文章编号: 1001-0920 (2015) 12-2168-07 DOI: 10.13195/j.kzyjc.2014.1869

# 多目标自适应混沌粒子群优化算法

杨景明, 马明明, 车海军, 徐德树, 郭秋辰

(燕山大学 工业计算机控制工程河北省重点实验室，河北 秦皇岛 066004)

**摘 要:** 提出一种多目标自适应混沌粒子群优化算法(MACPSO). 首先, 基于混沌序列提出一种新型动态加权方法选择全局最优粒子; 然后, 改进NSGA-II拥挤距离计算方法, 并应用到一种严格的外部存档更新策略中; 最后, 针对外部存档提出一种基于世代距离的自适应变异策略. 以上操作不仅提高了算法的收敛性, 而且提高了Pareto最优解的均匀性. 实验结果表明了所提出算法的有效性.

**关键词:** 多目标优化；粒子群；混沌Logistic映射；拥挤距离；自适应变异

**中图分类号:** TP18 **文献标志码:** A

# Multi-objective adaptive chaotic particle swarm optimization algorithm

*YANG Jing-ming*, *MA Ming-ming*, *CHE Hai-jun*, *XU De-shu*, *GUO Qiu-chen*

(Key Lab of Industrial Computer Control Engineering of Hebei Province，Yanshan University，Qinhuangdao 066004, China．Correspondent：MA Ming-ming，E-mail：mamming1990@sina.com)

**Abstract:** A multi-objective adaptive chaotic particle swarm optimization(MACPSO) algorithmis proposed. Firstly, on the basis of the chaotic sequence, a new dynamic weighting method is proposed to select the global optimum particle. Then, the calculation method of crowding distance in NSGA-II is improved and applied to a rigorous external archive updating strategy. Finally, an adaptive mutation strategy based on the generational distance is presented for the external archive. The operations above mentioned not only enhance the convergence performance of the proposed algorithm, but also improve the uniformity of the Pareto optimal solution. The experimental results show the effectiveness of the proposed method.

**Keywords:** multi-objective optimization；particle swarm；chaotic Logistic map；crowding distance；adaptive mutation

## 0 引 言

在工程实践和科学研究中, 很多问题是由多个相互影响、相互冲突的目标构成, 人们希望能够求得一组解使得多个目标在可行域内尽可能达到最优, 此时就要面临求解多目标优化问题(MOP).

进化算法在多目标优化领域的应用中获得了广泛的研究, 其中SPEA2[1]和NSGA-II[2]为比较经典的多目标进化算法. 近年来, 多目标研究呈现新的特点, 新的占优机制被提出, 比如$\varepsilon$-占优[3]、部分占优[4]等. 同时, 一些新型进化范例被应用于多目标优化问题, 比如粒子群优化[5-6]、差分进化算法[7]、基于分解的进化算法[8]等.

粒子群优化算法[5]具有收敛速度快、原理简单、易于编程实现的特点, 在多目标优化方面得到了广泛的研究. Hu等[9]提出了基于分解的多子群多目标粒子群优化算法, 将多目标优化问题分解成多个标量子问题, 通过相应的子种群对每个标量子问题进行寻优; Elloumi等[10]提出了H-MOPSO-FACO, 将模糊蚁群算法的最优粒子作为粒子群算法的全局最优粒子进行寻优; 胡旺等[11]通过Pareto熵评估种群多样性以及当前进化状态, 以此作为反馈信息来设计进化策略, 使得算法能够兼顾收敛性和多样性.

混沌序列由于具有遍历性、随机性的特点而被应用到多目标粒子群优化算法中. 文献[12]通过混沌序列对种群进行初始化, 使得初始种群能够更均匀地分布于决策空间; 文献[13]利用混沌序列产生新的粒子, 进行混沌搜索, 以提高解的多样性; 文献[14]利用混沌序列进行混沌变异, 使种群跳出局部最优. 以上改进不同程度地改善了Pareto前沿解的质量.

受此启发, 本文提出一种多目标自适应混沌粒子

**收稿日期:** 2014-12-08；**修回日期:** 2015-04-13.

**基金项目:** 河北省高等学校创新团队领军人才培育计划项目(LJRC013)；河北省科技支撑计划项目(13211817)；国家冷轧板带及装备工程研究中心开放课题项目(2012005).

**作者简介:** 杨景明(1957–), 男, 教授, 博士生导师, 从事冶金机械综合自动化、先进控制及工程应用等研究；马明明(1990–), 男, 硕士生, 从事冶金机械综合自动化、多目标决策的研究.

群优化算法(MACPSO). 该算法将混沌序列引入到粒子群优化算法中, 提出一种基于混沌序列的动态加权方法, 用以选择全局最优粒子; 在NSGA-II拥挤距离计算方法的基础上, 改进了拥挤距离的计算公式, 以提高Pareto解集的分布性, 并应用到一种严格的外部存档更新策略中获取最优解集; 对外部存档提出一种自适应变异规模的变异策略. 通过与其他3种算法进行对比研究, 表明所提出的算法在保证收敛性的前提下, 具有更好的分布性, 表现出优良的寻优性能.

