文章编号：1674-8085(2018)03-0032-06

一种求解大规模问题的自学习协同粒子群算法

*肖根福[1]，刘　欢[2]，李东洋[3]，欧阳春娟[2]

(1. 井冈山大学机电工程学院，江西，吉安 343009；2. 井冈山大学电子与信息工程学院，江西，吉安 343009；
3. 同济大学电子与信息工程学院，上海 201804)

摘　要：随着工程技术要求的提高，许多实际优化问题从低维问题发展成高维的大规模优化问题，自然计算算法在面对该类问题时容易陷入局部最优，而协同粒子群算法是解决大规模优化问题的重要手段之一。本文将子种群划分自学习策略和惯性权重自适应策略引入到协同粒子群算法中，增强了算法的自学习能力，提高了算法的全局寻优能力。实验结果表明，所提算法的性能超过了传统协同粒子群等算法，具有求解大规模问题的较大潜力。

关键词：大规模优化；协同粒子群算法；学习自动机；自学习

中图分类号：TP301.6　　**文献标识码**：A　　DOI:10.3969/j.issn.1674-8085.2018.03.008

AN SELF-LEARNING COOPERATIVE PARTICLE SWARM OPTIMIZATION ALGORITHM FOR SOLVING LARGE SCALE PROBLEMS

XIAO Gen-fu[1], LIU Huan[2], LI Dong-yang[3], OUYANG Chun-juan[2]

(1. School of Mechanical and Electrical Engineering, Jinggangshan University, Ji'an, Jiangxi 343009, China;
2. School of Electronics and Information Engineering, Jinggangshan University, Ji'an, Jiangxi 343009, China;
3. School of Electronics and Information Engineering, Tongji University, Shanghai 201804, China)

Abstract: With the improvement of engineering requirements, many practical optimization problems have been developed from low dimensional problems to large-scale optimization problems. The natural computing algorithm is prone to fall into the local optimal in the face of this kind of problem. Cooperative particle swarm optimization (CPSO) is one of the important means to solve large-scale optimization problems. In this paper, sub population partition self-learning strategy and inertia weight self-adaptive strategy are introduced into cooperative particle swarm optimization algorithm, which enhances the self-learning ability of the algorithm and improves the global optimization ability of the algorithm. The experimental results show that the performance of the proposed algorithm exceeds the traditional cooperative particle swarm optimization algorithms, and has great potential for solving large-scale problems.

Key words: large scale global optimization; cooperative particle swarm optimization; learning automata; self-learning

收稿日期：2018-03-07；修改日期：2018-04-18
基金项目：国家自然科学基金项目（61462046）；江西省自然科学基金项目（2016BAB202049）
江西省教育厅科技项目(GJJ160741,GJJ170633,GJJ170632)；江西省艺术规划项目（YG2016250,YG2017381）
作者简介：*肖根福(1980-)，男，江西赣州人，讲师，博士，主要从事智能控制、建模与优化研究(E-mail:xiaogenfu@163.com);
刘　欢(1981-)，女，江西吉安人，副教授，博士，主要从事图像处理、计算机视觉研究(E-mail:liuhuan816618@163.com);
李东洋(1992-)，男，河南南阳人，博士生，主要从事自然计算算法研究(E-mail:lidongyang0412@163.com);
欧阳春娟(1974-)，女，江西吉安人，副教授，博士，主要从事智能优化、信息隐写研究(E-mail:oycj001@163.com).

0 引言

近年来，自然计算在优化问题中取得了巨大的成功，进行了较多的实际应用。与传统的梯度下降法相比，自然计算算法在面对不可微、非线性问题时有较大优势。自然计算算法种类很多，如粒子群算法、微分进化算法、蚁群算法、萤火虫算法等等。每一种算法又有许多改进的形式，以 1995 年由 kennedy 提出的粒子群算法为例，有聚类粒子群[1]、散射学习粒子群[2]、自适应粒子群[3]和综合学习粒子群（CLPSO）[4]等等。

尽管自然计算算法取得了巨大的成功，但自然计算算法在求解大规模优化问题时性能将迅速恶化，遭遇“维数诅咒”[5]。协同进化策略是解决大规模问题的重要手段之一，协同进化采用“分而治之”的策略将大规模问题分解成小的子问题。自从 Potter[6]提出协同进化算法框架以来，研究者们提出了多种协同进化算法，如协同进化遗传算法[7]，协同进化粒子群算法[8]，协同进化微分进化算法[9]。

