第 37 卷 第 14 期 计 算 机 工 程 2011 年 7 月
Vol.37 No.14 Computer Engineering July 2011

·人工智能及识别技术· 文章编号：1000—3428(2011)14—0152—03 文献标识码：A 中图分类号：TP301.6

基于均匀设计的聚类多目标粒子群优化算法

刘衍民[1,2]，牛 奔[3]，赵庆祯[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：为更有效地求解多目标优化问题，提出一种基于均匀设计的聚类多目标粒子群算法 UCMOPSO。采用基于均匀设计的交叉操作尽可能地获得目标空间中均匀分布的非劣解，帮助种群跳出局部最优解，并通过一种新的聚类操作选择外部存档中有代表性的非劣解，从而控制外部存档规模，降低计算复杂度。对基准函数的测试结果表明，UCMOPSO 算法相比同类算法在收敛性和分布性方面具有优势。

关键词：均匀设计；多目标优化；聚类；粒子群优化算法；外部存档

Clustering Multi-objective Particle Swarm Optimization Algorithm Based on Uniform Design

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

(1. Department of Math, Zunyi Normal College, Zunyi 563002, China; 2. School of Management and Economics, Shandong Normal University, Jinan 250014, China; 3. College of Management, Shenzhen University, Shenzhen 518060, China)

【Abstract】In order to solve multi-objective problems efficiently, this paper proposes a clustering multi-objective Particle Swarm Optimization (PSO) algorithm based on uniform design named UCMOPSO. Crossover operation based on uniform design is adjusted to get uniformly distributed solutions in objective space to help swarm to escape from local optima, and a new clustering operator is introduced to select the representative non-dominated solutions, which decreases the computation complexity and limits the size of the external archive. Experimental results based on benchmark functions indicate that UCMOPSO has superiority in convergence and distribution compared with other algorithms.

【Key words】uniform design; multi-objective optimization; clustering; Particle Swarm Optimization(PSO) algorithm; external archive

DOI: 10.3969/j.issn.1000-3428.2011.14.050

1 概述

粒子群优化(Particle Swarm Optimization, PSO)[1]算法是一类基于种群的进化算法，它具有高速收敛和易于实现的特点，因此，可以通过扩展 PSO 算法来求解多目标优化问题[2-4]。本文提出一种基于均匀设计的聚类多目标粒子群优化算法 UCMOPSO 用以提高 PSO 算法求解多目标问题的效率。

2 基于均匀设计的聚类 MOPSO

2.1 外部存档

PSO 算法每一次迭代都会产生一组非劣解。因此，在算法运行过程中采用外部存档存储每一代产生的非劣解。值得注意的是，随着迭代的进行，每一代所产生的非劣解用来更新外部存档，随着迭代次数的增加，外部存档规模将逐渐增加。如不控制存档规模，将极大地增加计算复杂度。同时，由于PSO每一次迭代所产生的非劣解在某种程度上具有类似的特性，因此寻求一种限制外部存档规模并且能够删除具有相同特性的粒子(称为同质粒子)将会极大地提升非劣解在 Pareto 前沿的均匀分布性。本文由此提出一种基于聚类的删除策略。

聚类中心粒子数量的确定方法如下：

$$C=\arg\{sort(Pos_{front(i)}{}^{t}),n\} \tag{1}$$

其中，$Pos_{front(i)}{}^{t}$ 表示在迭代时刻 t 与粒子 i 最接近的真实的 Pareto 解；$sort(\cdot)$ 表示对元素从小到大进行排序；$\arg(\cdot)$ 表示识别元素在数组中的位置；n 表示所确定的聚类中心数量，在本算法中，n 等于粒子总数的 1/3。

聚类中心的半径确定方式如下：

$$r=\frac{\sum_{i=1}^{n}D((f_i^{\min},f_{i+1}^{\max}),(f_i^{\max},f_{i+1}^{\min}))}{\sigma N} \tag{2}$$

其中，N 表示外部存档中所包含的非劣解个数；σ 表示半径调整参数，用以调整同质粒子个数；n 表示目标函数个数；$f_i^{\min}$ 和 $f_i^{\max}$ 分别表示第 i 个目标对应的最小值和最大值；$D(\cdot)$ 表示欧式距离。

