文章编号: 1001-0920(2011)01-0089-07

基于 ε 占优的自适应多目标粒子群算法

刘衍民[1,2], 赵庆祯[2], 牛 奔[3], 邵增珍[2]

(1. 遵义师范学院 数学系，贵州 遵义 563002；2. 山东师范大学 管理与经济学院，济南 250014；3. 深圳大学 管理学院，广东 深圳 518060)

摘 要: 针对粒子群算法求解多目标问题极易收敛到伪 Pareto 前沿 (等价于单目标优化问题中的局部最优解), 并且收敛速度较慢的问题, 提出一种 ε 占优的自适应多目标粒子群算法 (εDMOPSO). 在 εDMOPSO 算法中, 每个粒子的邻居根据粒子的运行动态地组建, 且粒子的速度不由其邻居中运行最好的粒子来调整, 而是由其所有邻居共同调整. 同时, 采用外部存档保存非劣解, 并利用 ε 占优更新非劣解. 模拟结果表明了 εDMOPSO 算法的有效性.

关键词: 多目标优化；粒子群算法；ε占优；动态邻居

中图分类号: TP301.6 **文献标识码:** A

Adaptive multi-objective particle swarm optimizer based on ε dominance

LIU Yan-min[1,2], *ZHAO Qing-zhen*[2], *NIU Ben*[3], *SHAO Zeng-zhen*[2]

(1. Department of Mathematics，Zunyi Normal College，Zunyi 563002，China；2. School of Management and Economics，Shandong Normal University，Ji'nan 250014，China；3. College of Management，Shenzhen University, Shenzhen 518060，China. Correspondent：LIU Yan-min，E-mail：yanmin7813@163.com)

Abstract: Multi-objective particle swarm optimizers(MOPSOs) easily converge to a false Pareto front (the equivalent of a local optimum in single objective optimization), and converge slowly when applied to solve multi-objective optimization problems(MOPs). Therefore, this paper presents a self-adaptive multiobjective particle swarm optimizer based on ε-domination(εDMOPSO) to handle MOPs. In the εDMOPSO algorithm, the neighborhood of each particle is dynamically changed in terms of the performances of the particles, and the velocity of each particle is not adjusted by the best performing particle in its neighborhood, but by all particles in its neighborhood including itself. Finally, external archive is employed to store the nondominated solutions and ε-dominance is applied to update non-dominated solutions in external archive. Simulation results show the effectiveness of the proposed εDMOPSO algorithm.

Key words: multi-objective optimization；particle swarm optimizer；ε-domination；dynamic neighbor topology

1 引 言

利用扩展粒子群算法 (PSO) 处理多目标优化问题 (MOPs) 是粒子群算法领域的一个研究热点. PSO 算法是一种基于种群的进化算法, 在每次迭代过程中, 均能产生一组非劣解. 由于 PSO 操作简单, 收敛速度快, 许多学者提出采用 PSO 来求解 MOPs. 这些研究都基于以下 3 个目标: 1) 所得非劣解集尽可能接近真实的 Pareto 前沿; 2) 所得非劣解集要尽可能保持多样性分布; 3) 所得非劣解的数量要尽可能的多. 但是, 由于解的多样性分布会减慢算法的收敛速度, 甚至会降低解的质量, 造成两个目标之间的冲突[1]. 另外, PSO 算法的搜索行为是通过对每个粒子的速度进行调整 (根据粒子自身的历史最优位置和其邻居中表现最好的粒子) 来搜索整个空间, 直至收敛到真实的 Pareto 前沿, 这就需要每个粒子能够自适应地调整其邻居拓扑结构, 进而产生均匀分布的 Pareto 前沿.

基于此, 本文提出一种基于 ε 占优的自适应多目标粒子群算法 (MOPSO), 以提升 PSO 求解 MOPs 的有效性.

2 相关研究

PSO 是一种进化计算技术[2], 一些研究已表明, PSO 经过修改后可以求解 MOPs, 且产生的非劣解逼近真实的 Pareto 前沿. 文献 [3] 提出了一种 MOPSO 算

收稿日期: 2009-10-16；**修回日期:** 2010-03-21.

基金项目: 广东省自然科学基金项目(9451806001002294)；深港创新圈基金项目(200810220137A)；贵州省教育厅社科基金项目(2007018).

作者简介: 刘衍民(1978−), 男, 讲师, 博士, 从事运筹学理论、进化计算的研究；赵庆祯(1943−), 男, 教授, 博士生导师, 从事运筹学理论、进化计算等研究.

