文章编号: 1001-0920(2011)07-1019-07

基于 K-均值聚类的动态多种群粒子群算法及其应用

刘衍民[1,2], 隋常玲[1], 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)

摘 要: 针对粒子群算法在求解复杂的多峰问题时极易陷入局部最优解的问题, 提出一种基于 K-均值聚类的动态多种群粒子群算法(KDMSPSO). 在该算法中, 利用 K-均值聚类算法将种群分成若干个子群(聚类); 为了增强子群间的信息交流, 对子群进行动态重组; 在每个子群中, 粒子的速度由它所在子群的中心粒子和该粒子所有邻居的信息共同调整. 在基准函数测试和实际应用中, 其结果显示 KDMSPSO 算法相比其他 PSO 算法具有一定的优势.

关键词: 粒子群算法；K-均值；动态多种群

中图分类号: TP301.6 **文献标识码:** A

Dynamic multi-swarm particle swarm optimizer based on K-means clustering and its application

LIU Yan-min[1,2], *SUI Chang-ling*[1], *ZHAO Qing-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. Correspondent：LIU Yan-min，E-mail：yanmin7813@163.com)

Abstract: Particle swarm optimizer(PSO) may easily get trapped in a local optimum, when it comes to solving complex multimodal problems. Therefore, this paper presents dynamic multi-swarm particle swarm optimizer based on *K*-means clustering(KDMSPSO). In KDMSPSO, the population is divided into several sub-swarms by using *K*-means clustering. In order to increase the message exchange of sub-swarms, the sub-swarm is dynamically constructed, and the velocity of each particle is adjusted by clustering center that it belongs to and all particles in its neighborhood including itself. In benchmark function and actual application, the experimental results show that the KDMSPSO algorithm can achieve better solutions than other PSO algorithms.

Key words: particle swarm optimizer；*K*-means；dynamic multi-swarm

1 引 言

粒子群算法(PSO)[1]起源于对简单社会系统(鸟群觅食过程)的模拟. PSO 算法概念简单, 控制参数少, 寻优结果与初值无关, 具有一定的并行性等特点, 自提出以来便受到了学术界的广泛关注. 经验显示, PSO 算法在求解大多数优化问题时表现十分出色. 但是, PSO 算法在求解复杂的多峰问题时极易陷入局部最优解而导致典型的早熟收敛. 为了克服这个缺点, 提升算法的运行效率, 一些研究者提出了许多改进算法[2-9]. 对于邻居拓扑结构的改进方面, 文献 [2] 提出了一种动态邻居拓扑结构粒子群算法, 该算法在运行前期采用局部邻居拓扑结构(每个粒子的邻居由与其直接相连的粒子构成), 而运行的后期则采用全局邻居拓扑结构(每个粒子的邻居由群体中除自己以外的所有粒子构成), 通过调整邻居的拓扑结构来提升粒子的搜索范围. Mohais 等人[3]提出了一种随机图的邻居拓扑结构(采用随机图理论构建邻居拓扑结构)来提升粒子的搜索能力. 文献 [4] 提出了动态环境下的双子群 PSO 算法, 针对不同的目标, 用 2 个子群根据一定的规则优化各自的目标, 进而提升种群跳出局部最优解的能力. 还有一些研究者针对粒子速度更新时粒子学习样本的选择进行了改进, 如适应距离比例的带有近邻合作 PSO 算法(FDR-PSO)[5], 完全信息粒子群算法[6], 共协作粒子群最优法(CPSO-H)[7], 动态多种群 PSO 算法[8]和自适应扩散混合变异 PSO 算法[9]等. 这些改进策略对于提升算法的运行效率, 跳

收稿日期: 2010-04-16；**修回日期:** 2010-05-22.

基金项目: 贵州教育厅社科项目(0705204)；遵义师范学院基础教育课题(基 07017, 基 07015)；遵义市科技局项目([2008]21).

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.

出局部最优解有一定作用. 但是, 在求解复杂的多峰问题时, 对于规避早熟以及在收敛速度及其精度上仍存在不足.

本文在总结前人研究的基础上, 结合文献[8]提出的动态多种群PSO算法(DMO-PSO), 提出一种*K*-均值聚类的动态多种群粒子群算法(KDMSPSO). 该算法通过*K*-均值聚类算法将种群划分为多子群; 并每间隔若干代重新构建新的子群, 以增加粒子间的信息交流; 在学习样本选择上, 每个粒子不是向它的邻居中表现最好的粒子学习, 而是向其所在子群的中心粒子和它的所有邻居学习. 这些策略可大大提升粒子跳出局部最优解的能力.

