高技术通讯 2009 年 第 19 卷 第 8 期:825 ~ 832

doi:10.3772/j.issn.1002-0470.2009.08.010

多目标进化算法的一种基于生成树的分布性维护方法①

李密青② 郑金华

(湘潭大学信息工程学院 湘潭 411105)

摘 要 针对多目标进化算法的种群维护和运行效率相矛盾的问题,提出了一种基于生成树的分布性维护方法,即对整个种群构造一棵生成树,定义一种密度估计指标——树聚集距离,并结合树中的最短树枝和个体度数对种群进行维护。由于树聚集距离和度数具有动态性,每移出一个个体,种群中与之相连个体的信息都会发生相应的变化,因而可即时反映出种群的分布情况。与三个著名的算法 NSGA-Ⅱ、SPEA2 和 C-NSGA-Ⅱ的比较实验表明,该方法能在得到良好分布性解集的同时,能以较快的速度对种群进行维护,具有较好的时间效率。

关键词 多目标优化,进化算法,分布性维护,生成树,种群

0 引 言

进化算法(evolutionary algorithm, EA)已被证明是一种解决多目标优化问题(multi-objective optimization problem, MOP)的有效方法[1]。用进化算法解决多目标优化问题的算法称为多目标进化算法(multi-objective evolutionary algorithm, MOEA)。学者们已提出了多种有效的多目标进化算法[2],具有代表性的有 Deb 等提出的 NSGA-II[3], Knowles 等提出的 AGA[4], Zitzler 和 Thiele 提出的 SPEA2[5]等。

多目标优化解集的三个基本评价指标为收敛性、分布性和运行时间。其中分布性是指解集分布的均匀程度和广泛程度。如何找寻一种能够在较短的时间内求得一个分布性较好的解集的算法是多目标进化研究的重要课题。NSGA-II[3]用聚集距离的方法对种群进行维护,拥有很快的速度,其时间复杂度为 $O(MN\log N)$, 其中 N 为种群规模, M 为目标数;SPEA[1]用聚类的方法对种群进行维护,实验表明在三维以上问题上解集的分布性要好于 NSGA-II,但时间复杂度达到 $O(MN^3)$。最近研究者们又提出很多维护方法, Kukkonen 和 Deb[6]提出一种改进的聚集距离方法,其时间复杂度同样为 $O(MN\log N)$,该方法在二维上取得很好的效果,但随着目标维数的增加,分布性能急剧下降; Kuang 和 Zheng[7]提出一种用极坐标对种群进行维护的方法,时间复杂度为 $O(MN^2)$, 分布结果优于 NSGA-II,但通常只适合于 Pareto 面为球面的测试问题; Kukkonen 等[8]用主成分分析法(PCA)进行分布性维护,时间复杂度同样为 $O(MN\log N)$, 但由于只投影到第一主成分方向,结果并不十分准确,通常只适应于三维以下的目标函数。另一方面,一些能很好维护分布性的方法具有较高的时间复杂度。SPEA2[5]用第 k 小距离对种群进行维护,取得了很好的结果,但时间复杂度为 $O(MN^2\log N)$; MST-MOEA[9]用最小生成树对种群进行维护,解集拥有很好的均匀性和广泛性,但时间复杂度可达 $O(N^3)$。

本文提出了一种利用生成树对种群进行快速维护的方法,定义了一种密度估计指标——树聚集距离,并结合生成树的最短树枝和个体度数对种群进行维护,且通过多个函数实验,测试和讨论了算法的性能。

1 基本概念

最小化与最大化问题可以相互转化,因此,仅以最小化多目标问题为研究对象。多目标优化问题(MOP)的一般描述为:

给定决策向量 $X = (x_1, x_2, \cdots, x_n)$, 它满足下列约束:

① 国家自然科学基金(60773047),863 计划(2001AA114060)和湖南省自然科学基金(05JJ30125)资助项目。
② 男,1981 年生,硕士,助教;研究方向:多目标进化算法;联系人,E-mail: limit1008@126.com
(收稿日期:2008-06-10)

$$g_i(X) \geqslant 0 \quad i = 1,2,\cdots,k \tag{1}$$

$$h_i(X) = 0 \quad i = 1,2,\cdots,l \tag{2}$$

设有 r 个优化目标,且这 r 个优化目标是相互冲突的,优化目标可表示为

$$\vec{f}(X) = (f_1(X), f_2(X), \cdots, f_r(X))$$

寻求 $X^* = (x_1^*, x_2^*, \cdots, x_n^*)$, 使 $\vec{f}(X^*)$ 在满足约束(1)和(2)的同时达到最小。MOEA 经常用到如下几个基本概念:

定义 1 个体的 Pareto 支配关系。设 p 和 q 是进化群体 Pop 中的任意两个不同的个体,我们称 p 支配(dominate) q,则必须满足下列二个条件:

Ⅰ 对所有的子目标,p 不比 q 差,即 $f_k(p) \leqslant f_k(q)(k = 1,2,\cdots,r)$;

Ⅱ 至少存在一个子目标,使 p 比 q 好。即 $\exists l \in \{1,2,\cdots,r\}$, 使 $f_l(p) < f_l(q)$。其中 r 为子目标的数量。此时称 p 为非支配的(non-dominated), q 为被支配的(dominated),表示为 $p \succ q$, 其中"$\succ$"是支配关系(dominate relation)。

定义 2 Pareto 最优解。给定一个多目标优化问题 $\mathrm{Min}\vec{f}(X)$, 称 $X^* \in \Omega$ 是最优解,若 $\forall X \in \Omega$, 则或者满足 $\bigwedge_{i \in I}(f_i(X) = f_i(X^*))$, 或者至少存在一个 $j \in I$, $I = \{1,2,\cdots,r\}$, 使 $f_j(X) > f_j(X^*)$。其中 Ω 是满足式(1)和式(2)的可行解集,即:

$$\Omega = \{X \in R^n \mid g_i(X) \geqslant 0,\ h_i(X) = 0;\ (i = 1,2,\cdots,k; j = 1,2,\cdots,l)\}$$

定义 3 Pareto 最优边界(前沿)。给定一个多目标优化问题 $\mathrm{Min}\vec{f}(X)$ 和它的最优解集 $\{X^*\}$, 它的 Pareto 最优面定义为

$$PF^* = \{\vec{f}(X) = (f_1(X), f_2(X), \cdots, f_r(X)) \mid X \in \{X^*\}\} \tag{3}$$

定义 4 非支配集(非劣集)。设有解集 P, P 中的个体 q 不被任何其它个体支配,则 q 是 P 中的非支配个体; P 的非支配个体构成的子集称为 P 的非支配集 $NDset$, $NDset = \{q \mid q \in P$ 且 $\bar{\exists} p \in P$, 使 $p \succ q\}$。

2 基于生成树的分布性维护方法

2.1 树聚集距离

在介绍维护方法之前,我们引进一种密度估计指标——树聚集距离。

定义 5 设 T 为一连接种群 P 中所有个体的生成树。对于 P 的任一个体 i, 设连接它的边为 l_1, $l_2, \cdots, l_r$, 则定义 i 的树聚集距离为

$$T(i) = (\sum_{j=1}^{r} l_j)/r \tag{4}$$

2.2 算法

我们用树聚集距离结合生成树的其它信息对种群进行维护,算法的具体流程如下:

参数设置: Q——待维护的非支配集; N——种群规模。

步骤 1: 计算 Q 中所有个体之间的欧氏距离,并根据欧氏距离生成一棵最小生成树 T,同时记录下每个个体的树聚集距离和度数。

步骤 2: 找出 T 中最短的树枝 $L_{i,j}$, i, j 为树枝 $L_{i,j}$ 的端点。

步骤 3: 若 $d_i = d_j$,则转步骤 4;若 $d_i > d_j$,淘汰 i,反之淘汰 j,转步骤 5,其中 d_i 为结点 i 的度数。

步骤 4: 比较个体 i, j 的树聚集距离,若 $T(i) > T(j)$,淘汰 j,反之淘汰 i。

步骤 5: $|Q| = |Q| - 1$,若 $|Q| = N$,结束算法。

步骤 6: 设 i 是被淘汰的个体, i_0, $i_1, \cdots, i_k$ 为原树中与个体 i 相连的个体,显然当 i 被删除后原树将变为由 k 棵树组成的森林。现在对 i_0, $i_1, \cdots, i_k$ 生成一个最小生成树。此时,整个种群恢复为树 T'。

步骤 7: $T = T'$,并更新树中的树聚集距离和个体度数,转步骤 2。

图 1(a)-(f)为种群维护的一个例子,图 1 中非支配集 $|Q| = 8$, 种群规模 $N = 3$。A,B,C,D,E,F,G,H 为三维空间中的非支配个体。图 1(a)首先对非支配集生成一棵最小生成树,然后找出最短树枝 L_{AB},由于 $d_A = 1 < d_B = 2$,根据步骤 3,淘汰 B,对连接 B 的个体 A,C 生成最小生成树得图 1(b)(如果被删除的个体的度数为 2,生成最小生成树等于直接相连),相应调整 A,C 的树聚集距离和度数;然后,对图 1(b)中的树,找出最短树枝 L_{CD},由于 $d_D = 1 < d_C = 4$,根据步骤 3,C 被淘汰,对连接 C 的个体 A,D,F,H 生成最小生成树得图 1(c),并调整 A,D,F,H 的树聚集距离和度数;然后,对图 1(c)中的树,找出最短树枝 L_{DF},由于 $d_F = d_D = 3$, $T(D) < T(F)$,根据步骤 4,D 被淘汰,对连接 D 的个体 A,H,F 生成最小生成树得图 1(d),调整 A,H,F 的树聚集距离和度数。重复此操作直到 $|Q| = 3$,此时 A,E,G 为剩下的个体。需要指出的是步骤 6 中的生成树 T' 不

