动态概率粒子群优化模型及实验分析

倪庆剑[1] 邢汉承[1] 张志政[1,2] 王蓁蓁[1]
(东南大学计算机科学与工程学院 南京 210096)[1]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[2]

摘 要 对传统PSO算法中种群产生方式的特点,结合根据历史信息直接取样生成新种群的思想,抽象出动态概率粒子群优化(DPPSO)模型,并给出了该模型的形式化描述;同时提出了DPPSO模型可以采用的几种动态概率进化算子,最后通过常用Benchmark函数优化问题对DPPSO模型采用不同进化算子时的性能进行了实验分析。实验结果验证了DPPSO模型及所提出的进化算子的有效性,同时根据实验结果提出了进化算子设计与选择的指导性建议,并对相关参数设置也做了分析和讨论。

关键词 群智能,动态概率粒子群优化模型,种群生成方式,速度属性,动态概率进化算子
中图法分类号 TP18

Experiment and Analysis on Dynamic Probabilistic Particle Swarm Optimization Model

NI Qing-jian[1] XING Han-cheng[1] ZHANG Zhi-zheng[1,2] WANG Zhen-zhen[1]
(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)[1]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[2]

Abstract The dynamic probabilistic particle optimization model (DPPSO) was constructed based on the new inspiration of population generation method according to historical information about particles, and the formal description of DPPSO model was proposed, several dynamic probabilistic evolution operators were also designed for realization of DPPSO model. DPPSO algorithms with several evolution operators were tested on five benchmark functions which are common in evolutionary computation. Experiment results demonstrate the effectiveness of DPPSO model and the proposed evolution operators. The suggestion on designing evolution operators was provided based on experimental data and some configurations of relative parameter setting were also discussed and recommended.

Keywords Swarm intelligence, Dynamic probabilistic particle swarm optimization, Population generation method, Velocity, Dynamic probabilistic evolution operator

1 引言

1995年,Kennedy和Eberhart受到生物群体行为机制的启发,同时融入进化算法的思想,提出了粒子群优化算法(以下简称PSO算法)[1,2]。PSO算法求解优化问题时一般采用实数值编码,易于实现,作为一种有效的优化工具已经被成功地应用到诸多领域[3]。

近10年来,国内外的研究者们分别从不同角度对PSO算法进行了改进[4-9],这些改进一般通过调控粒子的速度进而更新粒子的位置,最终达到寻优的目的,可以归纳为轨迹方法(trajectory method)。

PSO算法易于编码实现,但难于解释其工作机制。PSO算法有较多的可变模块,各个模块之间也存在相互作用,这些都增加了理解各个模块对寻优所做贡献的难度。Kennedy解构了传统PSO算法中的各个模块,并将其重新组合,指出可以尝试用概率的方式生成新的种群,并做了初步的尝试,提出了高斯动态粒子群优化算法(GDPS)[10,11]。目前对这类方法的尝试与分析并不深入,理论分析及实验分析的文献均不多。本文对根据历史信息直接取样生成下一代种群的方法作了进一步的研究,提出了动态概率粒子群优化(dynamic probabilistic particle swarm optimization,以下简称DPPSO)模型,分析并给出了几种可以采用的动态概率进化算子,最后通过标准测试问题对DPPSO算法采用不同动态概率进化算子时的性能进行了分析和比较,同时提出了DPPSO模型中设计与选择动态概率进化算子的指导建议。

2 经典粒子群优化算法

PSO算法通常被作为一种进化算法。与其它的进化算法一样,PSO算法从一组随机的初始解出发,通过反复迭代进化粒子的位置,搜索解空间,达到寻优的目的。经典PSO算法[5]的速度更新和位置更新由式(1)和式(2)决定。

$$v(t+1)=\chi*[v(t)+c_1*rand()*(P_i-x(t))+c_2*$$

到稿日期:2008-03-04 本课题得到国家自然科学基金重大研究计划项目(90412014)资助。
倪庆剑 博士研究生,主要研究领域为群智能方法、机器学习,E-mail:nqj@seu.edu.cn;邢汉承 教授,博士生导师,主要研究领域为人工智能和逻辑程序设计;张志政 讲师,博士,主要研究领域为知识表示和推理;王蓁蓁 博士研究生,主要研究领域为增强学习。

