一种改进的基于密度的多目标进化算法

王 鹏,张长胜,张 斌,刘婷婷
(东北大学信息科学与工程学院,辽宁沈阳 110819)

摘 要: 多目标密度驱动进化算法(MODdEA)利用非支配等级信息和分区密度信息求解多目标优化问题,该算法在与其他多目标进化算法的比较中有着出色的表现.在其基础上本文提出了一种改进的多目标进化算法MODdEA+,首先在该算法中基于搜索空间的分区机制提出了克隆操作,该操作不但能在进化前期增强算法的全局搜索能力,还能在进化后期提高算法的局部精化能力;其次引入一种基于Pareto信息表中个体支配及被支配信息的评价策略以使对信息表个体的排序结果更加精确;最后对变异操作进行了改进以降低出现不必要越界情况的概率.为验证改进算法的有效性,在对其进行分析的基础上针对多个测试问题将其与原算法进行了实验比较,结果表明改进算法的求解质量明显优于原算法.

关键词: 进化算法;密度驱动;克隆操作;粗适应度值;变异操作
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2016)05-1071-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.05.009

An Improved Density-Driven Multi-objective Evolutionary Algotithm

WANG Peng, ZHANG Chang-sheng, ZHANG Bin, LIU Ting-ting
(*College of Information Science & Engineering, Northeastern University, Shenyang, Liaoning* 110819, *China*)

Abstract: Multi-objective evolutionary algorithm that diversifies population by its density (MODdEA) solve multi-objective optimization problem according to the non-dominated sorting information and spatial density information, the algorithm has a good performance in the comparison with other multi-objective evolutionary algorithm. In this paper, we propose an improved multi-objective evolutionary algorithm MODdEA + based on MODdEA. Firstly, we propose a operator named clone operator based on the partition mechanism in search space, this operator could not only improve the global search capabilities in the early stage of evolution, but also enhance the local refinement capabilities in the late stage of evolution; secondly, we introduce a evaluation strategy which evaluate the individuals in Pareto information list based on the dominate and dominated information, this strategy provide a more accurate sorting result; finally, we improve the mutation operator in order to reduce the probability of overstep of the boundary. To demonstrate the effectiveness of the improved algorithm, we compare it with MODdEA on multiple testing problems, the experimental results show that the improved algorithm's solving quality is much better than the original algorithm's.

Key words: evolutionary algorithm; density-driven; clone operator; raw fitness; mutation operator

1 引言

最优化问题是工业生产和科学研究中主要的问题形式之一,当多个目标函数需要同时处理时,最优化问题称为多目标优化问题(MOPs).对于多目标优化问题,通常一个解对于某个目标来说可能较好,而对于其他目标来讲可能是较差的,因此多目标优化问题通常求解一个折中解的集合,该集合称为Pareto最优解集.

多目标进化算法(MOEA)在每一代进化过程中精炼种群的最优解来实现全局搜索,该类算法可以有效的完成对多目标优化问题的Pareto最优解集的搜索.自从1985年第1种多目标进化算法提出以来,MOEA已发展成为求解MOPs的主流方法之一;同时,MOEA也已成为进化算法领域最热门的研究方向之一.以NSGA-II[1],SPEA2[2],PAES[3],IBEA[4,5],MOEA/D[6]等为代表的MOEA算法在众多应用领域获得了广泛的应用.

收稿日期:2014-10-28;修回日期:2015-03-26;责任编辑:蓝红杰
基金项目:宁夏回族自治区自然科学基金(No. NZ13265);中央高校东北大学基本科研专项基金(No. N120804001, No. N120204003)

多目标密度驱动进化算法[7](MODdEA)克服了邻域假设,并可以有效的处理不连通问题(TYD-MOP).该算法将所有已生成过的解都存储在 BSP 树上,根据该树存储的对搜索空间的分区信息,经过计算可以得到搜索空间任一点的解密度信息;算法综合利用该密度信息配合解的非支配信息选择交叉个体,然后利用多样性变异算子和扩展算术交叉算子生成新的个体.实验证明相比于已有的 MOEA,MODdEA 不仅在处理不连通问题时有着出众的表现,同时在处理连通问题时也达到了高水准.

目前该算法主要在以下三个方面有待提高:

(1)算法在进化后期的局部精化能力有待改进.算法 MODdEA 的个体选择操作、变异操作、交叉操作都从不同角度加强了算法在进化期间的全局多样化能力,全局多样化的能力对于进化算法在进化过程中的全局搜索有着至关重要的意义.但是由于搜索资源的有限性,在进化过程的后期强调全局多样化的能力,将削弱算法的局部精化能力,局部精化能力的不足将直接影响算法求得解的精确度.因此有必要提高算法后期的局部精化能力.

