• 计算机工程与应用 •

基于深度确定性策略梯度的粒子群算法

鲁华祥[1,2,3,4]，尹世远[1]，龚国良[1*]，刘 毅[1]，陈 刚[1]

(1. 中国科学院半导体研究所 北京 海淀区 100083；2. 中国科学院大学微电子学院 北京 海淀区 100089；
3. 中国科学院脑科学与智能技术卓越创新中心 上海 松江区 200031；
4. 半导体神经网络智能感知与计算技术北京市重点实验室 北京 海淀区 100083)

【摘要】在传统的粒子群优化算法 (PSO) 中，所有粒子都遵循最初设定的一些参数进行自我探索，这种方案容易导致过早成熟，且易被困于局部最优点。针对以上问题，该文提出了一种基于深度确定性策略梯度的粒子群优化算法 (DDPGPSO)，通过构造神经网络分别实现了动作函数和动作价值函数，且利用神经网络可以动态地生成算法运行所需要的参数，降低了人工配置算法的难度。实验表明 DDPGPSO 相比 9 种同类算法在收敛速度和寻优精度上均有较大的提升。

关 键 词 自适应惯性权值； 收敛因子； 深度确定性策略梯度算法； 强化学习； 群体智能； 粒子群优化算法
中图分类号 TP18 文献标志码 A doi:10.12178/1001-0548.2020420

A Particle Swarm Optimization Algorithm Based on Deep Deterministic Policy Gradient

LU Hua-xiang[1,2,3,4], YIN Shi-yuan[1], GONG Guo-liang[1*], LIU Yi[1], and CHENG Gang[1]

(1. Institute of Semiconductors, CAS Haidian Beijing 100083;
2. School of microelectronics, University of Chinese Academy of Sciences Haidian Beijing 100089;
3. Center for Excellence in Brain Science and Intelligence Technology, CAS Songjiang Shanghai 200031;
4. Beijing Key Laboratory of Semiconductor Neural Network Intelligent Sensing and Computing Technology Haidian Beijing 100083)

Abstract In the traditional particle swarm optimization (PSO) algorithm, all particles follow some initial parameters to explore themselves. This scheme is easy to lead to premature maturity, and easy to be trapped in the local optimum. To solve the above problems, a particle swarm optimization algorithm based on deep deterministic policy gradient (DDPGPSO) is proposed. The action function and action value function are realized by constructing neural network. The parameters required by the algorithm can be generated dynamically by using the neural network, which reduces the difficulty of manual configuration of the algorithm. The experimental results show that DDPGPSO has a great improvement in convergence speed and optimization accuracy compared with nine similar algorithms.

Key words adaptive inertia weight; convergence factor; deep deterministic policy gradient; reinforcement learning; swarm intelligence; swarm PSO

群智能优化算法属于随机优化算法的一种，包括经典粒子群算法 (particle swarm optimization, PSO)[1]、鲸鱼群算法 (whale optimization algorithm, WOA)[2]、灰狼算法 (gray wolf optimization algorithm, GWO)[3]、蝙蝠算法 (BAT)[4] 等，这些算法都是模拟大自然中某种生物的生活习性而创造的算法。

PSO[1] 是一种群智能优化算法，其原理是模仿鸟群捕食和聚集的行为，并采用计算机模拟的方式创造一种随机优化的算法。该算法自首次提出开始就受到广泛关注，其优势主要在于它简单的基本原理、较少的参数和快速的收敛速度。该算法已经广泛应用于工业生产[5]、矿山[6]、煤炭[7]、水电调度[8]等领域。

但 PSO 算法也有一些不足。传统 PSO 算法可

收稿日期：2020－11－21；修回日期：2020－12－23
基金项目：国家自然科学基金 (U19A2080, U1936106)；北京市科技计划 (Z181100001518006)；高技术项目 (31513070501, 1916312ZD00902201, XDA27040303)
作者简介：鲁华祥 (1965－)，男，博士，研究员，博士生导师，主要从事神经计算芯片、类脑神经计算技术和应用系统方面的研究.
通信作者：龚国良，E-mail：gongmianjie@semi.ac.cn