法, 该算法合并了Pareto占优的概念, 采用外部存档控制器来存储和决定每一代中哪些粒子将会成为非劣解的成员, 利用这些成员来引导其他粒子的飞行(多目标优化问题的概念: 非劣解, Pareto解, Pareto前沿, Pareto占优等, 见[4]). [5]提出了一种动态邻居拓扑结构粒子群算法, 该算法包含动态邻居、全局最好粒子更新策略和1-D优化3个标准, 但它仅能处理具有较少目标函数的问题. [6]提出了非占优排序粒子群算法(NSPSO), 该算法采用NSGA-II[7]的外部存档维持方法. [8]介绍了一种sigma(sMOPSO)方法, 该方法是粒子根据外部存档中的成员计算sigma值来选择全局领导者, 但这种寻找全局领导者的方法会导致求解一些MOPs时出现早熟现象. [9]提出将广义粒子群算法与Pareto占优相结合来处理MOPs. [10]提出了ε占优的方法, 以控制外部存档规模并且缩减了计算时间. [11]在[3]的基础上, 提出用变异因子来提升粒子的探索能力. [12]提出一种新的MOPSO算法(OMOPSO), 该算法将整个群分成3个规模相同的子群, 每个子群采用不同的变异因子, 算法极大地提升了粒子的探索能力. [13]提出了精英MOPSO(EMOPSO)算法, 其中合并了精英变异的因子, 以增加粒子的探索和搜索能力. 在最近的研究中, [14]提出了动态人口多种群MOPSO算法, 该算法应用自适应局部外部存档来提升每个种群的多样性, 进而提升每个粒子收敛到真实Pareto前沿的能力. [15]提出了一种类似于[14]的动态多群MOPSO算法, 通过分配一定数量的子群来保证算法在运行过程中收敛到真实的Pareto前沿. 总之, 不论哪种改进算法, 其搜索能力均由粒子的邻居拓扑结构来确定, 且收敛速度取决于外部存档中非劣解更新所采用的占优关系. 除了[10]所提出的算法外, 其他算法均采用Pareto占优关系更新外部存档中的非劣解, 没有采用[16]提出的ε占优方法. 在[10]中, 已经证实该方法对于提升算法的运行效率具有积极的作用; 同时, 上述算法中每个粒子的学习样本仅是它邻居中最好运行的一个粒子, 没有充分向其邻居中其他粒子学习, 以上均是本文研究的动机.

3 基于ε占优的自适应多目标粒子群算法

3.1 动态邻居拓扑结构

为了保证每个粒子都能更好地向它的邻居学习, 搜索一组均匀分布且更加接近真实Pareto前沿的非劣解, 本文采用文献[17]提出的邻居构建策略, 即每个粒子根据自身的运行自适应动态组建邻居的策略. 该策略能够增加整个群体的多样性, 使得粒子能够搜索更多的可行区域. 本文规定每个粒子的邻居选取规则为粒子间的欧氏距离, 于是当前粒子i的邻居可表示为

$$\begin{cases} L_i(t)=\{d_{ij}(t)|d_{ij}(t)=\|x_i(t)-x_j(t)\|\}, \\ \text{neighbor}_i(t)=\arg(\min(\text{sort}(L_i(t)),n)). \end{cases} \tag{1}$$

其中: $j\neq i,\ j\in \text{ps}$, ps为种群规模; $L_i(t)$为在迭代时刻t, 当前粒子i与种群中其他粒子的欧氏距离集合; $\text{neighbor}_i(t)$为在迭代时刻t, 当前粒子i所有邻居的集合; $\arg(C)$为识别C中元素的位置; n为粒子i拥有的邻居个数, 其取值通常为粒子规模的$1/4\sim 1/3$, 本文中$n=\text{ps}/3$; sort*(A)*表示对集合A中元素从小到大进行排序. 为了充分利用每个粒子邻居的信息, 规定若一个粒子的自身最优位置连续T代没有得到提升, 则需对粒子i重新构建邻居拓扑结构. 经过反复试验, 当T=7时, 取得了较好的运行结果.

3.2 粒子的学习样本选择

若一个粒子的每一维都向同一个粒子的相应维数学习, 则会造成"two steps forward, one step back"现象[18]. 因此在文献[17]提出的广义学习策略的基础上, 提出了本文的学习策略, 即一个粒子的学习样本不是它邻居中表现最好的一个粒子, 而是该粒子的所有邻居, 包括它自己. 因此, 每个粒子的速度更新过程为

$$\begin{aligned} \overline{v_i}^{t+1}&=\omega\overline{v_i}^{(t)}+\varphi_1 r_1(\overline{p_g}-\overline{x_i}^{(t)})+ \\ &\quad \varphi_2 r_2(\overline{p_{\text{bin}(i)}}-\overline{x_i}^{(t)}), \end{aligned} \tag{2}$$

$$p^d_{\text{bin}(i)}=\arg\Big\{\max\Big[\frac{A}{|p_{jd}-x_{id}|}\Big]\Big\},$$

$$A=\text{sum}(\text{Fitness}(p_j)-\text{Fitness}(p_i)),$$

$$d\in(1,2,\cdots,n),\ i=1,2,\cdots,\text{ps},\ j\in\text{neighbor}_i. \tag{3}$$

其中: $\overline{p_g}$表示群中表现最好的粒子(即在外部存档中随机选择的粒子); $\overline{p_{\text{bin}(i)}}$表示粒子$i$的每一维学习样本; neighbor_i表示粒子i的邻居构成的集合; $p^d_{\text{bin}(i)}$表示$\overline{p_{\text{bin}(i)}}$的第$d$维. 这里, $\omega=0.729$, $\varphi_1=\varphi_2=1.494\,45$. 这种粒子的学习策略使得整个群体具有更广阔的搜索空间. 在文献[17]中已经证实, 如果一个粒子速度的每一维都向其邻居中表现最好的一个粒子的相应维数学习, 则将仅有少数的粒子成为学习样本, 这极大地减少了群体的多样性及潜在的搜索空间; 反之亦然. 称这种学习策略为广泛学习策略.