(2)非支配排序算法考虑的信息不够全面.算法 MODdEA 采用的非支配排序算法的核心思想计算每个解 p 支配的解数 n_p,及一个该解支配的解的集合 S_p,递归的通过操作所有解的这两个变量计算出所有解的非支配等级.这种算法可以高效的求解出一个解集合所有解的非支配等级,但是在排序过程中该算法只考虑该解支配解的情况,并没有考虑有多少解支配该解.这种排序产生的结果并不能全面的反映解与解之间的支配与被支配的关系.

(3)变异操作可能出现不必要的越界.MODdEA 中的变异操作 DM 在选定变异维度 p_d 后,生成步长的规则如下:生成一个随机准步长,从准步长、上界-原值、原值-下界中选择一个最小值,作为步长;随后从值与步长之间生成一个高斯随机数作为变异维度上的最终值.这种设置可以使变异操作在全局搜索与局部精化间相互切换,保持算法在整个进化过程中的全局搜索能力.但是由于一旦生成的准步长过大,即使选择上下界作为步长也很容易出现越界的情况.

针对上述不足,本文进行三处改进,从而提出改进算法 MODdEA + :

(1)提出了一种克隆操作,并将其结合到该算法中.由于该操作以 BSP 树存储系统的分区机制为基础,所以能够在进化前期增强算法的全局搜索能力,在进化后期增强算法的局部精化能力,从而提高算法的求解精度.

(2)针对非支配排序算法考虑的信息不够全面问题,本文根据个体的粗适应度值(raw fitness)对进行排序.在计算每个个体的粗适应度值的过程中充分考虑了该个体支配与被支配的信息.因此这种方法所产生的排序结果可以全面的反映解与解之间的支配关系.

(3)针对变异操作可能出现不必要的越界,对变异操作进行改进,提出了一种新的越界处理策略.一旦随机生成的准步长越界,不再将上下界与原值的差作为备选步长,而是将越界的步长减去上下界与原值的差,使越界的后的替换步长减小,降低变异操作越界的概率.

2 问题描述

不失一般性,一个具有 n 个决策变量 m 个目标函数的多目标优化问题(MOP)可以定义为:

$$\min \boldsymbol{F}(x) = [f_1(x), \cdots, f_m(x)]^{\mathrm{T}}, \text{for all } x \in \boldsymbol{S} \subset \Re^n$$

其中,$\boldsymbol{S}$ 是 n 维决定空间(decision space);$\boldsymbol{F}: \boldsymbol{S} \rightarrow \boldsymbol{\Omega}$ 属于 $\Re^m$ 包含 m 个目标函数(objective problems);$\boldsymbol{\Omega}$ 是 m 维目标空间(objective space).MOP 的目标函数之间通常相互冲突,这种情况下往往不存在一个最优解满足所有的目标函数.因此,MOP 的最优解并不是一个解,而是一个解集,相关定义如下.

定义 1 Pareto 强支配:设 $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{\Omega}$,对于一个最小化问题,当且仅当 $u_i < v_i$ 对 $i = 1, 2, \cdots, m$,称 $\boldsymbol{u}$ Pareto 强支配 $\boldsymbol{v}$.

定义 2 Pareto 最优解:对于上述多目标优化问题的解集 P,对解集中的一点 $x_0 \in P$,如果 x_0 不被 P 中的其他点 $x \in P$ 所强支配的话,则称 x_0 为 P 的 Pareto 最优解(Pareto optimal solution).

定义 3 Pareto 最优解集:所有 Pareto 最优解的集合称为解集 P 的 Pareto 最优解集(Pareto set,PS).

定义 4 Pareto 最优向量:解集 P 的 Pareto 最优解集在目标空间的映射称为解集 P 的 Pareto 最优向量(Pareto optimal vector).

定义 5 Pareto 最优前沿:所有 Pareto 最优向量的集合称为解集 P 的 Pareto 最优前沿(Pareto front,PF).

3 相关工作

MOEA 的目标是找到一个对应的解向量集接近、密集并且均匀的分布于实际 Pareto 最优前沿的逼近前沿.这就要求 MOEA 在进化过程中既需要不断的精化已存在的优秀解,同时还必须在搜索空间中搜索新的解.受限于进化次数,好的 MOEA 应该在精化已有解与搜索新解这两项工作之间得到理想的平衡.为了实现这一平衡,大多数 MOEA 都在个体选择的过程中兼顾局部收敛及全局多样化.