以很好地工作在低维问题上，但是在高维问题上经常表现很差。另外，PSO 算法在求解复杂问题时，经常会被困在局部最优点中，过早地收敛[9]。研究者对 PSO 算法存在的问题进行了很多改进。文献 [10] 首次提出了惯性权重，文献 [11] 首次提出了收缩系数。这些方法都通过控制算法的一些运行参数较好地平衡了算法的探索能力和挖掘已有信息的能力。自组织层次化粒子群算法[12] 通过采用一个随时间变化的参数，提升了算法的收敛能力。文献 [13] 提出了一种自适应粒子群优化算法，通过进化状态估计将种群状态划分为探索、开发、收敛和跳出 4 种状态，然后根据这 4 种状态自适应地调整惯性权重和加速度系数。文献 [14] 通过将粒子位置的各维度进行逐通道学习避免了 PSO 算法在优化过程中经常出现震荡的问题。文献 [15] 提出了无速度约束多群粒子群优化，通过加入约束处理技术，在优化可行域之前将总体搜索引向搜索空间的可行区域，使得 PSO 算法可以解决约束优化问题。文献 [16] 提出了基于末端弹性机制和交叉转向机制的粒子群算法，通过引入非线性惯性权重和弹性机制，更好地平衡了算法的利用与开发。文献 [17] 提出了一种具有变异算子的粒子群优化算法，通过将种群分为不同的亚群，引入非均匀突变算子，增加高斯变异操作，提高了算法的探索能力。

这些改进方法具有一个共同的特点，即通过人工对算法的超参数进行设置，只有经过多次反复尝试才能得到一个相对较优的参数组。近年来，随着机器学习和强化学习的兴起，神经网络逐渐展现出强大的参数搜索能力。优化问题的目标是优化一个函数的值，而每步之间该值的差非常适合作为强化学习中的奖励基准来使用。这使得在群智能算法领域应用强化学习成为可能。

但是，在传统的连续控制领域，动作空间十分巨大并且动作值均为连续的实数值。这使得强化学习智能体直接从像素级别信息进行学习变得十分困难。近些年 Google DeepMind 团队提出了一种新型的深度强化学习算法：深度确定性策略梯度算法 (deep deterministic policy gradient, DDPG)[18]，其在很多连续控制问题上表现突出，比如 Atari 电子游戏[18-19]，但尚未有人将该算法与群体智能算法相结合来进行研究。

本文提出了一种基于深度确定性策略梯度的粒子群优化算法，在原有的粒子群算法中引入神经网络来动态调节算法的收敛速度。通过提前训练，获得一个动作网络，然后应用动作网络求解优化问题。对改进的算法进行仿真验证，在多种测试函数上与其他优化函数进行对比，得出 DDPGPSO 有了更高的寻优速度和精度。

1 提出算法的基本原理

本文首先介绍基本的粒子群算法，然后介绍 DDPG 的基本思想，最后介绍融合了 DDPG 的粒子群优化算法 DDPGPSO。

1.1 粒子群优化算法

粒子群优化算法是一种群体智能的算法。这种算法源于对鸟群捕食和聚集行为的研究，通过设计一种无质量的粒子来模拟鸟群中的鸟，粒子仅具有两个属性：速度和位置，速度代表移动的快慢，位置代表移动的方向。每个粒子在搜索空间中单独的搜寻最优解，将其记为当前个体极值，并将个体极值与整个粒子群里的其他粒子共享，找到最优的个体极值作为整个粒子群的当前全局最优解，粒子群中的所有粒子根据自己找到的当前个体极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。其具体流程如下：