2 基本粒子群算法

在粒子群算法领域, 根据邻居拓扑结构的不同, 可以将粒子群算法分为局部版本粒子群算法(LPSO)和全局版本粒子群算法(GPSO). GPSO和LPSO算法的速度更新方程分别描述如下:

$$\overline{v}_i^{t+1}=\overline{v}_i^{(t)}+\varphi_1 r_1(\overline{p}_i^{(t)}-\overline{x}_i^{(t)})+\varphi_2 r_2(\overline{p}_g^{(t)}-\overline{x}_i^{(t)}),\tag{1}$$

$$\overline{v}_i^{t+1}=\overline{v}_i^{(t)}+\varphi_1 r_1(\overline{p}_i^{(t)}-\overline{x}_i^{(t)})+\varphi_2 r_2(\overline{p}_{\mathrm{neighbor}(i)}^{(t)}-\overline{x}_i^{(t)}).\tag{2}$$

其中: $\overline{p}_i^{(t)}$表示在迭代时刻t粒子i所经历的最好位置, 称作粒子的"认知部分"; $\overline{p}_g^{(t)}$表示在迭代时刻t群体中的粒子所经历的最好位置, 称作粒子的"社会部分"; $\overline{p}_{\mathrm{neighbor}(i)}^{(t)}$表示在迭代时刻$t$粒子$i$的邻居中所有粒子经历的最好位置, 也称为"社会部分". 这两种算法本质上是相同的, 只是粒子速度更新时, "社会部分"学习样本的选择范围不同. GPSO是在整个粒子群中选择, 而LPSO是在事先指定的粒子的邻居中选择.

3 基于*K*-均值动态多种群PSO算法

3.1 基于*K*-均值的多种群构建

文献[8]指出, 多种群PSO算法运行效率和运行精度往往优于单种群PSO算法, 但文中所提出的多种群的构建策略是通过随机选取种群中的粒子, 这种子群构建策略具有一定的盲目性. 现实鸟群中, 鸟类的聚合往往通过一定的规则以"头鸟"为中心飞向食物所在位置. 同时, 考虑到基于种群的进化算法的特性, 即当群体多样性降到很低时, 群体将在局部极值点停滞, 失去进一步开拓新区域的能力, 图1(b)和图1(c)以及图2(b)和图2(c)便证实了PSO算法无论是早熟收敛还是全局收敛, 种群中的粒子都会出现"聚集"现象, 进而说明种群多样性的匮乏将导致早熟收敛. 因此, 寻找一种能够增加种群多样性以及多种群构建策略, 对于提升算法的运行效率非常重要. 为此, 提出一种基于*K*-均值聚类算法来构建子群, 这种策略可以满足上述要求. 为了检测*K*-均值聚类算法构建子群(*K*-means construct)相比于随机选取粒子构建子群(randomly construct)的优势所在, 选取一个单峰(Rosenbrock)和一个多峰(Griewanks)函数来检测种群多样性与函数值的关系. 实验参数设置如下: 种群规模为30个粒子, 6×10^4函数评价, 检测函数为30维, 独立运行30次取平均值. 图1和图2给出了对于不同的子群构建策略种群多样性之间的关系. 其中: I表示基于*K*-均值的子群构建, II表示基于随机策略

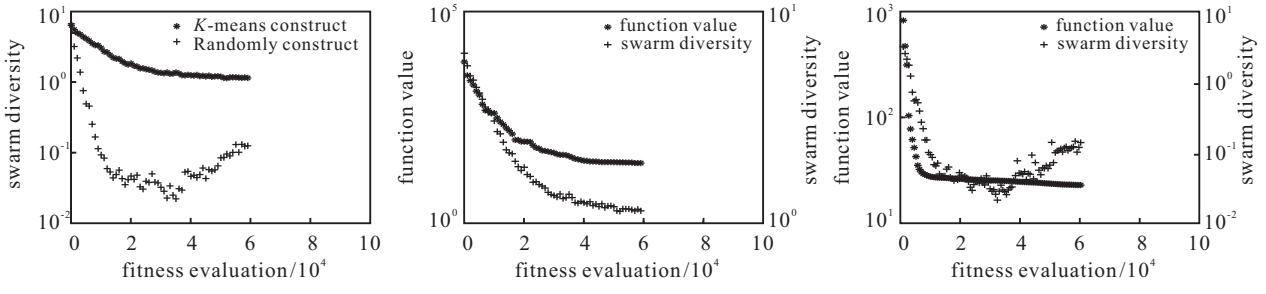

图 1 Rosenbrock函数种群多样性与适应函数值的关系

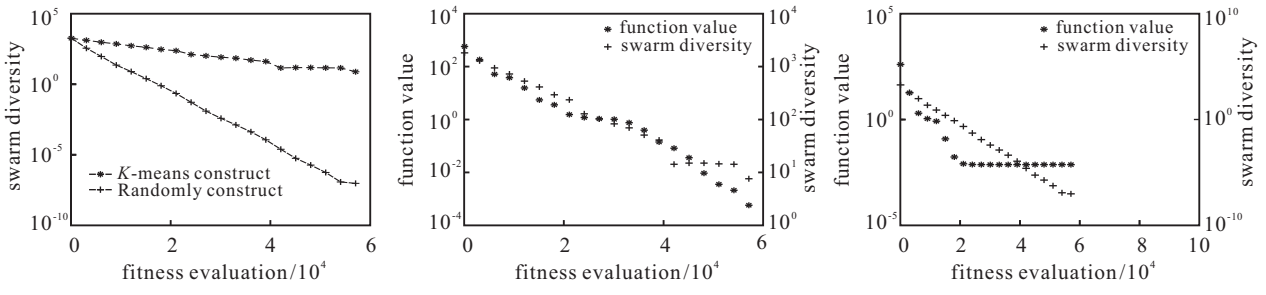

图 2 Griewanks函数种群多样性与适应函数值的关系

的子群构建. 这里采用文献[10]提出的种群多样性的度量方法, 即

$$D(t)=\frac{\sum_{i=1}^{N}\sqrt{\sum_{d=1}^{n}(x_{id}^{t}-\overline{p}_d)^2}}{NL}. \tag{3}$$