3.3 基于ε占优的非劣解存储

运行效率问题是MOPSOs的研究热点之一. 文献[16]提出了一种经典Pareto占优关系的改进方法, 即ε-dominace, 它极大地提升了算法的运行效率. 其思想如下: 首先, 将整个搜索空间离散成一些方格(box), 每个非劣解属于一个特定的方格. 方格划分如下:

$$b_i = \left\lfloor \frac{\log f_i}{\log(1+\varepsilon)} \right\rfloor, \tag{4}$$

其中 $\lfloor\cdot\rfloor$ 表示 floor 函数. 在每个格子内使用 Pareto 占优关系, 这样算法总能维持一套非劣解格子. 在每个格子内, 非劣解仅能被一个优于它的解所替换. 因此根据下式:

$$|A| \leqslant \left(\frac{\log f_i}{\log(1+\varepsilon)}\right)^{m\text{-}1} \tag{5}$$

能够保证非劣解在指定的范围内. 其次, 计算外部存档中非劣解的拥挤距离[7]. 随机选取 2 个粒子, 拥挤距离大的作为学习样本.

文献 [10] 已经证实了 ε 占优关系比其他方法有更快的收敛速度. 因此, 可利用 ε 占优来更新外部存档. 最初外部存档是空的, 随着迭代的进行, 每一代所产生的非劣解 (ns) 用来更新外部存档, 每个非劣解将与当前存档文件中存在的非劣解逐一比较. 算法 1 显示了这一更新过程.

算法 1　利用 ε 占优更新 Archive 成员.

Initialize External Archive(EA).

Initialize non-dominated solutions using ε-dominance.

If new solution(ns) dominates a set of solutions $A(i)$

in EA in terms of ε-dominance.

Delete $A(i)$ and insert ns in EA

Else If ns is dominates by any solution in EA.

Discard ns.

Else If ns and the solutions of EA are nondominated

If ns and anther solution in EA share the same box and neither dominates the other.

Choose the one closer to the box.

End

End If

3.4　基于 ε 占优的自适应 MOPSO

由于外部存档中的每一个成员都是非劣解, 无法确定哪一个非劣解最优. 对于如何选取外部存档中的一个非劣解作为粒子学习样本, 学者们提出了不同的策略. 在本文提出的算法中, 随机选取一个非劣解, 这种方法操作简单而且计算复杂度小. 算法 2 显示了 εDMOPSO 的算法流程.

算法 2　εDMOPSO 算法.

Staynum=zeros(1,ps).

Calculate crowding distance of each particle.

While stopping criterion is not satisfied,

For i =1:ps

If staynum$(i) \geqslant 7$

staynum(i) =0

update particle i.s neighbor in terms of Eq.(1).

End

End

For i =1:ps

Randomly select an exemplar from EA.

Assign each dimension examplar using Eq(3).

Update particle velocity and position using Eq(2).

Update pbest.

Maintain particles in search space.

Evaluate the fitness values.

End For

Update the external archive using 算法 1.

End While

3.5　算法复杂度和收敛性分析

设 ps 是一个大小为 N 的进化群体 (粒子), 并假设 N 有 m 个非支配个体. 在一般情况下, 假设进行了 $k(m \leqslant k \leqslant N)$ 次比较, 根据 ε 占优, 每次比较过程中, 均会淘汰一个粒子, 则每次迭代共淘汰 k 个粒子, 其中包括 m 个非支配个体, $(k-m)$ 个支配个体. 经过 k 次比较后, 在 N 个个体中, 将有 $(N-k)$ 个支配个体被淘汰. 为了分析的方便, 假设每次比较 $(N-k)$ 个支配个体的概率相同, 则 k 次比较的计算复杂度为

$$\begin{aligned}
&(N-1)+\left(N-2-\frac{(N-k)}{k}\right)+\\
&\left(N-3-\frac{2(N-k)}{k}\right)+\cdots+\\
&\left(N-k-\frac{(k-1)(N-k)}{k}\right)=\\
&\frac{[(N-1)+(N-k)]k}{2}-\\
&\left[\frac{(N-k)}{k}+\frac{k(N-k)}{k}\right]\frac{k}{2}+(N-k)=\\
&\frac{k(N-1)}{2}+\frac{(N-K)}{2}\leqslant N^2.
\end{aligned}$$

同时, 在应用 ε 占优更新外部存档过程中, 其拥挤距离的计算复杂度为 $O(M(2N)\log(2N))$, 则总的计算复杂度约为 $O(MN^2)$, 其中 M 是目标的个数. 表 6 表明了 εDMOPSO 算法的计算复杂度优于其他算法.