根据 MOEA 采用的基本思想的不同,大致可以分

为以下四类:基于 Pareto 占优关系的 MOEA;基于评估指标的 MOEA;基于分解技术的 MOEA 和基于运行过程中历史信息的 MOEA 算法.

4　改进的 MODdEA

针对算法 MODdEA 的三处不足,本节提出的三处改进,并提出改进算法. 本小节将详述这三处改进与算法 MODdEA + 的算法描述.

4.1　变异克隆算子

算法 MODdEA 所提到的多样变异算子和扩展的算术交叉算子,在进化过程中都在全局搜索与局部精化之间随机变动. 但是在整个的进化的过程中,尤其是进化后期,局部精化比全局搜索更能提高求解精度. 本文在使用原有交叉变异操作的基础上,提出一种新的操作称为克隆操作.

该操作以 BSP 树结构存储的搜索空间分区信息为基础,每轮进化在种群的优秀个体的子区域内随机生成一个新的个体. 由于搜索空间整体的超体积不变,BSP 树结构的分区数量随着进化的进行逐渐增加,因此分区的平均超体积在进化过程中由大到小递减. 进化前期,在较大的区域进行克隆操作可以增强算法的全局搜索能力;进化后期,在较小的区域进行克隆操作可以提高搜索的局部精化能力.

具体操作如下:对每一个本代优秀个体 $m = population\ size$,p_i属于 $P = \{p_1, p_2, \cdots, p_m\}$,在交叉变异生成新一代的同时. 在 BSP 树存储系统搜索 p_i的区域,并在该区域内随机生成一个 p_i的克隆解 c_i,得到 $\boldsymbol{P}$ 的克隆解集 $\boldsymbol{C} = \{c_1, c_2, \cdots, c_n\}$,并将插入 BSP 树存储系统. 将 C 与 N 一同加入到 PIL 中,更新 PIL.

具体的变异克隆算子算法描述如算法 1.

算法 1　Clone

Input:(1) search space $S = \prod_{i=1}^{n} [L_i, U_i] \subset \Re^n$,
(2) parent individual $p = [p_1, p_2, \cdots, p_n] \in S$,
(3) the BSP tree T,
Output:Cloning Mutation individual c of p
Begin
Search $h(p)$:the sub-region of p from T
For $i = 1$ to n
　c_i = Random($h(p)[i][0], h(p)[i][1]$)
Next
$\boldsymbol{c} = [c_1, c_2, \cdots, c_n]$
End

其中函数 Random(a,b)是在实数 a、b 之间生成一个随机数. 实际上,克隆算子所做的对上一代所选出的种群进行操作:对每一个种群中的个体,从 BSP 树系统中搜索该个体所在的子区域,并在该区域内随机生成一个新的个体,并将该个体插入到 BSP 树系统. 由于在进化过程中,BSP 树系统中的子区域由少到多,整个的区域的超体积是不变化的,易见在进化过程中子区域的超体积是一个由小到大的过程,克隆操作所做的操作针对本代种群所做的操作在进化前期由于子区域的超体积相对较大,使用克隆算子可以增强算法的全局搜索能力;而在进化后期,由于子区域已经变小,对优秀解进行克隆操作可以精炼这些优秀解,增加算法的局部精化能力.

4.2　PIL 的支配关系排序方法

PIL 表结构是算法 MODdEA + 维护的一个外部集合,该表保存进化过程中产生的优秀个体,避免个体选择的随机性丢失这些个体. 在进化过程中每当有新的个体产生时,算法 MODdEA + 都要都要将这些个体并入 PIL 中然后与 PIL 中原有的个体重新根据支配关系排序. 上文提到算法 MODdEA 采用的非支配排序方法考虑的信息不够全面,下面将介绍算法 MODdEA + 采用的支配关系排序方法.

排序方法如下:

(1)计算所有解支配的个体数,即力度(strength)值,公式如下:

$$strength(i) = |\{j | j \in PIL \wedge i > j\}|$$

其中,>表示支配关系;

(2)根据力度值,计算所有解的粗适应度值,如下:

$$raw\ fitness(i) = \sum_{j \in PIL, j > i} strength(j)$$

(3)按 $raw\ fitness(i)$ 从小到大对所有个体排序,值相同的为一级.

易见,与算法 MODdEA 采用的非支配排序算法不同,算法 MODdEA + 根据个体的粗适应度值进行的排序. 个体的粗适应度值不仅考虑了个体支配个体的数量信息,同时也考虑了支配该个体的个体的数量信息,全面的考虑这两种信息可以更全面的产生排序结果. 根据粗适应度值排序产生的排序结果可以更全面的反映个体间支配的优先关系.

