ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.6, June 2007, pp.1287−1297
DOI: 10.1360/jos181287

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

用擂台赛法则构造多目标 Pareto 最优解集的方法*

郑金华[1+], 蒋 浩[1], 邝 达[1], 史忠植[2]

[1](湘潭大学 信息工程学院,湖南 湘潭 411105)

[2](中国科学院 计算技术研究所,北京 100080)

An Approach of Constructing Multi-Objective Pareto Optimal Solutions Using Arena's Principle

ZHENG Jin-Hua[1+], JIANG Hao[1], KUANG Da[1], SHI Zhong-Zhi[2]

[1](Institute of Information Engineering, Xiangtan University, Xiangtan 411105, China)

[2](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-732-8293457, E-mail: jhzheng@xtu.edu.cn, http://www.xtu.edu.cn

Zheng JH, Jiang H, Kuang D, Shi ZZ. An Approach of Constructing Multi-Objective Pareto Optimal Solutions Using Arena's Principle. *Journal of Software*, 2007,18(6):1287−1297. http://www.jos.org.cn/1000-9825/18/1287.htm

Abstract: This paper proposes an approach, namely the arena's principle (AP), to construct the Pareto optimal solutions by utilizing features of the multi-objective evolution. It is proved that the AP works correctly and its computational complexity is $O(rmN)$ ($0<m/N<1$). Theoretically, when AP is compared with Deb's algorithm and Jensen's algorithm (their computational complexity are $O(rN^2)$ and $O(N\log^{(r-1)}N)$ respectively), AP is better than Deb's, and is also better than Jensen's when the objective number r is relatively large (such as $r\geq 5$). Moreover, AP performs better than the other two algorithms when m/N is relatively small (such as $m/N\leq 50\%$). Experimental results indicate that AP performs better than the other two algorithms on the CPU time efficiency. In applications, AP can be integrated into any Pareto-based MOEA to improve its running efficiency.

Key words: multi-objective evolution; arena's principle; non-dominated set; Pareto optimal solutions; running efficiency

摘 要: 针对多目标进化的特点,提出了用擂台赛法则(arena's principle,简称 AP)构造多目标 Pareto 最优解集的方法,论证了构造方法的正确性,分析了其时间复杂度为 $O(rmN)$($0<m/N<1$).理论上,当 AP 与 Deb 的算法以及 Jensen 的算法比较时(它们的时间复杂度分别为 $O(rN^2)$和 $O(N\log^{(r-1)}N)$),AP 优于 Deb 的算法;当目标数 r 较大时(如 $r\geq 5$),AP 优于 Jensen 的算法;此外,当 m/N 较小时(如 $m/N\leq 50\%$),AP 的效率与其他两种算法比较具有优势.对比实验结果表明,AP 具有比其他两种算法更好的 CPU 时间效率.在应用中,AP 可以被集成到任何基于 Pareto 的 MOEA 中,并能在

* Supported by the National Natural Science Foundation of China under Grant Nos.60435010, 69974043 (国家自然科学基金); the Scientific Research Foundation for the Returned Overseas Chinese Scholars, Ministry of Education (教育部留学回国人员科研启动基金); the Natural Science Foundation of Hu'nan Province of China under Grant Nos.01JJY2060, 05JJ30125 (湖南省自然科学基金); the SRF of Hu'nan Provincial Education Department (湖南省教育厅重点科研项目)

Received 2006-07-05; Accepted 2006-08-18

较大程度上提高 MOEA 的运行效率.

关键词: 多目标进化;擂台赛法则;非支配集构造方法;Pareto 最优解集;运行效率
中图法分类号: TP301 文献标识码: A

进化算法(evolutionary algorithm,简称 EA)因其具有解决高度复杂的非线性问题的能力,引起了科学研究和实际应用各个层面的极大兴趣.现实世界中的许多实际问题一般都是多属性的,通常是对多个目标的同时优化,因此,近年来,多目标进化算法(multi-objective evolutionary algorithm,简称 MOEA)的研究进入了快速发展阶段.

文献[1]等将多目标进化算法分为 3 大类:Priori 技术,Progressive 技术和 Posteriori 技术.Priori MOEA 技术在 MOEA 搜索之前就输入决策信息,然后通过 MOEA 运行产生一个解提供给决策者;而 Posteriori MOEA 技术则通过运行 MOEA 产生一组解供决策者选择;Progressive MOEA 技术则是通过决策者与 MOEA 搜索过程的不断交互实现的.目前,MOEA 的研究主要集中在 Posteriori 技术[1,2]方面.Posteriori 技术又分 4 类方法:Independent sampling,Criterion selection,Aggregation selection,和 Pareto sampling.其中,基于 Pareto 的方法目前最为活跃.这类方法的共同特点是,在进化过程中,通过构造当前进化群体的 Pareto 最优解集(即非支配集),并使之不断逼近 True Pareto 最优解集来实现;这类方法的优点是通用性较好,其不足是运行效率不高.

基于 Pareto 的典型算法主要有:PESA(the Pareto envelope-based selection algorithm for multi-objective optimization)[3],PAES(the Pareto archived evolution strategy)[4],IS-PAES(inverted-shrinkable PAES)[5],Fonseca 和 Fleming 的 MOGA(multi-objective genetic algorithm)[6],Horn 和 Nafpliotis 的 NPGA(a niched Pareto genetic algorithm for multiobjective optimization)[7],Zitzler 和 Thiele 的 SPEA(strength Pareto evolutionary algorithm)[8,9],Deb 的 NSGA-II(non-dominated sorting in genetic algorithm)[10].这些算法构造非支配集的时间复杂度均不低于 $O(rN^2)$,其中,r 为目标数,N 为群体规模.Deb 提出了多种构造非支配集的方法[10,11],其时间复杂度均为 $O(rN^2)$.此外,Jensen 于 2003 年提出了一种采用递归构造非支配集的方法[12],其时间复杂度为 $O(N\log^{(r-1)}N)$.目前,Deb 在 NSGA-II 中提出的构造非支配集的方法和 Jensen 的方法被认为是最具代表性的两类方法.

由于在基于 Pareto 的方法中每一次迭代操作(即每一代进化)都要构造当前进化群体的非支配集,因此,构造非支配集的速度直接影响到 MOEA 的运行效率.为此,本文讨论一类基于擂台赛法则的构造进化群体 Pareto 最优解集的方法.