其中: L 为搜索空间对角最大长度, N 为种群规模大小, n 为解空间维数, x_{id}^{t} 表示在迭代时刻 t 第 i 个粒子的第 d 维坐标值, $\overline{p}_d$ 表示所有粒子的第 d 维坐标值的平均值. 平均粒子间距描述了种群中各粒子相互之间分布的离散程度, $D(t)$ 越小表示种群越集中, 种群的多样性越低.

从图 1(a) 和图 2(a) 可以看出, 基于 *K*-均值的种群多样性大于基于随机策略的种群多样性, 因此在本文提出的算法中采用 *K*-均值聚类算法重新构建子群. 值得注意的是: 在每个子群中, 粒子通过跟随该子群的中心粒子进行搜索, 如果子群的中心粒子处在局部最优解的位置, 则该子群极易出现早熟收敛. 因此, 为了避免出现早熟现象, 增加每个子群间的信息交流, 子群在搜索过程中不是固定的, 而是动态地进行组建. 组建规则为: 每间隔 R 代, 将种群重新划分为若干子群, 这里 R 称作重组间隔代数. 这种策略使得每个子群的信息互相交流, 进而提升粒子跳出局部最优解的能力.

3.2　子群构建间隔代数 (R) 与子群数量 (n) 的关系

文献[8]指出, 重新构建子群的间隔代数 (R) 和子群数量 (n) 将直接影响算法的运行效率, 但文中没有给出它们取值的方法. 一般而言, 如果 R 的取值较小, 则计算复杂度将会增加 (例如每间隔一代); 而 R 的取值较大, 则种群的多样性将减少 (极端情况不进行重新构建子群). 因此, 为了检测重新构建子群的间隔代数 R 与子群数量 n 之间的关系, 选择 2 个检测函数 (单峰 Rosenbrock 和多峰 Griewanks 检测函数) 在如下进化方程:

$$\begin{cases}\overline{v}_i^{t+1}\rightarrow\overline{v}_i^{(t)}+\varphi_1 r_1(\overline{p}_{C(i)}^{(t)}-\overline{x}_i^{(t)}),\\ \overline{x}_i^{(t+1)}=\overline{x}_i^{(t)}+\overline{v}_i^{(t+1)}\end{cases} \tag{4}$$

上独立运行 30 次, 每次运行 1 000 次迭代, 种群规模为 30 个粒子, 取 30 次运行的平均值测试这 2 个参数之间的关系. 表 1 和表 2 给出了不同参数 (R,n) 的运行结果, 其中"2(15)"表示有 2 个子群, 每个子群含有 15 个粒子.

表 1　不同参数在 Rosenbrock 函数的运行结果

R	2(15)	3(10)	5(6)	6(5)	10(3)	15(2)
4	28.91	21.45	15.49	9.03	16.66	28.09
7	17.35	16.88	14.19	7.95	15.45	25.43
10	16.69	15.80	13.05	6.70	10.31	11.80
13	20.90	18.04	12.39	6.87	12.78	17.84
16	23.28	19.99	13.77	7.21	13.63	27.76
20	26.46	19.89	14.74	9.58	12.72	25.78

表 2　不同参数在 Griewanks 函数的运行结果

R	2(15)	3(10)	5(6)	6(5)	10(3)	15(2)
4	7.81e-006	5.78e-006	5.11e-007	7.49e-007	5.05e-006	5.85e-006
7	8.78e-006	5.01e-006	4.03e-007	6.42e-007	5.75e-006	9.97e-006
10	4.42e-006	9.41e-007	2.46e-007	3.37e-007	3.06e-006	8.70e-006
13	9.11e-006	8.17e-006	5.15e-006	1.89e-006	8.22e-006	8.78e-006
16	9.68e-006	8.98e-006	8.80e-006	1.20e-006	8.37e-006	9.96e-006
20	9.49e-005	9.22e-006	8.76e-006	6.73e-006	8.43e-006	9.66e-005

从表 1 和表 2可以看出: 当 R 固定时, n 取值较大和较小的运行结果较差; 当 n 固定时, R 取值较大和较小的运行结果也较差. 因此, 在本文提出的算法中, 当种群规模为 30 个粒子时, 选取参数 $R=10$, $n=6$.