在协同策略框架中，有两个关键问题值得探讨，一是将大规模问题合理地划分为子问题，而划分子问题的关键是找到变量间的内在关系，将耦合关系强的变量归入同一子群；二是平衡子群的全局搜索能力和局部搜索能力，平衡这两种能力的关键是惯性权重的合理选取。

本文针对协同粒子群算法在求解大规模优化问题时容易陷入局部最优的特点，采取两个策略来改进协同粒子群算法，一是利用学习自动机挖掘出问题子维间的耦合关系，将关系相近的子维放入到一个子种群当中；二是将子种群自适应地分成两个部分，对优化效果差的粒子赋予新的惯性权重，令其跳出局部最优，加快算法收敛。

1 大规模优化问题

优化问题可以用下式描述：

$$\begin{aligned} &\min/\max F(x) = f(x_1, x_2, \cdots, x_D) \\ &x_i \in [x_{\min}, x_{\max}], \quad i = 1, 2, \cdots, D \end{aligned} \tag{1}$$

其中，$F(x)$为优化问题的目标函数，有最小化和最大化两类问题，本文算法针对的是最小化问题。$[x_{\min}, x_{\max}]$为边界约束，x_i 为决策变量，D 为决策变量的个数，即优化问题的维数。

一般来说，所谓大规模优化问题是指决策变量的个数 D 大于 100 的问题[5]。大规模优化问题的求解主要有两个难点：一是决策变量的增加使解空间呈指数式增长，在计算资源有限的情况下很难找到最优解；二是决策变量增加使多峰函数的局部最优解个数呈指数式增长，算法易陷入局部最优。

2 自学习协同粒子群算法

2.1 协同粒子群算法

粒子群算法是受飞鸟觅食行为的启发而提出的，每个粒子代表一个可行解，粒子追随个体极值（pBest）和全局极值（gBest）进行寻优。在标准粒子群算法中，粒子用下式更新自己：

$$V_{i+1} = w \cdot V_i + c_1 \cdot r_1 \cdot (p_b - p_i) + c_2 \cdot r_2 \cdot (p_g - p_i) \tag{2}$$

$$p_{i+1} = p_i + V_{i+1} \tag{3}$$

其中 V_i 为当代粒子的移动速度，V_{i+1} 是下一代粒子的移动速度，p_i 是当代粒子的位置，p_{i+1} 是下一代粒子的位置，p_b 为个体最优位置，p_g 为全局最优位置，r_1，r_2 是随机数，c_1，c_2 是学习因子，一般为常数，w 为惯性权重。

在标准粒子群算法中，每个粒子代表一个完整的解向量，包含了所有决策变量，包含了问题的每一子维。每次粒子位置的更新都要更新所有的决策变量，这容易造成“两步向前、一步向后”的问题，也就是说，虽然在优化过程中适应值函数是逐渐减小的，但问题的每一个子维并不都是向最优位置靠近，部分子维会远离最优位置，从而削弱算法的全局收敛性，使算法容易陷入局部最优。

为了保证粒子的每个子维都向正确的方向运动，Van den Bergh[8]提出了协同粒子群算法，该算法的基本思想为：将粒子群划分成多个相互独立的子种群，分别在问题的不同子维上搜索，每个子种群解决问题的一个子目标，同时，子群体之间保持信息沟通和知识共享，协同进化。在协同粒子群算法中，子种群是依次更新的，当某一子种群的粒子朝某一子维方向移动一次后，与其它子种群获得的当前最优位置结合，构造出一个完整的解向量，并

用评价函数计算适应度，之后下一个子种群再朝另一子维移动，直至所有子种群都完成更新。

2.2 子种群划分自学习

大规模优化问题变量众多，当大量的变量间存在相关性时问题求解变得更加困难。在协同进化过程中，子种群划分也就是问题分解是协同进化的最关键一步。一般来说，不考虑变量耦合的问题分解策略对可分离问题是有效的，但对不可分离问题，它们的性能将会变差。子种群划分策略的目标是将独立的变量分入不同的群，而非独立的、相关性强的变量则分入同一个子种群。本文利用学习自动机根据环境反馈，自学习地选择合理的种群划分。

近年来，强化学习在推动人工智能技术的发展中做出了重要贡献，人工智能围棋程序 AlphaGo Zero 正是依靠强化学习实现了无师自通[10]。学习自动机属于强化学习范畴，由 Tsetlin 于 1961 年提出，学习自动机的特点是能够在未知环境中找出最优参数。