$$Rand() * (P_g - x(t))] \quad (1)$$

$$x(t+1) = x(t) + v(t+1) \quad (2)$$

其中，χ 一般取经验值 0.729，c_1 和 c_2 一般取经验值 2.05。式(1)可以转换为式(3)：

$$v(t+1) = W1 * v(t) + W2 * rand() * (P_i - X(t)) + W2 * Rand() * (P_g - X(t)) \quad (3)$$

由式(2)和式(3)经过变换可得式(4)：

$$x(t+1) = x(t) + (\text{Ⅰ}) \; W1 * (x(t) - x(t-1)) + (\text{Ⅱ}) \; W2 * rand() * (P_i - X(t)) + W2 * Rand() * (P_g - X(t)) \; (\text{Ⅲ}) \quad (4)$$

在变换后的位置更新式(4)中，(Ⅰ)项表明粒子当前位置是搜索的起点，每一代的更新都依赖上一代的位置；(Ⅱ)项为速度属性，粒子具有上代位置的记忆，速度属性使粒子保持原有的运动趋势，当然这种趋势被加以权值 $W1$；(Ⅲ)项包含粒子个体和群体关于历史最优位置的信息，并综合考虑粒子的当前位置与个体经验及群体经验的差异，引导粒子向历史最优位置靠近。

3 动态概率粒子群优化模型

3.1 非轨迹方法的尝试

已有的研究表明，PSO 算法中，粒子在群体最优位置和个体最优位置之间振荡。已经有学者开始研究如何用直接取样的方法，代替轨迹方法根据历史最优信息生成新一代的种群[11-14]。大体上，这些方法可以分为以下 3 类：

(A) 粒子位置更新时将式(4)中的(Ⅱ)项(即速度属性项)去掉，同时将影响粒子的信息源推广到更大的邻域空间或者整个群体，而非推广到单个的邻域最优粒子，这种改进后的 PSO 算法性能一般。

(B) 在以粒子当前位置为中心的一定范围内依某种抽样方法进行搜索，从而确定粒子的新位置，而在多大范围内进行搜索则根据历史最优信息通过计算确定，但是这种尝试最后的效果也较为一般。

(C) 在一个中心点附近一定范围内依某种抽样方法进行搜索，确定粒子的新位置、中心点由粒子当前位置、粒子的运动趋势以及粒子个体的历史最优位置共同确定，在多大范围内进行搜索则根据群体中粒子的历史最优信息通过计算获得，Kennedy 提出的 GDPS 和 TUPS[10] 本质上就是这种生成新种群的方法，获得了较好的性能。

关于非轨迹方法的尝试，方法(A)和(B)获得的效果一般，而方法(C)的效果比较令人满意，因此对这一类方法值得更为深入的探讨。

3.2 动态概率粒子群优化模型

基于在解空间中根据历史信息直接取样生成粒子新位置的思想，本文在分析了几种概率分布函数特性以及随机数发生函数的基础上，提出动态概率粒子群优化模型。

定义 1(粒子，particle) 算法中用于寻优的个体，是多维解空间中的候选解，具有位置属性(X_i，如果解空间为 D 维，则 X_i 为 D 维向量)，算法通过进化粒子的位置寻优，同时粒子具有自身所经历的最好位置的记忆。

定义 2(种群，swarm) 种群是 N 个粒子的集合 $S=(X_1, X_2, \cdots, X_i, \cdots, X_{N-1}, X_N)$，算法通过种群中 N 个粒子间的相互协作进化粒子的位置。

定义 3(种群拓扑，swarm topology) 种群拓扑是种群中粒子之间的信息共享方式，通常可以用无向图的形式来表示。对于单个粒子来说，拓扑定义了粒子的邻域结构。

