DOI: 10.3724/SP.J.1219.2011.00733

基于 Predator-Prey 行为的双种群粒子群优化算法

秦全德[1]，牛 奔[1,2]，李 丽[1]，李荣钧[3]

（1. 深圳大学管理学院，广东 深圳 518060；2. 中国科学院合肥智能机械研究所，安徽 合肥 230031；
3. 华南理工大学工商管理学院，广东 广州 510640）

摘 要：根据生物的捕食-食饵（predator-prey）行为的规律，提出了一种双种群粒子群优化（DPPSO）算法. 将粒子分成 predator 和 prey 两个种群，其中 predator 种群每间隔一定的迭代次数后排斥 prey 种群. 在排斥的过程中，predator 种群采用“擒贼先擒王”的策略，逐步向 prey 种群的群体最优位置靠近，同时每个 prey 粒子尽量逃离距离最近的 predator 粒子. 采用了一种速度变异的方法提高 prey 种群在停滞状态时摆脱局部最优的能力. 基准函数的仿真结果表明 DPPSO 具有速度收敛快和全局搜索能力强的特点.

关键词：粒子群优化；predator-prey 行为；双种群；基准函数

中图分类号：TP18 **文献标识码**：A **文章编号**：1002-0411(2011)-06-0733-07

A Double-Population Particle Swarm Optimization Algorithm Based on Predator-Prey Behavior

QIN Quande[1], NIU Ben[1,2], LI Li[1], LI Rongjun[3]

(1. *College of Management, Shenzhen University, Shenzhen* 518060, *China*;
2. *Hefei Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei* 230031, *China*;
3. *School of Business Administration, South China University of Technology, Guangzhou* 510640, *China*)

Abstract: According to biological rules of predator-prey behavior, a double-population particle swarm optimization (DPPSO) algorithm is proposed. The particles are divided into two populations, the predator population and the prey population. The particles in the predator population exclude those in prey population in a certain interval of iterations. During the course of exclusion, particles in predator population adopt the strategy of “catching the ringleader first in order to capture all his bandit followers”, which means that all predator particles chase after the best position in prey population and particles in prey population try their best to escape from the nearest predator particles. In order to enhance the capacities of escaping from local optimum of the particles in prey population with stagnation state, a speed mutation method is used. The experiment results on benchmark functions show that DPPSO algorithm has the properties of fast convergence rate and strong global searching capability.

Keywords: particle swarm optimization; predator-prey behavior; double populations; benchmark function

1 引言（Introduction）

粒子群优化（particle swarm optimization，PSO）算法是 Kennedy 和 Eberhart 模拟鸟群觅食行为规律而提出来的一种基于种群搜索的全局优化算法[1]. 由于其编码简单，需要调整的参数少，且易于编程实现[2]，目前已经成功应用于函数优化、神经网络训练、流水线调度和金融优化等众多领域[3-6]. 相关研究表明，PSO 算法在优化复杂多维函数时，虽然前期的收敛速度较快，但在迭代的后期，由于群体的多样性丧失较快，容易出现早熟收敛现象[7]. 研究人员给出了多种改进方法来克服这个问题，这些方法可以简单归纳为算法参数调节[3]、群体拓扑结构的调整[8]、混合算法[9-10]和新的学习策略[11].

PSO 算法源于对社会型群居生物的行为模拟，因而将大自然中其它一些生物行为机制嵌入到 PSO 算法中是一条潜在可行的改进途径. 国内外学者在这方面已开展了一些相关的研究[12-13]. 但是，根据已报道的文献来看，目前的研究成果相对较少且不够系统. 受自然界中 predator-prey 行为的竞争协同进化的启发，Silva 等[13]提出了一种改进 PSO 算法. 在该算法中，群体中了增加了一个 predator 粒子，该粒子在搜索过程中迫使陷入局部最优点的 prey 粒

基金项目：国家自然科学基金资助项目（71071057，71001072）；中国博士后基金资助项目（20100480705）.
通讯作者：牛奔，drniuben@gmail.com 收稿/录用/修回：2010-09-09/2011-05-06/2011-07-12

子逃离，受排斥后重组的 prey 种群的粒子增加了开采（exploitation）能力并逐步靠近全局最优解. 用 φ 表示 [0,1] 之间均匀分布的随机数，p_f 代表 prey 粒子的恐惧率，对于 prey 粒子的每一维，若 $\varphi < p_f$，prey 粒子的这一维将受到排斥. 从文 [13] 的仿真实验中可以看出，该算法的参数设置具有较大的问题依赖性（对于不同的问题，参数 a 和 p_f 的设置有所差别），这在一定程度上降低了其实用性. 2006 年，Higashitani 等 [14] 提出了一种基于 predator-prey 行为的改进 PSO 算法，该算法增加了 predator 粒子的数量，并重新设计了 predator 和 prey 种群中粒子的速度更新公式，遗憾的是算法参数设置依然存在问题依赖性（如算法中的参数 P）.