典型的学习自动机由一个四元组{A,R,P,T}定义。其中 A={A1,A2,⋯,An}为学习自动机的动作集合，R={R1,R2,⋯,Rn}为环境的反馈集合，反馈分奖赏和惩罚两种情况，P={P1,P2,⋯,Pn}是与动作 A 一一对应的一组概率值，记录 A 集合中每个动作被选中的概率，T 是更新规则，不同类型的学习自动机有不同的更新规则。学习自动机模型如下图所示。

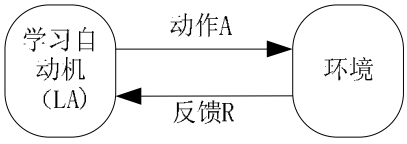

图 1 学习自动机模型

Fig.1 Model of learning automata

子种群划分要解决的问题是子种群应该包含哪些问题子维，在标准的协同粒子群算法中，种群划分在程序初始化时已经确定，不能随问题的改变而修改子种群与问题子维的对应关系。本文通过群表来自学习地确定子种群与问题子维的关系，群表如下图所示，SWARM 表示子种群，D 表示问题子维，LA 则是用来分配问题子维的学习自动机。0/1 表示学习自动机的行动（Action），0 表示该问题子维不属于该子种群，1 表示问题子维属于该子种群，同一列只有一个子种群为 1。

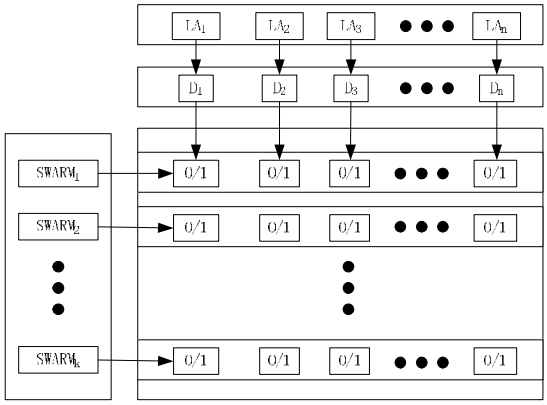

图 2 群表

Fig.2 Swarm table

每个子种群进化时，如果协同粒子群算法的全局最优（gBest）有提高，则将奖励信号反馈至学习自动机。获得反馈信号后，采取连续线性奖惩-不活跃策略（L_{R-I}）更新学习自动机的概率矩阵。通过概率矩阵，学习自动机可以选择对应行动（0/1），明确问题子维与子种群的对应关系。

2.3 惯性权重自适应

惯性权重是协同粒子群算法中最重要的可调参数之一。若惯性权重较大，则算法的全局搜索能力较强，找到全局最优解的概率较大；反之，则局部搜索能力较强，收敛速度较快。大规模优化问题的解空间大，容易使自然计算算法陷入局部最优，如果能合理选择惯性权重策略，平衡全局搜索能力和局部搜索能力，可以加快大规模优化问题的收敛速度，快速找到全局最优解。

Y.Shi 于 1998 年在 IEEE 国际进化计算学术会议中建议在粒子群算法中采用线性递减惯性权重策略[11]，即：

$$w^k = w_{ini} - \frac{w_{ini} - w_{end}}{K_{\max}} \times k \qquad (4)$$

其中 k 为当前迭代次数，K_{max} 为最大迭代次数，w_{ini} 为惯性权重的初始值，通常取 0.9，w_{end} 为惯性权重的最终值，通常取 0.4。这种策略对一些小规模优化问题会得到较好的效果，但对于不确定较大的大规模优化问题，单纯使用线性调整策略有一定局限性。

为了使粒子群算法跳出局部最优，惯性权重的非线性微分变化策略被提出，具体表达式如下:

$$w^k = w_{ini} + \frac{(w_{ini} - w_{end})}{K_{\max}} \times k - \frac{2(w_{ini} - w_{end})}{K_{\max}^2} \times k^2 \quad (5)$$

在这种策略中，w 先逐渐增加到某个较大的数值，然后再微分递减。该策略与线性递减策略相比可以在算法初期跳出局部最优，加强搜索力度。

协同粒子群算法搜索最优值时，子种群中的一部分粒子会到达适应值更优的位置，而另一部分则可能到达适应值更差的位置。因此根据迭代时适应值的差异，动态地调整惯性权重策略，使适应值较好的粒子保持原来的惯性权重策略，而适应值更差的粒子则采取新的惯性权重策略，将会增强协同粒子群算法的全局寻优和快速收敛的能力。