## 1　多目标自适应混沌粒子群优化算法

### 1.1　多目标模型

多目标优化问题一般可以表述成如下模型:

$$\min f(x) = (f_1(x), \cdots, f_m(x), \cdots, f_M(x)),$$
$$\text{s.t. } x \in \Omega^n. \tag{1}$$

其中: $x = [x_1, x_2, \cdots, x_n]$ 为 $n$ 维决策变量, $\Omega^n$ 为决策变量可行解空间, $f_m(x)$ 为第 $m$ 维目标函数, $M$ 为目标函数总维度.

求解多目标问题, 最终得到一组解集. 为详细说明, 引入以下定义.

**定义1**　$x_a$、$x_b$ 为两个决策变量, 当 $\forall f_m(x_a) \leqslant f_m(x_b)\ (1 \leqslant m \leqslant M)$, 且 $\exists f_m(x_a) < f_m(x_b)(1 \leqslant m \leqslant M)$ 时, 称 $x_a$ 占优 $x_b$, 记作 $f(x_a) \succ f(x_b)$.

**定义2**　对于可行解集中的解 $x_i$, 当且仅当 $\neg\exists f_m(x_j) \succ f_m(x_i)$ 时, $x_i$ 为Pareto最优. 相应地, 所有最优解 $x_i$ 的集合称为Pareto最优解集, 最优解集中所有的解对应的目标矢量所组成的曲线(面), 称为Pareto前沿.

求解多目标问题最终得到的是一组尽可能逼近Pareto前沿且分布均匀的解集.

### 1.2　基本粒子群优化算法

粒子群优化算法速度和位置更新公式如下:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1(\mathrm{pb}_{id} - x_{id}(t)) + c_2 r_2(\mathrm{gb}_d - x_{id}(t)), \tag{2}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1). \tag{3}$$

其中: $1 \leqslant i \leqslant N$, $1 \leqslant d \leqslant D$; $i$ 表示粒子标号, $N$ 表示种群规模, $d$ 表示决策变量维度标号, $D$ 表示决策变量总维度, $x$ 表示粒子位置(即决策变量), $v$ 表示速度, $t$ 表示进化代数, pb表示粒子个体历史最优位置, gb表示粒子全局历史最优位置, $\omega > 0$ 表示惯性因子, $c_1$ 表示个体加速因子, $c_2$ 表示全局加速因子, $r_1$ 和 $r_2$ 表示 $[0,1]$ 间的随机数.

### 1.3　基于混沌序列的全局最优粒子选择

粒子群算法中, 全局最优粒子gb引导着整个种群的进化方向, 直接影响着最终解集的优劣. 本文提出一种基于混沌序列的动态加权法对全局最优粒子进行选择.

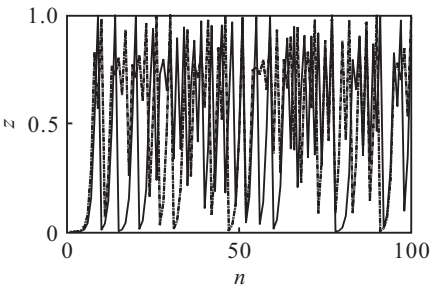
#### 1.3.1　混沌序列

混沌是非线性动力学系统中的一种运动形式, 具有随机性、遍历性及对初始值敏感性的特点. 将确定性方程得到的具有随机性的运动状态称为混沌, 呈现混沌状态的变量称为混沌变量. Logistic映射是一种典型混沌系统, 其有限差分方程如下:

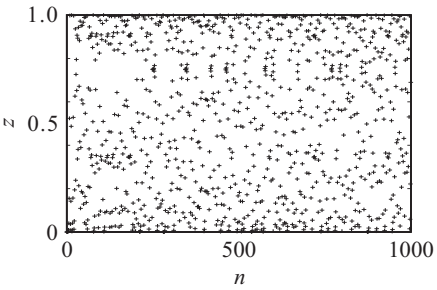
$$z_{n+1} = u z_n (1 - z_n). \tag{4}$$

其中: $u$ 为控制参数; $z_0 \in [0,1]$, 且 $z_0 \neq 0.5$. 当 $u = 4$ 时, 该方程呈现完全混沌状态, 得到的 $z$ 序列为 $[0,1]$ 区间上的满映射.

图1(a)表示在初始值 $z_0$ 分别取0.000 01(实线)和0.000 02(虚线)时, 通过Logistic映射迭代100次所得序列; 图1(b)表示在初始值 $z_0$ 取0.000 01时, 通过Logistic映射迭代1 000次所得序列.