本文根据生物 predator-prey 行为的规律，提出了一种新的双种群的粒子群优化算法（DPPSO）. 在 DPPSO 中粒子分成 predator 和 prey 两个种群，predator 种群间隔一定的迭代次数定期对 prey 种群进行排斥. 本文采用了一种速度变异的方法提高 prey 种群的粒子处于停滞（stagnation）状态时逃离局部最优的能力. 基准函数的仿真实验结果表明了 DPPSO 算法的有效性.

2 基本粒子群优化算法（Basic PSO）

PSO 算法的基本思想是将每个粒子视为求解问题的一个解，其在搜索过程中的状态属性由 2 个向量描述：位置向量 $\boldsymbol{X}_i^t = [x_{i1}^t, x_{i2}^t, \cdots, x_{iD}^t]$ 和速度向量 $\boldsymbol{V}_i^t = [v_{i1}^t, v_{i2}^t, \cdots, v_{iD}^t]$. 其中，$i$ 和 D 分别表示粒子的编号和求解问题的维数，t 是当前的迭代次数. 粒子的速度决定了粒子飞行的方向和速率，而位置体现了粒子所代表的解在搜索空间的位置. 设 L_d 和 U_d 分别为第 $d(d=1,2,\cdots,D)$ 维搜索空间的下限和上限，即 $x_{id}^t \in (L_d, U_d)$，速度大小的区间设置为 $[v_{\min}, v_{\max}]$. 在执行 PSO 算法时，首先在设定的范围内随机初始化粒子的位置和速度，并设置相应的参数；然后，通过评价各粒子的适应度大小，确定 t 次迭代时粒子 i 的个体历史最优位置，记为 $\boldsymbol{P}_i^t = [p_{i1}^t, p_{i2}^t, \cdots, p_{iD}^t]$，以及迄今为止整个群体所发现的最优位置，记为 $\boldsymbol{P}_g^t = [p_{g1}^t, p_{g2}^t, \cdots, p_{gD}^t]$. 在算法的迭代过程中，每个粒子的速度和位置按照式 (1) 和式 (2) 进行更新.

$$v_{id}^{t+1} = \omega v_{id}^t + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{id}^t) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (2)$$

其中，r_1 和 r_2 为均匀分布在 [0,1] 区间的随机数；c_1 和 c_2 称为加速系数；ω 称为惯性权重，表示继承先前速度的比例大小. 一般设定，ω 依式 (3) 随迭代次数的增加而线性递减 [3].

$$\omega = (\omega_{\text{start}} - \omega_{\text{end}}) \times \frac{t_{\max} - t}{t_{\max}} + \omega_{\text{end}} \quad (3)$$

上式中，$t_{\max}$ 表示最大迭代次数；ω_{start} 和 ω_{end} 是开始和结束迭代时的惯性权重大小. Shi[3] 经过多组实验，建议采用 $\omega_{\text{start}} = 0.9$，$\omega_{\text{end}} = 0.4$. 本文将这种惯性权重递减的 PSO 算法称为标准粒子群优化（SPSO）算法. 2002 年，Clerc 等 [15] 采用了压缩因子 λ 来研究 PSO 算法的收敛行为，其中速度更新采用式 (4).

$$v_{id}^{t+1} = \lambda (v_{id}^t + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{id}^t)) \quad (4)$$

式 (4) 中参数含义同式 (1)，λ 按式 (5) 计算.

$$\lambda = \frac{2}{\left| 2 - \varphi + \sqrt{\varphi^2 - 4\varphi} \right|}, \quad \varphi = c_1 + c_2 \quad (5)$$

当 $c_1 = c_2 = 2.05$，即 $\lambda = 0.729$ 时，算法具有较好的性能 [15].

3 DPPSO 算法框架（The framework of DPPSO algorithm）

3.1 DPPSO 基本思想

在 DPPSO 算法中，粒子被划分为 predator 和 prey 两个种群. 与 Silva 提出的算法相比，DPPSO 中 predator 粒子数量较多. 随着 predator 粒子数量的增加，prey 粒子被分散到更大范围内，增加了群体多样性，有利于算法的全局搜索. 但太多的 predator 粒子频繁排斥 prey 种群，会使 prey 种群较难有效组合协作，从而使得算法接近于随机搜索. 为了平衡群体的多样性和算法的全局搜索能力，在 DPPSO 算法中采用了两个种群间隙性模拟 predator-prey 行为的策略，即在一般的情况下，两个种群的粒子没有任何干扰，按照各自的方式进行搜索寻优；在间隔一定的迭代次数 k 后，prey 种群就会受到 predator 种群的排斥而逃离.