图 1 给出了有 2 个目标函数的聚类操作过程。

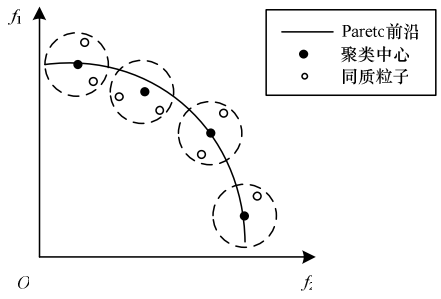


图 1 聚类操作过程

基金项目：广东省自然科学基金资助项目(9451806001002294)；贵州省教育厅社科基金资助项目(0705204)；山东省科技攻关计划基金资助项目(2009GG10001008)

作者简介：刘衍民(1978－)，男，讲师、博士，主研方向：进化计算，运筹学；牛 奔，副教授、博士；赵庆祯，教授、博士生导师

收稿日期：2010-12-20 **E-mail：**yanmin7813@sohu.com

2.2　基于均匀设计的交叉操作

为了使算法获得的非劣解尽可能均匀地分布在 Pareto 前沿，本文提出一种基于均匀分布设计的交叉操作，该操作主要集中在 Pareto 前沿的边界部分，这样有助于扩展 Pareto 前沿，进而产生均匀分布的非劣解。首先确定 Pareto 前沿的边界粒子，然后借鉴文献[3]提出的变异操作思想，对边界点粒子按式(3)产生边界点粒子的邻居，最后对边界点和它的邻居进行基于均匀设计的交叉操作。

$$N(t+1)=\begin{cases}\boldsymbol{x}_1(t)+(\boldsymbol{x}_1^{U}(t)-\boldsymbol{x}_1(t))\cdot r\cdot k & \text{if } \boldsymbol{x}_1<\boldsymbol{x}_2\\ \boldsymbol{x}_2(t)+(\boldsymbol{x}_2^{L}(t)-\boldsymbol{x}_2(t))\cdot r\cdot k & \text{if } \boldsymbol{x}_1\geqslant\boldsymbol{x}_2\end{cases} \tag{3}$$

其中，$N(t+1)$表示在迭代时刻 t 边界点的邻居；$\boldsymbol{x}_i(t),i=1,2$ 表示在迭代时刻 t 当前边界点的粒子位置；$\boldsymbol{x}_i^{U}(t)$ 和 $\boldsymbol{x}_i^{L}(t)$ 分别表示 $\boldsymbol{x}_i(t)$的上界和下界；r 是粒子 i 的扰动项，它是[0,1]内的随机数；k 是一个参数，用以调整边界点粒子的邻居数量，本算法中 k=0.2, 0.3。

图 2 给出了有 2 个目标问题的边界粒子及边界点粒子邻居的产生过程。

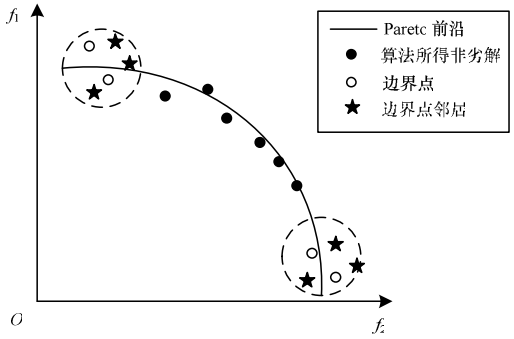


图 2　边界点识别

均匀设计[5]是一种非常有效的多因素多水平试验设计优化方法，是一种将数论和多元统计相结合的新的试验方法，它以均匀分散为准则，在试验范围内均匀分布试验点，使每个试验点都有充分的代表性，从而以较少的试验获得较多的信息。因此，本文在基于均匀设计的思想上，对 Pareto 前沿的边界点及其邻居进行交叉操作，使所产生的非劣解尽可能均匀地分布于 Pareto 前沿。均匀设计是通过均匀设计表进行试验设计的。这里利用 Pareto 边界点和它的邻居进行交叉操作来产生均匀分布的非劣解，进而更加有效地探索可行区域，具体过程参阅文献[5]。

2.3　学习样本的选取

PSO 在求解单目标问题时，每次迭代会产生一个全局最优粒子；而在求解多目标问题时，在每次迭代过程中产生一组非劣解，从众多的非劣解中随机选择一个非劣解作为全局最优解，这种策略明显降低了计算复杂度。为充分利用每个粒子的历史信息 *Pbest*，这里规定当粒子的 *Pbest* 对应的适应函数值连续 N 代没有得到提升时，才对 *Pbest* 进行更新，在本算法中，N=5。