定义 4(适应度值函数，fitness function) 适应度值函数与最优化问题的目标函数相关，用于评价粒子的“优劣”，指导粒子群的进化。PSO 算法结束时，群体中适应度值最优的解即为 PSO 算法所获得的最优解。

定义 5(个体最优位置，particle best position)P_i 个体最优位置是粒子对自身历史信息的记忆，记录自身经历，获得最优适应度值的位置 P_i。

定义 6(集群趋势，centralized tendency)CT_i 集群趋势表明粒子 i 向邻域粒子靠近的趋势，由粒子当前位置、邻域粒子的最优位置等信息确定，表达群体中邻域粒子对粒子的吸引作用。其每一维 CT_{id} 通常可由式(5)定义：

$$CT_{id} = \sum_{k=1}^{K} P_{kd} / K - X_{id} \quad (5)$$

其中，i 为粒子的索引号，k 是粒子邻域粒子的索引，K 是粒子邻域粒子的个数，P_k 是邻域粒子的个体最优位置。

定义 7(离群趋势，outlier trend)OT_i 离群趋势表明粒子根据自身经验远离邻域粒子的趋势，由粒子个体最优位置以及邻域粒子的个体最优位置确定，表达粒子与邻域粒子之间信念的分歧。其每一维 OT_{id} 通常由式(6)定义：

$$OT_{id} = \sum_{k=1}^{K} |P_{kd} - P_{id}| / K \quad (6)$$

其中，i 为粒子的索引号，k 是粒子邻域粒子的索引，K 是粒子邻域粒子的个数，P_k 是邻域粒子的个体最优位置。

定义 8(动态概率进化算子，dynamic probabilistic evolution operator)$GEN(RNG, parameter)$ 动态概率进化算子定义了 PSO 算法在解空间中的取样方式，算法在生成粒子新位置时，先根据粒子当前位置、运动趋势和集群趋势等信息确定了粒子下一步搜索的大致区域，动态概率进化算子则再结合离群趋势以概率形式在解空间中取样。对于 $GEN(RNG, parameter)$，RNG 为服从特定概率分布的随机数发生函数，$parameter$ 是其参数序列。

对于基于 DPPSO 模型的算法，粒子的位置更新公式如式(7)所示。

$$X_i(t+1) = X_i(t) + \alpha * (X_i(t) - X_i(t-1)) + \beta * CT_i + \gamma * Gen() * OT_i \quad (7)$$

其中，α, β, γ 均为权值，$parameter$ 是其参数序列。进化算子在模型中仅为抽象的模式，基于 DPPSO 模型的算法具体实现时需要指定 $Gen()$ 的实现形式。

基于以上定义，动态概率粒子群优化模型的框架如图 1 所示。

Framework of Dynamic Probabilistic Particle Swarm Optimization Model:

```
Initialize particle swarm randomly;
Do{
    For each particle Xi{
    Calculate centralized tendency;
    Calculate outlier trend;
    Search the solution space according to the dynamic probabilistic evolution operator;
```

```
}
For each particle Xi{
Evaluate particle using fitness function;
Update the particle best position;
}
}While termination criteria are not satisfied.
```

图 1　动态概率粒子群优化模型流程

在 DPPSO 模型中，粒子没有传统 PSO 算法中的速度属性，算法根据粒子个体及邻域粒子的历史信息计算出粒子当前的集群趋势和离群趋势，再结合动态概率进化算子在解空间生成下一代粒子的位置。粒子的进化由集群趋势及离群趋势依据动态概率进化算子所推动。其中，集群趋势由粒子当前位置、粒子邻域粒子的历史最优位置确定；粒子先根据粒子的运动趋势以及集群趋势，同时结合当前位置对下一步搜索区域的中心位置做出判断，然而这只是粒子根据历史经验的一个预测值；离群趋势则表明当前粒子与邻域粒子对最优位置信念的差异程度。而动态概率进化算子则根据粒子与邻域粒子的分歧程度在集群趋势所指示的中心位置周围进行随机搜索，它在 PSO 算法中以概率的形式体现。值得注意的是，在基于 DPPSO 模型的算法中，粒子的进化充分关注了粒子的所有邻域粒子的历史经验，并非如传统 PSO 算法中仅仅关注邻域中获得最优位置的粒子的经验，而当粒子群的邻域拓扑结构是全连接型拓扑（Gbest 模型）时，算法关注了整个种群中所有粒子的经验。