3.3　基于 K-均值的动态多种群粒子群算法

文献[7]推断出, 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学习, 将会造成"two steps forward, one step back"现象, 导致种群陷入局部最优解, 出现早熟现象. 因此, 在 KDMSPSO 算法中, 采用文献[11]提出的广义学习策略, 即粒子的"认知部分"学习样本不是其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). 由于 PSO 算法是对简单社会系统 (鸟群觅食过程) 的模拟, 在这个生物系统中, 领导者未必一定是在学识上 (检测函数的适应函数值) 最优的, 但领导者一定是群体的核心力量, 是其他成员学习的目标[13]. 因此, 每个粒子的"社会部分"学习样本是该粒子所在子群 (聚类) 的中心粒子. 根据上述分析, 每个粒子的速度和位置更新过程可按下式进行:

$$\begin{aligned}\overline{v}_i^{t+1}=&\overline{v}_i^{(t)}+\varphi_1 r_1(\overline{p}_{C(i)}(t)-\overline{x}_i^{(t)})+\\ &\varphi_2 r_2(\overline{p}_{\mathrm{bin}(i)}(t)-\overline{x}_i^{(t)}),\end{aligned} \tag{5}$$

$$p_{\mathrm{bin}(i)}^{d}=\arg\left\{\max\left[\frac{\mathrm{fitness}(p_j)-\mathrm{fitness}(p_i)}{|p_{jd}-x_{id}|}\right]\right\}. \tag{6}$$

其中: $d\in(1,2,\cdots,n)$, $i=1,2,\cdots,\mathrm{ps}$, $j\in\mathrm{neighbor}_i$, ps 表示种群规模 (粒子总数); $\overline{p}_{C(i)}(t)$ 表示在迭代时刻 t 粒子 i 所在子群的聚类中心; $\overline{p}_{\mathrm{bin}(i)}(t)$ 表示在迭代时刻 t 粒子 i 每一维学习对象; $\mathrm{neighbor}_i$ 表示粒子 i 的邻居构成的集合; $p_{\mathrm{bin}(i)}^{d}$ 表示 $\overline{p}_{\mathrm{bin}(i)}$ 的第 d 维; φ_1 和 φ_2 表示粒子加速常数, $\varphi_1=\varphi_2=2$; $\overline{p}_j=(p_j^1,p_j^2,\cdots,p_j^n)$, $j\in\mathrm{neighbor}_i$, 表示粒子 i 的邻居中任何一个粒子的 p_{best}; $\mathrm{fitness}(p)$ 表示向量 p 的函数值; r_1 和 r_2 表示在 (0,1) 之间的随机数. 式 (6) 的学习策略, 使得整个群体有更广阔的搜索空间, 被称为广泛学习策略. 算法 1 给出了 KDMSPSO 算法的伪代码.

算法 1　KDMSPSO 算法

Initialize positions and velocities of all particles.

Set k for the number of sub-swarm.

While stopping criterion is not satisfied

For ($i = 1 : \mathrm{ps},\ j = 1 : k$)

Construct sub-swarm using K-means algorithm.

Find the center position of sub-swarm(i).

End for

For each particle($i = 1 : \mathrm{ps}$)

Updating particle velocity, position, $p\mathrm{best}$, R.

Evaluate fitness values of the current particle i.

End for

If $R == 7$

Reconstruct sub-swarm using K-means algorithm.

End

End While

3.4 算法的计算复杂性分析

为了研究算法的计算复杂性, 用T表示最大迭代次数, N表示粒子总数, D表示决策变量的维数, k表示子群数目 (聚类数目). KDMSPSO算法在标准PSO算法的基础上, 增加了基于K-均值聚类算法的动态子群构建策略和广义学习策略. 其中K-均值聚类算法对种群进行划分的计算复杂度为$T_1(N)=O(N\times D\times k)$, 广义学习策略计算复杂度为$T_2(N)=O(N\times D)$, 基本PSO算法的时间复杂度为$T_3(N)=O(N\times D\times T)$, 则KDMSPSO计算复杂度$T(N)=O(N\times D\times k)+O(N\times D)+O(N\times D\times T)$. 因为$k$是一个常数, 所以$T(N)\approx O(N\times D\times T)$. 因此理论上KDMSPSO算法与基本PSO算法的复杂度在同一数量级上.

4 仿真实验及分析

4.1 检测函数和算法参数设置

为了测试所提出的算法的运行效率, 选择2个单峰和5个多峰Benchmark函数,分别为

$\mathrm{Sphere}(f_1)$, $\mathrm{Rosenbrock}(f_2)$, $\mathrm{Ackley}(f_3)$,

$\mathrm{Griewanks}(f_4)$, $\mathrm{Rastrigin}(f_5)$, $\mathrm{Schewfel}(f_6)$,

$\mathrm{Rastrigin\text{-}noncont}(f_7)$,

函数表达式见文献[7]. 为了增加这些检测函数的优化难度, 除单峰函数外, 其他所有函数都进行旋转, 旋转方法见文献[13], 用“$*$”表示对函数进行旋转. 选取5种有代表性的改进的PSO算法与本文提出的算法进行比较, 这些算法分别是: 局部版本PSO算法(CF-LPSO)[14], 基于适应距离比PSO算法(FDR-PSO)[5], 基于完全信息的PSO算法(FIPS)[6], 基于共协作的PSO算法(CPSO-H)[7].