在 predator 种群排斥 prey 种群的过程中，假定 predator 种群采用“擒贼先擒王”的策略是合理的. 在 DPPSO 算法中，该策略模拟表现为 predator 种群的粒子向其个体最优位置、群体最优位置以及 prey 种群的群体最优位置 3 个方向靠近；同时，prey 种群的粒子为避免被捕食，可以放弃其个体的最优位置和群体最优位置的记忆，尽量逃离距离自己最近的 predator 粒子.

DPPSO 中粒子速度的更新采用带压缩因子的方式. 综上分析，在 predator 种群没有干扰 prey 种群的时候，每个种群的粒子按照式 (4) 迭代搜索；在

发生排斥的过程中，predator 种群的粒子速度按照式 (6) 更新.

$$v_{id}^{t+1}=\lambda(v_{id}^{t}+c_{11}r_{11}(p_{id}^{t}-x_{id}^{t})+c_{12}r_{12}(\tilde{p}_{gd}^{t}-x_{id}^{t})+c_{13}r_{13}(\hat{p}_{gd}^{t}-x_{id}^{t})) \tag{6}$$

其中 $\tilde{p}_{gd}^{t}$ 和 $\hat{p}_{gd}^{t}$ 分别代表在 t 次迭代时 predator 和 prey 种群的群体最优位置的第 d 维. prey 种群受到排斥后依式 (7) 逃离. prey 种群逃离后，将会增加粒子分散的多样性，从而有利于全局搜索.

$$v_{id}^{t+1}=\lambda(p_{id}^{t}-c_{21}r_{21}(\zeta_{id}^{t}-x_{id}^{t})) \tag{7}$$

在 (7) 式中 $\zeta_i^t=[\zeta_{i1}^t,\zeta_{i2}^t,\cdots,\zeta_{iD}^t]$ 表示 t 次迭代时 prey 种群中距离粒子 i 最近的 predator 粒子位置. 式 (6) 和式 (7) 中的 c_{11}、c_{12}、c_{13} 和 c_{21} 分别表示加速系数，r_{11}、r_{12}、r_{13} 和 r_{21} 是均匀分布在 $[0,1]$ 之间的随机数.

当 prey 种群中的粒子处于停滞状态时，与其距离较近的 predator 粒子对其排斥效果并不明显，这将影响算法的性能. 本文采用了速度变异方法来增加排斥效果，即在算法搜索过程中，如果连续 30 代两个种群中最佳的群体最优位置没有发生变化，随机选择任意一个种群中的一个粒子 i 在任一维 d 上按式 (8) 发生速度变异.

$$v_{id}^{t}=\begin{cases}\gamma\times v_{\max}\times r_3, & r_4<0.5\\ -\gamma\times v_{\max}\times r_3, & r_4\geqslant 0.5\end{cases} \tag{8}$$

在式 (8) 中，$v_{\max}$ 表示粒子飞行的最大允许速度，r_3 和 r_4 为均匀分布在 $[0,1]$ 之间的随机数. 在算法迭代的前期，发生变异的范围大利于摆脱局部最优；而在算法迭代的后期，则发生的变异较少有利于局部搜索. 因此，式 (8) 中 γ 的值设定为随迭代次数的增加从 1 线性递减到 0.1. 由于只选择任意一个种群内的一个粒子中的一维速度发生变异，所以几乎不会破坏群体的结构，但是从统计意义上来说，两个种群的每一个粒子的每一维的速度被选择发生变异的概率是相同的.

3.2 DPPSO 算法流程

DPPSO 算法的流程可简述为：

步骤 1　初始化两个种群中粒子的位置和速度，设定相应的参数，并且 $t=0$.

步骤 2　判断两个种群是否满足发生排斥的条件，即如果 $H_{\mathrm{mod}}(t,k)\neq 0\ \&\ t\geqslant k$ 成立，执行步骤 3；否则直接执行步骤 4，其中 H_{mod} 表示取余运算.

步骤 3　两个种群的粒子按式 (4) 进行速度更新后，转到步骤 5.

步骤 4　predator 种群按式 (6) 更新速度，prey 种群依式 (7) 更新速度，然后转步骤 5.

步骤 5　分别更新两个种群中粒子的位置，以及它们的个体最优位置和群体最优位置.

步骤 6　判断是否发生变异，如果满足变异条件，按式 (8) 进行变异操作，否则转到步骤 7.

步骤 7　$t=t+1$，判断是否满足程序终止条件：若满足则算法终止，输出优化解；否则转步骤 2.