假设种群中含有 m 个粒子，解空间的维度是 n，每个粒子代表解空间中的一个点。粒子 i 包含一个 n 维的位置向量 $(x_1,x_2,\cdots,x_n)$ 和速度向量 $(v_1,v_2,\cdots,v_n)$。每个粒子通过学习它的最优历史位置和全局最优位置来更新速度和位置，更新公式如下：

$$v_{ij}=v_{ij}+c_1r_{1j}(x_{pbestij}-x_{ij})+c_2r_{2j}(x_{gbest}-x_{ij}) \quad (1)$$

$$x_{ij}=v_{ij}+x_{ij} \quad (2)$$

式中，i=1,2,$\cdots$,m，j=1,2,$\cdots$,n，m 和 n 分别是种群数量和解空间的维度；v_{ij} 和 x_{ij} 分别代表粒子 i 的第 j 维的速度和位置；$x_{pbestij}$ 代表粒子 i 的历史最佳位置；x_{gbest} 代表种群内的历史最佳位置；c_1 和 c_2 是控制加速的参数；r_{1j} 和 r_{2j} 是两个均匀分布于 [0,1] 区间的随机数。最大速度参数 v_{maxj} 被用来将每个粒子的速度控制到一个合适的区间中。如果 $|v_{ij}|>v_{maxj}$，则令 $v_{ij}=v_{ij}v_{maxj}/|v_{ij}|$。为了平衡种群的探索和挖掘能力，文献 [9] 提出了速度惯性权重来控制速度，式 (1) 被修改成：

$$v_{ij}=\omega v_{ij}+c_1r_{1j}(x_{pbestij}-x_{ij})+c_2r_{2j}(x_{gbest}-x_{ij}) \quad (3)$$

式中，ω一般设置为从 0.4～0.9 随时间线性递减。在搜索过程中，惯性参数ω的这种变化确保算法在早期拥有强大的全局探索能力，在后期拥有强大的局部搜索能力。

1.2　深度确定性策略梯度原理

在标准的强化学习环境中，每一个智能体与环境进行交互，最终目标是最大化环境收益。这种互动过程被格式化地描述为马尔科夫决策过程(MDP)，可以通过四元组 (S, A, R, P) 来描述这个过程。S 是状态空间，A 是动作空间，R：$S\times A\to R$ 是奖励函数，P：$S\times A\times S\to[0,1]$ 是转移概率。在这个环境下，一个智能体会去学习一个策略 π：$S\to A$ 来最大化在环境中的收益：

$$R_0=\sum_{i=0}^{T} r(s_i,a_i) \tag{4}$$

式中，T 是交互结束时前进的步数；$r(s_i,a_i)$是在环境 s_i 中执行 a_i 所获得的收益。通常情况下环境中的长期收益会通过参数γ来缩减：

$$R_0^{\gamma}=\sum_{i=0}^{T}\gamma^i r(s_i,a_i)\qquad \gamma\in(0,1) \tag{5}$$

一般用动作价值函数代表在 s 环境中执行 a 动作的长期收益：

$$Q(s_t,a_t)=\mathrm{E}[R_t^{\gamma}|s=s_t,a=a_t]=\mathrm{E}\left[\sum_{i=t}^{T}\gamma^{i-t}r(s_i,a_i)\right] \tag{6}$$

通常使用贝尔曼方程来寻找这种最优的动作价值函数：

$$Q*(s_t,a_t)=\mathrm{E}[r(s_t,a_t)+\gamma\max_{a'_{t+1}}Q*(s_{t+1},a'_{t+1})] \tag{7}$$

然而，这种方式只适合于那些动作和状态空间都是离散情况的场合。为了应用强化学习到动作空间和状态空间为连续情况的问题，DDPG 设计了两个深度神经网络，分别是动作价值网络$Q(s_t,a_t|\theta^Q)$和动作网络$\mu(s_t|\theta^{\mu})$，其中θ^Q和θ^{μ}为网络参数。动作网络$\mu(s_t|\theta^{\mu})$就是一个对应状态空间和动作空间的映射，可以直接根据状态产生需要的动作。动作价值网络$Q(s_t,a_t|\theta^Q)$用于接近动作价值函数，并且可以为动作网络的训练提供梯度。