为了使不同的算法具有可比性, 实验的相关设置如下: 所有的PSO算法的种群规模都为30个粒子, 每个检测函数独立运行30次, 对于非旋转的检测函数每次运行3×10^4函数评价, 而对于旋转的检测函数则每次运行6×10^4函数评价, 其他参数设置与算法提出时所用参数一致.

4.2 实验结果及分析

4.2.1 收敛特性仿真实验

为了检测不同算法的收敛特性, 给出在确定的函数评价次数下每种算法的运行特征. 图3和图4分别给出了各种PSO算法在3×10^4和6×10^4函数评价后的收敛特征曲线图. 同时, 为了检测各种算法的运行差异, 采用Wilcoxon秩和检验对KDMSPSO算法所得结果与其他5种PSO算法中最好的结果进行检验, 以验证KDMSPSO算法所得结果与其他5种PSO算法中最好结果的差别是否有统计学意义. 表3底部给出了检验结果. 其中$h=1$表示结果有差别, $h=0$表示结果没有差别.

表 3 达到停止标准时各种算法运行时间 s

算法	f_1	f_2	f_3	f_4	f_5	f_6	f_3^*	f_4^*	f_5^*	f_7^*
CF-LPSO	7	12	19	29	50	36	431	262	421	589
CF-GPSO	8	10	18	27	46	37	409	286	452	546
FDR-PSO	9	9.8	20	35	45	35	387	268	399	654
FIPS	9	8.9	16	40	70	32	374	363	488	653
CPSO	10	8.5	24	43	54	56	452	544	515	468
KDMSPSO	21	28	39	74	86	132	499	570	549	792
h	0	0	1	0	1	1	1	1	1	1

从图3可以看出, 对于所有的非旋转检测函数, 除Sphere和Griewanks函数外, KDMSPSO算法取得了最好的运行效果, 尤其对于Ackley, Rastrigin-noncont和Schewfel函数, 表现出明显优势. 而FDR-PSO和FIPS算法也表现出求解多峰问题的能力, 这主要是因为它们的学习策略不同于CF-GPSO和CF-LPSO算法, 使之能够有效地规避早熟收敛. 从图4可看出, 对于所有的旋转函数, 尤其Ackley, Griekwanks和Rastrigin函数, KDMSPSO算法表现出较好的收敛特性, 这主要是由于采用了基于K-均值聚类算法来构建子群和广泛学习策略, 进而增加了种群的多样性, 提升了种群跳出局部最优解的能力. 从表3底部的Wilcoxon秩和检验结果分析, 除了检测函数f_1, f_2和f_4, KDMSPSO算法所得结果与其他5种PSO算法中最好结果的差别在95%的置信区间内有统计学意义.

另外, KDMSPSO算法引入了K-均值聚类算法对种群进行划分和广泛学习策略, 因此探讨这些策略是否增加了算法的计算复杂度极其重要. 这里利用Matlab软件中的(tic,toc)命令在型号为ThinkPad-SL400的电脑上度量达到停止标准时, 每种算法所需时间. 停止标准为: 非旋转的检测函数为3×10^4函数评价, 旋转的检测函数为6×10^4函数评价. 从表3可以看出, KDMSPSO算法在所有检测函数上的运行时