4.3　改进的多样变异算子

针对变异操作可能出现不必要的越界问题,本小节对变异操作如下改进:一旦随机生成的准步长越界,不再将上下界与原值的差作为备选步长,而是将越界的步长减去上下界与原值的差,使越界的步长减小.

处理过程如下:

(1)首先给定父代 $p = [p_1, p_2, \cdots, p_n]$;将子代 o 初始化为 p,即 $o = p$.

(2)从 $\{1, 2, \cdots, n\}$ 中随机生成一个变异维度 d;

(3)从 $[0, U_d - L_d]$ 随机生成步长标准差 r,当 $r > \max(p_d - L_d, U_d - p_d)$ 时,$r = r\text{-}\max(p_d - L_d, U_d - p_d)$.

(4)最后在 p_d与 r 之间生成一个高斯随机数,将子

代 d 维度替换为该随机数,完成多样变异.

改进后的多样化变异算法如算法 2.

算法 2 Diversified Mutation

Input:(1) search space $S=\prod_{i=1}^{n}[L_i,U_i]\subset\Re^n$

(2) parent individual $p=[p_1,p_2,\cdots,p_n]\in S$

Output:Offspring individual o of p under diversified mutation bound

Begin

Select the mutation dimension

$d:=$ Random ($\{1,2,\cdots,n\}$)

$r=$ Random ($(0,U_d-L_d)$)

If $r>\max(p_d-L_d,U_d-p_d)$

then $r=r$-$\max(p_d-L_d,U_d-p_d)$

$m:=$ GaussianRandom(p_d,r)

while ($m\notin[L_d,U_d]$) {

then $r=r$-$\max(p_d-L_d,U_d-p_d)$

$m:=$ GaussianRandom(p_d,r)}

$o:=[o_1,o_2,\cdots,o_n]$ where

$$o_j=\begin{cases}m, & \text{if } j=d\\ p_j, & \text{otherwise}\end{cases}$$

End

其中函数 GaussianRandom(a,b)是在实数 a、b 之间生成一个高斯随机数. 尽管只是在步长越界的替换策略做了改进,但是这种处理方式首先保证了变异操作在进化过程中仍然可以在全局搜索与局部精化随机切换;其次,易见算法只会在准步长越界之后出现步长越界情况,降低准步长越界后替换准步长的上界可以有效的降低步长越界的概率,减少不必要的资源浪费.

4.4 主循环

算法 MODdEA + 主要由进化算法模块和存储器模块两部分组成.

进化算法模块包括:多样化变异操作(DM)、扩展的算术交叉操作(EAX)、克隆操作(Clone)和个体选择操作(SDPD).

(1)经典的变异算子的步长由大到小,使算法在进化过程中从前期的扩张到后期的收敛;DM 在一定范围内随机生成步长,使在算法进化过程中在扩张与收敛之间随机切换,算法在进化后期收敛的同时兼顾扩张.

(2)经典的交叉算子的交叉权重从 0 到 1 取值,使子代相较于父代越来越收敛,影响了算法后期的收敛能力;EAX 的交叉权重从 -1 到 2 取值,使算法在进化过程中都可以在扩张和收敛之间相互切换.

(3)经典的个体选择操作仅在目标空间根据当前代的解信息估计解密度,并且一旦 Pareto 最优解集超过种群容量就将舍弃一部分,浪费这部分搜索到的优秀解;SDPD 根据所有以生成解得信息在搜索空间估计解密度,并且将超过种群容量的解存储在 PIL 中,供下一代继续使用.

(4)DC 的作用是在进化过程的前期增强扩张,后期针对已选出的优秀解,加强收敛能力.

存储器模块包括:BSP 树结构和 PIL 表结构. BSP 树结构存储着进化过程中已生成的所有解的空间分割信息,根据空间分割信息中可以求得决定空间任意一点的解密度. PIL 表结构存储着进化过程中已经生成的优秀解,并按非支配等级排序,该结构可以保证生成的优秀解不会因为选择配对池(mating pool)的随机性而丢失.

本算法的处理过程如下:

(1)进行一系列的初始化工作:首先,随机生成一个种群 P;然后,将 BSP 树初始化为一棵只含根节点的树;最后,将 PIL 初始化为一个空表;(2)算法调用用交叉变异算子生成子代;(3)算法调用用克隆算子生成克隆代;(4)将子代和克隆代存入 BSP 树,并且更新 PIL;(5)算法调用 ISDPD 从 PIL 中选取个解;(6)重复 2、3、4 和 5 直至迭代次数足够.