2.4　UCMOPSO 算法

粒子的速度和位置更新公式如下：

$$\boldsymbol{v}_i(t+1)=\boldsymbol{v}_i(t)+\varphi_1\cdot r_1(\boldsymbol{p}_i(t)-\boldsymbol{x}_i(t))+\varphi_2\cdot r_2(\boldsymbol{rep}_h(t)-\boldsymbol{x}_i(t)) \tag{4}$$

$$\boldsymbol{x}_i(t+1)=\boldsymbol{x}_i(t)+\boldsymbol{v}_i(t+1) \tag{5}$$

其中，$\boldsymbol{rep}_h(t)$ 表示在迭代时刻 t 选取的全局最优粒子；$\boldsymbol{p}_i(t)$ 表示粒子的历史最优位置；$\varphi_1=\varphi_2=2$。算法流程如下：

```
Initialize positions and velocities of all particles
While stopping criterion is not met
    For each particle (i=1:ps)
        Select an exemplar from external archive
        Update Pbest,
velocity and position
Construct uniform design list
Find boundary particles
        Execute crossover operation
        Evaluate the fitness values of the current particle i
    End for
End while
```

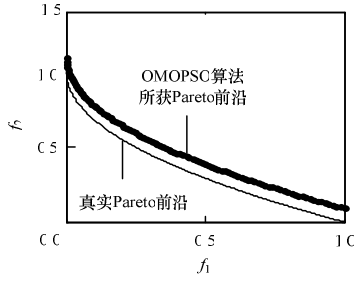
3　仿真实验与分析

3.1　检测函数及参数设置

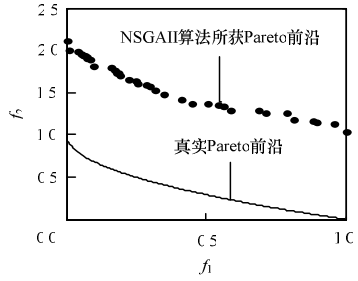
实验选取 3 个函数[3]ZDT1、ZDT2 和 ZDT3 来检测算法的性能，并选择 2 种比较有代表性的算法 NSGAII[6]和 OMOPSO[2]与本文的 UCMOPSO 算法进行比较。所有算法的迭代次数为 200，存档规模为 100，粒子规模为 100，其他参数设置与各种算法提出时所用参数一致。

3.2　仿真结果及分析

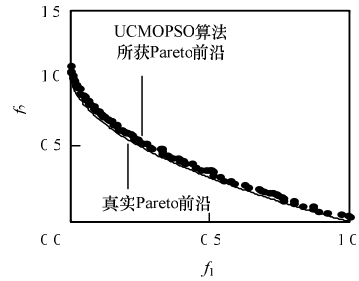
仿真实验平台为 ThinkPad-SL400 上的 Matlab7.7.0 (R2008b)，每个函数独立运行 20 次。图 3~图 5 给出了 3 种算法在 ZDT1、ZDT2 和 ZDT3 上所得的 Pareto 前沿(f_1 和 f_2 表示适应度函数值)。可以看出，本文算法得到的 Pareto 前沿相比 OMOPSO 和 NSGAII 更接近真实的 Pareto 前沿，说明引入的基于均匀设计的交叉操作和聚类操作大幅提升了算法运行效率。