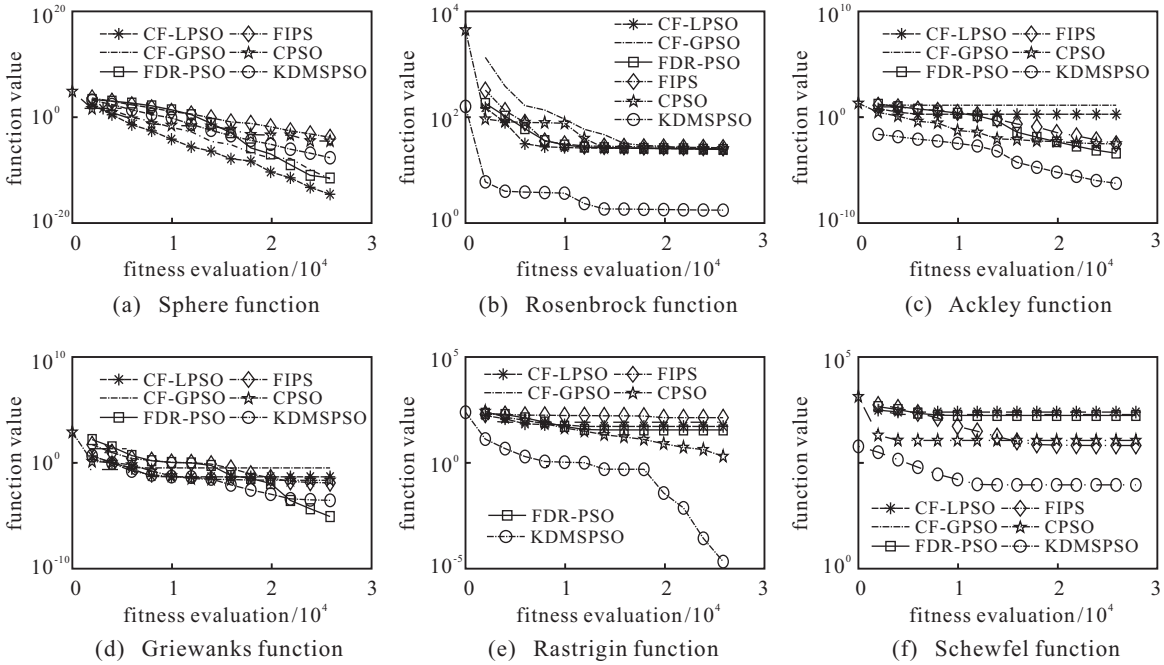

图 3　非旋转检测函数收敛特征图

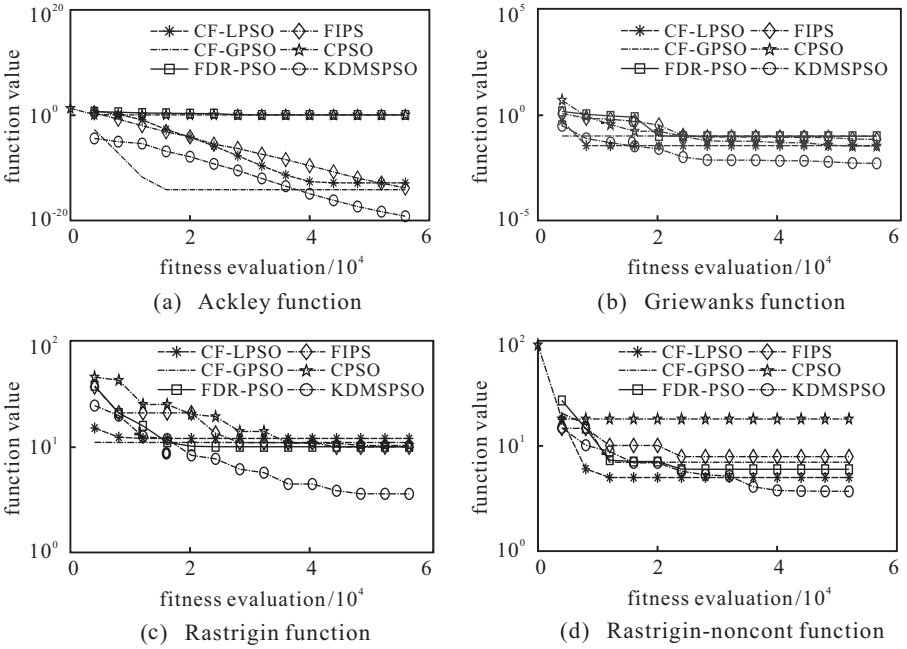

图 4　旋转检测函数收敛特征图

间与其他算法在同一数量级上, 引入的策略没有增加算法的计算复杂度, 这与计算复杂性分析结论一致(见 3.4 节).

4.2.2　鲁棒性分析

为了测试各种改进的 PSO 算法对于同一检测函数在不同的环境下 (函数旋转和不旋转)算法运行是否具有稳定性, 选择 3 个 Benchmark 函数 (Ackley, Griewanks 和 Rastrigin), 研究算法在这些检测函数上的稳定性. 表 4 给出了鲁棒性分析结果, 这里“鲁棒性”指在 60 次独立运行过程中算法成功达到给定阈值的比例, “FEs”表示达到阈值时的函数评价次数, “S”表示达到给定阈值的比例 (单位 %). 其中各函数的阈值分别为 0.05, 0.02 和 50.

从表 4 可看出: 对于Ackley函数, CF-LPSO, FDR-PSO 和 KDMSPSO 在非旋转的情况下 100%达到了给定阈值, 而旋转情况下只有 KDMSPSO 算法成功地达到了阈值. 对于 Griewanks 函数, 不论函数是否旋转, 仅仅 KDMSPSO 算法成功地达到了阈值, 但 CPSO 算法在非旋转的情况下显示其较为稳定. 对于 Rastrigin 函数, 在非旋转情况下, FDR-PSO, CPSO 和 KDMSPSO 算法成功地达到了阈值, 在旋转情况下 FIPS 有较高的成功比例, 仅 CPSO 和 KDMSPSO 算法 100%达到

表 4 各种算法的鲁棒性分析

	f_3		f_4		f_5		f_3^*		f_4^*		f_5^*	
	FEs	S	FEs	S	FEs	S	FEs	S	FEs	S	FEs	S
CF-LPSO	21 081	100	14 337	43	13 145	31	4 270	20	12 017	20	2 670	26
CF-GPSO	20 080	70	9 288	40	5 400	20	3 331	19	6 148	25	3 200	20
FDR-PSO	19 761	100	8 648	70	13 674	100	15 525	45	24 173	40	1 980	90
FIPS	19 832	90	8 643	72	13 654	94	10 642	90	37 202	39	3 100	95
CPSO	19 325	90	25 109	86	919	100	2 474	90	24 162	41	3 560	100
KDMSPSO	1 023	100	17 428	100	784	100	1 945	100	23 786	100	1 847	100

了阈值, 但 KDMSPSO 比 CPSO 用了较少的函数评价.