3.3　可行的几种动态概率进化算子

根据 DPPSO 模型的定义及上面的分析，DPPSO 算法根据集群趋势和离群趋势结合动态概率进化算子，以概率的方式生成新一代的种群，其核心与实质在于选择合适的随机数发生函数（*RNG*）。关于 *RNG* 的选择，基于前人的研究[12]及我们的分析，粒子在全局极值点以及个体极值点之间振荡，其图形呈对称的钟形分布，因此可以考察服从具有类似特征的概率分布的随机数发生函数，即在中心点处概率值较大，往两边延伸时逐渐变小。而根据 DPPSO 模型的定义和分析，Kennedy 所提出的 GDPS 和 TUPS[10]可以看作该模型的具体实现版本，GDPS 所采用的是服从高斯分布的 *RNG*，TUPS 则使用了修正过的均匀分布。

本文考察了几种可行的服从对称分布的 *RNG*，其概率分布是 Cauchy 分布、Hyperbolic Secant 分布、Laplace 分布、Logistic 分布、Semicircle 分布、U 形分布。在这些概率分布函数中，Gaussian 分布、Cauchy 分布、Logistic 分布、Hyperbolic Secant 分布这 4 个分布的概率密度函数的图像均为钟型曲线，如图 2 所示。

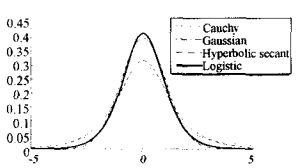

图 2　几种概率密度函数分布图

以基于 Logistic 分布的动态概率进化算子为例，其粒子位置更新根据式(5)、(6)、(8)确定：

$$X_i(t+1)=X_i(t)+\alpha*(X_i(t)-X_i(t-1))+\beta*CT_i+\gamma*Gen(LogisticRNG,parameter)*OT_i \quad (8)$$

其中，α,β,γ 均为权值，t 为进化代数，i 为粒子的索引号，k 是粒子邻域粒子的索引，K 是粒子邻域粒子的个数，P_k 是邻域粒子的个体最优位置，$Gen(LogisticRNG,parameter)$ 产生参数为 *parameter*，服从 Logistic 分布的随机数。基于 DPPSO 模型的 PSO 算法称为 DPPSO 算法，而对于指定了具体的动态概率进化算子的 DPPSO 算法，可以将相关的概率分布作为其名称的一部分，例如，如果动态概率进化算子基于 Logistic 分布设计，则相应算法称为 Logistic 动态粒子群优化算法（LDPSO）或者 DPPSO-Logistic。DPPSO-Logistic 算法流程如下：

(1) 随机初始化粒子群中粒子的位置；

(2) 用适应度值函数评估粒子群中的粒子；

(3) 将粒子的个体最优位置设置为粒子的当前位置；

(4) 判断算法的结束条件是否满足，如果满足，转向步骤(9)；否则，执行步骤(5)；

(5) 对粒子群中的所有粒子执行以下操作：

A)根据式(5)计算粒子的集群趋势；

B)根据式(6)计算粒子的离群趋势；

C)根据基于 Logistic 分布的动态概率进化算子依式(8)更新粒子的位置；

(6) 用适应度值函数评估粒子群中的粒子；

(7) 更新粒子群的粒子的个体最优位置；

(8) 判断算法的结束条件是否满足，如果满足，执行步骤(9)；否则，转向步骤(5)；

(9) 输出记录的运行结果，算法运行结束。

其它版本的 DPPSO 算法在实现时主要区别在于集群趋势、离群趋势的计算以及动态概率进化算子中概率分布函数的选择。在实验部分，将给出 DPPSO 算法的工作参数。