一定为最小生成树,如图 1(b)中的情况,显然边 AD 的长度要小于 L_{AC}。

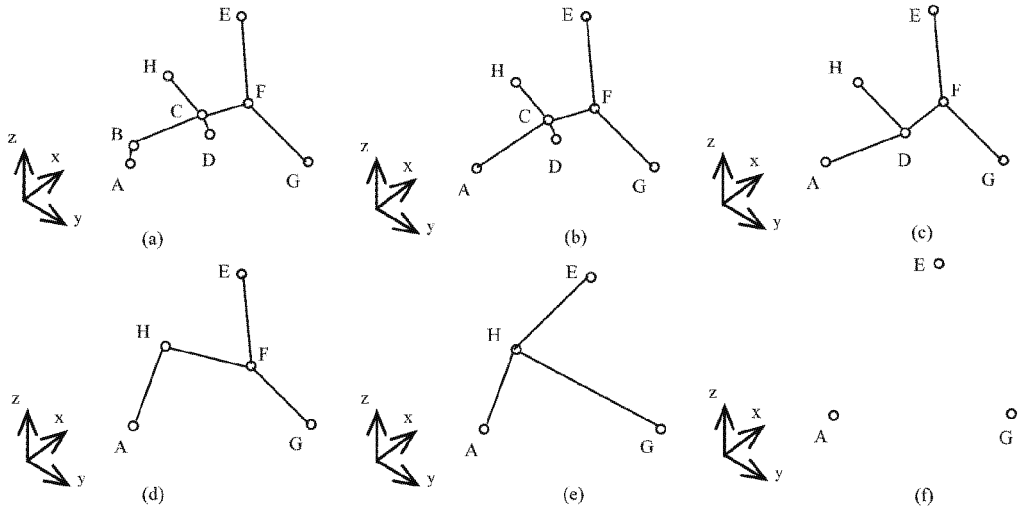


图 1　种群维护示例

在多目标进化算法中,边界个体保留是种群维护的一个重要的目标[10],它很大程度上影响着解集分布范围。在基于生成树的维护方法中,我们用生成树的度数对边界个体进行保留。个体在生成树中的度数不仅反映了个体的密集程度,也一定程度上反映了个体的位置关系。通常位于种群周边的个体度数较小,而种群内部的个体度数较大,在算法的步骤 3 中,度数较大的个体将首先被淘汰,这样边界个体得到一定程度的保留。

2.3　时间复杂度分析

算法中步骤 1 的时间耗费包括两个方面,一为计算种群中任意两个个体之间的欧氏距离,二为对种群生成最小生成树。计算任意两个个体之间欧氏距离的时间复杂度为 $O(MN^2)$,由 Prim 算法对种群生成最小生成树的时间复杂度为 $O(N^2)$。通常 $2N \geqslant |Q| \geqslant N$,因此步骤 2 到步骤 7 的循环最多运行 N 次。步骤 2 中的找到生成树中最短树枝的时间复杂度为 $O(N)$。步骤 3 – 7 为常数时间复杂度 $O(1)$,其中步骤 6 为对连接已删除个体的个体生成最小生成树,由于生成树中个体的度数是与 N 无关的常数,平均值为 $2\times(N-1)/N$,所以对其生成最小生成树的时间耗费为常数。综上,上述算法的总时间复杂度为 $O(MN^2)$。

3　实验设计与结果

3.1　实验参数设置

为检验所提分布性维护方法的有效性,我们将上述算法与 3 种著名的多目标优化算法 NSGA-Ⅱ[3]、C-NSGA-Ⅱ[11]、SPEA2[5]的分布性进行了实验比较。NSGA-Ⅱ拥有很快的速度,C-NSGA-Ⅱ为对 NSGA-Ⅱ的改进,其分布性要好于 NSGA-Ⅱ,SPEA2 是在保持分布性方面最好的算法之一。我们的方法在其它方面的设置(如适应度赋值等)与 NSGA-Ⅱ相同。这里我们选择了 12 个测试函数来比较来测试这 4 种方法的性能,这些多目标测试函数有二维的,高维的,连续的,非连续的,凸的,凹的等。

前面已经提到多目标进化算法中的分布性是指解集分布的均匀性和广泛性。在此我们用最近距离评价方法 SP[1]和均匀估计方法 UA[12]对 4 种算法得到的解集进行均匀性评价,用最大分布评价方法 D[10]对其进行广泛性评价。下面我们简要地介绍这 3 种评价方法。