4 实验分析 (Experimental analysis)

4.1 基准函数

本文选取了智能优化中 6 个常用的基准函数进行分析，详细描述见表 1. 其中 $f_1\sim f_2$ 是单峰函数，$f_3\sim f_6$ 属于多峰函数. 表 1 给出每个函数的表达式、搜索范围、初始化范围和阈值.

4.2 实验设置

本文将 DPPSO 的测试性能同 PSOPC (particle swarm optimizer with passive congregation)[12]、HPSO-TVAC (hierarchical particle swarm optimizer with time-varying acceleration coefficients)[16] 及 SPSO 进行比较，验证 DPPSO 算法的有效性. SPSO、PSOPC 和 HPSO-TVAC 的参数根据文 [3]、[12] 和 [17] 设定，各算法的参数设置详见表 2.

为避免对称初始化可能造成算法性能较好的假象，文中采用了非对称初始化的方法，即让初始解不包括最优解. 测试函数的维数 D 设置为 30，即 $D=30$. 参与比较的各算法中粒子的数量设置为 60，在 DPPSO 中 predator 种群和 prey 种群粒子的数量都设置为 30. 实际情况经常需要对变量的取值范围进行限制，本文采用文 [18] 的方法处理，即粒子 i 在更新位置时，如果 $x_{id}^t>U_d$ 或者 $x_{id}^t<L_d$，那么 $v_{id}^t=0$，$x_{id}^t=\min(\max(x_{id}^t,L_d),U_d)$.

在 DPPSO 中，如果参数 k 的设置过大，则会减少两个种群的交互，不利于全局搜索；反之，如果 k 值设置过小，predator 种群频繁排斥 prey 种群，则不利于每个种群的粒子相互协作. 为了选择合适的 k 值，本文采用不同的 k 值对 f_1、f_2、f_4 和 f_5 的 30 维函数进行仿真实验. 实验的最大迭代次数设为 3000 次，每个实验独立运行 30 次得到的结果如表 3. 从表 3 可以看出，对于单峰函数 f_1 和 f_2 来说，当 $k=350$ 时，算法的性能较好. 当 $k=200$ 时多峰函数 f_4 和 f_5 优化的性能较好. 在本文后续的仿真实验中，在优化单峰函数和多峰函数时，k 分别设置为 350 和 200. 一般来说，较少的 k 值在求解多峰函数时不容易陷入局部最优；较大的 k 值可以较好地维

表 1 基准函数及参数设置 [11,16]

Tab.1 Benchmark functions and the settings of their parameters

函数名称	函数数学表达式	搜索范围	初始化范围	阈值
Schwefel's problem 1.2	$f_1(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	$(-100,100)^D$	$(50,100)^D$	10
Rosenbrock	$f_2(x)=\sum_{i=1}^{D}\left(100(x_{i+1}-x_i)^2+(x_i-1)^2\right)$	$(-30,30)^D$	$(10,30)^D$	100
Ackley	$f_3(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	$(-32,32)^D$	$(10,20)^D$	0.1
Rastrigin	$f_4(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	$(-10,10)^D$	$(5,10)^D$	100
Griewank	$f_5(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$(-600,600)^D$	$(300,600)^D$	0.1
Weierstrass	$f_6(x)=\sum_{i=1}^{D}\sum_{k=0}^{20}\left(a^k\cos(2\pi b^k(x_i+0.5))\right)-D\sum_{k=0}^{20}\left(a^k\cos(2\pi b^k\times 0.5)\right),\ a=0.5,\ b=3$	$(-0.5,0.5)^D$	$(0.2,0.5)^D$	10

表 2 不同算法的参数设置

Tab.2 Parameter settings of different algorithms

算法	参数设置
SPSO	$\omega=(0.9-0.4)\times\frac{t_{\max}-t}{t_{\max}}+0.4,\quad c_1=c_2=2$
HPSO-TVAC	$\omega=0,\quad c_1=(2.5-0.5)\times\frac{t_{\max}-t}{t_{\max}}+0.5,\quad c_2=2.5-(2.5-0.5)\times\frac{t_{\max}-t}{t_{\max}}$
PSOPC	$\omega=(0.9-0.7)\times\frac{t_{\max}-t}{t_{\max}}+0.7,\quad c_1=c_2=0.5,\quad c_3=0.4+(0.6-0.4)\times\frac{t_{\max}-t}{t_{\max}}$
DPPSO	$\lambda=0.729,\quad c_1=c_2=2.05,\quad c_{11}=c_{12}=c_{13}=1.367,\quad c_{21}=4.1$

表 3 不同 k 值下函数 f_1、f_2、f_4 和 f_5 的测试结果

Tab.3 Testing results of functions f_1, f_2, f_4 and f_5 with different values of k