4　实验研究

4.1　实验设置及 Benchmark 函数问题

为了验证本文所提出的 DPPSO 模型的有效性，并对比采用不同进化算子时的性能差异，本文引入 5 个 Benchmark 问题进行分析，分别是 Sphere 函数、Schaffer F6 函数、Rosenbrock 函数、Rastrigin 函数、Griewank 函数，其具体的参数及相关设置如表 1 所列。对所有算法的种群规模 N 设为 20，拓扑结构为经典的全连接型拓扑。DPPSO 算法的参数设置如下：$\alpha=0.729,\beta=2.187,\gamma=0.5$。

表 1　五个常用的标准测试函数

函数	表达式	相关的参数
Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$	维数：30 初始范围：[−100,100] 误差范围：<0.01
Schaffer F6	$f(X)=\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1+0.001(x_1^2+x_2^2)]^2}-0.5$	维数：2 初始范围：[−100,100] 误差范围：<0.00001
Rosenbrock	$f(x)=\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	维数：30 初始范围：[−30,30] 误差范围：<100
Rastrigrin	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	维数：30 初始范围：[−5.12,5.12] 误差范围：<100
Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	维数：30 初始范围：[−600,600] 误差范围：<0.1

实验对用不同动态概率进化算子的 DPPSO 算法在 5 个 Benchmark 问题上进行了测试，采用了两种测试方法：(1)固定算法的最大迭代次数。重复实验，考察算法到达最大迭代次数时所获得的最优值、中值、均值、标准差以及最差值；(2)给定可接受解与理论最优解的误差范围。重复多次实验，测试每个算法满足误差范围的比率(成功率)以及满足误差范围时的平均迭代次数。第一种测试方法主要评估算法发现最优解的能力，第二种测试方法主要评估算法的稳定性以及收敛速度。

4.2 实验结果及讨论

固定最大进化代数为 3000，重复实验 100 次后，各个算法所获得的最优值、中值、均值、标准差以及最差值等数据如表 2 所列。给定可接受的误差范围如表 1 中所列，设定最大迭代次数 3000，重复实验 100 次后，达到最大迭代次数之前满足误差范围的比率以及平均迭代次数如表 3 所列。

表 2 5 个标准测试问题的实验结果对比

函数	算法	最小值	中值	均值	标准差	最大值
Sphere	GDPS	4.63E-99	1.87E-28	0.039499	0.30312	2.9316
	TUPS	4.65E-29	2.71E-10	200.02	1407	10000
	Laplace	9.45E-64	2.61E-10	0.021869	0.11604	1.0321
	Cauchy	1.06E-33	8.33E-31	1.21E-26	1.19E-25	1.19E-24
	Semicircle	8.10E-98	1.62E-49	0.001613	0.015579	0.15581
	Ushape	1.49E-35	1.36E-34	1.91E-34	1.68E-34	8.24E-34
	Secant	2.15E-39	4.30E-38	6.17E-38	6.28E-38	3.21E-37
	Logistic	3.80E-77	7.64E-76	5.28E-69	5.28E-68	5.28E-67
Schaffer F6	GDPS	0 [66%]	0	0.001202	0.002881	0.009716
	TUPS	0 [65%]	0	0.003304	0.004625	0.009716
	Laplace	0 [58%]	0	0.001478	0.002811	0.009716
	Cauchy	0 [39%]	0.000812	0.003585	0.004245	0.009716
	Semicircle	0 [59%]	0	0.001493	0.002981	0.009716
	U shape	0 [63%]	0	0.000105	0.000972	0.009716
	Secant	0 [77%]	0	0.000303	0.001646	0.009716
	Logistic	0 [77%]	0	0.000434	0.001931	0.009716
Rosenbrock	GDPS	23.21	33.212	115.82	237.93	1747.3
	TUPS	4.0623	69.943	2889.1	15409	90007
	Laplace	24.448	49.593	213.13	624.19	5384.7
	Cauchy	21.52	25.613	25.613	0.81481	29.394
	Semicircle	16.746	29.307	99.832	190.55	1358.1
	Ushape	21.619	22.52	31.293	39.403	385.37
	Secant	21.817	22.611	29.208	19.194	99.189
	Logistic	21.999	27.418	51.417	74.21	468.71
Rastrigin	GDPS	12.056	20.894	21.524	5.9112	43.78
	TUPS	47.758	103.48	108.44	29.237	185.06
	Laplace	7.0016	22.884	22.796	6.8855	49.238
	Cauchy	8.9546	16.914	18.412	5.9388	34.824
	Semicircle	10.945	20.894	21.279	6.3117	45.78
	Ushape	11.94	25.391	25.507	7.223	52.762
	Secant	9.9496	20.894	21.056	5.3473	38.803
	Logistic	5.9698	15.919	16.963	4.9516	31.839
Griewank	GDPS	0 [43%]	1.04E-11	0.009677	0.036431	0.32446
	TUPS	0 [1%]	0.03687	1.8843	12.665	90.145
	Laplace	0 [14%]	1.28E-06	0.0168	0.085888	0.83599
	Cauchy	0 [35%]	0.012316	0.017171	0.018645	0.070662
	Semicircle	0 [53%]	0	0.003988	0.012719	0.0686
	Ushape	0 [96%]	0	0.000345	0.001752	0.012316
	Secant	0 [94%]	0	0.000468	0.001877	0.009865
	Logistic	0 [90%]	0	0.001059	0.003867	0.027037