SP 用来评价解集的均匀性。其评价函数定义如下:

$$SP=\sqrt{\frac{1}{|Q|-1}\sum_{i=1}^{|Q|}(\bar{d}-d_i)^2} \tag{5}$$

$$d_i=\min_{q_j\in Q\wedge q_j\neq q_i}\sum_{m=1}^{M}|f_m(q_i)-f_m(q_j)| \tag{6}$$

d_i 是指解集中非支配边界上两个连续向量的欧几里得距离,$\bar{d}$ 是这些距离的平均值。解集的 SP 值越小表明分布越均匀,理想的情况是 $SP=0$,但 SP 受一些其他因素影响(如种群规模、收敛性等)并不完全准确。UA 是一种鲁棒性很强的解集均匀性的评价方法,基本不受其它因素的影响,其评价结果在 0 到 1 之间,越接近 1 表明分布越均匀。D 用来评价解集的分布广度,值越大表明解集分布越广泛。

在实验中，这 4 种算法都采用实数编码，交叉概率为 0.9，变异概率为 1/nreal，nreal 为决策变量维数，二维测试函数的规模为 100，评价次数为 20000；三维测试函数的规模为 200，评价次数为 100000，四维测试函数的规模为 300，评价次数为 300000。算法的运行代数为评价个体的数目除以种群规模。每个算法对各个测试函数独立运行 20 次，结果取平均值。实验运行在 1.7GHz CPU，256M 内存，Windows XP 环境下。

3.2 二维测试函数

我们选取了 7 个二维测试函数。SCH[1]被认为是在 MOP 领域具有代表性的测试函数，FON[1]是一个可变决策变量维数的测试函数，ZDT 是由 Zitzler 等提出来的一系列测试函数[10]，这些函数是较具有代表性的二维测试函数。其中 SCH，ZDT1，ZDT4 为凸形最优边界，FON，ZDT2，ZDT6 为凹形最优边界，ZDT3 的最优边界为非连续的。

图 2 为 4 种算法在 ZDT2 上最终解集的分布。从图中可以看出，我们的方法与 SPEA2 有相似的均匀性，好于 C-NSGA-II 和 NSGA-II。二维测试函数的结果如表 1 所示，在每个单元格中，第 1 行为均值，第 2 行为标准差。在 SP，UA 上我们的方法与 SPEA2