(a) 对初始值敏感性




(b) 随机性、遍历性

**图1　混沌序列特性仿真**

从图1(a)可以看出, 初始值仅相差0.000 01时, 在有限的100次迭代中, 两个序列具有很大的差别; 从图1(b)可以看出, 在1 000次的迭代中, 混沌序列几乎随机地遍历了 $[0,1]$ 区间.

#### 1.3.2　全局最优粒子选择

全局最优粒子应当代表种群的最优进化信息, 故从当前得到的最优解集(即外部存档)中选择. 传统的选择方法有随机选择方法、基于密度的选择方法等. 这里提出一种简单合理的选择方法——基于混沌序列的动态加权法.

外部存档中每个粒子适应度计算如下:

$$\text{fitness} = 1 \Big/ \sum_{m=1}^{M} \omega_t^m f^m(x),\ t = 1, 2, \cdots, \tag{5}$$

其中 $\sum_{m=1}^{M} \omega_t^m = 1$ 为通过混沌序列产生的权值.

权值$\omega$产生方法如下:

Step 1: 对于$t=1$, $r=[r_1,r_2,\cdots,r_M]$, $r_1,r_2,\cdots,r_M$为非0.5的随机数, $\omega=r/\mathrm{sum}(r)$.

Step 2: 对于$t>1$, 分别以$r_1,r_2,\cdots,r_M$为初始值, 依据式(4)进行Logistic映射, 迭代$t$次, 得到新的$r$, 进而得到新的$\omega$.

按式(5)计算出每个粒子适应度后, 选择适应度最大的粒子作为下一代种群更新的全局最优粒子.

### 1.4 外部存档维护策略

#### 1.4.1 选择机制

为保证最终得到的外部存档中的解为Pareto最优解, 此处采用一种严格的外部存档选择机制. 外部存档更新策略如下:

Step 1: 对于新解$x$, 如果外部存档为空, 则将$x$存入, 更新结束, 否则继续下一步.

Step 2: 如果$x$被外部存档中所有解支配, 则外部存档不变, 更新结束, 否则继续下一步.

Step 3: 如果$x$支配外部存档中部分解, 且不被其他解支配, 则删除被支配解, 将$x$存入, 更新结束, 否则继续下一步.

Step 4: 如果$x$支配外部存档中部分解, 且被其他解支配, 则删除被支配解, 不存入, 更新结束, 否则继续下一步.

Step 5: 如果$x$与外部存档中的所有解互不支配, 则将$x$存入, 判断外部存档中解的个数是否达到最大值, 如果未达到, 则更新结束; 否则, 将计算每个解的拥挤距离, 删除拥挤距离最小的解, 更新结束.

#### 1.4.2 改进拥挤距离计算

NSGA-II中拥挤距离计算方法[2]具有简单快捷的特点. 但是由于按照一个目标函数排序计算拥挤距离, 导致最终保留的解集仅在该目标函数上是均匀分布的, 而在其他函数上存在不均匀的情况. 本文在其基础上, 改进了拥挤距离计算方法, 其计算公式如下:

$$\mathrm{dis}_i=\begin{cases}\inf,\ i=1\,\mathrm{or}\,i=N;\\ \sqrt{\displaystyle\sum_{\mathrm{m}=1}^{M}\left(\frac{f_{\mathrm{next}}^m-f_{\mathrm{previous}}^m}{f_{\max}^m-f_{\min}^m}\right)^2},\ \mathrm{else}.\end{cases}\tag{6}$$

其中: $f_{\mathrm{next}}^m$表示按第$m$维目标函数值从小到大排序后, 第$i$个粒子的下一个粒子的第$m$维目标函数值; $f_{\mathrm{previous}}^m$表示按第$m$维目标函数值从小到大排序后, 第$i$个粒子的前一个粒子的第$m$维目标函数值.

改进后的拥挤距离计算方法不但依然仅考虑前后两个粒子, 而且综合考虑了每一维目标函数, 更具合理性.

### 1.5 基于外部存档的自适应变异策略

粒子群算法是通过个体间经验的学习产生新个体, 具有渐变的特点, 收敛速度快, 但是易陷入局部最优; 遗传算法是通过个体交叉变异产生新个体, 具有跳变的特点, 具有较强的全局搜索能力, 但是收敛速度慢.

外部存档中的解包含当前最优的进化信息, 因此对外部存档中的解进行变异操作, 更容易产生优秀的个体. 这里使用Deb等[2]提出的多项式变异, 对于粒子$x$, 随机选择一维进行变异, 变异方式如下:

$$x_{\mathrm{new}}^d=x^d+(x_U^d-x_L^d)\delta.\tag{7}$$

其中: $x_U^d$、$x_L^d$表示粒子第$d$维上界和下界, $\delta$为粒子扰动项, 由以下公式产生:

$$\delta=\begin{cases}(2r)^{\frac{1}{\mu+1}}-1,\ r<0.5;\\ 1-[2(1-r)]^{\frac{1}{\mu+1}},\ r\geqslant 0.5.\end{cases}\tag{8}$$

$r$为$(0,1)$间均匀分布随机数, $\mu$为变异分布指数.