1 多目标进化个体之间的关系

一个多目标优化问题可描述如下:

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

$$g_i(X)\geq 0,\ i=1,2,\ldots,k \tag{1}$$

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

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

$$\mathrm{Min}\vec{f}(X)=(f_1(X),f_2(X),\ldots,f_r(X)) \tag{3}$$

寻求 $X^*=(x_1^*,x_2^*,\ldots,x_n^*)$,使 $\vec{f}(X^*)$ 在满足约束(1)和约束(2)的同时达到最优.

我们这里所讲的最优,是由 Vilfredo Pareto 在 1896 年提出来的,故称之为 Pareto 最优解(Pareto optimum solution),定义如下:

我们称 $X^*\in F$ 是最优解,若$\forall X\in F$ 满足

或者

$$\bigwedge_{i\in I}(f_i(X)=f_i(X^*)),\ I=\{1,2,\ldots,r\} \tag{4}$$

或者至少存在一个 $j\in I$,使

$$f_j(X)>f_j(X^*) \tag{5}$$

其中,F 是满足式(1)和式(2)的可行解集,即

$$F=\{X\in R^n|g_i(X)\geq 0,i=1,2,\ldots,k;h_j(X)=0,j=1,2,\ldots,l\}.$$

通常情况下,最优解不只一个,而是一个最优解集(Pareto optimal solutions).我们所做的工作是,构造非支配集(non-dominated solutions 或 non-inferior solutions),并使非支配集不断逼近 Pareto 最优解集,最终达到最优.值得说明的是,本文所用到的术语“最优解”指的是 Pareto 最优解,“非支配个体”不一定是 Pareto 最优解.只有当算法收敛时,“非支配个体”才是 Pareto 最优解.

在多目标进化群体中,或对一个给定的多目标可行解集,个体之间的关系不同于单目标情况下的大小关系.对于个体 x 与个体 y,或者 x 支配 y,或者 y 支配 x,或者 x 与 y 之间相互不被支配.为了描述方便起见,下面给出有关定义.

定义 **1**. 设 P 为一个集合,其大小为 N,P 中每个个体均有 r 个属性,$f_k()$是每个属性的评价函数(k=1,2,…,r),P 中个体之间的关系定义为:

(1) 支配关系:$x,y\in P$,若 $f_k(x)\leq f_k(y),k=1,2,\ldots,r$;且$\exists l\in\{1,2,\ldots,r\}$,使 $f_l(x)<f_l(y)$,则称 x 支配 y,表示为 $x\succ y$.此时,称 x 为非支配的(non-dominated),y 为被支配的(dominated),其中,“$\succ$”是支配关系(dominated relation).

(2) 不相关:$x,y\in P$,若 x 和 y 之间不存在支配关系,则称 x 和 y 不相关或无关.

定义 **2**. 对于给定个体 $x\in P$,若 $\nexists y\in P$,使 $y\succ x$,则 x 称为集合 P 的非支配个体.由所有 P 的非支配个体组成的集合,称为 P 的非支配集.

定义 **3**. 设 Nds 是 P 的非支配集,$Nds\subseteq P,\forall x\in P$,若 x 是 P 的非支配个体,必有 $x\in Nds$,则称 Nds 是 P 的最大非支配集.

定义 **4**. 称 x^*是偏序集$(P,\succ)$中的最小元素,若在 P 中不存在任何其他 x 比 x^*更小,即 $\nexists x\in P$,使 $x\succ x^*$.所有最小元素的集合表示为 $M(P,\succ)$.

定义 **5**. 设 $x\in M(P,\succ)$,$Cluster(x)=\{y|x\succ y,y\in P\}$是以 x 为最小元的族.

在此需要说明的是,$M(P,\succ)$即为 P 的非支配集,$M(P,\succ)$中所有个体之间是互不相关的[13].

2 Deb 和 Jensen 的工作

Deb 在文献[11]中提出了一类构造非支配集的算法,其时间复杂度为 $O(rN^2)$,其中,r 为目标数,N 为群体规模.Deb 在文献[10]中提出了改进方法,见算法 1.虽然其时间复杂度仍为 $O(rN^2)$,但实际运行效率有较大提高.

算法 **1**. 构造进化群体的非支配集,结果放在 P'中.

```
P'=find-nondominated-front(P)
P'={1}                                  将第 1 个个体 p 放入 P'中
For each p∈P∧p∉P'                       每次取 1 个个体
    P'=P'∪{P}                           将个体 p 放入 P'中(临时)
    For each q∈P'∧q≠P                   比较 p 与 P'中其他个体 q 之间的支配关系
        If P≻q, then P'=P'\{q}          若 p 支配 q,则删除 q
        Else q≻p, then P'=P'\{q}        若 q 支配 p,则删除 p
```

Jensen 基于分治技术(divide and conquer),采用递归的方法(recursive procedure),于 2003 年提出了一类构造非支配集的算法,其时间复杂度为 $O(N\log^{(r-1)}N)$,其中,N 为进化群体的规模,r 为目标数.Jensen 的算法比较复杂,由 4 个模块组成:主过程 *Non-dominated-sort*(S,r),第 1 层递归模块 *ND-helper_A*(S,r),第 2 层递归模块 *ND-helper_B*(L,H,r),以及针对二维情况的 *Non-dominated-sort-on-2D*(S),这里,S 为进化群体.

在主过程中,对每个个体 $s\in S$ 设置一个函数值 $f[s]$(初始时,$f[s]$=1),用于表示该个体所对应的边界层次.通过调用 *ND-helper_A*(S,r),使 $f[s]$返回个体 s 所对应的边界层次.显然,所有 $f[s]$=1($s\in S$)的个体 s 均为非支配的.但当待优化问题只有两个目标时(即 r=2),则作为特殊情况处理,直接调用算法 *Non-dominated-sort-on-2D*(S),而无须递归调用.