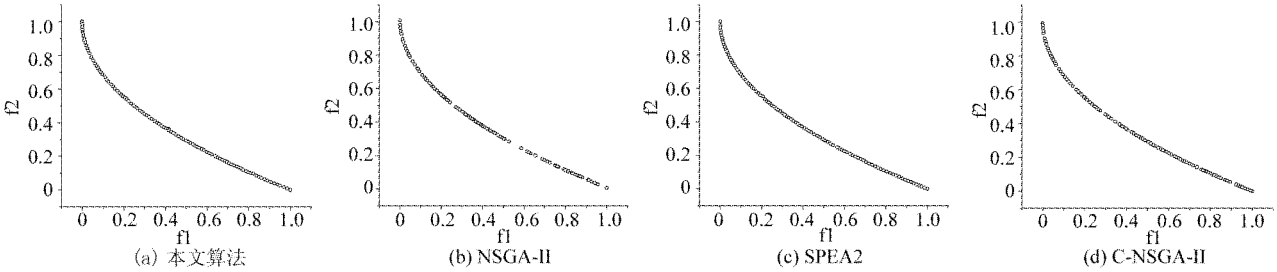


图 2 四种维护方法在 ZDT1 上的最终解集分布

表 1 四种算法在二目标函数下的性能比较

测试函数	方法	SP	UA	D	运行时间(s)
SCH	本文算法	1.2648E-02 ± 7.5676E-04	8.4579E-01 ± 1.2667E-02	**5.6581E+00 ±** 3.9386E-03	1.2284E+00 ± 4.4572E-02
	NSGA-II	3.8312E-02 ± 5.3120E-03	5.0282E-01 ± 3.5652E-02	5.6574E+00 ± 3.7574E-03	6.9251E-01 ± 4.4691E-02
	SPEA2	**1.2637E-02 ±** 1.3859E-03	**8.5192E-01 ±** 1.4950E-02	5.6566E+00 ± 1.3622E-03	7.5607E+00 ± 7.9327E-02
	C-NSGA-II	2.2621E-02 ± 1.8967E-03	5.5255E-01 ± 2.6412E-02	5.6466E-00 ± 2.1023E-03	1.4974E+00 ± 2.6855E-02
FON	本文算法	3.2648E-03 ± 3.8435E-04	7.7108E-01 ± 2.2457E-02	1.3807E+00 ± 3.5307E-03	1.7205E+00 ± 1.6231E-01
	NSGA-II	6.5605E-03 ± 7.3453E-04	4.4242E-01 ± 5.9127E-02	1.3881E+00 ± 3.0227E-04	**8.8441E-01 ±** 2.7925E-02
	SPEA2	**3.0437E-03 ±** 2.5083E-04	**7.9274E-01 ±** 1.4522E-02	**1.3883E+00 ±** 4.8125E-04	8.3999E+00 ± 1.3016E-01
	C-NSGA-II	5.1366E-03 ± 2.6602E-04	5.6157E-01 ± 1.6359E-02	1.3729E+00 ± 6.4992E-03	2.0197E+00 ± 8.1159E-02
ZDT1	本文算法	**3.2223E-03 ±** 3.5124E-04	**7.6343E-01 ±** 1.0976E-02	**1.4151E+00 ±** 3.3323E-04	2.1718E+00 ± 5.8146E-02
	NSGA-II	7.5588E-03 ± 1.1264E-03	4.1389E-01 ± 4.7899E-02	1.4134E+00 ± 8.4719E-04	**1.4017E+00 ±** 3.1219E-02
	SPEA2	3.3599E-03 ± 3.1526E-04	7.6007E-01 ± 1.6212E-02	1.4140E+00 ± 1.0859E-03	8.8484E+00 ± 1.5339E-01
	C-NSGA-II	5.3985E-03 ± 4.9801E-04	5.7705E-01 ± 3.1117E-02	1.4041E+00 ± 4.3405E-03	2.4283E+00 ± 6.8198E-02

(接上页)

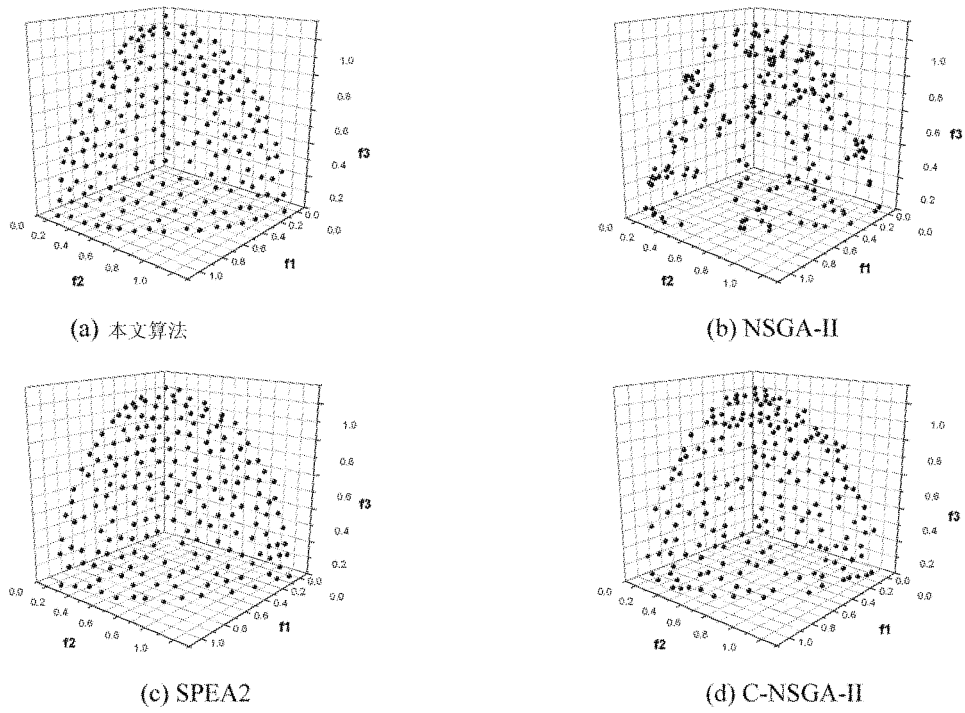
ZDT2	本文算法	3.3859E-03 ± 3.6483E-04	7.6867E-01 ± 1.6113E-02	**1.4163E+00 ±** 3.9197E-03	2.1489E+00 ± 1.3949E-01
	NSGA-II	8.1965E-03 ± 1.2146E-03	4.1459E-01 ± 5.1063E-02	1.4141E+00 ± 3.8041E-03	**1.4644E+00 ±** 1.1284E-01
	SPEA2	**3.3848E-03 ±** 3.6575E-04	**7.6995E-01 ±** 1.9642E-02	1.4123E+00 ± 7.0245E-04	8.4405E+00 ± 1.7342E-01
	C-NSGA-II	5.4475E-03 ± 3.0255E-04	5.7737E-01 ± 2.1506E-02	1.4123E+00 ± 1.0839E-02	2.2657E+00 ± 1.2775E-01
ZDT3	本文算法	**3.854E-03 ±** 4.7131E-04	**7.3511E-01 ±** 1.7705E-02	**1.9717E+00 ±** 6.6882E-03	2.3532E+00 ± 7.5910E-02
	NSGA-II	7.9885E-03 ± 1.1776E-03	4.2442E-01 ± 3.0389E-02	1.9305E+00 ± 1.1230E-02	**1.4609E+00 ±** 3.9710E-02
	SPEA2	4.1186E-03 ± 4.1561E-04	7.0687E-01 ± 2.4697E-02	1.9691E+00 ± 3.1101E-03	8.9965E+00 ± 1.2581E-01
	C-NSGA-II	5.6612E-03 ± 6.6094E-04	5.4367E-01 ± 1.9907E-02	1.9541E+00 ± 1.1568E-02	2.6578E+00 ± 8.5513E-02
ZDT4	本文算法	**3.4159E-03 ±** 2.9703E-04	**7.8954E-01 ±** 1.2116E-02	1.4512E+00 ± 4.3425E-02	2.0314E+00 ± 1.5045E-01
	NSGA-II	7.9517E-03 ± 1.2036E-03	4.3030E-01 ± 1.7350E-02	**1.4635E+00 ±** 1.7407E-01	**1.0813E+00 ±** 4.4824E-02
	SPEA2	4.1902E-03 ± 5.0794E-04	7.8050E-01 ± 1.9035E-02	1.4443E+00 ± 1.1482E-01	9.1454E+00 ± 2.0044E-01
	C-NSGA-II	5.8356E-03 ± 7.2791E-04	5.4051E-01 ± 2.0009E-02	1.4320E+00 ± 1.3268E-01	2.2188E+00 ± 2.1715E-01
ZDT6	本文算法	**3.0396E-03 ±** 2.7072E-04	**7.3899E-01 ±** 1.7647E-02	**1.0567E+00 ±** 1.5927E-03	1.3264E+00 ± 9.6551E-02
	NSGA-II	5.1495E-03 ± 4.6986E-04	4.5949E-01 ± 3.7466E-02	1.0461E+00 ± 6.9547E-03	**1.1469E+00 ±** 9.0027E-02
	SPEA2	3.2875E-03 ± 3.0328E-04	6.9201E-01 ± 1.4607E-02	1.0404E+00 ± 4.0961E-03	7.7724E+00 ± 7.6176E-02
	C-NSGA-II	3.9843E-03 ± 4.1080E-04	5.6189E-01 ± 2.9189E-02	1.0415E+00 ± 1.9781E-03	1.7704E+00 ± 8.9926E-02