算法 3 Main Loop

Input:1) MOP $F=\{f_i(.)\}$

2) search space S

3) population size μ

4) number of generations N_g

Output:Pareto-optimal set found by MODdEA +

Begin

Initialize the current population $P=\{x_i\}_{i=1,2,\cdots,\mu}$ where $x_i\in S$

Initialize BSP tree T to consist of a root node only

Pre-set PIL Ψ to an empty list

For $i:=1$ to μ

BSPTreeNodeInsert(x_i,T)

Next i

$\Psi:=P$

For $g:=2$ to N_g

For $i:=1$ to μ

$m_i:=$ DiversifiedMutation(S,x_i)

Next i

For $i:=1$ to μ

$a:=$ Random($\{m_i\}$)

$b:=$ Random($\{m_i|m_i!=a\}$)

$n_i=$ ExtendedArithmeticCrossover($S,\{a,b\}$)

Next i

For $i:=1$ to μ

$c_i:=$ Clone (S,x_i)

Next i

For $i:=1$ to μ

BSPTreeNodeInsert(n_i,T)

BSPTreeNodeInsert(c_i,T)

Next i

$\Psi:=$ PILUpdate($\Psi\cup N\cup C$)

$P:=$ SDPD(Ψ,T,μ)

Next g

End

其中函数 BSPTreeNodeInsert(x_i,T)是将个体 x_i 插入到 BSP 树 T 中,并为该个体划分出新区域;函数 ExtendedArithmeticCrossover($S,\{a,b\}$)是在搜索空间 S

中,对个体 a,b 进行算术扩展交叉操作;函数 PILUpdate($\Psi \cup N \cup C$)是将集合 $\Psi \cup N \cup C$ 加入到 PIL 表中,并对表进行更新和如果成员个数超过规定容量则调用表的截断操作;函数 SDPD(Ψ,T,μ)的功能是根据 BSP 树 T 中的信息从种群 Ψ 中选出 μ 个个体. 与本文没有详述的部分与算法 MODdEA 一致,详细内容参考文献[7].

4.5　复杂度分析

计算算法 MODdEA + 每一轮迭代的复杂度需要考虑以下基本操作:

(1)将产生的 u 个后代个体插入 BSP 树系统;(2)根据产生的 u 个后代个体更新 PIL 表系统;(3)在运行 SDPD 过程中搜索 PIL 表中的子区域;(4)SDPD 的概率选择模式;(5) 运行克隆算子过程中搜索上一代种群的子区域;

设已经产生的解的数量为 n_e,种群容量为 u,PIL 表长为 n_p,目标数为 M. 基本操作 1 的算法复杂度为 $O(u\log(n_e))$.

在基本操作 1,对所有个体的粗适应度值排序,u 个个体的目标向量的为得到非支配等级所做的比较是 u^2M. 在基本操作 2 中,u 个解每个解都要与 PIL 表中的解进行一次比较,额外需要 $n_p u^2 M$. 基本操作 2 的时间复杂度 $O(u(u+n_p)M)$.

因为后代个体的子区域在基本操作 1 中已经获得,所以基本操作 3 与基本操作 5 不需要额外的操作.

基本操作4 的平均时间复杂度为 $O(u\log(n_p))$. 一种合理的估计是设 $n_p \approx 10u$. 因此,基本操作 2 和基本操作4 的时间复杂度分别是 $O(11u^2M)$ 和 $O((u\log(10u))$. 因此算法 MODdEA + 的时间复杂度为 $O(u\log(n_e)+u^2M)$.

5　仿真实验

由于算法 MODdEA 与已有的大部分进化算法已经进行了实验比较分析,结果表明在绝大部分测试问题上算法 MODdEA 明显优于其他相比较的算法[7]. 因此,本文在实验部分只将提出的算法与算法 MODdEA 进行比较分析. 为了评价算法的有效性,本文采用 IGD[10] 作为评估指标. IGD(P_A,P')可以计算所得解集 P_A 与最优解集 P'之间的距离,从而反映两种算法的求解效果.

$$\mathrm{IGD}(P_A,P') = \frac{1}{|P'|}\sum_{v\in P'} d(v,P_A)$$

测试的两种算法均为少参数算法,可设置的参数只有种群规模(population size),在测试中两个算法的种群规模与文献[7]相同均设置为 10.

5.1　测试问题

在实验过程中,本文采用3 类问题作将提出的算法与算法 MODdEA 进行比较实验:文献[1]中提出的 TDY1-TDY6;文献[8]中提出的 ZDT1-ZDT4,ZDT6;文献[9]中提出的 UcP1-UcP10. 从中选取有代表性的 10 个问题作为本文的测试问题:TYD01,TYD02,TYD05,ZDT1,ZDT2,ZDT6,CEC02,CEC03,CEC04,CEC05.

