(4)	7.31E+00	7.81E+00(6)	1.67E−03
	Δ	4.64E−01(6)	4.16E−02	4.35E−01(6)	2.46E−02	5.76E−01(5)	5.08E−03	4.79E−01(4)	9.84E−03	6.44E−01(6)	3.50E−02
MOEA/D	r	7.79E−04(3)	6.50E−04	5.28E−04(5)	3.73E−04	5.23E−02(6)	3.45E−03	3.25E−00(5)	8.49E−05	**7.32E−04(1)**	1.41E−04
	Δ	3.78E−01(4)	1.23E−01	3.03E−01(4)	1.68E−01	8.86E−01(6)	1.48E−02	6.23E−01(5)	3.27E−01	2.98E−01(2)	3.10E−02
MODE-RMO	r	6.17E−04(2)	5.51E−04	4.55E−04(4)	3.03E−04	4.12E−02(4)	1.45E−04	3.36E−00(6)	2.19E−02	7.12E−03(5)	1.41E−04
	Δ	4.25E−01(5)	2.43E−01	2.93E−01(3)	2.54E−01	5.71E−01(4)	1.16E−03	6.39E−01(6)	2.57E−01	5.88E−01(5)	3.10E−02
OMODE	r	1.41E−03(6)	2.40E−04	1.10E−05(3)	6.30E−05	1.38E−03(3)	7.30E−05	6.22E−02(3)	8.33E−02	5.68E−03(4)	1.02E−03
	Δ	3.34E−01(3)	2.58E−02	3.11E−01(5)	1.81E−02	5.03E−01(3)	2.63E−02	3.97E−01(3)	1.01E−01	3.90E−01(4)	2.15E−01
MDEOO	r	1.31E−03(5)	2.35E−04	1.08E−05(2)	5.41E−04	1.21E−03(2)	1.17E−05	5.11E−03(2)	8.36E−02	4.65E−03(3)	2.14E−04
	Δ	3.24E−01(2)	3.59E−02	2.01E−01(2)	2.23E−02	3.51E−01(2)	8.12E−03	3.28E−01(2)	5.58E−01	3.58E−01(3)	1.32E−01

从表 2 可以看到，GOMCDE 算法明显优于其他算法，绝大部分结果都排在第一的位置；而且 OMODE，MDEOO 算法比 NSGA-Ⅱ，MOEA/D，MODE-RMO 算法也略占优势，这是由于 OMODE 和 MDEOO 算法都采用了 OBL 机制的原因. 另外还可以看出，MDEOO 算法比 OMODE 算法稍占优势，这是因为虽然它们都采用了 OBL 机制，但 OMODE 算法只是在种群初始化阶段采用 OBL 机制，而 MDEOO 算法在种群初始化和代跳跃阶段都采用了 OBL 机制. GOMCDE 算法比 MDEOO 算法较优是因为 GOMCDE 算法在种群初始化和代跳跃阶段采用了 GOBL 机制.

对于无约束优化问题,我们随机地选取了ZDT2测试问题做了Pareto前沿的测试,图4显示了这6种算法的Pareto前沿测试结果.从测试结果来看,GOMCDE算法和OMODE算法、MDEOO算法虽然都达到了最优Pareto前沿,但是GOMCDE算法在分布性方面明显要优于OMODE算法和MDEOO算法;NSGA-Ⅱ,MOEA/D,MODE-RMO算法无论是在收敛性还是分布性方面均稍弱于OMODE算法和MDEOO算法,更弱于GOMCDE算法.