函数名称	不同 k 值下求解结果的均值和标准差（不加括号的数值为均值，括号内数值为标准差）						
	100	150	200	250	300	350	400
f_1	4.74e−03	3.18e−03	2.44e−03	1.93e−03	8.17e−04	4.08e−03	5.24e−02
	(2.62e−03)	(1.92e−03)	(3.23e−03)	(1.64e−03)	(6.26e−04)	(3.11e−03)	(4.63e−02)
f_2	2.74e+01	2.38e+01	1.92e+01	1.32e+01	1.28e+01	7.82e+00	1.39e+01
	(3.03e+01)	(1.75e+01)	(1.07e+01)	(5.72e+00)	(7.49e+00)	(7.21e+00)	(8.29e+00)
f_4	7.43e+00	6.30e+00	4.81e+00	1.38e+01	1.43e+01	1.58e+01	1.69e+01
	(3.95e+00)	(5.14e−01)	(4.07e−01)	(3.29e+00)	(4.73e+00)	(3.55e+00)	(5.41e+00)
f_5	3.92e−03	3.37e−03	1.57e−03	8.18e−03	1.19e−02	8.73e−02	1.68e−02
	(9.22e−03)	(8.04e−03)	(4.23e−03)	(1.31e−02)	(1.75e−02)	(2.19e−02)	(1.94e−02)

护群体中粒子的结构，有利于求解单峰函数.

4.3 实验结果

采用 2 种性能指标来测试算法性能：固定迭代次数内算法平均最优结果和到达确定阈值的平均迭代次数.

对于 6 个基准函数，采用 SPSO、HPSO-TVAC、PSOPC 和 DPPSO 四种粒子群算法分别独立运行 30 次，最大迭代次数为 6000 次，得到的实验结果整理后如表 4 所示. 表 4 列出了各算法对每个测试函数优化的平均值和标准差，其中实验结果的最好值加粗表示. 图 1 ～ 6 描述了每种算法求解 6 个基准函数的收敛曲线，横坐标和纵坐标表示迭代次数和平均最优值（取以 10 为底的对数）. 表 5 列出了各算法在每个基准函数的搜索精度达到阈值

表 4　各种算法对不同基准函数的优化结果

Tab.4　Optimization results of benchmark functions on different algorithm

	函数维数	最大迭代次数	评价指标	比较的 PSO 算法			
				SPSO	PSOPC	HPSO-TVAC	DPPSO
f_1	30	6000	平均值	5.74e-01	6.80e-01	4.90e-06	**7.08e-09**
			标准差	4.65e-01	6.24e-01	5.44e-06	**2.21e-08**
f_2	30	6000	平均值	4.08e+01	2.05e+01	1.17e+00	**4.78e-01**
			标准差	3.32e+01	4.97e+00	1.03e+00	**9.14e-01**
f_3	30	6000	平均值	7.40e-15	3.23e-14	3.00e-14	**5.51e-15**
			标准差	**2.90e-15**	3.43e-14	7.50e-15	4.50e-15
f_4	30	6000	平均值	2.17e+01	2.97e+01	1.99e+00	**9.95e-02**
			标准差	3.80e+00	8.09e+00	8.12e-01	**3.15e-01**
f_5	30	6000	平均值	1.96e-02	1.85e-02	3.02e-03	**2.30e-12**
			标准差	2.52e-02	1.83e-02	4.83e-03	**4.61e-12**
f_6	30	6000	平均值	2.25e-02	1.76e+00	6.96e-13	**4.26e-15**
			标准差	1.64e-02	1.49e+00	4.26e-13	**1.12e-14**

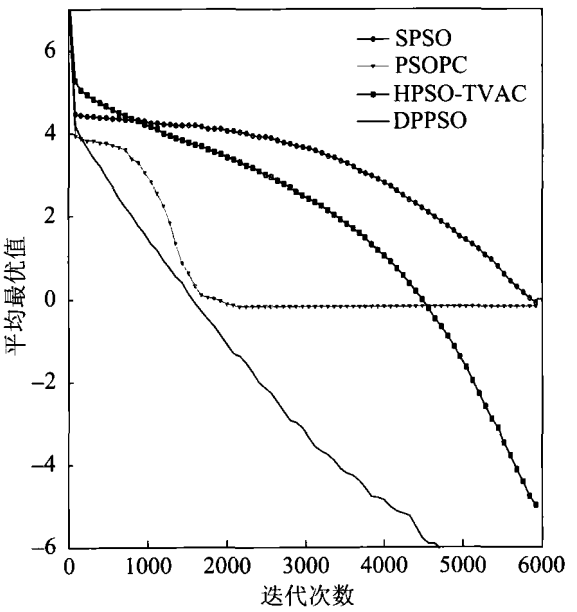

图 1　各算法求解 f_1 的收敛曲线

Fig.1　Convergence curves of f_1 solved by the algorithms

时的平均迭代次数，其中最小迭代次数加粗表示.