表 3 平均迭代次数以及成功率

算法	平均迭代次数及成功率									
	Sphere		Schaffer F6		Rosenbrock		Rastrigin		Griewank	
GDPS	198	96%	1400	73%	147	82%	337	100%	182	97%
TUPS	680	92%	405	65%	735	82%	222	42%	481	78%
Laplace	193	92%	1275	60%	142	71%	315	100%	180	98%
Cauchy	445	100%	1078	39%	109	100%	218	100%	423	100%
Semicircle	201	99%	1419	64%	144	81%	408	100%	188	100%
Ushape	507	100%	1921	93%	412	99%	206	100%	519	100%
Secant	463	100%	1644	92%	394	100%	484	100%	481	100%
Logistic	249	100%	1627	90%	176	91%	893	100%	233	100%

图 3—图 7 分别显示了 5 个 Benchmark 函数的最优适应度值进化曲线。为较直观地考察各种算法最优适应度值的进化趋势，同时为避免出现对 0 取对数的情况，本文将最优适应度值加上 10^{-12} 后再取对数。

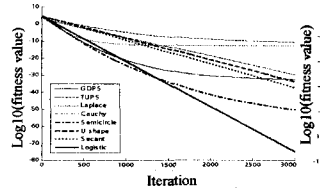

图 3 Sphere 函数最优适应度值进化曲线

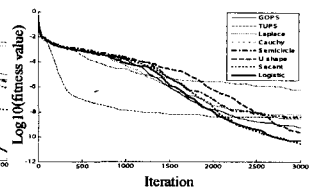

图 4 Schaffer F6 函数最优适应度值进化曲线

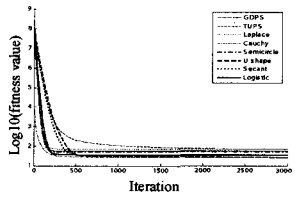

图 5 Rosenbrock 函数最优适应度值进化曲线

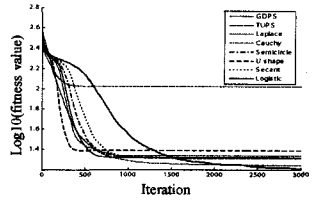

图 6 Rastrigin 函数最优适应度值进化曲线

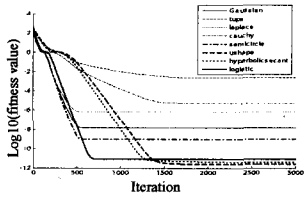

图 7 Griewank 函数最优适应度值进化曲线

从表 2 可以看出，实验中所对比的几种版本的 DPPSO 算法，除 TUPS 表现较不稳定外，基本都获得了可以接受的解，再结合表 3 中的成功率数据，这些 DPPSO 算法的表现是可以接受的。