对于多目标优化问题, 如果满足下述条件: 1) Pareto 最优边界的格子计数维数不大于 $r-1$(r 为目标数); 2) $P_{\mathrm{know}(0)}$, $P_{\mathrm{know}(1)}$,$\cdots$ 是单调递增的, 则 εDMOPSO 以概率 1 收敛, 即

$$\mathrm{Prob}(\lim_{t\to\infty}\{P_{\mathrm{true}} = P_{\mathrm{know}(t)}\}) = 1.$$

原因如下: 将算法的运行过程看作一个Markov链, 包含两种状态, 即 $P_{\text{known}(t)} = P_{\text{true}}$ 成立或不成立. 由条件2)可知, 从第1个状态到第2个状态的转移概率是0; 由条件1) 可知, 从第2个状态到第1个状态的转移概率大于0, 表明此状态是瞬时的. 在这两种情况下, εDMOPSO完全满足Markov定理[4], 即εDMOPSO以概率1收敛.

4 仿真实验及其分析

4.1 检测函数及算法的参数设置

为了测试εDMOPSO算法, 选取5个检测函数: ZDT1, ZDT2, ZDT3, ZDT4和ZTD6, 每个检测函数含有两个目标函数和30维变量[13-14]. 将本文提出的εDMOPSO算法与MOPSO[11], OMOPSO[12], EM-MOPSO[13], sMOPSO[8], NSPSO[6]进行比较, 所有算法的迭代次数均为200, 除εDMOPSO外存档规模均为100. MOPSO, OMOPSO, EM-MOPSO和εDMOPSO的粒子数为100, sMOPSO和NSPSO的粒子数为200, 其他参数设置与各种算法提出时所用参数设置一致[6,8,11-12].

4.2 评价指标

为了验证算法的有效性, 利用分布指标(Δ)[7]和收敛性指标(γ)[11]对不同的算法运行评价, 有

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{N-1} |d_i - \overline{d}|}{\sum_{m=1}^{M} d_m^e + (N-1)\,\overline{d}}, \tag{6}$$

$$\gamma = \sqrt{\sum_{i=1}^{N} d_i^2} \Big/ N. \tag{7}$$

其中: N为算法所得非劣解个数; d_i为每个非劣解与最接近的真实Pareto解间的欧式距离; d_m^e为Pareto前沿极端解和算法所得非劣解间所对应的第m个目标函数的欧式距离; $\overline{d}$为d_i的平均值. γ越小表示算法的收敛性越好, Δ越小表示所得非劣解有越好的分布.

4.3 实验结果分析

表1~表5给出了各种算法的评价指标, 表6给出了在γ =0.001时, 每种算法需要的函数评价次数(FEs)和运行时间, 运行最好的结果用黑斜体字标出. 采用Wilcoxon秩和检验对εDMOPSO算法所得结果与其他5种算法中最好的结果进行检验, 以验证εDMOPSO算法所得结果与其他5种算法所得结果的差别是否具有统计学意义. 检验结果列于表1~表5右端, 其中$h = 1$表示两种算法在95%的置信区间内结果具有差别; $h = 0$表示两种算法的结果没有差别. 图1~图5给出了6种算法在不同检测函数上的

表 1 ZDT 1评价参数比较

	MOPSO		OMOPSO		EM-MOPSO		sMOPSO		NSPSO		εDMOPSO		***h***	
	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ
Best	0.073	0.58	0.061	0.56	0.002 3	0.22	0.071	0.56	0.081	0.66	0.003 2	0.21		
Worst	0.231	0.94	0.123	0.67	0.005 8	0.47	0.163	0.77	0.243	0.77	0.004 8	0.46		
Average	0.098	0.66	0.069	0.59	0.005 1	0.39	0.089	0.64	0.139	0.69	***0.003 9***	***0.37***	1	1
Median	0.097	0.66	0.067	0.58	0.004 9	0.39	0.087	0.63	0.138	0.68	0.003 8	0.37		
Variance	6.17e-04	7.23e-03	4.12e-05	6.36e-03	6.82e-07	1.57e-04	5.61e-04	3.31e-04	4.17e-03	6.23e-03	4.61e-07	2.36e-05		

表 2 ZDT 2评价参数比较

	MOPSO		OMOPSO		EM-MOPSO		sMOPSO		NSPSO		εDMOPSO		***h***	
	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ
Best	0.114	0.58	0.005 9	0.56	0.003 8	0.26	0.069	0.34	0.071	0.46	0.004 6	0.26		
Worst	0.581	0.88	0.014 6	0.67	0.005 4	0.33	0.121	0.74	0.137	0.73	0.011 3	0.63		
Average	0.273	0.87	0.007 3	0.59	***0.004 6***	***0.28***	0.076	0.59	0.099	0.63	0.006 1	0.35	0	0
Median	0.246	0.86	0.007 2	0.58	0.004 5	0.27	0.075	0.58	0.098	0.62	0.006 1	0.35		
Variance	5.77e-02	4.75e-02	4.63e-04	5.63e-03	6.83e-08	5.77e-06	5.94e-04	4.64e-03	5.21e-03	6.18e-03	4.58e-06	2.27e-06		