从表 4、表 5 以及图 1 ～ 6 可以看出，与其它算法相比，DPPSO 算法表现出了较快的收敛速度和持续的搜索寻优能力，不易陷入局部最优. 在对单峰函数 f_1 优化中，DPPSO 算法求解的平均值、标准差均显著优于 SPSO 和 PSOPC. 与 HPSO-TVAC 比较，二者都得到了较满意的优化结果，但 DPPSO 收敛速度却远远快于 HPSO-TVAC. f_2 是一个经典复杂优化问题，其全局最优点位于一个平滑、狭长的抛物线山谷内，算法很难辨别搜索方向，因此找到全局最优点的机会很小. 在对 f_2 的优化中，DPPSO

表 5　各种算法的搜索精度达到阈值时的平均迭代次数

Tab.5　Average numbers of iterations to reach the threshold of searching accuracy of the algorithms

函数名称	SPSO	PSOPC	HPSO-TVAC	DPPSO
f_1	5324	1427	4112	**826**
f_2	3593	825	1503	**428**
f_3	3275	671	1762	**313**
f_4	2831	1833	1487	**197**
f_5	975	3108	957	**287**
f_6	2679	1137	2035	**364**

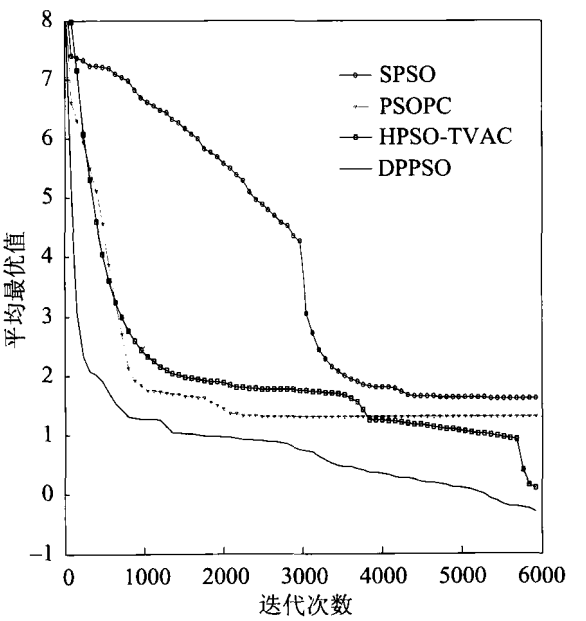

图 2　各算法求解 f_2 的收敛曲线

Fig.2　Convergence curves of f_2 solved by the algorithms

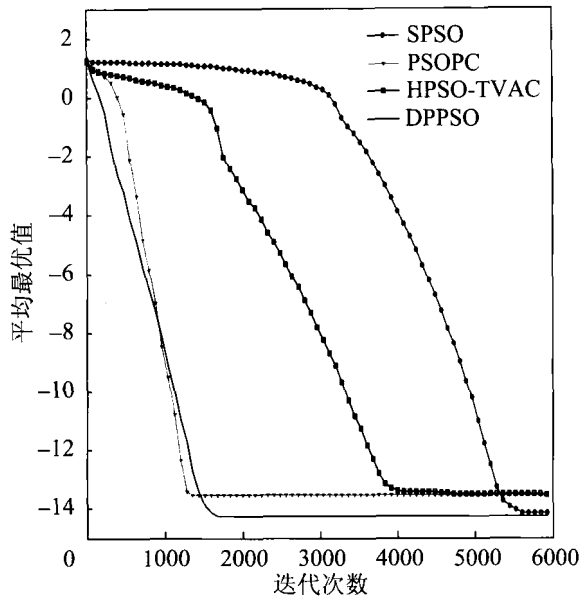

图 3 各算法求解 f_3 的收敛曲线

Fig.3 Convergence curves of f_3 solved by the algorithms

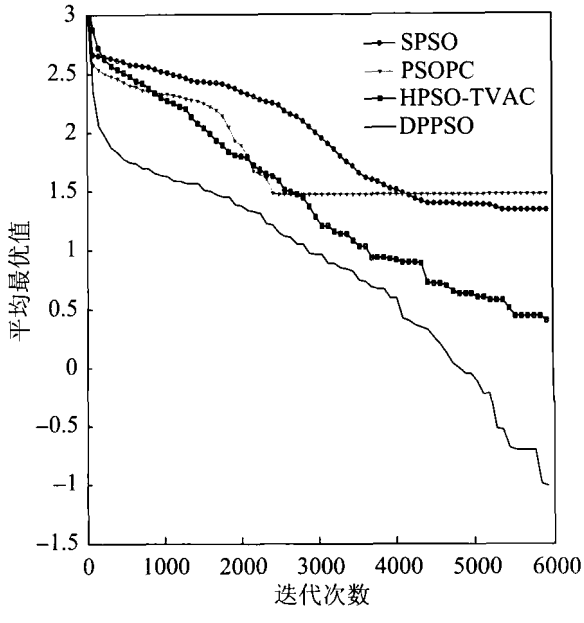

图 4 各算法求解 f_4 的收敛曲线