在递归过程 *ND-helper_A*(*S*,*r*)中,*r* 个子目标分别表示为 $x_1...x_r$.当 *S* 中只有两个个体时(即|*S*|=2),则将这两个个体进行比较.若 s_1 支配 s_2,则令 $f[s_2]=\max(f[s_1]+1,f[s_2])$;若 s_2 支配 s_1,则令 $f[s_1]=\max(f[s_1],f[s_2]+1)$.当 *S* 中个体大于 2 时,则将 *S* 分割为两个大小相等的子集 *L* 和 *H*,即 $set(L,H)=split(S, x_r^{split}, r)$.分割时,取第 *r* 个目标的中值 x_r^{split},然后将所有第 *r* 个目标值小于等于 x_r^{split} 的个体放入 *L* 中,高于 x_r^{split} 的则放入 *H* 中.由此可知,*L* 中的任何一个个体均不被 *H* 中个体所支配,因为有:$\forall s_1\in L\wedge\forall s_2\in H\Rightarrow x_r(s_1)<x_r(s_2)$.这样,通过进一步递归调用 *ND-helper_A*(*L*,*r*)来构造 *L* 的非支配集,且在构造 *L* 的非支配集时就不需要考虑 *H*.但 *H* 中的个体也有可能不被 *L* 中的个体所支配,因此在构造 *H* 的非支配集时必须考虑 *L*.要判断 $s_2\in H$ 是否被 $s_1\in L$ 支配,需要调用 *ND-helper_B*(*L*,*H*,*r*−1)来比较 *L* 和 *H* 在前(*r*−1)个子目标上的支配关系.若 $s_1\in L$ 支配 $s_2\in H$,则令 $f[s_2]=\max(f[s_1]+1,f[s_2])$.此外,还需要判断 *H* 中的个体是否被 *H* 中其他个体所支配,因此需要调用 *ND-helper_A*(*H*,*r*).

在递归过程 *ND-helper_B*(*L*,*H*,*r*)中,通过对 *H* 中个体与 *L* 中个体进行比较,从而实现对 *H* 中个体进行分类(注意:这只是部分分类,要在此语句后调用 *ND-helper_A*(*H*,*r*),才能实现对 *H* 的完全分类).若 *L* 或 *H* 中只有 1 个个体,则将 *H*(或 *L*)中所有个体与 *L*(或 *H*)中这(唯一的)一个个体比较,如果 $s_1\in L$ 支配 $s_2\in H$,则令 $f[s_2]=\max(f[s_1]+1, f[s_2])$.如果此时 *r*=2,则直接调用一个二维(2D)分类算法(类似于算法 *Non-dominated-sort-on-2D*(*S*)),通过与 *L* 中个体比较其支配关系,给 *H* 中个体分配边界数(即调整 $f[s],s\in H$),即如果 $s_1\in L$ 支配 $s_2\in H$,则令 $f[s_2]=\max(f[s_1]+1, f[s_2])$.

在其他情况下,处理要复杂一些:

(1) 如果$(\max(x_r(l_1),\ldots,x_r(l_{|L|}))\le\min(x_r(h_1),\ldots,x_r(h_{|H|}))$,表明在第 *r* 个目标上,*L* 中个体比 *H* 中个体具有较小的值,则直接调用 *ND-helper_B*(*L*,*H*,*r*−1).

(2) 如果$(\min(x_r(l_1),\ldots,x_r(l_{|L|}))>\max(x_r(h_1),\ldots,x_r(h_{|H|})))$,表明在第 *r* 个目标上,*H* 中个体比 *L* 中个体具有较小的值,这种情况下算法无须做任何处理.

(3) 如果$(\max(x_r(l_1),\ldots,x_r(l_{|L|}))>\min(x_r(h_1),\ldots,x_r(h_{|H|}))$,且$(\min(x_r(l_1),\ldots,x_r(l_{|L|}))\le\max(x_r(h_1),\ldots,x_r(h_{|H|})))$,表明在第 *r* 个目标上,*H* 中部分个体的值比 *L* 中个体要小,同时,*H* 中部分个体的值又比 *L* 中个体要大,即 *H* 中个体与 *L* 中个体的大小关系存在着交迭.这种情况下,需要在 *H* 或 *L* 中,针对第 *r* 个目标选取一个中值 x_r^{split},并将 *H* 分割为 H_1 和 H_2,将 *L* 分割为 L_1 和 L_2.然后分别递归调用 *ND-helper_B*(L_1,H_1,*r*),*ND-helper_B*(L_1,H_2,*r*−1)和 *ND-helper_B*(L_2,H_2,*r*).需要说明的是,这里不需要针对 L_2 来调整 H_1,即不需要调用 *ND-helper_B*(L_2,H_1,*r*),是因为 H_1 中个体不可能被 L_2 中个体所支配.

在 *Non-dominated-sort-on-2D*(*S*)中,首先将 *S* 中个体排序为 $s_1,s_2,\ldots,s_N$,要求满足条件:$i<j\Rightarrow((x_1(s_i)<x_1(s_j))\vee((x_1(s_i)=x_1(s_j))\wedge(x_2(s_i)<x_2(s_j)))$.用 F_1 保存第 1 层非支配个体,用 F_2 保存第 2 层非支配个体(即 *S* 中删除第 1 层非支配个体后的非支配个体).类似地,用 F_A 保存第 *A* 层非支配个体(即 *S* 中删除前 *A*−1 层非支配个体后的非支配个体).初始时,令:$F_1=\{s_1\}$,*A*=1.然后依次判断个体 $s_2,\ldots,s_N$ 与 $F_1,\ldots,F_A$ 中个体之间的关系,并按它们之间的支配关系,将其保存到相应的 F_i 中(i=1…*A*,*A*≥1).

3 用擂台赛法则构造多目标进化群体的最优解集

基于 Pareto 的优化,多目标进化群体的最优解集是通过构造当前进化群体的非支配集来实现的.采用擂台赛法则(arena's principle,简称 AP)构造非支配的基本思路是,在每一轮比较时,从构造集中选出一个个体出任擂主(一般为当前构造集的第 1 个个体),由擂主与构造集中其他个体进行比较,败者被淘汰出局,胜者成为新的擂主,并继续该轮比较;一轮比较后,最后的擂主个体即为非支配个体.按照这种方法进行下一轮比较,直至构造集为空.

3.1 用擂台赛法则构造非支配集的方法

设 *P* 为进化群体,*Q* 为构造集,初始时 *Q*=*P*;*Nds* 为非支配集,初始时为空.从 *Q* 中任取一个个体 *x*,依次与 *Q* 中所有其他个体 *y* 比较,如果 *x* 支配 *y*,则将个体 *y* 从 *Q* 中清除;如果 *y* 支配 *x*,则用 *y* 代替 *x*(即产生了新的擂主),

并继续进行比较.一轮比较后,形成族 $Cluster(x)=\{y|x\succ y,$且 $x,y\in P\},x$ 为族长,将 x 并入非支配集 Nds 中,而 $Cluster(x)$中个体是被 x 所支配的,必须从构造集 Q 中清除.依此类推,直至 Q 为空.构造非支配集的具体过程见算法 2.

算法 **2**. 用擂台赛法则构造非支配集.