为与算法 MODdEA 保持一致,本文的关于问题的参数设置与终止条件与文献[7]相同. TYD01,TYD02,TYD05 问题的维度为 30,适应度进化次数设定为 30000. ZDT1,ZDT2 问题的维度为 30,ZDT6 问题的维度为 10,适应度进化次数都设定为 30000. 每一个问题都独立运行 100 次,统计其运行结果. CEC02,CEC03,CEC04,CEC05 问题的维度为 30,适应度进化次数设定为 300000. 每一个问题都独立运行 30 次,统计其运行结果.

5.2　IGD 值比较

将两个算法算法在 3 类的 10 个测试问题中进行实验测试,对测试结果的 IGD 值进行比较. 表 1 给出了两个算法在所有问题上所得实验结果的最大值、最小值和均值,加粗字体显示的是两算法中较小的数据. 图 1 给出了 10 个问题中具有代表性的 6 个问题(每类 2 个)实验结果的盒子图.

表 1　两种算法针对 10 个问题 IGD 值对比(最大/最小/均值(标准差))

问题	MODdEA	MODdEA +
ZDT1	4.266e-3/2.912e-3/3.603e-3(9.656e-4)	**4.711e-3/2.503e-3/3.407e-3(7.589e-4)**
ZDT2	3.972e-3/**1.667e-3**/2.836e-3(8.036e-4)	**3.375e-3**/1.856e-3/**2.468e-3(5.753e-4)**
ZDT6	**2.560e-4**/1.955e-4/2.200e-4(4.026e-5)	2.708e-4/**1.409e-4/2.020e-4(2.098e-5)**
CEC02	8.649e-3/6.512e-3/7.189e-3(4.666e-3)	**7.860e-3/5.901e-3/7.015e-3(4.423e-3)**
CEC03	7.777e-2/6.367e-2/7.083e-2(5.467e-3)	**6.943e-2/5.678e-2/6.321e-2(3.691e-3)**
CEC04	**3.163e-2**/2.815e-2/3.016e-2(**5.326e-3**)	3.239e-2/**2.712e-2/2.947e-2(6.020e-3)**
CEC05	1.786e-1/1.218e-1/1.480e-1(**1.864e-2**)	**1.656e-1/1.053e-1/1.366e-1(2.209e-2)**
TYD01	**5.459e-3**/1.614e-3/2.892e-3(3.206e-3)	5.511e-3/**1.188e-3/2.491e-3(3.047e-3)**
TYD02	7.279e-2/4.486e-2/5.611e-2(8.477e-3)	**5.651e-2/3.454e-2/4.179e-2(7.170e-3)**
TYD05	1.932e-2/7.183e-3/1.178e-2(4.489e-3)	**1.411e-2/3.877e-3/8.130e-3(3.131e-3)**

从表中易见算法 MODdEA + 在是 10 个测试问题中的 IGD 平均值都好于算法 MODdEA. 因此认为算法 MODdEA + 比算法 MODdEA 有更好的收敛能力,可以看做是增加了克隆算子的结果. 在问题 ZDT1、CEC02、CEC03、CEC05、TYD02 和 TYD05 中算法 MODdEA + 的 IGD 最大值、最小值和平均值均小于算法 MODdEA,其他 4 个问题也至少有两个值小于,因此可以认为算法 MODdEA + 相比算法 MODdEA 有更好的稳定性. 同样如表 2 所示,算法 MODdEA + 所求解集的 IGD 值的各统计量比算法 MODdEA 的相应值更优,所以认为算法 MODdEA + 比算法 MODdEA 更有效. 两个算法的稳定性在图 1 中得到进一步的显示,图中选取了 10 个问题中具有代表性的 3 类 6 个问题,明确显示了两个算法的 IGD 值分布图. 它给出了两种算法的 IGD 值分布,包括最小观察值、低四分位值、中位值、高四分位值、

最大观察值和平均值. 易见算法 MODdEA + 在所显示问题上 IGD 值的显著优越性. 产生这种优势的原因是:克隆算子的加入增强了算法后期的收敛能力与前期的探索能力,整体上提高了算法的寻优能力;更新的非支配排序策略提高了算法的求解效率.