Fig.4 Convergence curves of f_4 solved by the algorithms

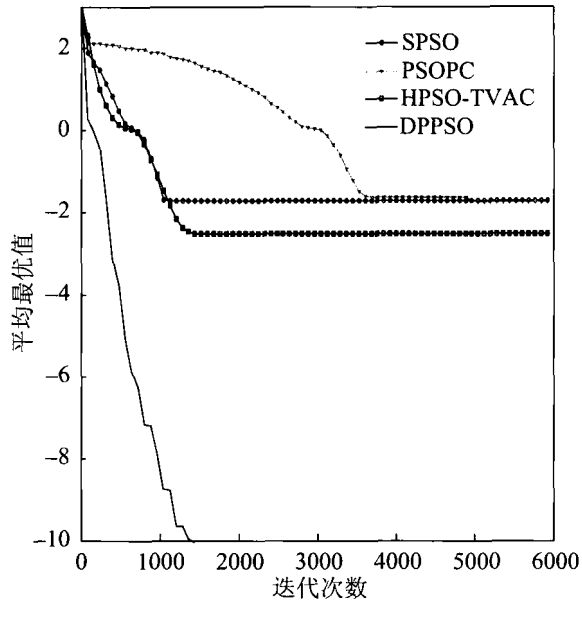

图 5 各算法求解 f_5 的收敛曲线

Fig.5 Convergence curves of f_5 solved by the algorithms

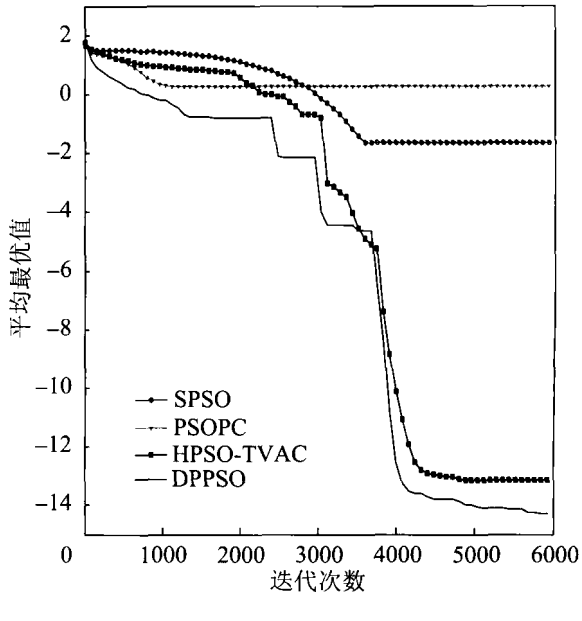

图 6 各算法求解 f_6 的收敛曲线

Fig.6 Convergence curves of f_6 solved by the algorithms

的搜索精度高于 HPSO-TVAC、SPSO 和 PSOPC，表现出较强的持续优化能力和快速的收敛速度. 对多峰函数 f_3 的优化结果可以看出，DPPSO 的优化结果略好于其他 3 种算法，但差别不大. 对于有具有大量局部最优点的复杂多峰函数 f_4，尽管 HPSO-TVAC 取得了较好的优化效果，但 DPPSO 明显比 HPSO-TVAC 具有更快的收敛速度和更高的搜索精度. 多峰函数 f_5 变量之间存在相关性，一般的算法非常容易陷入局部最优点. 从图 5 上可以看出，SPSO、PSOPC 和 HPSO-TVAC 在迭代的前期就陷入局部最优，DPPSO 在迭代的在过程中没有陷入局部最优，求解结果非常优异. f_6 中，DPPSO 与 HPSO-TVAC 的搜索精度相差不大，但收敛更加快速. 从以上分析可以看出，相较其他 3 种算法，DPPSO 的收敛速度非常快，且对于一般算法难于优化的复杂问题，表现出了优秀的性能.

在 DPPSO 中，两个种群之间定期的排斥作用实际上是在群体协作的秩序上增加了扰动性，这种机制有利于群体的多样性维护和全局搜索. 在 Silva 提出的算法中，predator 粒子只是单纯用于排斥 prey 粒子，而在 DPPSO 中，predator 种群不仅排斥了 prey 种群促使其逃离局部最优，而且 predator

种群自身的粒子也能够相互协作搜索. 从一定程度上 DPPSO 提高了算法的全局搜索能力和鲁棒性.