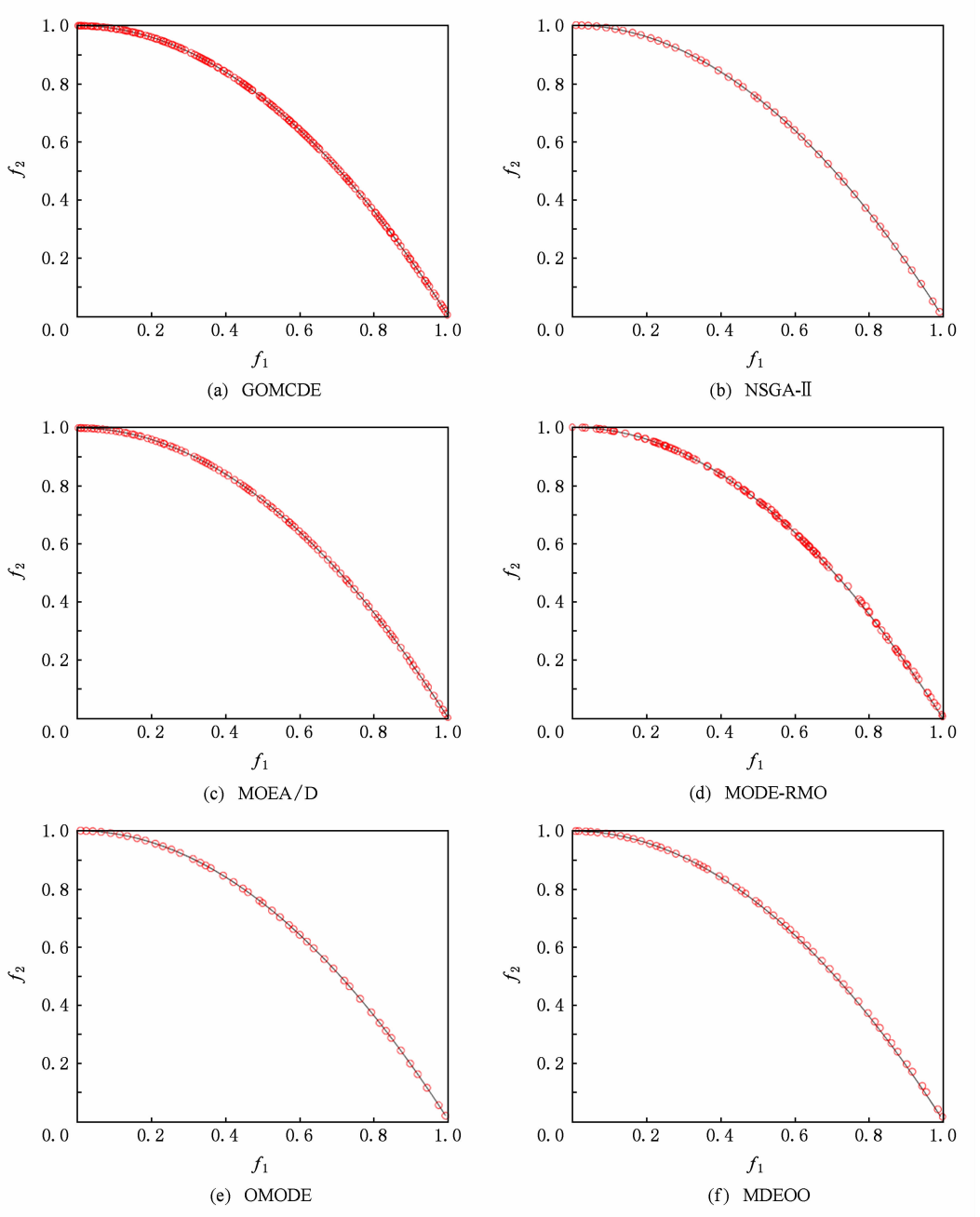


Fig. 4 Pareto front on ZDT2 for six algorithms.

图4 6种算法针对ZDT2测试问题的Pareto前沿

4 结束语

GOBL是一种泛化的OBL机制,把GOBL机制应用于进化算法被证明能够加快算法的收敛速度和提高算法的求解精度.然而GOBL机制却从来没有应用于多目标优化问题中,更没有用来解决多目标约束优化问题.针对此情况,本文结合GOBL机制,提出了GOMCDE算法.该算法首先利用GOBL机制生成变换种群;然后采用Pareto非支配排序、

拥挤距离排序和约束处理技术从初始种群和其变换种群中选取最优的个体组成新的初始种群继续迭代;最后为了加速收敛速度,采用 GOBL 机制代跳跃方法产生子代种群的变换种群,再次使用 Pareto 非支配排序、拥挤距离排序和约束处理技术从父代种群、子代种群及其变换种群中选择最优的个体组成新的子代种群继续迭代. 为了验证本文算法的优势,针对多目标测试函数集进行了实验测试. 测试结果表明,与其他相关算法相比,GOMCDE 算法具有更强的全局搜索性能、更快的收敛速度和更好的 Pareto 前沿.