根据上述原理，本文根据迭代后适应值的变化情况，将协同粒子群算法的每一个子种群均划分为两个部分，一个是适应值比上一次迭代时更优的部分，另一个是适应值比上一次迭代更差的部分。各粒子惯性权重的取值由下式决定：

$$\omega_i^k = \begin{cases} \text{式（4），} & f(X_i^k) - f(X_i^{k-1}) \le 0 \\ \text{式（5），} & f(X_i^k) - f(X_i^{k-1}) > 0 \end{cases} \quad (6)$$

式中，x_i 表示第 i 个粒子，k 为当前迭代次数，f 为适应值函数。

也就是说，对于适应值比上一步更优的粒子，将其惯性权重设置为线性递减策略，即式(4)，确保快速收敛；对于适应值更差的粒子，将其惯性权重设置为非线性微分变化策略，在算法初期获得更大的惯性权重值，使子种群保持探索，跳出局部最优。

2.4 算法实现

自学习协同粒子群算法（SLCPSO）利用学习自动机找出问题子维之间的关系，自学习划分子种群，并引入惯性权重自适应策略，具体算法步骤如下：

1）参数设定。包括粒子群算法参数设定，如子种群规模、迭代次数、学习因子等；学习自动机参数如奖励参数等；

2）初始化。包括粒子群算法初始化，如粒子的初始位置、初始速度等；学习自动机初始化，如初始化群表、概率矩阵等；

3）根据群表划分子种群；

4）计算粒子适应值，更新 gbest，pbest；

5）如果 gbest 适应值改善，输出奖励信号；反之输出惩罚信号；

6）利用式(6)确定粒子的惯性权重在动态调整后的取值；

7）是否子种群内的粒子都迭代完成？没有则更新粒子序号，返回第 4 步;

8）根据奖惩信号，更新概率矩阵；

9）是否子种群都迭代完成？没有则更新子种群序号，返回第 4 步；

10）根据概率矩阵更新群表；

11）是否迭代次数或收敛精度达到？没有则返回第 3 步；

12）输出结果。

3 算法验证

为了评价本文提出的 SLCPSO 算法的性能，用表 1 所示 6 个经典的测试函数进行系列实验。其中，F1-F2 为单峰函数，F3-F6 为多峰函数，D 为问题维数。

表 1 测试函数

Table 1 Benchmark functions

序号	函数名	表达式	最值	定义域
F_1	Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	0	$[-100,100]^D$
F_2	Rosenbrock	$f_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2$	0	$[-0.048,2.048]^D$
F_3	Ackley	$f_3(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	0	$[-32.768,32.768]^D$
F_4	Griewanks	$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$	0	$[-600,600]^D$
F_5	Rastrigin	$f_5(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10$	0	$[-5.12,5.12]^D$
F_6	Schwefel	$f_8(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	0	$[-500,500]^D$

采用本文提出的SLCPSO算法与下面3种粒子群算法进行比较：惯性权重线性下降的粒子群算法（LWPSO）[11]，综合学习粒子群算法（CLPSO）[4]和H型协同粒子群算法(CPSO-H)[8]。

实验中SLCPSO算法参数设置：子种群数量为10个，每个子种群包含50个粒子，最大迭代次数为10000次，惯性权重初值为0.9，终止为0.4，学习自动机的奖励参数为0.1，测试函数的维数分成小规模问题（10维）和大规模问题（100维）两种情况，通过20次计算得出平均值与标准差。其他三种算法参数参照对应文献进行设置。

从表2可以看出，对10维的小规模优化问题来说，所有种类的粒子群算法都取得了比较好的效果，本文提出的SLCPSO算法总体来看效果最好，但CLPSO算法在F3、F4函数的计算中取得了比其他所有算法都好的结果，可见对小规模问题来说，协同类算法（CPSO-H、SLCPSO）并没有压倒性优势。而对表3中的100维大规模问题来说，所有粒子群算法的性能都明显下降，协同类算法（CPSO-H、SLCPSO）在求解大规模问题上有较大优势，本文提出的SLCPSO算法对所有的测试函数均取得了比其他3种粒子群算法更好的计算结果。

表2 算法均值与标准差(10维)