相似,SPEA2 在 SCH,FON,ZDT2 上略好于我们的方法,而在 ZDT1,ZDT3,ZDT4,ZDT6 上要略逊于我们的方法。对分布广度评价指标 D,我们的算法在大多数函数上(SCH, ZDT1, ZDT2, ZDT3, ZDT6)拥有最优值,SPEA2 在 FON 上,NSGA-II 在 ZDT4 上拥有最优值。

3.3 三维测试函数

我们选取了 5 个三维测试函数 DTLZ 系列,DTLZ 系列是 Deb, Thiele, Laumanns, Zitzler 共同提出来的[13],它们有一个很重要的特点就是目标个数是规模可变的。其中 DTLZ1 的最优边界为线性超平面,DTLZ2 的最优边界为球形,DTLZ4 与 DTLZ2 的最优边界形状相同,只是 DTLZ4 在 f2 = 0、f3 = 0 上有更密集的解个体,DTLZ5 的最优边界为一曲线,而 DTLZ7 具有一组不连续的最优边界。

图 3 为 4 种算法在 DTLZ2 的分布。从图中可以看出,我们的算法在解集均匀性上略逊于 SPEA2,但远好于 C-NSGA-II 和 NSGA-II。三维测试函数的结果如表 2 所示。对均匀性评价指标 SP 和 UA,在大多数测试函数上,我们的方法略逊于 SPEA2。在 DTLZ1 上,注意到我们的算法在 SP 上好于 SPEA2,但在 UA 上逊于 SPEA2,这是可能是由于 SPEA2 对 DTLZ1 函数的收敛性不稳定,而解集的收敛性一定程度上影响 SP 的结果[12]。对分布广度评价指标 D,除了在 DTLZ2 上,C-NSGA-II 有最优值外,在其余的测试函数上,我们的方法均有最佳结果。