对外部存档中粒子进行变异后, 产生新的解集, 对于每一个新解, 对外部存档进行1.2节的更新操作, 保留优秀粒子.

算法运行初期, 种群收敛速度较快, 可以适当减小变异规模, 以提高算法运算效率. 算法运行后期, 种群收敛速度减慢, 甚至可能陷入局部最优, 算法停滞, 因此应增大变异规模, 使种群跳出局部最优. 这里选择改进的世代距离GD对变异规模进行自适应调整.

GD指标[15]定义如下:

$$\mathrm{GD}=\sqrt{\sum_{i=1}^{s}d_i^2}\Big/s.\tag{9}$$

其中: $s$为待评价解集中解的个数, $d_i$为第$i$个解与最接近的真实Pareto解间的欧氏距离.

然而, 在实际问题中, 在算法求解过程中, 不可能知道真实的Pareto前沿, 因此取每代外部存档与前一代之间的世代距离, 记作$\mathrm{GD}^*$, 相邻两代解集$\mathrm{GD}^*$的差值记作$\Delta_{\mathrm{GD}^*}$, 即

$$\Delta_{\mathrm{GD}^*}(t)=\mathrm{GD}^*(t)-\mathrm{GD}^*(t-1),\ t\geqslant 3.\tag{10}$$

计算某一代的$\Delta_{\mathrm{GD}^*}$, 需要用到最近三代的外部存档解集. $\Delta_{\mathrm{GD}^*}$表征了算法的收敛速度, $\Delta_{\mathrm{GD}^*}>0$表示算法收敛速度变快, 应适当减小变异规模, 提高算法的效率; $\Delta_{\mathrm{GD}^*}<0$表示算法收敛速度减慢, 应当适当增大变异规模, 提高算法的收敛性, 同时可以提高多样性.

变异规模调整如下:

$$Q(t+1)=\begin{cases}\mathrm{integer}(0.75Q(t)),\ \Delta_{\mathrm{GD}^*}(t)>0;\\ Q(t),\ \Delta_{\mathrm{GD}^*}(t)=0;\\ \mathrm{integer}(1.25Q(t)),\ \Delta_{\mathrm{GD}^*}(t)<0.\end{cases}\tag{11}$$

其中: $Q(t)$为第$t$代外部存档变异规模, 且需满足$0<Q(t)\leqslant z$, $z$为当前外部存档规模; $t=1,2,3$时, $Q(t)$

$= z$.

### 1.6 算法流程

Step 1: 初始化种群规模 $N$, 外部存档最大规模 $Z$, 总的评价次数 FEAS, 代数计数器 $t = 0$, 随机初始化种群 $x_0$, 速度 $v_0$, 个体历史最优 $\mathrm{pb} = x_0$, 随机给定 $r$, 计算出 $\omega$ 值, 计算每个粒子在当前 $\omega$ 下的适应度, 选择 gb.

Step 2: 根据式 (2) 和 (3) 对每个粒子进行速度、位置更新操作, 检查是否越界, 若大于上界取最大值, 小于下界取最小值.

Step 3: 对于每个新粒子, 依据 1.4 节进行外部存档更新操作.

Step 4: 依据 1.5 节的自适应变异策略, 对外部存档进行变异操作, 得到新的解集, 依据 1.4 节进行外部存档更新操作.

Step 5: 依据支配关系更新 pb, 依据 1.3 节更新 gb.

Step 6: 判断是否达到最大评价次数, 未达到则跳转 Step 2; 达到则结束, 输出外部存档.

## 2 算法测试及结果分析

为了验证所提出的算法求解多目标问题的性能, 将其与 IMOPSO、MOEA/D 和 NSGA-II 进行对比研究, 选取 ZDT 系列的测试函数进行性能测试.

### 2.1 性能指标

本文选取 4 个性能指标[15]对算法性能进行评价.

1) 世代距离 GD. GD 衡量算法获得的最优解集与真实 Pareto 前沿的逼近程度, GD 越小表示算法收敛性越好, 其表达式见式 (9).

2) 多样性 $\Delta$. $\Delta$ 衡量获得的最优解集的分布情况, $\Delta$ 越小表示最优解集分布性越好, 其表达式如下:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{s-1} |d_i - \bar{d}|}{d_f + d_l + (s-1)\bar{d}}. \tag{12}$$

其中: $\bar{d} = \dfrac{\sum_{i=1}^{s-1} d_i}{(s-1)}$ 为距离平均值, $d_i$ 为所得非劣解间欧式距离, $d_f$、$d_l$ 为边界解与极端解间的距离.