具体地，对于 Sphere 函数，从最小值、中值、均值等指标来看，Logistic 版本以及 Secant 版本的结果较好。而且从标准差来看，算法性能也非常稳定，占较大优势。同时，图 3 显示这两个版本一直保持着向理论最优值进化的趋势，且较为领先。Semicircle 版本和 GDPS 虽然在最小值上占优，但是表现不稳定。TUPS 的表现较差，多次实验的结果差异较大，性能较不稳定。

对于 Schaffer F6 函数，从最小值、中值、均值数据来看，各版本的性能差异并不明显，但 Logistic 版本和 Secant 版本在多次实验获得最优值 0 的比率达到 77%，以较大优势领先其它版本的 DPPSO 算法，而且从成功率指标来看，这两个版本也占优。图 4 显示了各个版本 DPPSO 算法的进化趋势。TUPS 虽然在进化初期收敛迅速，但其后便很难进一步接近理论最优解。而 Logistic 版本和 Secant 版本则保持着一致向最优解靠近的趋势，且在进化的后期领先于其它版本。

对于 Rosenbrock 函数，从最小值、中值、均值数据来看，除 TUPS 性能不稳定表现较差外，其它几种版本的性能差异不大，结合图 5 的最优适应度值进化曲线，并没有占据绝对优势的版本，而再结合表 3 中所列的数据，发现 Cauchy，Semi-

circle 以及 Logistic 这 3 个版本的 DPPSO 算法不仅成功率相比其它较高，而且平均迭代次数较少，均少于 200。因此，综合来看，Cauchy，Semicircle 以及 Logistic 这 3 个版本略微占优。

对于 Rastrigin 函数，从表 2 中的数据来看，Cauchy，Secant 以及 Logistic 这 3 个版本的 DPPSO 算法相比其它算法略微占优。其它版本除 TUPS 相对较差外，性能差异并不明显。从表 3 中所列的数据来看，除 TUPS 外，其它版本的成功率均达到 100%。图 6 的最优适应度值进化曲线显示，Logistic 和 Cauchy 版本虽然在进化初期收敛的速度优势并不明显，但它们在进化的末期仍然有向理论最优解靠近的趋势，并最终优于其它版本。

对于 Griewank 函数，从最小值、中值、均值指标来看，Logistic，Secant 以及 U shape 版本的优势非常明显，而且它们在 100 次实验中找到最优值的比率均在 90%以上，远优于其它几种。从成功率指标来看，除 TUPS 较低外，其它几种均接近或达到 100%。再结合图 7 的进化曲线，仍然是 Logistic，Secant 以及 U shape 版本占优。

综合以上的分析，我们发现 DPPSO 模型采用本文所提出的动态概率进化算子时都获得了较为满意的解。这充分说明了新的种群产生方式对算法性能有着积极的影响，同时，也验证了 DPPSO 模型的有效性，值得进一步的研究和拓展。同时我们注意到在实验中对比的几种动态概率进化算子中，性能较为突出的是 Logistic 版本以及 Secant 版本，它们所采用的 RNG 的概率密度函数具有相似的特征，都呈钟形，如图 2 所示；而且相对高斯分布来说具有“厚尾”的特征，这使得粒子在搜索解空间时可以到离集群趋势较远的地方搜索。Cauchy 分布的尾部也较长，但仅在部分 Benchmark 函数上占优，这说明集群趋势对粒子进化趋势具有指导作用。

Kennedy 最早提出的 GDPS 已经获得了相当好的性能[10]。实验数据显示本文所提出的几种动态概率进化算子同样相当有效，其中 Logistic 版本和 Secant 版本更显示了突出的优势。这些都表明，本文根据新种群产生方式所抽象出的 DPPSO 模型，具备进一步研究的价值。