5 结论与展望（Conclusion and prospect）

将大自然的其它生物机制嵌入到 PSO 算法中是提高其性能的一条可行途径. 本文根据生物的 predator-prey 行为的规律，提出了一种新的改进算法 DPPSO. 在 DPPSO 中，两个种群之间定期模拟生物的 predator-prey 行为，即间隔一定的迭代次数 k 后，predator 种群对 prey 种群进行排斥. 为了提高 prey 种群在停滞状态时摆脱局部最优的能力，本文采用了一种速度变异的方法. 基准函数的实验结果表明 DPPSO 具有快速的收敛速度和优秀的全局持续搜索能力. 未来的研究将对 DPPSO 算法的理论进行较深一步的探讨并将其应用到实际的经济管理和工程优化中.

参考文献（References）

[1] Eberhart R C, Kennedy J. Particle swarm optimization[C]// IEEE International Conference on Neural Networks. Piscataway, NJ, USA: IEEE, 1995: 1942-1948.

[2] 谢晓锋，张文俊，杨之廉. 微粒群算法综述 [J]. 控制与决策，2003，18(2)：129-133.
Xie X F, Zhang W J, Yang Z L. Overview of particle swarm optimization[J]. Control and Decision, 2003, 18(2): 129-133.

[3] Shi Y, Eberhart R C. A modified particle swarm optimizer[C]// IEEE Congress on Evolutionary Computation. Piscataway, NJ, USA: IEEE, 1998: 69-73.

[4] Natarajan U, Saravanan R, Periasamy V M. Application of particle swarm optimization in artificial neural network for the prediction of tool life[J]. International Journal of Advanced Manufacturing Technology, 2007, 31(9/10): 871-876.

[5] Liu B, Wang L, Jin Y H. An effective PSO-based memetic algorithm for flow shop scheduling[J]. IEEE Transactions on Systems, Man and Cybernetics: B, 2007, 37(1): 18-27.

[6] Chen W, Zhang W G. The admissible portfolio selection problem with transaction costs and an improved algorithm[J]. Physica A: Statistical Mechanics and Its Applications, 2010, 389(10): 2070-2076.

[7] Angeline P J. Evolutionary optimization versus particle swarm optimization and philosophy and performance difference[C]//IEEE Annual Conference on Evolutionary Programming. Piscataway, NJ, USA: IEEE, 1998: 601-610.

[8] 倪庆剑，张志政，王蓁蓁，等. 一种基于可变多簇结构的动态概率粒子群优化算法 [J]. 软件学报，2009，20(2)：339-349.
Ni Q J, Zhang Z Z, Wang Z Z, et al. Dynamic probabilistic particle swarm optimization based on varying multi-cluster structure[J]. Journal of Software, 2009, 20(2): 339-349.

[9] Angeline P J. Using selection to improve particle swarm optimization[C]//IEEE World Congress on Computational Intelligence. Piscataway, NJ, USA: IEEE, 1998: 84-89.

[10] 栾丽君，谭立静，牛奔. 一种基于粒子群优化算法和差分进化算法的新型混合全局优化算法 [J]. 信息与控制，2007，36(6)：708-714.
Luan L J, Tan L J, Niu B. A novel hybrid global optimization algorithm based on particle swarm optimization and differential evolution[J]. Information and Control, 2007, 36(6): 708-714.

[11] Liang J J, Qin K, Suganthan P N. Comprehensive learning particle swarm optimization for global optimization of multimodal functions[J]. IEEE Transactions on Evolutionary Computation, 2006, 6(3): 281-295.

[12] He S, Wu Q H, Wen J Y, et al. A particle swarm optimizer with passive congregation[J]. BioSystems, 2004, 78: 135-147.

[13] Silva A, Neves A, Costa E. An empirical comparison of particle swarm and predator prey optimisation[C]//13th Irish Conference on Artificial Intelligence and Cognitive Science. New York, NY, USA: Springer, 2002: 103-110.

[14] Higashitani M, Ishigame A, Yasuda K. Particle swarm optimization considering the concept of predator-prey behavior[C]//IEEE Congress on Evolutionary Computation. Piscataway, NJ, USA: IEEE, 2006: 1541-1544.

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

[16] Yao X, Liu Y, Lin G M. Evolutionary programming made faster[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82-102.

[17] Ratnaweera A, Halganmuge S. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 240-255.

[18] Carlisle A, Dozier G. An off-the-shelf PSO[C]//Proceedings of the workshop on Particle Swarm Optimization. Indianapolis, IN, USA: Purdue School of Engineering and Technology, 2001: 1-6.

作者简介：

秦全德（1979–），男，博士. 研究领域为智能计算及其应用.

牛　奔（1980–），男，博士，副教授. 研究领域为优化理论与方法.