3) 覆盖率 MS. MS 衡量最优解集在 Pareto 前沿上的覆盖范围, 越接近于 1 表示所得最优解集的覆盖范围越大, 其表达式如下:

$$\mathrm{MS} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left\{\frac{\min(f_m^{\max}, F_m^{\max}) - \max(f_m^{\min}, F_m^{\min})}{F_m^{\max} - F_m^{\min}}\right\}^2}. \tag{13}$$

其中: $f_m^{\max}$、$f_m^{\min}$ 分别为所得最优解集在第 $m$ 维目标函数上的最大最小值, $F_m^{\max}$、$F_m^{\min}$ 分别为真实 Pareto 前沿在第 $m$ 维目标函数上的最大最小值.

4) 错误率 ER. ER 衡量最优解集在 Pareto 前沿上的比率, ER 越小表示最优解的质量越高, 其表达式为

$$\mathrm{ER} = \sum_{i=1}^{s} e_i \Big/ s. \tag{14}$$

其中: $e_i = 0$ 表示最优解集中第 $i$ 个解在真实 Pareto 前沿上; $e_i = 1$ 则表示最优解集中第 $i$ 个解不在真实 Pareto 前沿上.

### 2.2 测试结果及分析

为了保证公平性, 所有算法均设置种群规模 $N = 100$, 外部存档最大规模 $Z = 100$, 最高评价次数 $\mathrm{FEAS} = 50\,000$ 次, 4 种算法分别独立运行 50 次, 分别统计 4 个性能指标. 表 1 为性能指标统计数据, 图 2 为算法性能指标统计盒图, 图 3 为算法性能指标随评价次数变化曲线, 图 4 为 ZDT4 函数仿真曲线.

**表 1　性能指标统计**

| 性能指标 | 测试函数 | 算法 | | | |
|---|---|---|---|---|---|
| | | MACPSO | IMOPSO | MOEA/D | NSGA-II |
| GD | ZDT1 | 3.898e-06 | 7.727e-04 | 5.846e-04 | 5.928e-03 |
| | ZDT2 | 0.000e+00 | 7.054e-04 | 8.211e-05 | 4.304e-03 |
| | ZDT3 | 6.890e-05 | 2.824e-03 | 6.873e-04 | 1.064e-02 |
| | ZDT4 | 1.370e-04 | 4.266e-01 | 1.573e-02 | 2.016e+00 |
| | ZDT6 | 0.000e+00 | 0.000e+00 | 1.403e-02 | 1.027e-02 |
| $\Delta$ | ZDT1 | 7.557e-02 | 3.056e-01 | 2.976e-01 | 6.127e-01 |
| | ZDT2 | 8.194e-02 | 2.746e-01 | 1.427e-01 | 5.696e-01 |
| | ZDT3 | 4.161e-01 | 9.371e-01 | 8.996e-01 | 7.860e-01 |
| | ZDT4 | 7.264e-02 | 1.066e+00 | 4.317e-01 | 9.580e-01 |
| | ZDT6 | 1.919e-01 | 2.917e-01 | 5.791e-01 | 6.926e-01 |
| MS | ZDT1 | 1.000e+00 | 9.972e-01 | 9.988e-01 | 9.927e-01 |
| | ZDT2 | 1.000e+00 | 8.755e-01 | 9.995e-01 | 9.500e-01 |
| | ZDT3 | 9.999e-01 | 9.991e-01 | 9.770e-01 | 9.823e-01 |
| | ZDT4 | 9.999e-01 | 1.657e+00 | 9.585e-01 | 1.171e+01 |
| | ZDT6 | 1.000e+00 | 9.997e-01 | 9.996e-01 | 1.000e+00 |
| ER | ZDT1 | 2.860e-02 | 4.680e-01 | 1.000e+00 | 1.888e-01 |
| | ZDT2 | 0.000e+00 | 1.986e-01 | 9.704e-01 | 1.400e-01 |
| | ZDT3 | 1.468e-01 | 6.712e-01 | 9.882e-01 | 3.124e-01 |
| | ZDT4 | 1.000e+00 | 1.000e+00 | 1.000e+00 | 1.000e+00 |
| | ZDT6 | 0.000e+00 | 0.000e+00 | 8.880e-02 | 1.600e-03 |

从表 1 和图 2 可以看出, MACPSO 的 GD 指标在所有测试函数上均小于其他算法, 且接近于 0, 表明算法收敛性良好, 所得 Pareto 解集基本收敛到真实 Pareto 前沿; MACPSO 的 $\Delta$ 指标也接近于 0, 且明显优于其他算法, 表明所提出的算法有效地提高了算法的分布性, 获得了在 Pareto 前沿上分布更均匀的解集; MACPSO 的 MS 指标基本为 1, 或者接近于 1, 明显优于其他算法, 表明所得解集基本覆盖了全部的 Pareto 前沿; 对于 ER 指标, 虽然在 ZDT4 测试函数上, 所有算法 ER 指标显示为 1, 但是由 GD 指标可以看出, MACPSO 已基本接近于 0, 且优于其他算法 2～4 个数量级. 同时, 由图 4 可以直观地看到, MACPSO 获得的解已经很接近真实的 Pareto 前沿, 而其他算法所得