{设 P 为 r 个目标的进化群体,初始时令非支配集 $Nds=\varnothing$,令 $Q=P$}

(1) 从 Q 中任选一个个体 x,作为比较对象;

(2) 令 $Q=Q-x,RK=\varnothing,R=\varnothing$;

(3) 若 Q 空,则转(5),否则转(4);

(4) 依次从 Q 中取一个个体 y,与 x 比较其相互关系:

(4.1) 若 $x\succ y$,则令 $Q=Q-y$,转(3);

(4.2) 若 $y\succ x$,则令 $x=y,Q=Q-y,RK=RK\cup R,R=\varnothing$,转(3);

(4.3) 若 x 与 y 无关,则令 $R=R\cup\{y\},Q=Q-y$,转(3);

(5) 令 $RK'=\{y\in RK|\text{not }(x\succ y)\},Nds=Nds\cup\{x\}$;

(6) 令 $Q=RK'\cup R$,若$|Q|>1$,则转(1);否则,令 $Nds=Nds\cup Q$,结束.

如果在一轮比较中,出现了替代操作(一次以上),即产生了新的擂主,或者说擂主被更换了一次以上.这种情况下,需要记录最后一次替换操作(或更新操作)的有关信息,如更新操作发生时的构造集、新的擂主等.这是因为,在更新操作之前被保留下来的个体有可能是被最新的擂主所支配的,如果其中一些个体只被当前这个最新擂台主所支配而不被任何其他个体所支配,这种情况下,这个最新的擂主必须回过头去找出这样的被它所支配的个体并清除之.否则,只被某一个新擂主所支配的个体就会被保留下来,并被当成是非支配个体.如例 1 的 C_8,它只被 C_1 所支配,且 C_1 是第 1 轮比较的最新擂主,如果 C_1 不回过头去找出 C_8 并清除之,C_8 将被保留到非支配集中.这显然是不允许的.

一般情况下,设在构造非支配集的某一轮比较中共出现了 k 个新的擂主,表示为 $X_{k+1}fX_kf\ldots fX_2fX_1$,其中,$X_1$ 为第 1 个擂主,X_{i+1} 表示第 i 次出现的新擂台主.R_j 表示第 j 次出现新擂主 X_{j+1} 时,没有被此前的旧擂台主 X_j 在比较时所清除的所有个体的集合.这样一来,当第 i 个新的擂主出现时,X_{i+1} 所需要回过头去进行比较的所有个体的集合为 $R_1\cup R_2\cup\ldots\cup R_i$.由此可知,任何被 X_{i+1} 所支配的个体 $Y\in R_1\cup R_2\cup\ldots\cup R_i,(1\le i<k)$,也必然被 X_{k+1} 所支配;同时,k 次新旧擂主的更替,所有被 $X_{i+1}(1\le i\le k)$所支配的个体的集合为 $R_1\cup R_2\cup\ldots\cup R_k$.从而可知,当多次出现新旧擂主的更替时,我们只要将最后一个擂主与其前被保留下来的所有个体进行比较,并清除被它支配的所有个体.

3.2 构造方法的正确性与时间复杂性分析

下面我们讨论用擂台赛法则(AP)构造非支配集的正确性,主要包括两个方面:一方面讨论采用 AP 所构造的 Nds 就是 P 的非支配集,另一方面也讨论 Nds 也是 P 的最大非支配集.

定理 **1**. 用擂台赛法则所构造的 Nds 就是 P 的非支配集,且为最大非支配集.

证明:

(1) 首先证明第 1 个进入 Nds 的个体是 P 的非支配个体.

设第 1 个进入 Nds 的个体为 x_1,令此时的构造集为 Q_1,由构造方法可知,$\nexists y\in Q_1$,使 $y\succ x_1$,即 x_1 不被 Q_1 中任意个体所支配;同时有:$\nexists y\in Q_1,x_1\succ y$,即 x_1 也不支配 Q_1 中任意个体;故而可知 x_1 与 Q_1 中个体不相关.

此外,当产生 x_1 后,形成了族 $Cluster(x_1)=\{y|x_1\succ y,$且 $x_1,y\in P\}=P-Q_1-\{x_1\}$.

从而可得,$\nexists y\in P$,使 $y\succ x_1$.即 x_1 不被 P 中任意个体支配,亦即第 1 个进入 Nds 的个体 x_1 是 P 的非支配个体.

(2) 证明第 $s(N\ge s\ge 2)$个进入 Nds 的个体 x_s 是 P 的非支配个体.

设当 x_s 进入 Nds 后构造集变为 Q_s,由构造方法可知,此时 x_s 与 Q_s 中任意个体无关.即 $\nexists y\in Q_s$,使 $y\succ x_s$;同时,$\nexists y\in Q_s,x_s\succ y$.

设此时已形成的 s 个族分别为 $Cluster(x_1),Cluster(x_2),\ldots,Cluster(x_{s-1})$及 $Cluster(x_s)$,且 $\nexists y\in\{x_1,x_2,\ldots,x_{s-1}\}$,使 $y\succ x_s$;否则在第 s 轮比较之前,x_s 已被 y 从构造集中清除.显然,$\nexists y\in Cluster(x_1)\cup Cluster(x_2)\cup\ldots\cup Cluster(x_{s-1})\cup$

$Cluster(x_s)$,使 $y \succ x_s$;其中,$Cluster(x_k)=Q_{k-1}-Q_k-\{x_k\}$,$k=2,3,\ldots,s$.故 $\nexists y\in P-Q_s$,使 $y \succ x_s$.由此可得,$\nexists y\in P$,使 $y \succ x_s$.即 x_s 不被 P 中任意个体所支配,也就是说,第 $s(N\geq s\geq 2)$个进入 Nds 的个体 x_s 是 P 的非支配个体.

综合(1)和(2),Nds 是 P 的非支配集.

(3) 证明 Nds 是 P 的最大非支配集.

若$\exists x_k\in Nds$,不是 P 的非支配个体,则必$\exists y\in P$,使 $y \succ x_k$.即或者$\exists y\in Q_k$,使 $y \succ x_k$,则 x_k 将被 y 从构造集 Q 中清除并进入下一轮比较,这种情况下,x_k 不可能被并入 Nds 中;或者$\exists y\in P-Q_k$,使 $y \succ x_k$,则 x_k 早在 y 被清除之前已从构造集 Q 中被清除.