(a)OMOPSO

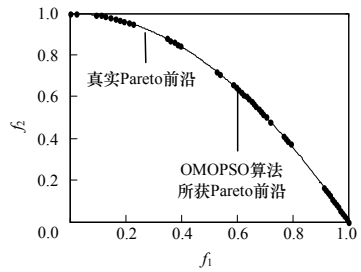


(b)NSGAII

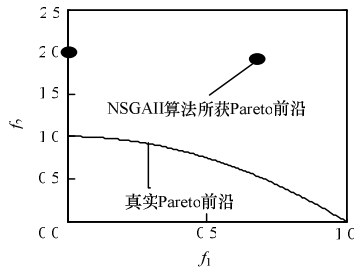


(c)UCMOPSO

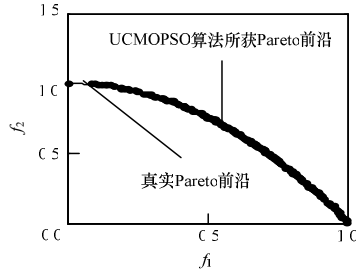
图 3　3 种算法在 ZDT1 函数上的 Pareto 前沿



(a)OMOPSO

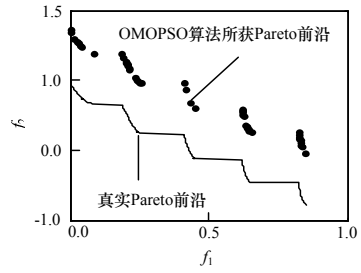


(b)NSGAII

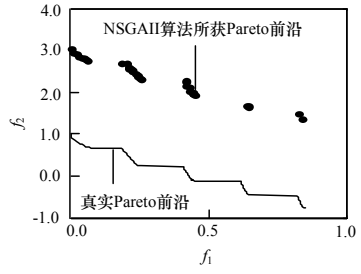


(c)UCMOPSO

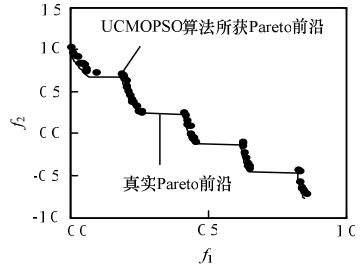
图 4 3 种算法在 ZDT2 函数上的 Pareto 前沿



(a)OMOPSO



(b)NSGAII



(c)UCMOPSO

图 5 3 种算法在 ZDT3 函数上的 Pareto 前沿

同时，采用分布指标 Δ 和收敛性指标 γ[2]评价不同算法的运行效率。其中，γ 越小，算法的收敛性越好；Δ 越小，非劣解的分布性越好。表 1 为 3 种算法 20 次独立运行的结果。由表 1 可以看出，UCMOPSO 算法在分布指标和收敛性指标方面都优于其他算法，而 OMOPSO 优于 NSGAII。

表 1 3 种算法的性能比较

函数	最终值	UCMOPSO		OMOPSO		NSGAII	
		γ	Δ	γ	Δ	γ	Δ
ZDT1	最优值	0.014 7	0.29	0.063 0	0.58	0.071 0	0.59
	最差值	0.098 6	0.47	0.131 0	0.69	0.234 0	0.87
	平均值	0.047 3	0.38	0.078 0	0.61	0.096 0	0.66
ZDT2	最优值	0.013 9	0.32	0.015 6	0.54	0.198 0	0.63
	最差值	0.041 6	0.53	0.162 8	0.69	0.619 0	0.81
	平均值	0.021 6	0.37	0.049 7	0.61	0.352 0	0.69
ZDT3	最优值	0.048 2	0.28	0.268 0	0.51	0.114 0	0.43
	最差值	0.132 8	0.43	0.531 0	0.64	0.286 0	0.74
	平均值	0.079 3	0.33	0.383 0	0.55	0.185 0	0.56

为了检验本文算法是否增加了计算复杂度，利用 Matlab 软件中的 tic 和 toc 函数计算当 γ=0.001 时各种算法所需运行时间。表 2 为 3 种算法的运行时间比较结果。从中可以看出，UCMOPSO 算法所需计算时间与其他算法在同一数量级上，表明本文算法所引入的策略并没有增加计算复杂度。

表 2 3 种算法的运行时间比较 s

函数	UCMOPSO	OMOPSO	NSAGII
ZDT1	0.97	0.51	1.47
ZDT2	0.79	0.61	2.15
ZDT3	21.77	15.53	17.81

4 结束语

本文提出的聚类多目标PSO算法通过引入均匀设计的交叉操作和聚类操提升了 PSO 算法求解多目标问题的能力，对检测函数的模拟结果证明UCMOPSO算法有效提高了求解多目标问题的效率。将来的工作主要集中在：(1)将 UCMOPSO 算法应用于工程实践中，以检测算法的性能；(2)研究调整边界点粒子邻居数量的参数 K 对算法的影响。

参考文献

[1] Kennedy J, Eberhart R C. Particle Swarm Optimization[C]// Proceedings of IEEE International Conference on Neural Networks. Piscataway, USA: IEEE Press, 1995: 1942-1948.

[2] Sierra M R, Coello C A C. Improving PSO-based Multi-objective Optimization Using Crowding, Mutation and ε-dominance[C]// Proc. of the 3rd International Conference on Evolutionary Multi-criterion Optimization. Guanajuato, Mexico: [s. n.], 2005: 505-519.

[3] Coello C A C, Pulido G T. Handling Multiple Objectives with Particle Swarm Optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 256-279.

[4] 汪文彬，钟 声. 基于改进拥挤距离的多目标进化算法[J]. 计算机工程, 2009, 35(9): 211-213.

[5] Fang Kai Tai, Wang Yuan, Number-theoretic Method in Statistics[M]. London, UK: Chapman and Hall, 1994.

[6] Deb K, Pratap A. A Fast and Elitist Multiobjective Genetic Algorithm: NSGAII[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

编辑 张 帆