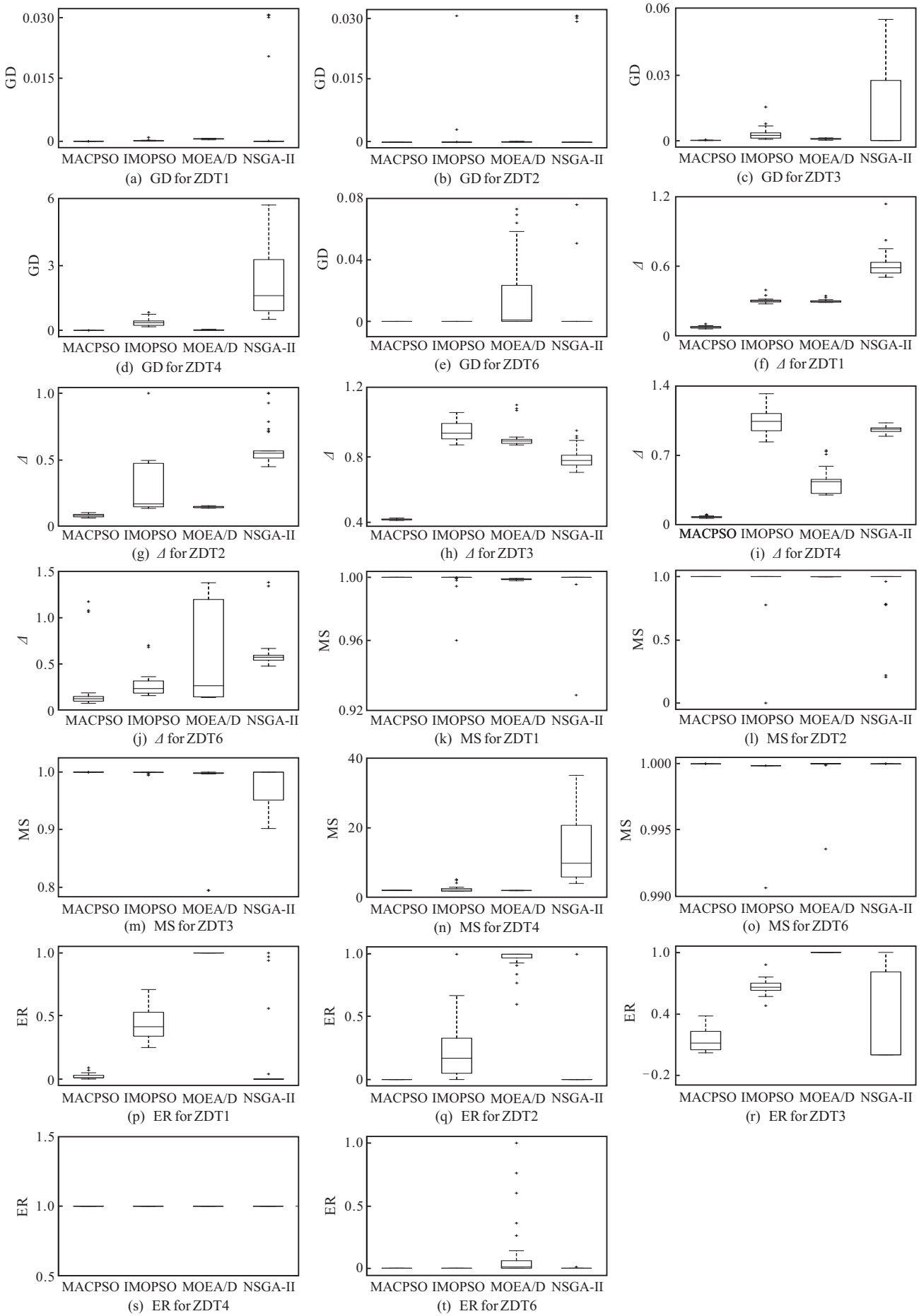

GD
MACPSO IMOPSO MOEA/D NSGA-II
(a) GD for ZDT1
(b) GD for ZDT2
(c) GD for ZDT3
(d) GD for ZDT4
(e) GD for ZDT6
Δ
(f) Δ for ZDT1
(g) Δ for ZDT2
(h) Δ for ZDT3
(i) Δ for ZDT4
(j) Δ for ZDT6
MS
(k) MS for ZDT1
(l) MS for ZDT2
(m) MS for ZDT3
(n) MS for ZDT4
(o) MS for ZDT6
ER
(p) ER for ZDT1
(q) ER for ZDT2
(r) ER for ZDT3
(s) ER for ZDT4
(t) ER for ZDT6