结束语 本文结合新的种群产生方式，抽象出动态概率粒子群优化模型，给出了 DPPSO 模型的形式化定义，并在前人研究的基础上，提出了几种可行的动态概率进化算子。实验结果显示，采用这些进化算子的 DPPSO 算法性能良好，达到了预想的效果，其中 Logistic 版本和 Secant 版本的优势较为突出。DPPSO 模型根据群体历史信息计算出集群趋势及离群趋势，再根据集群趋势与离群趋势以概率的方式产生新一代粒子群体。这种新的粒子群进化模型以一种新的描述形式诠释了 PSO 算法的工作过程，比传统的描述形式更为简洁和直观，同时也为理解 PSO 算法的工作机制进而分析其收敛性和稳定性提供了一个全新的角度。

基于已有的研究，进一步的工作主要有如下方面：

(1) 动态概率进化算子的设计，Logistic 版本与 Secant 版本的效果较好，但仍然有需要改进的地方。比如对于个别测试函数，在进化的初期，最优适应度值的下降较慢。而 Cauchy 版本，虽然综合性能一般，但针对个别测试函数效果较好，这些都值得进一步地关注。

(2) 动态概率进化算子的选择。实验表明，进化算子中的 RNG，其概率密度函数具备某些特征时效果较好。比如 Logistic 版本和 Cauchy 版本，其 RNG 的概率分布广义上都属于广义的 Tukey-Lambda 分布，可以进一步考察根据这类分布所设计的动态概率进化算子的有效性。

(3) DPPSO 算法的改进策略研究。实验结果显示，综合指标较好的 Logistic 版本以及 Secant 版本，并非在任何指标上都占优。可以考虑混合优化策略，融合其它优化算法的思想，提高 DPPSO 算法的高效性和稳定性。

(4) DPPSO 算法中的邻域拓扑研究。DPPSO 模型中，集群趋势和离群趋势的计算涉及所有的邻域粒子，并非传统 PSO 算法速度更新时只考虑群体最优或邻域最优粒子，邻域结构受到了更大的关注，因此有必要进一步对 DPPSO 算法中的邻域拓扑进行研究。

参 考 文 献

[1] Kennedy J，Eberhart R C. Particle swarm optimization // Proceedings of the IEEE International Conference on Neural Networks. Perth，Australia，1995

[2] Eberhart R C，Kennedy J. A new optimizer using particle swarm theory // Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya，Japan，1995

[3] Shi Y. Particle swarm optimization. IEEE Connections，2004，2(1)：8-13

[4] Shi Y H，Eberhart R C. A modified particle swarm optimizer // Proceedings of the IEEE International Conference on Evolutionary Computation. Anchorage，AK，USA，1998

[5] Clerc M，Kennedy J. The particle swarm - explosion，stability，and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation，2002，6(1)：58-73

[6] Kennedy J. Small worlds and mega-minds：effects of neighborhood topology on particle swarm performance // Proceedings of the 1999 Congress on Evolutionary Computation. Washington，DC，USA，1999

[7] 赫然，王永吉，王青，等. 一种改进的自适应逃逸微粒群算法及实验分析. 软件学报，2005，16(12)：2036-2044

[8] 高海兵，周驰，高亮. 广义粒子群优化模型. 计算机学报，2005，28(12)：1980-1987

[9] 崔志华，曾建潮. 基于微分模型的改进微粒群算法. 计算机研究与发展，2006，43(4)：646-653

[10] Kennedy J. Dynamic-probabilistic particle swarms // Proceedings of the Conference on Genetic and Evolutionary Computation. Washington DC，USA，2005

[11] Kennedy J. In search of the essential particle swarm // Proceedings of the 2006 Congress on Evolutionary Computation. Vancouver，BC，Canada，2006

[12] Kennedy J. Bare bones particle swarms // Proceedings of the IEEE Swarm Intelligence Symposium. Indianapolis，IN，USA，2003

[13] James K. Probability and dynamics in the particle swarm // 004 IEEE Congress on Evolutionary Computation. Portland，Oregon，USA，2004