反之,若$\forall x\in P-Nds$,是 P 的非支配个体,则 $\nexists y\in P$,使 $y \succ x$.因此,P 中任意一个体都不能将 x 从构造集 Q 中清除,即 $\nexists y\in Q\subseteq P$,使 $y \succ x$,故 x 必定被并入 Nds 中.由此可知,按上述方法构造的非支配集 Nds 是 P 的最大非支配集. □

现在对算法 2 进行时间复杂度分析,设集合 P 的大小为 N,P 中共有 m 个非支配个体,算法总共执行了 m 轮比较操作,每一轮产生一个非支配个体.产生第 1 个非支配个体时做了$(N-1)$次比较操作,设清除了 k_1 个 P 的支配个体,其中 $k_1\geq 0$;产生第 2 个非支配个体时做了$(N-k_1-2)$次比较操作,设清除了 k_2 个 P 的支配个体,其中 $k_2\geq 0$;以此类推,产生第 m 个非支配个体时做了$(N-k_1-k_2-\ldots-k_{m-1}-m)$次比较操作,设清除了 k_m 个 P 的支配个体,其中 $k_m\geq 0$.由于 m 个非支配个体全部产生后,所有的支配个体被清除,故$(k_1+k_2+\ldots+k_m)=N-m$.因此,算法在最坏情况下的时间复杂度为

$$\begin{aligned}T(N)&=(N-1)+(N-k_1-2)+\ldots+(N-k_1-k_2-\ldots-k_{m-1}-m)\\&=(N-1)+(N-2)+\ldots+(N-m)-[k_1+(k_1+k_2)+\ldots+(k_1+k_2+\ldots+k_{m-1})]\\&<(2N-m-1)m/2<Nm,\end{aligned}$$

即算法在最坏情况下的时间复杂度为 $O(rmN)$,其中,r 为优化目标的数目.

由于在一般情况下有 $m<N$,因此,AP 在 m 较小时具有比 $O(rN^2)$[10]和 $O(N\log^{(r-1)}N)$[12]更好的运行效率.

3.3 构造非支配集实例

下面我们给出一个具体实例,说明用擂台赛法则是如何构造非支配集的.

例 1:考虑一个具有两个目标,20 个个体的进化群体,这 20 个个体的定义如下:

$C_1=(9,1)$,$C_2=(7,2)$,$C_3=(5,4)$,$C_4=(4,5)$,$C_5=(3,6)$,$C_6=(2,7)$,$C_7=(1,9)$,$C_8=(10,1)$,$C_9=(8,5)$,$C_{10}=(7,6)$,

$C_{11}=(5,7)$,$C_{12}=(4,8)$,$C_{13}=(3,9)$,$C_{14}=(10,5)$,$C_{15}=(9,6)$,$C_{16}=(8,7)$,$C_{17}=(7,9)$,$C_{18}=(10,6)$,$C_{19}=(9, 7)$,$C_{20}=(8,9)$.

这里,个体 $C_i=(f_1,f_2)$的两个目标值分别为 f_1 和 f_2,$i=1,2,\ldots,20$.给定初始集$\{C_{15},C_{16},C_{11},C_6,C_8,C_{13},C_1,C_9,C_{17},C_{10},C_7,C_3,C_{12},C_2,C_{14},C_{18},C_4,C_{20},C_5,C_{19}\}$,用擂台赛法则构造其非支配集的过程描述如下:

Round 1#:将 C_{15} 选为擂主,依次与其他个体比较.在此过程中发现 C_{15} 被 C_1 所支配,因此,C_1 代替 C_{15} 成为新的擂主,同时发现 C_1 也是该轮比较的最后一个擂主.第 1 轮比较后,得到 C_1 是非支配个体,同时得到新的构造集为$\{C_{16},C_{11},C_6,C_8,C_{13},\boxed{C_1},C_9,C_{17},C_{10},C_7,C_3,C_{12},C_2,C_4,C_{20},C_5\}$.

在下一轮比较时,C_1 将与前 5 个个体进行比较,并清除被它所支配的个体.值得注意的是,这里 C_1 如果不与当前进化群体的前 5 个个体比较,个体 C_8 就不可能被清除,这样一来,C_8 就被当作非支配个体加入到非支配集中.造成这种情况的主要原因是 C_8 只被 C_1 一个个体所支配,同时,因为在上一轮比较中发生了由 C_{15} 到 C_1 的替代操作.

Round 2#:将 C_{16} 选为擂主,依次与其他个体比较.在该轮比较中,C_1 必须与当前进化群体的前 5 个个体进行比较:$\{C_{16},C_{11},C_6,C_8,C_{13}\}$,$C_8$ 被清除,因它被 C_1 所支配.该轮比较出现了两次替代操作,第 1 次为 C_{11} 替代 C_{16},第 2 次为 C_6 替代 C_{11}.第 2 轮比较后的结果为$\{\boxed{C_6},C_9,C_{10},C_7,C_3,C_2,C_4,C_5\}$,$C_6$ 为非支配个体,加入到非支配集中.由于最后一轮替代操作发生时,其前没有被保留下来的个体,所以 C_6 在下一轮不必与其他个体比较.

Round 3#:将 C_9 选为擂主,依次与其他个体比较.该轮比较中发现 C_9 被 C_3 所支配,所以 C_9 由 C_3 替代,这是唯一的一次替代操作,替代操作前有两个个体$\{C_{10},C_7\}$.第 3 轮比较后的结果为$\{C_{10},C_7,\boxed{C_3},C_2,C_4,C_5\}$,$C_3$ 为非支配个体,被加入到非支配集中.

Round 4#:因为上一轮的最后一个擂台主 C_3 要加入该轮比较,选取该轮第 1 个擂主的原则是第 1 个不被 C_3 所支配的个体,在此为 C_7.该轮比较没有发生替代操作,结果为{C_2,C_4,C_5},C_7 为该轮产生的非支配个体.

Round 5#:将 C_2 选为擂主,依次与其他个体比较.该轮比较没有发生替代操作,结果为{C_4,C_5},C_2 为该轮产生的非支配个体.

Round 6#:将 C_4 选为擂主,此时,构造集中只剩下个体 C_5,比较时互不相关,从而得出结果:C_4 和 C_5 均为非支配个体.结束.

在以上 6 轮共 46 次比较后(比 mN=7×20 少),得到非支配集为{C_1,C_6,C_3,C_7,C_2,C_4,C_5}.