表 3 ZDT 3评价参数比较

	MOPSO		OMOPSO		EM-MOPSO		sMOPSO		NSPSO		εDMOPSO		***h***	
	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ
Best	0.141	0.46	0.124	0.57	0.007 1	0.56	0.137	0.31	0.162	0.46	0.001 1	0.23		
Worst	0.274	0.78	0.243	0.68	0.009 6	0.66	0.262	0.88	0.357	0.57	0.002 9	0.42		
Average	0.189	0.51	0.159	0.59	0.007 3	0.57	0.174	0.54	0.231	0.51	***0.001 7***	***0.32***	1	1
Median	0.184	0.51	0.153	0.58	0.007 3	0.56	0.169	0.51	0.198	0.51	0.001 6	0.31		
Variance	3.51e-03	1.16e-03	1.45e-03	2.56e-03	6.8e-06	1.29e-03	1.59e-03	5.12e-03	1.97e-02	8.65e-03	6.13e-08	1.68e-04		

表 4　ZDT 4 评价参数比较

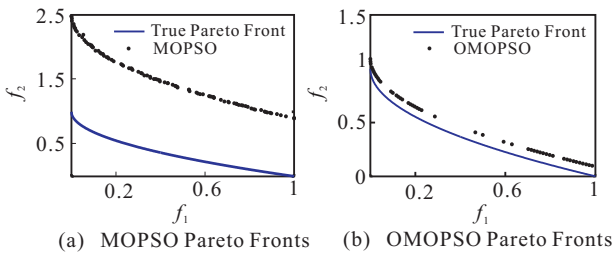
	MOPSO		OMOPSO		EM-MOPSO		sMOPSO		NSPSO		εDMOPSO		***h***	
	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ
Best	0.364	0.78	0.092	0.76	0.099	0.46	0.281	0.65	0.391	0.75	0.003 3	0.18		
Worst	0.872	0.97	0.255	0.96	0.601	0.86	0.734	0.89	0.794	0.94	0.006 8	0.67		
Average	0.867	0.92	0.198	0.81	0.427	0.74	0.656	0.81	0.676	0.84	***0.006 3***	***0.41***	1	1
Median	0.855	0.91	0.197	0.81	0.424	0.73	0.653	0.81	0.633	0.83	0.006 3	0.41		
Variance	6.53e-02	8.61e-02	6.43e-03	8.92e-02	6.77e-03	9.46e-02	7.28e-03	1.52e-02	9.84e-03	7.16e-02	9.42e-07	1.02e-05		

表 5　ZDT 6 评价参数比较

	MOPSO		OMOPSO		EM-MOPSO		sMOPSO		NSPSO		εDMOPSO		***h***	
	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ	γ	Δ
Best	0.159	0.69	0.008 8	0.79	0.005 9	0.39	0.102	0.41	0.202	0.42	0.002 6	0.23		
Worst	0.645	0.98	0.010 2	0.93	0.009 9	0.93	0.143	0.77	0.248	0.76	0.007 8	0.65		
Average	0.447	0.87	0.009 9	0.69	0.006 4	0.64	0.115	0.65	0.215	0.68	***0.006 3***	***0.41***	1	1
Median	0.446	0.83	0.009 1	0.68	0.006 3	0.63	0.114	0.62	0.214	0.66	0.006 3	0.41		
Variance	5.28e-02	6.32e-02	5.76e-06	7.25e-05	4.66e-07	7.52e-05	6.37e-02	5.78e-02	4.37e-02	5.48e-02	1.17e-07	9.78e-06		

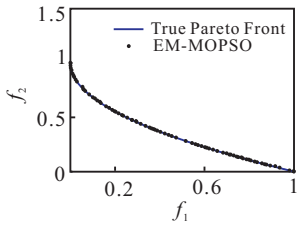
表 6　MOPSOs 算法达到 $\gamma = 0.001$ 时所需的平均函数评价次数和时间

	γ =0.001	MOPSO	OMOPSO	EM-MOPSO	sMOPSO	NSPSO	εDMOPSO
ZDT1	ANE	751 0	16 140	2 100	500 000	500 000	***2 060***
	Time(min	0.95	0.51	0.28	91.06	19.69	***0.25***
ZDT2	ANE	4 572	9 060	***4 360***	500 000	500 000	5 100
	Time/min	0.74	0.61	***0.58***	86.80	20.85	0.98
ZDT3	ANE	500 000	500 000	2 706.7	500 000	500 000	***2 130***
	Time/min	84.77	15.53	0.351	86.73	22.02	***0.29***
ZDT4	ANE	9 580	500 000	500 000	500 000	500 000	***9 200***
	Time/min	3.42	13.57	8.17	87.91	20.77	***3.29***
ZDT6	ANE	5 340	23 880	6 776.7	500 000	500 000	***3 760***
	Time/min	0.7	0.63	0.937	82.19	24.42	***0.61***