参 考 文 献

[1] Zhou A, Qu B, Li H. Multiobjective evolutionary algorithms: A survey of the state of the art [J]. Swarm and Evolutionary Computation, 2011, 1(1): 32-49

[2] He Z, Yen G, Zhang J. Fuzzy-based Pareto optimality for many-objective evolutionary algorithms [J]. IEEE Trans on Evolutionary Computation, 2014, 18(2): 269-285

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

[4] Yen G G, Leong W F. A multiobjective particle swarm optimizer for constrained optimization [J]. Recent Algorithms and Applications in Swarm Intelligence Research, 2011, 2(1): 1-23

[5] Woldesenbet G, Yen G, Tessema G. Constraint handling in multiobjective evolutionary optimization [J]. IEEE Trans on Evolutionary Computation, 2009, 13(3): 514-525

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

[7] Zheng Jinhua, Liu Lei, Li Miqing, et al. Difference selection strategy for solving complex multi-objective problems [J]. Journal of Computer Research and Development, 2015, 52(9): 2123-2134 (in Chinese)
(郑金华, 刘磊, 李密青, 等. 差分选择策略在复杂多目标优化问题中的研究[J]. 计算机研究与发展, 2015, 52(9): 2123-2134)

[8] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition [J]. IEEE Trans on Evolutionary Computation, 2007, 11(6): 712-731

[9] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359

[10] Abbass H A, Sarker R, Newton C. PDE: A Pareto frontier differential evolution approach for multi-objective optimization problems [C] //Proc of IEEE Congress on Evolutionary Computation, vol 2. Piscataway, NJ: IEEE, 2001: 831-836

[11] Madavan N K. Multiobjective optimization using a Pareto differential evolution approach [C] //Proc of IEEE Congress on Evolutionary Computation, vol 2. Piscataway, NJ: IEEE, 2002: 1145-1150

[12] Xue F, Sanderson A C, Graves R J. Pareto-based multi-objective differential evolution [C] //Proc of IEEE Congress on Evolutionary Computation, vol 2. Piscataway, NJ: IEEE, 2003: 862-869

[13] Robic T, Filipic B. DEMO: Differential evolution for multiobjective optimization [G] //LNCS 3410: Proc of Evolutionary Multi-Criterion Optimization. Berlin: Springer, 2005: 520-533

[14] Al-Qunaieer F S, Tizhoosh H R, Rahnamayan S. Opposition based computing—A survey [C] //Proc of the 6th IEEE World Congress on Computational Intelligence. Piscataway, NJ: IEEE, 2010: 1-7

[15] Xu Q Z, Wang L, Wang N, et al. A review of opposition-based learning from 2005 to 2012 [J]. Engineering Applications of Artificial Intelligence, 2014, 29: 1-12

[16] Wang H. Opposition-based barebones particle swarm for constrained nonlinear optimization problems [J]. Mathematical Problems in Engineering, 2012, 2012: 1-12

[17] Wang H, Wu Z, Liu Y, et al. Space transformation search: A new evolutionary technique [C] //Proc of World Summit Genetic Evolutionary Computation. New York: ACM, 2009: 537-544

[18] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution algorithms [C] //Proc of IEEE Congress on Evolutionary Computation. Piscataway, NJ: IEEE, 2006: 2010-2017

[19] Ahandani M A, Alavi-Rad H. Opposition-based learning in the shuffled differential evolution algorithm [J]. Soft Computing, 2012, 16(8): 1303-1337