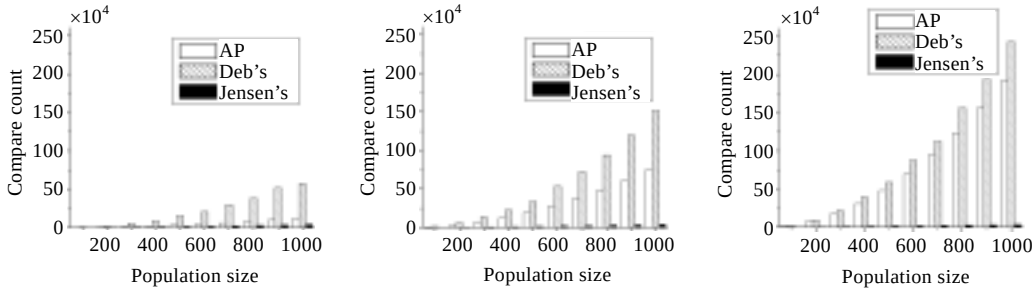
4 实验结果

为了验证前面的分析,我们做了 3 类实验:一类是构造非支配集的比较实验,主要是测试擂台赛法则在不同比例的非支配个体情况下构造非支配集的效率;第 2 类实验采用 NSGA-II 求解标准测试函数,测试进化过程中种群的非支配个体比例;第 3 类实验是求解标准测试函数的 CPU 时间的比较实验.实验环境为 Intel(R) Pentium(R) 4 CPU 1.70GHz, 256MB 内存;实验结果为 20 次重复实验的统计结果.

4.1 构造非支配集的比较实验

这里所讨论的构造非支配集的比较实验,主要是为了测试 AP 在不同比例的非支配个体情况下构造非支配集的效率,并分别与 Deb 及 Jensen 的方法进行比较.实验中,子目标数 r 分别取值为 2,5,8 和 10.针对不同的子目标,m/N 分别取值为 20%,50%和 80%,其中,N 为群体的规模,m 为进化群体中实际的非支配个体的数目.图 1~图 4 为不同子目标数的实验结果.子目标数相同的情况下,图的纵坐标(比较次数)取值区间是相同的,这样就可以很清楚地看到 3 种算法在不同非支配个体比例下(目标数、群体规模相等),关键操作的比较次数.

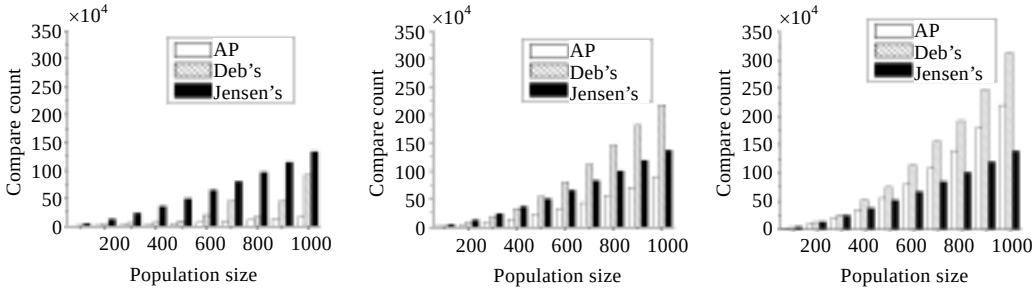
从图 1~图 4 中可以发现:(1) Jensen 的算法在不同非支配个体比例下,其比较次数基本上是一致的,而 AP 和 Deb 的算法则受非支配个体比例的影响较大,非支配个体比例越大,比较的次数越多;(2) 在子目标数较少时(r=2),Jensen 算法的比较次数明显小于其他两种算法,这是因为在两个目标下,其时间复杂度为 $O(N\log N)$;(3) 在 r=5,m/N=20%的情况下,AP 的比较次数是最少的,而且 Deb 的算法也比 Jensen 的要好,但是,随着非支配个体比例的增加,AP 和 Deb 的算法的比较次数逐步增加,所以 Jensen 的算法逐渐占据了优势;(4) Jensen 算法的时间复杂度随目标数的增加而增大,当子目标数 r 较大时(r=8,r=10),AP 和 Deb 的算法都要优于 Jensen 的算法,其中,AP 是 3 种算法中比较次数最少的,而且随着目标数的增加,AP 的优势会更加明显.