总之, 不论从算法的收敛特性还是从鲁棒性分析结果看, KDMSPSO 算法表现出了较好的收敛特性和稳定性, 所引入的基于 K-均值的聚类算法和广义学习策略没有增加计算复杂度.

5 PSO 算法在工程上的应用

为了测试 KDMSPSO 算法求解实际问题的能力, 对起重机箱型主梁进行优化. 该模型表达式如下:

$$\begin{cases} \min\ f(x) = x_1x_3 + x_2x_4; \\ \text{s.t. } 0.75L\Big[\dfrac{P_1 + 7.8\times 10^{-5}(x_1x_3 + x_2x_4)\times L}{3x_1x_2x_4 + x_1^2x_3}\Big] + \\ \qquad \dfrac{P_2}{3x_1x_2x_4 + x_1^2x_4} - 140 \leqslant 0, \\ \qquad \dfrac{P_1L^3}{1.68\times 10^6(3x_1^2x_2x_4 + x_1^3x_3)} - \dfrac{L}{700} \leqslant 0. \end{cases} \tag{7}$$

其中: $x = (x_1, x_3, x_3, x_4)^{\mathrm{T}}$ 为箱形梁的截面尺寸 ($700 \leqslant x_1 \leqslant 800$, $350 \leqslant x_2 \leqslant 400$, $5 \leqslant x_3, x_4 \leqslant 10$); $L = 10\,500$, $P_1 = 120\,000$, $P_2 = 12\,000$, $f(x)$ 为箱形梁单位长度的质量. 这里将各种 PSO 算法的优化结果与文献 [15] 给出的一些算法的实验结果进行对比, 以测试 KDMSPSO 算法解决实际问题的能力. 所有 PSO 算法的参数设置如下: 种群规模为 30 个粒子, 3×10^4 函数评价, 独立运行 30 次; 约束条件的处理方法采用内部罚函数法. 表 5 给出了不同算法的最优值比较, 其中各种 PSO 算法的结果是 30 次独立运行的平均值. 从表 5 可以看出, KDMSPSO 算法在对起重机箱型主梁优化时, 获得了较优的解.

表 5 不同算法优化结果比较

算 法	x_1	x_2	x_3	x_4	$f(x)$
常规算法	790.0	310.0	5.0	8.0	6.430e+003
神经网络	682.0	383.41	5.0	7.2	6.220 6e+003
混沌遗传	799.078 4	350.265 2	5.021 0	5.879 6	6.071 6e+003
MDE	772.328 5	350.0	5.0	5.833 3	5.903 3e+003
CF-LPSO	780.541 7	381.73	5.042	7.324 8	6.731 6e+003
CF-GPSO	745.654 9	376.92	5.041	6.586 3	6.241 4e+003
FDR-PSO	765.923 4	361.36	5.0	5.937 3	5.975 1e+003
FIPS	763.393 8	351.68	5.0	6.102 8	5.963 2e+003
CPSO	769.712 7	349.52	5.0	5.835 7	5.888 3e+003
KDMSPSO	760.456 2	348.37	5.0	5.812 6	**5.827 2e+003**

6 结 论

本文提出了一种基于 K-均值聚类的动态多种群 PSO 算法 (KDMSPSO). 在 KDMSPSO 算法中, 通过 K-均值聚类算法将种群分成若干个子群 (聚类); 为了增加种群的多样性, 避免种群陷入局部最优解, 每间隔若干代重新构建子群; 每个粒子的飞行方向由其所在子群的中心粒子和它所有邻居的各维信息共同调整 (包括它自己), 这种学习策略增加了粒子的探索空间. 从 Benchmark 函数模拟结果看, 这些策略提升了 PSO 算法求解多峰问题的能力, 并具有一定的稳定性. 在实际应用中, 通过对起重机箱型主梁优化设计模型进行优化, 表明了 KDMSPSO 算法较其他算法获得了质量更高的解. 因此, KDMSPSO 算法可以作为求解多峰问题的一种有效算法.

根据“没有免费的午餐理论 (no free lunch)” [16], 并不能说 KDMSPSO 算法是最好的算法. 但在现实世界, 对于所要解决的问题的曲线形状 (shape of the fitness landscape) 未知的情况, 选择一种能够较好地解决多峰问题的算法是一种明智的选择, 因为这种算法也能够解决单峰问题.

参考文献(References)

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

[2] Suganthan P N. Particle swarm optimizer with neighborhood operator[C]. Proc of IEEE Congress on Evolution Computation. Washington, 1999: 1958-1962.