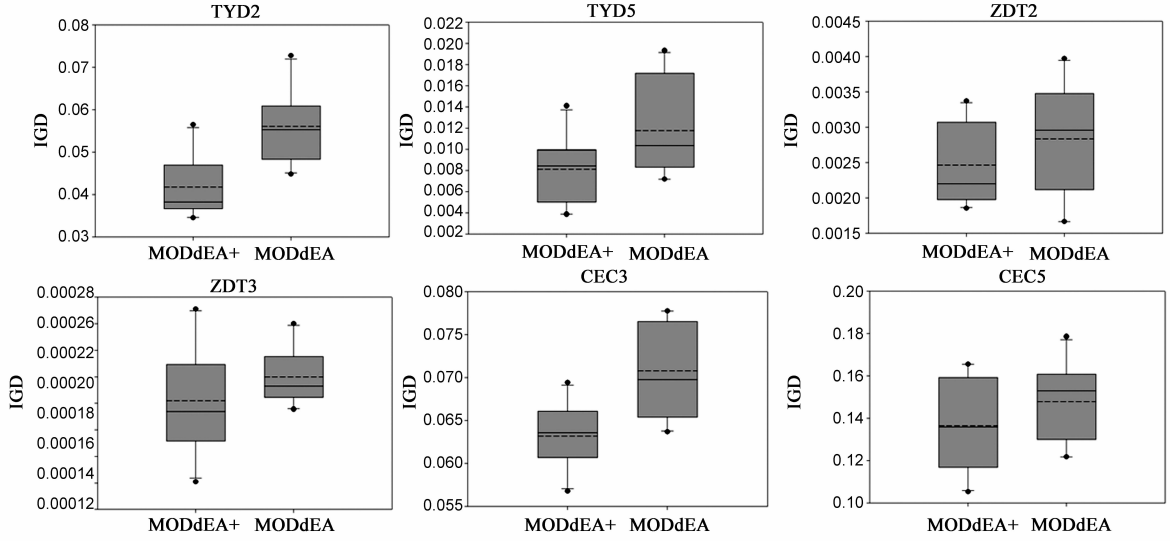

图1 IGD值分布图

5.3 收敛性比较

收敛性是进化算法的一个重要特征,当算法收敛后其各个性能指标都将趋于稳定. 随着迭代次数的积累到一定程度,算法都将会收敛于某一处. 为了客观地评价本文中提出的新算法在收敛后的各种性能,本小节通过 IGD 指标值来分析算法的收敛性,最终获取新

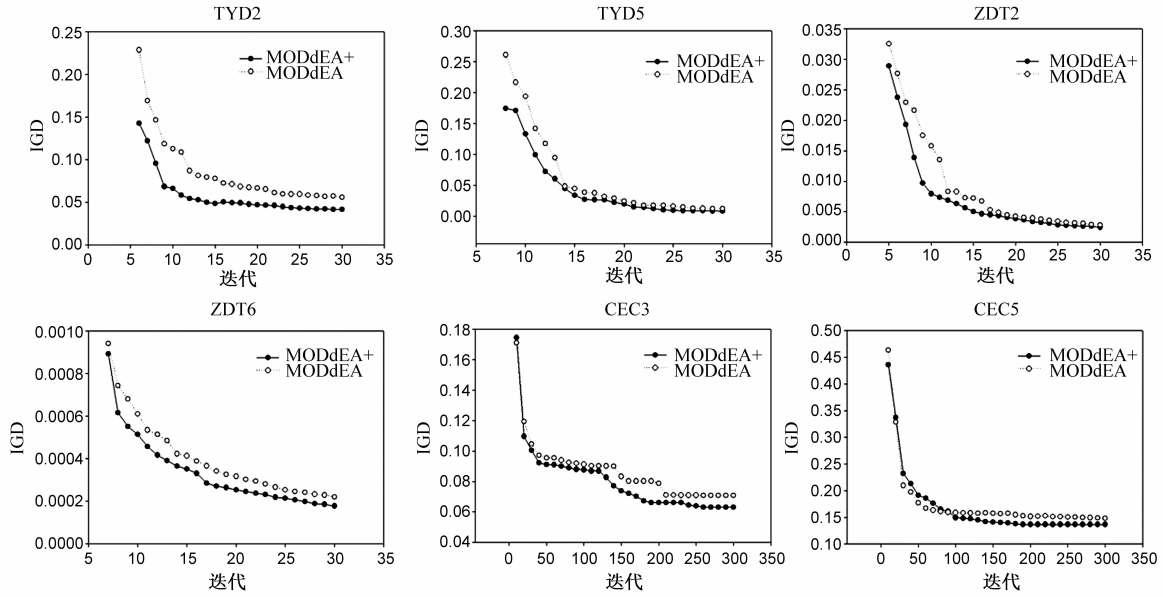

图2 两种算法在不同时刻求解测试问题的IGD值

算法在代表测试用例上的最大迭代次数(或最大评估次数).在收敛性分析中,本小节选取 2 个算法算法对于 3 类的 10 个测试问题中具有代表性的 6 个问题(每类 2 个)进行收敛性比较和分析.在迭代过程中每隔一定的代数,计算其运行结果的 IGD 值.图 2 给出了两种算法在规定的迭代次数内 IGD 值的变化情况.