**图 2　性能指标统计盒**

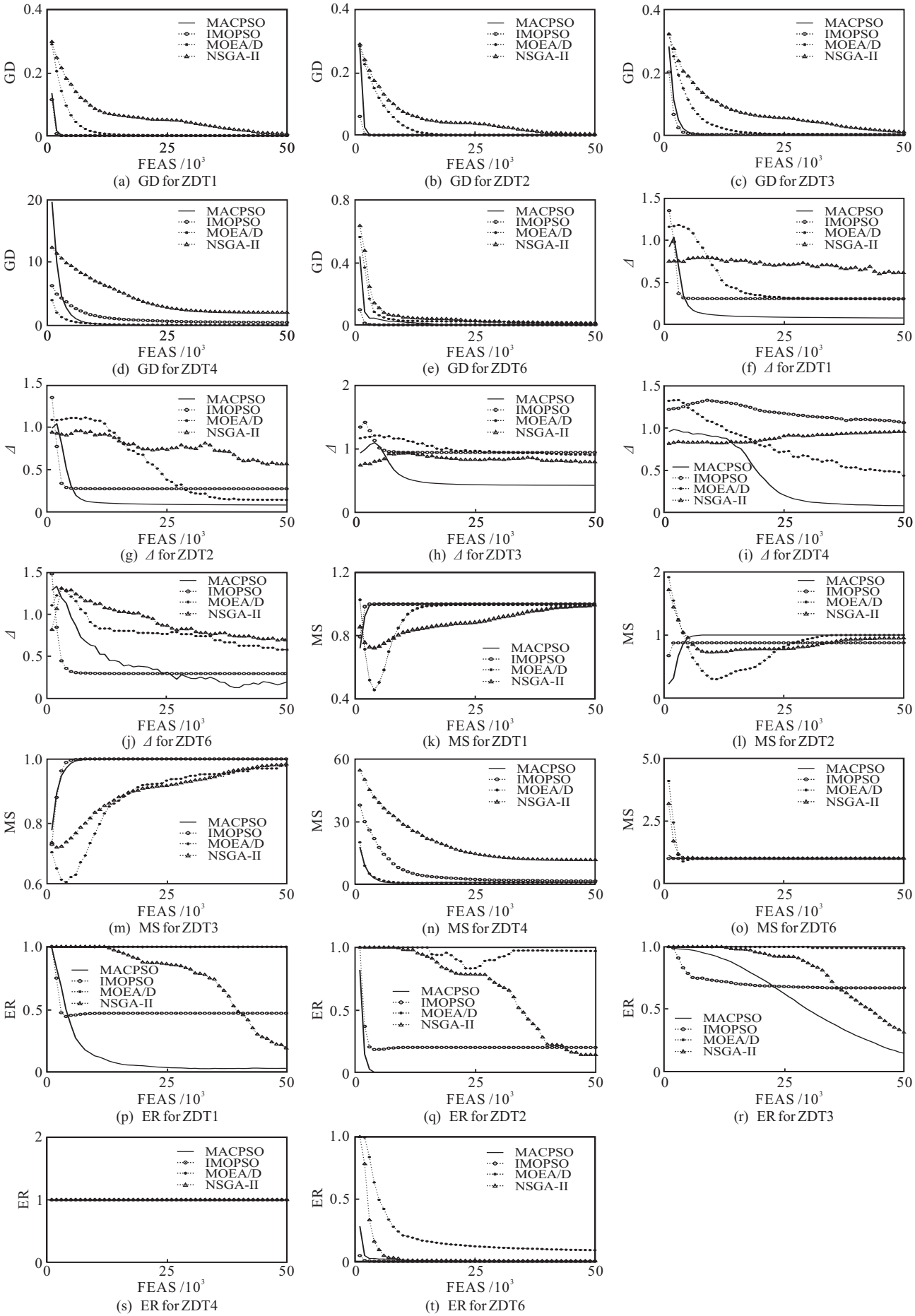
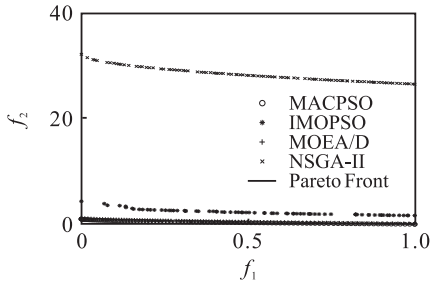
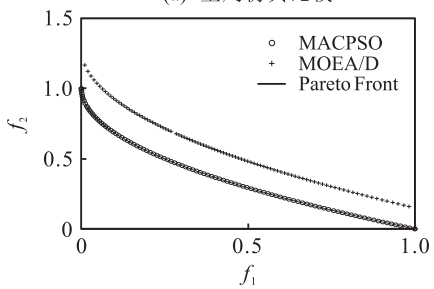