[20] Omran M G H. CODEQ: An effective metaheuristic for continuous global optimization [J]. International Journal of Metaheuristics, 2010, 1(2): 108-131

[21] Miao X F, Mu D J, Han X W, et al. A hybrid differential evolution for numerical optimization [C] //Proc of Int Conf on Biomedical Engineering and Informatics. Piscataway, NJ: IEEE, 2009: 1-5

[22] Wang H, Rahnamayan S, Wu Z J. Parallel differential evolution with self adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems [J]. Journal of Parallel Distributed Computing, 2013, 73(1): 62-73

[23] Omran M G H, Salman A. Constrained optimization using CODEQ [J]. Chaos, Solitons Fractals, 2009, 42(2): 662-668

[24] Peng L, Wang Y, Dai G. A novel opposition-based multi-objective differential evolution algorithm for multi-objective optimization [G] //LNCS 5370: Proc of Advances in Computation and Intelligence. Berlin: Springer, 2008: 162-170

[25] Dong N, Wang Y. Multiobjective differential evolution based on opposite operation [C] //Proc of Int Conf on Computational Intelligence and Security, vol 1. Piscataway, NJ: IEEE, 2009: 123-127

[26] Ray T, Tai K, Seow K C. An evolutionary algorithm for multiobjective optimization [J]. Engineering Optimization, 2001, 33(3): 399-424

[27] Wang Jianlin, Wu Jiahuan, Zhang Chaoran, et al. Constrained multi-objective particle swarm optimization algorithm based on self-adaptive evolutionary learning [J]. Control and Decision, 2014, 29(10): 1765-1770 (in Chinese)
(王建林,吴佳欢,张超然,等. 基于自适应进化学习的约束多目标粒子群优化算法[J]. 控制与决策, 2014, 29(10): 1765-1770)

[28] Binh T T, Korn U. MOBES: A multi-objective evolution strategy for constrained optimization problems [C] //Proc of the 3rd Int Conf on Genetic Algorithms. New York: ACM, 1997: 176-182

[29] Srinivas N, Deb K. Multi-objective function optimization using non-dominated sorting genetic algorithms [J]. Evolutionary Computation, 1994, 2(3): 221-248

[30] Tanaka M, Watanabe H, Furukawa Y, et al. GA-based decision support system for multi-criteria optimization [C] // Proc of Int Conf on Systems, Man and Cybernetics, vol 2. Piscataway, NJ: IEEE, 1995: 1556-1561

[31] Deb K, Pratap A, Meyarivan T. Constrained test problems for multi-objective evolutionary optimization [G] //LNCS 1993: Proc of Evolutionary Multi-Criterion Optimization. Berlin: Springer, 2001: 284-298

[32] Chen X, Du W, Qian F. Multi-objective differential evolution with ranking-based mutation operator and its application in chemical process optimization [J]. Chemometrics and Intelligent Laboratory Systems, 2014, 136: 85-96

[33] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results [J]. Evolutionary Computation, 2000, 8(2): 173-195

[34] Zitzler E, Thiele L, Laumanns M, et al. Performance assessment of multiobjective optimizers: An analysis and review [J]. IEEE Trans on Evolutionary Computation, 2003, 7(2): 117-132

[35] Fonseca C M, Knowles J D, Thiele L, et al. A tutorial on the performance assessment of stochastic multiobjective optimizers [G] //LNCS 3410: Proc of the 3rd Int Conf on Evolutionary Multi-Criterion Optimization, vol 216. Berlin: Springer 2005: 240-258

Wei Wenhong, born in 1977. Associate professor in Dongguan University of Technology. His main research interests include evolutionary algorithms, multi-objective optimization and machine learning.

Wang Jiahai, born in 1977. Associate professor in Sun Yat-sen University. His main research interests include computational intelligence and its applications.

Tao Ming, born in 1986. Associate professor in Dongguan University of Technology. His main research interests include intelligence computing and distributed computing.

Yuan Huaqiang, born in 1966. Professor in Dongguan University of Technology. His main research interests include intelligence computing.