[3] Mohais A S, Mendes R C. Neighborhood re-structuring in particle swarm optimization[C]. Proc of Australian Conf on Artificial Intelligence. Sydney, 2005: 776-785.

[4] 焦巍, 刘光斌. 动态环境下的双子群PSO算法[J]. 控制与决策, 2009, 24(7): 1083-1086.
(Jiao W, Liu G B. Two subpopulation swarm PSO algorithm in dynamic environment[J]. Control and Decision, 2009, 24(7): 1083-1086.)

[5] Peram T, Veeramachaneni K, Mohan C K. Fitness-distance-ratio based particle swarm optimization[C]. Proc of the IEEE Swarm Intelligence Symposium. Indianapolis, 2003: 174-181.

[6] Mendes R, Kennedy J, Neves J. The fully informed particle swarm: Simpler, maybe better[J]. IEEE Trans on Evolutionary Computation, 2004, 7(8): 204-210.

[7] Van D B F, Engelbrecht A P. A cooperative approach to particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 6(8): 225-239.

[8] Zhao S Z, Liang J J, Suganthan P N. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization[C]. Proc of IEEE Swarm Intelligence Symposium. Hong Kong, 2008: 3845-3852.

[9] 吕艳萍, 李绍滋, 陈水利, 等. 自适应扩散混合变异机制微粒群算法[J]. 软件学报, 2007, 18(11): 2740-2751.
(Lv Y P, Li S Z, Chen S L, et al. Particle swarm optimization based on adaptive diffusion and hybrid mutation[J]. J of Software, 2007, 18(11): 2740-2751.)

[10] Riget J, Vesterstroem J S. A diversity guided particle swarm optimizer-the ARPSO[R]. Aarhus: University of Aarhus, 2002.

[11] 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.)

[12] Charles S E. Animal ecology[M]. Chicago: University of Chicago Press, 2001.

[13] Salomon R. Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions[J]. BioSystems, 1996, 39(4): 263-278.

[14] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Trans on Evolutionary Computation, 2002, 2(6): 58-73.

[15] 卢青波, 张学良, 温淑花, 等. 差异演化算法改进与应用[J]. 农业机械学报, 2010, 41(2): 193-197.
(Lu Q B, Zhang X L, Wen S H, et al. Modified differential evolution and its application[J]. Trans on the Chinese Socitety for Agricultural Machinery, 2010, 41(2): 193-197.)

[16] Wolpert D H, Macready W G. No free lunch theorems for optimization[J]. IEEE Trans on Evolutionary Computation, 1997, 1(1): 67-82.

第 23 届中国控制与决策会议在绵阳召开

第 23 届中国控制与决策会议 (2011 CCDC)于 5 月 23 日～25 日在四川省绵阳市召开. 会议由东北大学、IEEE 新加坡工业电子分会和 IEEE 哈尔滨控制系统分会联合主办, 西南科技大学承办. 来自 28 个国家和地区的高等院校和科研机构的 450 多位代表参加了会议. 这是一次国际学术盛会, 大家齐聚一堂, 交流学术思想, 讨论学术问题, 充满了浓厚的学术气氛.

5 月 23 日上午举行了大会开幕式. 国际技术程序委员会主席杨光红教授主持会议, 东北大学副校长、大会总主席王福利教授致开幕辞, 西南科技大学校长肖正学教授致欢迎辞, 绵阳市副市长黄正良先生到会并讲话.

本届会议邀请了美国耶鲁大学 Morse 教授, IEEE 控制系统协会主席 Tempo 教授, 澳大利亚 Petersen 教授和中国工程院院士、清华大学吴澄教授等 16 位著名专家学者, 就当前控制与决策领域的热点问题和最新研究成果作了大会专题报告和准大会报告, 受到代表们的普遍欢迎.

会议分为 84 个专题组和 7 个张贴组进行学术交流. 代表们分别宣读了论文, 并就共同关心的问题进行了深入研讨. 每位作者都能认真回答其他代表提出的问题, 气氛非常热烈.

大会发行了 2011 中国控制与决策会议论文光盘. 光盘中的 855 篇论文将由 ISTP 收录, 并将进入 IEEE Xplore Data Base, 被 Ei 检索.

本届会议在评选张嗣瀛 (CCDC) 优秀青年论文奖的过程中, 有 5 位青年学者获得提名. 最终, 康卡迪亚大学的陈睿茹和北京交通大学的陈鹤楠凭借其优秀的论文和出色的报告双双赢得此奖项.

在张嗣瀛 (CCDC) 优秀青年论文奖颁奖仪式上, 大会副总主席徐心和教授主持仪式, 会议国际技术程序委员会主席杨光红教授介绍奖项评选过程, 中国科学院院士、大会顾问委员会主席张嗣瀛教授向获奖者颁发了获奖证书. 下届承办单位——太原理工大学信息学院党委书记李铁鹰介绍了第 24 届中国控制与决策会议的筹备情况, 并欢迎与会代表积极参加第 24 届中国控制与决策会议.