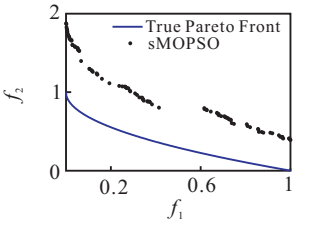


(a) MOPSO Pareto Fronts

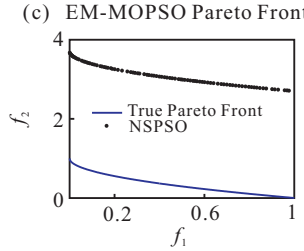
(b) OMOPSO Pareto Fronts



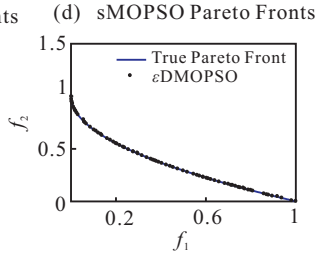
(c) EM-MOPSO Pareto Fronts



(d) sMOPSO Pareto Fronts

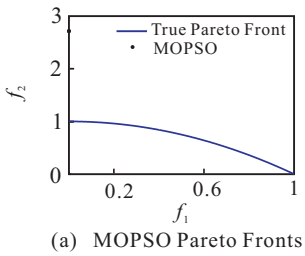


(e) NSPSO Pareto Fronts

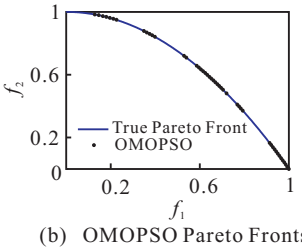


(f) εMOPSO Pareto Fronts

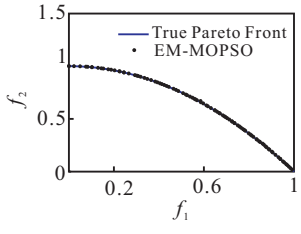
图 1　ZDT1 函数帕累托前沿



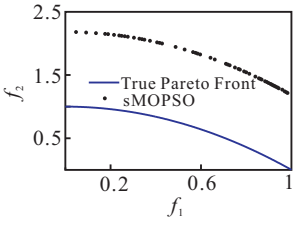
(a) MOPSO Pareto Fronts



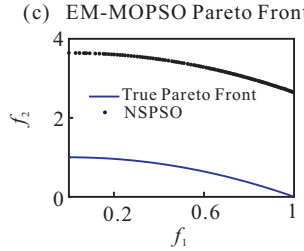
(b) OMOPSO Pareto Fronts



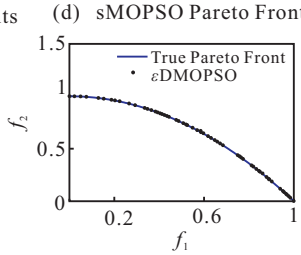
(c) EM-MOPSO Pareto Fronts



(d) sMOPSO Pareto Fronts

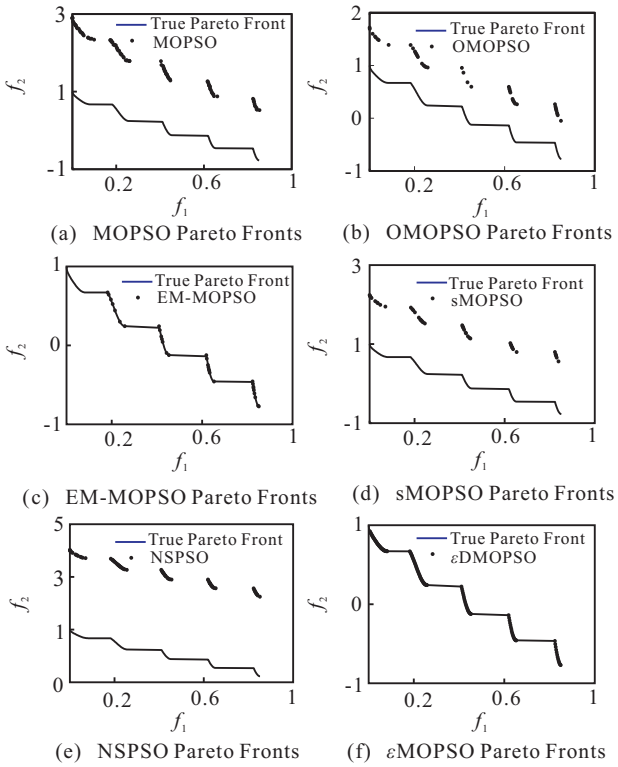


(e) NSPSO Pareto Fronts



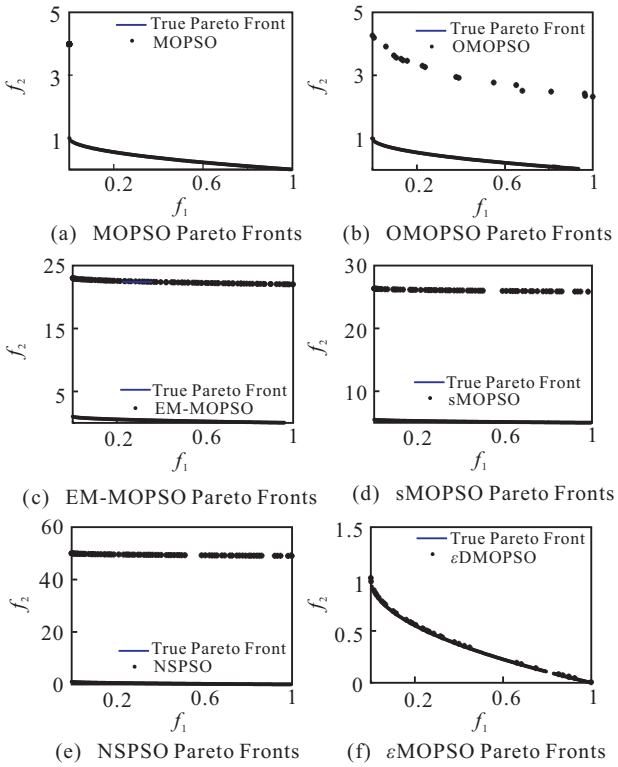
(f) εMOPSO Pareto Fronts

图 2　ZDT2 函数帕累托前沿



(a) MOPSO Pareto Fronts (b) OMOPSO Pareto Fronts
(c) EM-MOPSO Pareto Fronts (d) sMOPSO Pareto Fronts
(e) NSPSO Pareto Fronts (f) εMOPSO Pareto Fronts