(a) r=2,m/N=20% (b) r=2,m/N=50% (c) r=2,m/N=80%

Fig.1 Comparison of experimental results

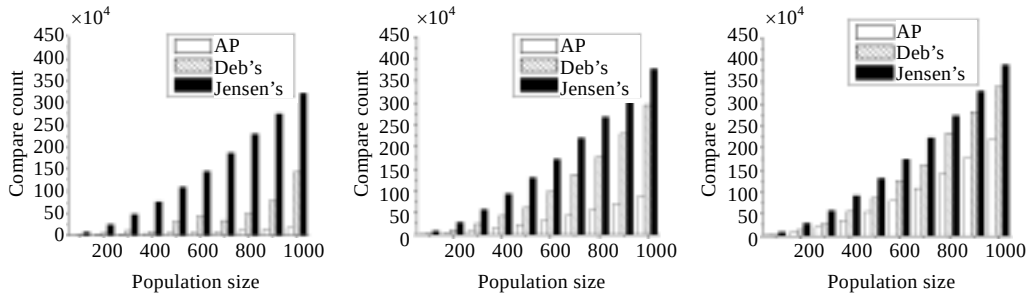
图 1 对比实验结果



(a) r=5,m/N=20% (b) r=5,m/N=50% (c) r=5,m/N=80%

Fig.2 Comparison of experimental results

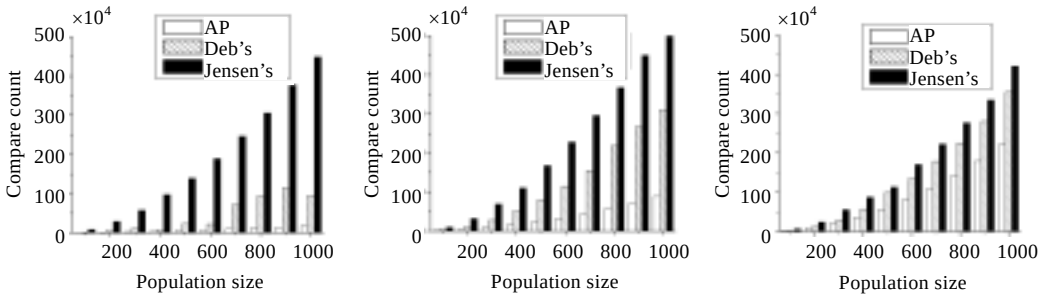
图 2 对比实验结果



(a) r=8,m/N=20% (b) r=8,m/N=50% (c) r=8,m/N=80%

Fig.3 Comparison of experimental results

图 3 对比实验结果



(a) r=10,m/N=20% (b) r=10,m/N=50% (c) r=10,m/N=80%

Fig.4 Comparison of experimental results

图 4 对比实验结果

4.2 Benchmark问题的测试实验

前面的分析及实验结果表明,擂台赛法则受非支配个体比例的影响.那么,我们先看看多目标进化算法在求解实际的测试问题中,每一代进化时实际的非支配个体比例.我们选用 NSGA-II[10]求解一系列标准测试函数(不同子目标数)并记录下每代中种群的非支配个体比例,测试函数选用的是 DTLZ 系列的 DTLZ1~DTLZ6[14],因为据我们所知,目前只有这个系列中的测试函数可以设置为含有任意的目标数.在文献[15]中给出了建议的归档集大小,当目标数增加时,要求归档集的大小相应地增大,目标数较大时计算量非常大.由于计算机性能的限制,对每个函数,我们只取 r=2,r=3,r=5 进行测试,基于文献[15]的建议,其归档集的大小分别取 100,800,5 000,结果如图 5 所示,我们发现,对于这些标准测试问题,2 个目标时,群体中非支配个体比例在 50%左右;在目标数较多的情

况下(r=3,r=5),群体中非支配个体的比例会有所上升,但是一般在 60%左右.值得说明的是,当 r=5 时,如果群体规模设置得更大一些(在文献[15]建议的范围内),非支配个体的比例会有所下降.

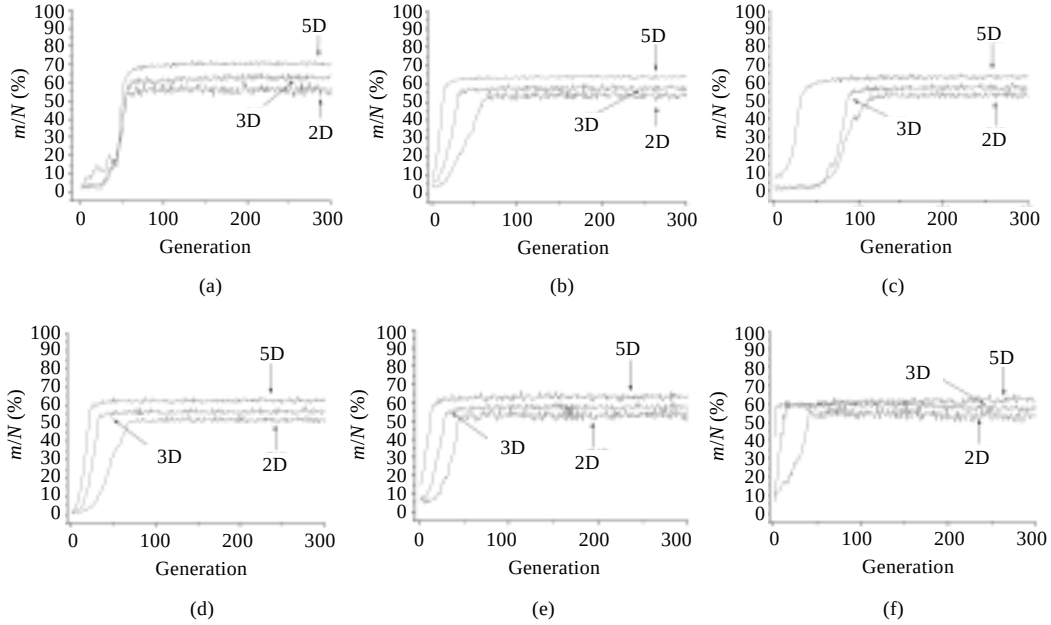


Fig.5 The proportion of non-dominated individuals of DTLZ1~DTLZ6 (from (a) to (f) respectively)

图 5 测试函数 DTLZ1~DTLZ6(从(a)到(f))计算过程中的非支配个体比例

4.3 Benchmark问题的比较实验

从前面两部分的实验我们可以得出如下结论:AP 受非支配个体比例的影响,而实际测试问题中每代的非支配个体比例不会达到 100%,一般在 60%左右.由此可见,AP 比 Deb 的算法要好.另外,随着子目标数的增加,AP 与 Jensen 算法的差距在减小,当子目标数较大时(即 r≥5 时),AP 构造非支配集的效率超过了 Jensen 的算法.

本节再基于 CPU 时间来比较 3 种算法的效率,我们将 AP 和 Jensen 的算法集成到 NSGA-II[10]中,然后比较(AP+NSGA-II)与(Jensen's+NSGA-II)及 NSGA-II 在 benchmark 测试问题上的 CPU 时间.这 6 个 benchmark 问题分别是我们上一节使用的 DTLZ1~DTLZ6[14].

对每一个问题,我们设置 3 种不同的目标数目,并且对每一种目标数配置相应的群体大小和总进化代数,见表 1.统计结果如图 6~图 8 所示,当目标数较少时(r=2,r=3),NSGA-II 的 CPU 时间较多,而(AP+NSGA-II)和(Jensen's+NSGA-II)的差别不大,当目标数较大时(r=5),(Jensen's+NSGA-II)的 CPU 时间最多,NSGA-II 居中,(AP+NSGA-II)的 CPU 时间最少.从该实验中可见,在 r=2 和 r=3 时,(Jensen's+NSGA-II)在 CPU 时间上也没有优势,而当 r=5 时,(Jensen's+NSGA-II)的 CPU 时间远高于(AP+NSGA-II),这主要是因为 Jensen 的算法是一个递归的函数.因此,从耗费的 CPU 时间来看,AP 的效率是最高的.