MACPSO
IMOPSO
MOEA/D
NSGA-II
GD
Δ
MS
ER
FEAS /10^3
(a) GD for ZDT1
(b) GD for ZDT2
(c) GD for ZDT3
(d) GD for ZDT4
(e) GD for ZDT6
(f) Δ for ZDT1
(g) Δ for ZDT2
(h) Δ for ZDT3
(i) Δ for ZDT4
(j) Δ for ZDT6
(k) MS for ZDT1
(l) MS for ZDT2
(m) MS for ZDT3
(n) MS for ZDT4
(o) MS for ZDT6
(p) ER for ZDT1
(q) ER for ZDT2
(r) ER for ZDT3
(s) ER for ZDT4
(t) ER for ZDT6

**图 3　性能指标变化曲线**




(a) 全局仿真比较




(b) 局部仿真比较

**图 4 ZDT4 仿真曲线**

解与Pareto前沿还有一定距离; 对于其他测试函数, MACPSO获得了更低的错误率; 从图3可以看出, 所提出的算法在提高了算法收敛性和分布性的前提下, 依然能够保持不错的收敛速度. 仿真结果表明, 所提出的算法有效改善了算法性能, 提高了求解效率.

## 3 结 语

为了提高求解多目标优化问题效率, 本文提出了一种多目标自适应混沌粒子群优化算法. 算法中给出了一种基于混沌序列的动态加权方法, 以选择全局最优粒子, 更好地引导种群的进化方向; 采用了一种严格的外部存档更新策略, 并改进了拥挤距离计算方法; 针对算法容易陷入局部最优的问题, 提出了基于外部存档的自适应变异策略. 对比测试结果表明, 所提出的算法能够获得收敛性更好、分布性更均匀、覆盖范围更广的高质量Pareto解集. 下一步的工作是考虑提高算法对更高维的多目标问题的求解性能, 并将其应用到工程设计中, 以实现其社会价值.

## 参考文献(References)

[1] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm[R]. Lausanne: Swiss Federal Institute of Technology Computer Engineering and Networks Laboratory, 2001.

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

[3] Laumanns M, Thiele L, Deb K, et al. Combining convergence and diversity in evolutionary multiobjective optimization[J]. Evolutionary Computation, 2002, 10(3): 263-282.

[4] Brockhoff D, Zitzler E. Are all objectives necessary? On dimensionality reduction in evolutionary multi-objective optimization[M]. Parallel Problem Solving from Nature-PPSN IX. Berlin: Springer Berlin Heidelberg, 2006: 533-542.

[5] Kennedy J. Particle swarm optimization[M]. Encyclopedia of Machine Learning. New York: Springer, 2010: 760-766.

[6] Zhang Y, Gong D W, Geng N. Multi-objective optimization problems using cooperative evolvement particle swarm optimizer[J]. J of Computational and Theoretical Nanoscience, 2013, 10(3): 655-663.

[7] 章萌, 章卫国, 孙勇. 多目标强度Pareto混沌差分进化算法[J]. 控制与决策, 2012, 27(1): 41-52.
(Zhang M, Zhang W G, Sun Y. Multi-objective strength Pareto chaotic differential evolution algorithm[J]. Control and Decision, 2012, 27(1): 41-52.)

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

[9] Hu P, Li R, Cao L L, et al. Multiple swarms multi-objective particle swarm optimization based on decomposition[J]. Procedia Engineering, 2011, 15: 3371-3375.

[10] Elloumi W, Baklouti N, Abraham A, et al. The multi-objective hybridization of particle swarm optimization and fuzzy ant colony optimization[J]. J of Intelligent and Fuzzy Systems, 2014, 27(1): 515-525.

[11] 胡旺, Yen G G, 张鑫. 基于Pareto熵的多目标粒子群优化算法[J]. 软件学报, 2014, 25(5): 1025-1050.
(Hu W, Yen G G, Zhang X. Multi-objective particle swarm optimization based on Pareto entropy[J]. J of Software, 2014, 25(5): 1025-1050.)

[12] 李娟, 杨琳, 刘金龙, 等. 基于自适应混沌粒子群优化算法的多目标无功优化[J]. 电力系统保护与控制, 2011, 39(9): 26-31.
(Li J, Yang L, Liu J L, et al. Multi-objective reactive power optimization based on adaptive chaos particle swarm optimization algorithm[J]. Power System Protection and Control, 2011, 39(9): 26-31.)

[13] 钱伟懿, 李阿军, 杨宁宁. 基于混沌的多目标粒子群优化算法[J]. 计算机工程与设计, 2008, 29(18): 4794-4796.
(Qian W Y, Li A J, Yang N N. Particle swarm optimization algorithm based on chaotic series for multi-objective optimization problems[J]. Computer Engineering and Design, 2008, 29(18): 4794-4796.)

[14] Pang S, Zou H, Yang W, et al. An adaptive mutated multi-objective particle swarm optimization with an entropy-based density assessment scheme[J]. Information & Computational Science, 2013, 4(10): 1065-1074.

[15] Tsai S J, Sun T Y, Liu C C, et al. An improved multi-objective particle swarm optimizer for multi-objective problems[J]. Expert Systems with Applications, 2010, 37(8): 5872-5886.

(责任编辑: 齐 霁)