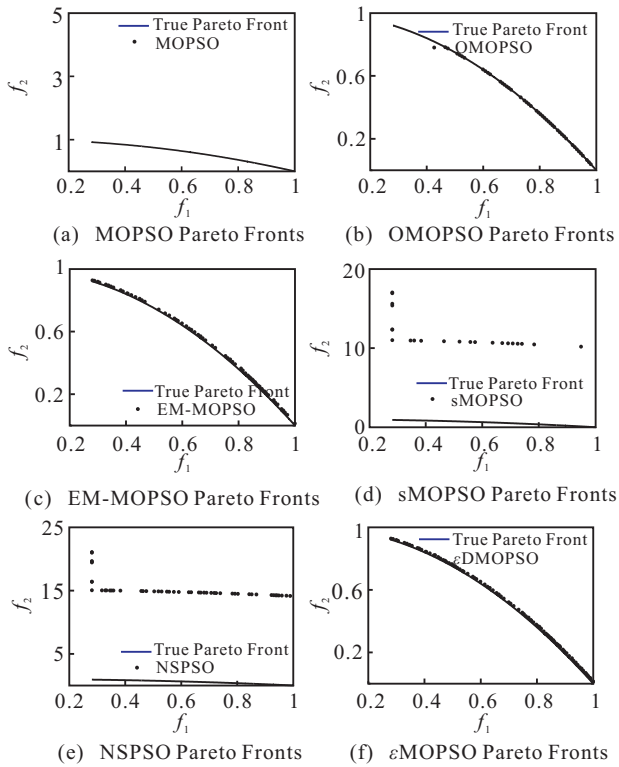
图 3 ZDT3 函数帕累托前沿



(a) MOPSO Pareto Fronts (b) OMOPSO Pareto Fronts
(c) EM-MOPSO Pareto Fronts (d) sMOPSO Pareto Fronts
(e) NSPSO Pareto Fronts (f) εMOPSO Pareto Fronts

图 4 ZDT4 函数帕累托前沿

Pareto 前沿图 (f_1, f_2 表示目标函数值). 算法采用标准 Matlab 语言实现, 每个检测函数独立运行 30 次. 由图 1～图 5 可以看出, εDMOPSO 算法在所有的检测函数上均能收敛到真实的 Pareto 前沿. 除了函数 ZDT4, EM-MOPSO 算法所产生的 Pareto 前沿类似于 εDMOPSO算法. OMOPSO算法在函数 ZDT2 和 ZDT6



(a) MOPSO Pareto Fronts (b) OMOPSO Pareto Fronts
(c) EM-MOPSO Pareto Fronts (d) sMOPSO Pareto Fronts
(e) NSPSO Pareto Fronts (f) εMOPSO Pareto Fronts

图 5 ZDT6 函数帕累托前沿

上产生了较好分布的 Pareto 前沿. 而算法 MOPSO, sMOPSO 和 NSPSO 表现最差. 同时, 从表 1～表5 也可以看出, εDMOPSO 算法的评价指标 (γ 和 Δ) 在检测函数 ZDT1, ZDT3, ZDT4 和 ZDT6 上均获得了最小值, 而 EM-MOPSO算法在 ZDT2函数上获得了最小值. 从 Wilcoxon 秩和检验来看, 除了函数 ZDT2, εDMOPSO 算法显著地优于其他 5 种算法.

为了测试 εDMOPSO 算法是否增加了计算复杂度, 应用 Matlab 函数 (tic 和 toc) 来计算当收敛性指标 γ =0.001 时, 各种算法所需要的 FEs 和运行时间. 由表 6 可以看出, εDMOPSO 算法在函数 ZDT1, ZDT3, ZDT4 和 ZDT6上, 需要最少的FEs和时间.