(a) 本文算法　(b) NSGA-II

(c) SPEA2　(d) C-NSGA-II

图 3　四种维护方法在 DTLZ2 上的最终解集分布

表 2　四种算法在三目标函数下的性能比较

测试函数	方法	SP	UA	D	运行时间(s)
DTLZ1	本文算法	**7.1364E-03 ±** 4.0706E-04	7.7786E-01 ± 1.0112E-02	**8.6872E-01 ±** 1.7229E-03	1.6756E + 01 ± 9.5145E-02
	NSGA-II	3.1992E-02 ± 2.1719E-02	4.0042E-01 ± 4.3178E-02	8.6132E-01 ± 1.9690E-03	**8.1047E + 00 ±** 8.7312E-02
	SPEA2	1.3653E-02 ± 1.5848E-02	**8.0726E-01 ±** 4.2639E-03	8.6652E-01 ± 1.0014E-02	1.1409E + 02 ± 1.0962E + 00
	C-NSGA-II	1.7768E-02 ± 1.0387E-04	6.9052E-01 ± 1.3064E-02	8.6059E-01 ± 5.5143E-02	2.5327E + 01 ± 9.6501E-01
DTLZ2	本文算法	1.7901E-02 ± 1.0748E-03	7.8171E-01 ± 1.3189E-02	1.7575E + 00 ± 1.0321E-02	2.7609E + 01 ± 1.3364E-01
	NSGA-II	4.1831E-02 ± 2.1634E-03	3.6837E-01 ± 1.1992E-02	1.7455E + 00 ± 6.7949E-03	**1.0600E + 01 ±** 4.0608E-02
	SPEA2	**1.6225E-02 ±** 1.1705E-03	**8.2226E-01 ±** 9.5991E-03	1.7712E + 00 ± 1.0455E-02	1.5123E + 02 ± 5.8966E-01
	C-NSGA-II	2.0334E-02 ± 1.3051E-03	6.9146E-01 ± 1.3551E-02	**1.7853E + 00 ± 4.4669E-02**	**4.2678E + 01 ± 4.1237E-01**
DTLZ4	本文算法	**2.4645E-02 ± 2.9058E-03**	**7.1378E-01 ± 1.8921E-02**	**1.9243E + 00 ±** 1.1205E-01	1.4657E + 01 ± 9.3137E-01
	NSGA-II	2.7077E-02 ± 2.5078E-02	3.6932E-01 ± 1.1042E-01	1.6429E + 00 ± 1.7169E-01	**7.8844E + 00 ±** 3.1829E-01
	SPEA2	**2.2087E-02 ±** 2.1847E-03	**7.7293E-01 ±** 2.1152E-02	1.8990E + 00 ± 4.1558E-02	7.9212E + 01 ± 1.6294E + 00
	C-NSGA-II	2.6563E-02 ± 5.6436E-03	6.6606E-01 ± 6.7239E-03	1.9033E + 00 ± 1.2866E-01	2.0862E + 01 ± 4.3824E-01

(接上页)

DTLZ5	本文算法	**2.3036E-03 ±** 2.0529E-04	**7.7935E-01 ±** 9.4632E-03	**1.4161E + 00 ±** 1.8362E-03	2.5416E + 01 ± 1.8362E-03
	NSGA-II	5.3861E-03 ± 8.1431E-04	4.6281E-01 ± 1.2015E-02	1.4146E + 00 ± 8.4298E-04	**8.9062E + 00 ±** 1.9923E-02
	SPEA2	2.3502E-03 ± 1.9935E-04	7.7168E-01 ± 4.6814E-03	1.4147E + 00 ± 3.8444E-04	1.2008E + 02 ± 1.5941E-01
	C-NSGA-II	3.6614E-03 ± 3.7886E-04	5.8333E-01 ± 1.5719E-02	1.4129E + 00 ± 1.0301E-02	3.1647E + 01 ± 2.0127E-01
DTLZ7	本文算法	2.4295E-02 ± 2.6566E-03	7.2236E-01 ± 1.0784E-02	**3.9751E + 00 ±** 2.6120E-01	2.2097E + 01 ± 1.5777E-01
	NSGA-II	3.2491E-02 ± 1.6232E-02	4.0483E-01 ± 1.3817E-02	3.5301E + 00 ± 4.6207E-02	**8.8854E + 00 ±** 3.0742E-02
	SPEA2	**2.1404E-02 ±** 2.1603E-03	**7.6141E-01 ±** 9.6525E-03	3.7593E + 00 ± 6.0109E-02	1.2093E + 02 ± 1.2815E-01
	C-NSGA-II	2.5098E-02 ± 2.0022E-03	6.6147E-01 ± 1.7277E-02	3.7529E + 00 ± 1.7491E-01	4.8428E + 01 ± 1.3242E-01

3.4 四维测试函数

下面我们对 DTLZ2 进行四维下的测试,实验结果如表 3 所示。从表中可以发现,我们的方法与 SPEA2 在分布性能上相当,好于 C-NSGA-II 和 NSGA-II。