6 个问题中算法 MODdEA + 均比算法 MODdEA 收敛到了一个更低的 IGD 值上,说明算法 MODdEA + 有更好的收敛能力.除问题 ZDT6 外,两种算法均在 25(* 1000 迭代)左右收敛,问题 ZDT6 的 IGD 值已经降到 1.0e-4 的数量级上,可以视为已经收敛.除问题 CEC5 之外的 5 个问题,算法 MODdEA + 的 IGD 值基本一直处于优势地位.因此可以认为算法 MODdEA + 相比算法 MODdEA 更加有效.主要原因是算法 MODdEA + 在算法 MODdEA 的基础上加入克隆算子,提高了收敛能力.

6　结语

针对 MODdEA 存在的三处不足,本文针对 MODdEA 进行了三处改进,并提算法 MODdEA + .针对算法后期缺乏收敛能力的问题,提出一种新的算子称为变异克隆算子.针对非支配排序算法求解效率问题,引入的粗适应度值,求出所有解的粗适应度值后,按粗适应度值排序.针对变异操作可能出现的越界问题,修改参数,降低出现越界的可能性.根据实验结果可以发现,相比算法 MODdEA,算法 MODdEA + 有着更好的求解精度,更快速的收敛到更精确的位置.由 BSP 树结构带来的时间空间资源的消耗过多,是下一步要解决的问题.

参考文献

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

[2] Zitzler E,Laumanns M,Thiele L. SPEA2:Improving the strength Pareto evolutionary algorithm for multiobjective optimization[A]. Proc of the Evolutionary Methods for Design,Optimisation and Control. Athens:International Center for Numerical Methods in Engineering[C]. Siitzerland:Technical report TIK-Report,2002. 95 - 100.

[3] Knowles J D,Corne D W. Approximating the nondominated front using the Pareto archived evolution strategy[J]. Evolutionary Computation,2000,8(2):149 - 172.

[4] Zitzler E,Künzli S. Indicator-based selection in multiobjective search[A]. Parallel Problem Solving from Nature-PPSN VIII[C]. Berlin Heidelberg:Springer. 2004. 832 - 842.

[5] Bader J,Zitzler E. HypE:An algorithm for fast hypervolume-based many-objective optimization[J]. Evolutionary Computation,2011,19(1):45 - 76.

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

[7] Chow C K,Yuen S Y. A multiobjective evolutionary algorithm that diversifies population by its density[J]. Evolutionary Computation,IEEE Transactions on,2012,16(2):149 - 172.

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

[9] Zhang Q,Zhou A,Zhao S,et al. Multiobjective optimization test instances for the CEC 2009 special session and competition[R]. University of Essex,Colchester,UK and Nanyang Technological University,2008. 1 - 30.

[10] Bandyopadhyay S,Bhattacharya R. NSGA-II based multiobjective evolutionary algorithm for a multi-objective supply chain problem[A]. Advances in Engineering,Science and Management (ICAESM),2012 International Conference on. IEEE[C]. Nagapattinam,Tamil Nadu:IEEE,2012. 126 - 130.

[11] He J,Mitavskiy B,Zhou Y. A theoretical assessment of solution quality in evolutionary algorithms for the knapsack problem[A]. Evolutionary Computation (CEC) 2014 IEEE Congress on[C]. Beijing:IEEE,2014. 141 - 148.

[12] Chow C K,Yuen S Y. A dynamic history-driven evolutionary algorithm[A]. Evolutionary Computation (CEC),2014 IEEE Congress on. IEEE. [C]. Beijing:IEEE. 2014. 1558 - 1564.

[13] Wang B,Xu H,Yuan Y. Quantum-inspired evolutionary algorithm with linkage learning[A]. Evolutionary Computation (CEC),2014 IEEE Congress on[C]. Beijing:IEEE,2014. 2467 - 2474.

作者简介

王　鹏　男,1987 年生于山东烟台.东北大学计算机应用技术专业博士研究生.研究方向为服务计算、人工智能算法.

张长胜　男,1980 年生于吉林长春.东北大学信息科学与工程学院副教授、硕士生导师.主要研究方向为智能信息处理.

张　斌(通信作者)　男,1964 年出生,东北大学信息科学与工程学院教授、博士生导师.主要研究方向为服务计算.

E-mail:zhangbin@ ise. neu. edu. cn