5 结 论

考虑 PSO 算法求解多目标优化问题极易收敛到伪 Pareto 前沿 (等价于单目标优化问题中的局部最优解) 和收敛速度较慢等缺陷, 本文提出基于 ε 占优的自适应 MOPSO 算法, 通过动态组建每个粒子的邻居和广泛的学习策略, 使得粒子有能力向真实的 Parero 前沿收敛. 同时, 通过 ε 占优关系的应用, 极大地提升了算法的运行效率. 从实验结果可以看出, εDMOPSO 算法是求解多目标优化问题的一种有效方法. 但是, PSO 算法是一种基于迭代的优化工具, 因此具有一定的随机性. 为了更好地证明 εDMOPSO 算法的有效性, 将来的工作主要集中于以下两点: 1) 从理论上证明算法的有效性; 2) 用更复杂的多维目标检测函数来验证 εDMOPSO 算法的有效性.

参考文献(References)

[1] Yen G G, Lu H. Dynamic multiobjective evolutionary algorithm: Adaptive cell-based rank and density estimation[J]. IEEE Trans on Evolutionary Computation, 2003, 7(3): 253-274.

[2] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Piscataway, 1995: 1942-1948,

[3] Coello C, Lechuga M S. MOPSO: A proposal for multiple objective particle swarm optimization[C]. Proc of IEEE Congress on Evolutionary Computation. Piscataway, 2002: 1051-1056.

[4] 郑金华. 多目标进化算法及其应用[M]. 北京: 科学出版社, 2007.
(Zheng J H. Multiobjective evolutionary algorithms and applications[M]. Beijing: Science Press, 2007.)

[5] Hu X, Eberhart R. Multiobjective optimization using dynamic neighborhood particle swarm optimizer[C]. Proc of the 2002 Congress on Evolutionary. Honolulu, 2002: 1677-1681.

[6] Li X. A non-dominated sorting particle swarm optimizer for multi-objective optimization[C]. Genetic and Evolutionary Computation Conf. Chicago, 2003: 37-48.

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

[8] Mostaghim M S, Teich J. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)[C]. IEEE Swarm Intelligence Symposium. Indianapolis, 2003: 26-33.

[9] Huang V L, Suganthan P N, Liang J J. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems[J]. Int J of Intelligent Systems, 2006, 21(3): 209-226.

[10] Mostaghim S, Teich J. The role of ε-dominance in multi objective particle swarm optimization methods[C]. IEEE Congress on Evolutionary Computation. Canberra, 2003: 1764-1771.

[11] Coello C, Pultdo G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 256-279.

[12] Sierra M R, Coello C. Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance[C]. Third Int Conf on Evolutionary Multi-Criterion Optimization. Guanajuato, 2005: 505-519.

[13] Reddy M J, Kumar D N. An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design[J]. Engineering Optimization, 2007, 39(1): 49-68.

[14] Leong W F, Yen G G. PSO-based multiobjective optimization with dynamic population size and adaptive local archives[J]. IEEE Trans on Systems, Man and Cybernetics－Part B: Cybernetics, 2008, 38(5): 1270-1293.

[15] Yen G G, Leong W F. Dynamic multiple swarms in multiobjective particle swarm optimization[J]. IEEE Trans on Systems, Man and Cybernetics－Part A: Systems and Humans, 2009, 39(4): 890-911.

[16] Laumanns M, Thiele L. Combining convergence and diversity in evolutionary multi-objective optimization[J]. Evolutionary Computation, 2002, 10(3): 263-282.

[17] 刘衍民, 赵庆祯. 一种基于动态邻居和变异因子的粒子群算法[J]. 控制与决策, 2010, 25(7): 968-974.
(Liu Y M, Zhao Q Z. Particle swarm optimizer based on dynamic neighborhood topology and mutation operator[J]. Control and decision, 2010, 25(7): 968-974).

[18] Van D B F, Engelbrcht A P. A cooperative approach to particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 225-239.

(上接第88页)

[8] Kotecha J H, Djurić P M. Gaussian sum particle filtering[J]. IEEE Trans on Signal Processing, 2003, 51(10): 2603-2613.

[9] Schon T, Gustafsson F, Nordlund P J. Marginalized particle filters for mixed linear/nonlinear state-space models[J]. IEEE Trans on Signal Processing, 2005, 53(7): 2279-2289.

[10] 鲍其莲, 周媛媛. 基于 UKF 的 GPS/SINS 伪距(伪距率)组合导航系统设计[J]. 中国惯性技术学报, 2008, 16(1): 78-82.
(Bao Q L, Zhou Y Y. Design of GPS/SINS pseudo-range(pseudo-range rate) integrated navigation system based on UKF[J]. J of Chinese Inertial Technology, 2008, 16(1): 78-82.)

[11] 华冰, 刘建业, 李荣冰, 等. 余度 MEMS-IMU/GPS 组合导航系统[J]. 南京航空航天大学学报, 2007, 39(5): 570-575.
(Hua B, Liu J Y, Li R B, et al. Redundant MEMS-IMU/GPS integrated navigation system based on improved unscented particle filter algorithm[J]. J of Nanjing University of Aeronautics and Astronautics, 2007, 39(5): 570-575.)