表 3 四种算法在四目标函数下的性能比较

测试函数	方法	SP	UA	D	运行时间(s)
DTLZ2	Proposed	**3.3213E-02 ±** 2.3650E-03	8.2093E-01 ± 5.8020E-03	2.3397E + 00 ± 4.0236E-02	2.4816E + 02 ± 6.0027E-01
	NSGA-II	7.6069E-02 ± 4.4408E-03	4.4569E-01 ± 2.5074E-02	2.1926E + 00 ± 5.7378E-02	**6.4635E + 01 ±** 6.3518E-01
	SPEA2	3.4801E-02 ± 2.0229E-03	**8.4328E-01 ±** 6.7074E-03	**2.3869E + 00 ±** 5.7339E-02	6.4842E + 02 ± 3.4001E + 00
	C-NSGA-II	3.7208E-02 ± 4.1537E-03	7.4536E-01 ± 7.7279E-03	2.2156E-01 ± 1.8586E-01	2.9948E + 02 ± 3.3765E + 00

3.5 运行时间比较

从表 1 – 3 中可以发现,NSGA-II 拥有最快的速度,我们的算法要略好于 C-NSGA-II,远好于 SPEA2。这是因为 NSGA-II 基于聚集距离维护方法时间复杂度为 $O(MN\log N)$, C-NSGA-II 基于聚类维护方法时间复杂度为 $O(MN^2)$, SPEA2 基于密度维护方法时间复杂度为 $O(MN^2\log N)$。

4 结 论

种群维护是多目标进化算法的重要组成部分,维护方法的好坏决定了最终解集的分布状况。然而通常维护方法的性能和运行时间存在矛盾,拥有较高性能的算法一般具有较慢的维护速度,运行效率较快的算法性能很难得到保证。本文尝试这方面的工作,提出一种基于生成树的种群维护方法,即在考虑最短树枝的同时,利用树聚集距离和度数对种群进行修剪。由于树聚集距离和度数的动态性,每移出一个个体,种群中与之相连个体的信息都会发生相应的变化,这就及时地反映了种群的分布情况。另外,度数的加入使边界个体基本得到保留,有利于种群获得广泛的非支配个体。通过与三个著名的算法 NSGA-II, SPEA2 和 C-NSGA-II 的比较实验,表明基于生成树的维护方法在均匀性、分布广泛性、运行时间方面有一个很好的折中,能以较快的速度对种群进行维护,并得到均匀、广泛的解集。

参考文献

[1] Deb K. Multi-Objective Optimization using Evolutionary Algorithms. Chichester, UK: John Wiley & Sons, 2001. 1-7

[2] 郑金华. 多目标进化算法及其应用. 北京: 科学出版社, 2007. 20-50

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

[4] Knowles J, Corne D W. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation*, 2003, 7(2): 100-116

[5] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm: [technical report]. Zurich, Switzerland: Computer Engineering and Network Laboratory, Swiss Federal Institute of Technology, 2001

[6] Kukkonen S, Deb K. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'2006), Vancouver, Canada, 2006. 3995-4002

[7] Kuang D, Zheng J H. Strategies based on polar coordinates to keep diversity in multi-objective genetic algorithm. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'2005), Edinburgh, Scotland, 2005. 1276-1281

[8] Kukkonen S, Deb K. A fast and effective method for pruning of non-dominated solutions in many-objective problems. In: Proceedings of the 9th International Conference on Parallel Problem Solving from Nature, Reykjavik, Iceland, 2006. 553-562

[9] Li M Q, Zheng J H, Xiao G X. An efficient multi-objective evolutionary algorithm based on minimum spanning tree. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'2008), Hong Kong, China, 2008. 617-624

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

[11] Deb K, Mohan M, Mishra S. Towards a quick computation of well-spread pareto-optimal solutions. In: Proceedings of the 2nd Evolutionary Multi-Criterion Optimization Conference (EMO' 2003), Faro, Portugal, 2003. 222-236

[12] Li M Q, Zheng J H, Xiao G X. Uniformity assessment for evolutionary multi-objective optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'2008), Hong Kong, China, 2008. 625-632

[13] Deb K, Thiele L, Laumanns M, et al. Scalable multi-objective optimization test problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'2002), Piscataway, New Jersey, USA, 2002. 825-830

Spanning tree-based diversity maintenance in multi-objective evolutionary algorithms

Li Miqing, Zheng Jinhua

(Institute of Information Engineering, Xiangtan University, Xiangtan 411105)

Abstract

The paper proposes a new method for maintaining the diversity of multi-objective evolutionary algorithms (MOEA) using the spanning tree. The method defines a density estimation metric, the spanning tree crowding distance. Moreover, it applies the shortest edge and the degree in the spanning tree combined with the spanning tree crowding distance to population truncation. The extensive comparative study with the three other classical methods of NSGA-Ⅱ, SPEA2 and C-NSGA-Ⅱ on four performance metrics and twelve test problems, indicates that the proposed method has a good balance among uniformity, extent and execution time.

Key words: multi-objective optimization, evolutionary algorithms, diversity maintenance, spanning tree, population