Table 2 The mean and standard deviation of algorithms (10 dimension)

函数	算法			
	LWPSO	CLPSO	CPSO-H	SLCPSO
F1	1.5E-53±5.9E-53	6.5E-33±1.3E-32	1.1E-15±4.3E-15	0±0
F2	3.4E+0±1.2E+0	3.3E+0±1.8E+0	6.8E+0±2.6E+1	7.9E-1±1.6E+0
F3	1.4E-14±7.8E-15	4.7E-15±1.7E-15	2.5E-9±4.3E-9	5.1E-15±2.6E-15
F4	9.5E-2±4.6E-2	4.6E-3±7.2E-3	4.0E-2±2.8E-2	5.1E-2±2.5E-2
F5	5.9E+0±2.6E+0	1.4E-1±3.6E-1	5.3E-16±2.3E-15	0±0
F6	3.6E+2±1.5E+2	5.3E+1±9.7E+1	3.0E+2±2.1E+2	4.1E+1±2.0E-1

表3 算法均值与标准差(100维)

Table 3 The mean and standard deviation of algorithms (100 dimension)

函数	算　　法			
	LWPSO	CLPSO	CPSO-H	SLCPSO
F1	7.9E+2±5.7E+2	1.7E+1±5.8E+0	3.8E-1±2.8E-1	1.8E-21±7.8E-21
F2	4.9E+2±1.0E+2	4.1E+2±6.0E+1	2.0E+2±4.3E+1	9.2E+1±1.6E+1
F3	1.0E+1±2.6E+0	2.9E+0±4.0E-1	1.1E-1±1.8E-2	2.7E-11±3.8E-11
F4	8.6E+0±5.6E+0	1.1E+0±8.0E-2	2.1E-1±8.5E-2	5.5E-3±1.6E-2
F5	5.0E+2±4.4E+1	6.6E+1±7.9E+0	1.9E+0±1.1E+0	6.9E-1±6.5E-1
F6	9.4E+3±8.8E+2	3.9E+3±6.8E+2	3.4E+3±4.9E+2	2.2E+2±7.8E+1

4 结论

针对协同粒子群算法在求解大规模优化问题过程中容易陷入局部最优、收敛速度慢的问题，在协同粒子群算法中引入子种群自学习划分策略和惯性权重自适应策略，提出一种自学习协同粒子群优化算法（SLCPSO）。该算法利用学习自动机挖掘问题子维之间的关系，动态确定问题子维和子种群的关系，并在适应值较差的部分粒子中采取非线性微分变化惯性权重策略，增强算法初期的寻优能力。SLCPSO算法与另外三种粒子群算法一起在测试函数集上进行了比较计算，由实验结果可知，SLCPSO算法在求解精度方面有显著的优势，表明该算法是一种求解大规模优化问题的较好方法。

参考文献：

[1] Kennedy J. Stereotyping: Improving particle swarm performance with cluster analysis[C]. Proceedings of IEEE Congress on Evolution. Computing. 2000, (2):1507-1512.

[2] Ren Z, Zhang A, Wen C, et al. A scatter learning particle swarm optimization algorithm for multimodal problems.[J]. IEEE Trans Cybern, 2014, 44(7):1127-1140.

[3] Hu M, Wu T, Weir J D. An Adaptive Particle Swarm Optimization With Multiple Adaptive Methods[J]. IEEE Transactions on Evolutionary Computation, 2013, 17(5):705-720.

[4] Liang J J, Qin A K, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(3):281-295.

[5] 刘睿,梁静. 求解大规模问题的协同进化动态粒子群优化算法[J].软件学报,2018,29(9):1-12

[6] Potter M A. The design and analysis of a computational model of cooperative coevolution[M]. George Mason University, 1997.

[7] Yu Y, Yu X. Cooperative Coevolutionary Genetic Algorithm for Digital IIR Filter Design[J]. IEEE Transactions on Industrial Electronics, 2007, 54(3):1311-1318.

[8] Frans V D B, Engelbrecht A P. A Cooperative approach to particle swarm optimization[J]. IEEE Trans.evol.comput, 2004, 8(3):225-239.

[9] Omidvar M N, Li X, Mei Y, et al. Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3):378-393.

[10] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676):354-359.

[11] Shi Y, Eberhart R C. A Modified Particle Swarm Optimization[C]. Proceedings of the Congress on Evolutionary Computation, 1998: 69-73.