Table 1 The setting of evolutionary parameters

表 1 进化参数设置

Objective	2	3	5
Population	100	800	5 000
Generation	200	300	400

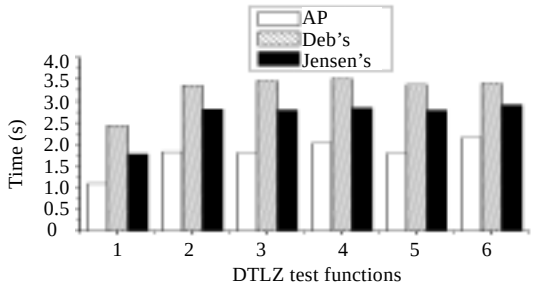


Fig.6 The CPU time of DTLZ1~DTLZ6 (from left to right) on 2 dimensions

图 6 3 种算法求解 2 维下 DTLZ1~DTLZ6 (从左到右)的 CPU 时间比较

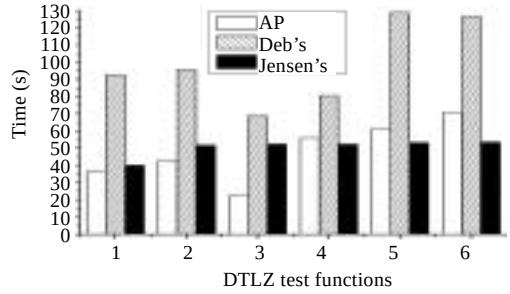


Fig.7 The CPU time of DTLZ1~DTLZ6 (from left to right) on 3 dimensions

图 7 3 种算法求解 3 维下 DTLZ1~DTLZ6 (从左到右)的 CPU 时间比较

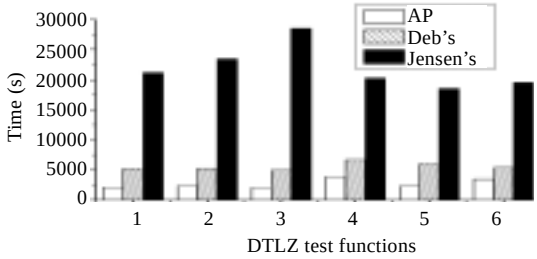


Fig.8 The CPU time of DTLZ1~DTLZ6 (from left to right) on 5 dimensions

图 8 3 种算法求解 5 维下 DTLZ1~DTLZ6(从左到右)的 CPU 时间比较

5 结 论

基于 Pareto 的多目标进化,是通过在每一步迭代(即每一代进化)中构造进化群体的非支配集,并使之不断逼近 True Pareto 最优解集来实现的.因此,构造非支配集的方法,直接影响到 MOEA 的运行效率.本文讨论了用擂台赛法则构造多目标进化群体的 Pareto 最优解集的方法,论证了构造方法的正确性,分析了其时间复杂性为 $O(rmN)$,其中 N 为进化群体的规模大小,m 为群体中实际的非支配个体的数目,r 为目标数.在多目标进化过程中,每一代进化实际的非支配个体数 m 是动态变化的,初始时比较小,随进化而不断增大(但不一定是线性的),进化到一定程度时趋向于稳定,第 4.2 节的实验结果表明,m/N 值一般稳定在 60%左右.理论上,AP 优于 Deb 的算法($O(rN^2)$),因为有 $m<N$;与 Jensen 的算法($O(N\log^{(r-1)}N)$)相比,当目标数较少时(如 r=2),Jensen 的算法明显优于 AP 和 Deb 的算法,但当目标数 r 增加时,AP 和 Deb 的算法逐步具有优势,当目标数 r 增加到一定值时,AP 和 Deb 的算法均将超过 Jensen 的算法(如第 4.1 节实验结果,r=8 时).将 AP 和 Jensen 的算法集成到 NSGA-II 中,比较(AP+NSGA-II)与(Jensen's+NSGA-II)及 NSGA-II 在 benchmark 测试问题上的 CPU 时间,实验结果表明,AP 具有最好的运行效率,即使在目标数 r 较小时(如 r=2),Jensen 的算法也比不上 AP,这主要是因为 Jensen 的算法是一个递归函数.

此外,AP 可以集成到任何基于 Pareto 的 MOEA 中,并能在较大程度上提高 MOEA 的运行效率.如果能将 AP 与一个好的保持进化群体分布性的方法有机地结合起来,将会进一步改善 MOEA 在运行效率与分布性等方面的性能.

References:

[1] Coello Coello CA, Van Veldhuizen DA, Lamont GB. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Acedemic/Plenum Publishers, 2002.

[2] Coello Coello CA, Lamont GB. Applications of Multi-Objective Evolutionary Algorithms. Singapore: World Scientific, 2004.

[3] Corne DW, Jerram NR, Knowles JD, Oates MJ. PESA-II: Region-Based selection in evolutionary multiobjective optimization. In: Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2001). Morgan Kaufmann Publishers, 2001. 283−290.

[4] Knowles JD, Corne DW. Approximating the nondominated front using the Pareto archived evolution strategy evolutionary computation. Evolutionary Computation, 2000. 149−172.

[5] Aguirre AH, Rionda SB, Coello Coello CA, Lizáraga GL, Montes EM. Handling constraints using multiobjective optimization concepts. Int'l Journal for Numerical Methods in Engineering, 2004,59(15):1989−2017.

[6] Fonseca CM, Fleming PJ. An overview of evolutionary algorithms in multi-objective optimization. Evolutionary Computation, 1995,3(1):1−16.

[7] Horn J, Nafpliotis N, Goldberg DE. A niched Pareto genetic algorithm for multiobjective optimization. In: Proc. of the 1st IEEE Conf. on Evolutionary Computation. Piscataway: IEEE Service Center, 1994. 82−87.

[8] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Trans. on Evolutionary Computation, 1999,3(4):257−271.

[9] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou K, *et al*., eds. Proc. of the EUROGEN 2001—Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems. 2001. 95−100.

[10] Deb K, Pratap A, Agrawal S, Meyrivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002,6(2):182−197.

[11] Deb K, Agrawal S, Pratab A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. KanGAL Report, 200001, Kanpur: Indian Institute of Technology, 2000.

[12] Jensen MT. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. IEEE Trans. on Evolutionary Computation, 2003,7(5):503−515.

[13] Zheng JH, Ling C, Shi ZZ, Xue J, Li XY. A multi-objective genetic algorithm based on quick sort. In: Tawfik AY, Goodwin SD, eds. Proc. of the 17th Canadian Conf. on Artificial Intelligence. London: Springer-Verlag, 2004. 175−186.

[14] Deb K, Thiele L, Laumanns M, Zitzler E. Scalable test problems for evolutionary multiobjective optimization. In: Yao X, ed. Proc. of the 2002 Congress on Evolutionary Computation. New Jersey: IEEE Service Center, 2002. 825−830.

[15] Knowles JD, Corne DW, Fleischer M. Bounded archiving using the lebesgue measure. In: Proc. of the 2003 Congress on Evolutionary Computation (CEC 2003), Vol. 4. Canberra: IEEE Press, 2003. 2490−2497.

郑金华(1963－),男,湖南邵东人,博士,教授,博士生导师,主要研究领域为遗传算法,多目标进化算法及其应用.

邝达(1980－),男,助教,主要研究领域为遗传算法,多目标进化算法及其应用.

蒋浩(1982－),男,硕士,主要研究领域为遗传算法,多目标进化算法及其应用.

史忠植(1941－),男,研究员,博士生导师,CCF 高级会员,主要研究领域为人工智能,机器学习,神经计算,认知科学.