对这种动作价值网络的训练是要最小化损失函数：

$$L(\theta^Q)=(r(s_t,a_t)+\gamma Q'(s_{t+1},a_{t+1}|\theta^{Q'})-Q(s_t,a_t|\theta^Q))^2 \tag{8}$$

式中，Q'是目标价值网络，从Q网络同步权重。而动作网络参数的更新需要策略梯度算法，其梯度更新方向为：

$$\nabla_{\theta^{\mu}}Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t,v)}=\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t,v)}\nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})|_{s=s_t} \tag{9}$$

1.3　基于深度确定性策略梯度的粒子群优化算法原理

1.3.1　迭代参数改进原理

标准的粒子群优化算法在运行过程中，参数都是确定的，如式 (3) 所示。也有一些文献采用提前设置参数变化形式的方式来自动修改参数[20]，但其实质仍是在开始运行时就已经确定，无法根据后续情况进行变化，不能达到最佳的算法效果。为了增加算法的自动调整能力，提高算法的寻优和收敛能力，本文引入了动态自适应参数调整算法，根据当前运行的情况来生成下次迭代所需的参数。改进后的速度更新公式为：

$$v_{ij}=\omega_t v_{ij}+c_{1t}r_{1t}(x_{\mathrm{pbest}ij}-x_{ij})+c_{2t}r_{2t}(x_{g\mathrm{best}}-x_{ij}) \tag{10}$$

$$\omega_t=G_{\omega}(s_{t-1}) \tag{11}$$

$$r_{1t}=G_{r1}(s_{t-1}) \tag{12}$$

$$r_{2t}=G_{r2}(s_{t-1}) \tag{13}$$

式中，t 为当前的迭代次数；G 为生成函数，本文使用神经网络作为生成函数，这样就可以在迭代过程中动态生成运行参数。

1.3.2　参数生成改进原理

随着 PSO 各种改进方式越来越复杂，越来越多的参数需要进行设定，但人工设定各种参数不仅麻烦，而且也不一定能获得最佳性能。加入了参数生成函数后，所需调整的参数量更加巨大，迫切地需要一种自动化的方式来自动配置算法。但是，由于这种优化效果本质上也是一个优化的问题，是不可求导的，也没有合适的损失函数可供使用，传统的深度学习算法无法应用到这种优化问题之中。因此，本文利用 DDPG 算法，创建一个智能体，通过跟算法的交互自动习得参数。

该算法所使用的状态空间为六维，前 5 个维度代表群智能过去的平均适应性函数的变化量，最后一个维度代表当前的迭代次数，其计算公式为：

$$\begin{gathered}s_{it}=\mathrm{fitmean}_{t+4-i}-\mathrm{fitmean}_{t+3-i}\quad 0\leqslant i\leqslant 4\\ s_{5t}=t/T_{\max}\end{gathered} \tag{14}$$

式中，$\mathrm{fitmean}_t$ 代表粒子群在第 t 轮循环时所有粒子解的平均值；$T_{\max}$ 代表迭代的最大次数；S_{it} 表示第 t 轮循环、第 i 维的状态向量。

算法的动作空间为三维，动作值介于−1～1 之间，通过线性变换将动作映射到合理的区间。三维动作分别对应式 (3) 中的 w，c_1，c_2，其中 c_1，c_2 变换到 0.5～2.5 区间，w 变换到 0.1～0.9 区间。线性变换公式为：

$$w = a_0/2.5 + 0.5$$
$$c_1 = a_1 + 1.5$$
$$c_2 = a_2 + 1.5 \tag{15}$$

式中，a_0，a_1，a_2 分别为动作的 3 个维度。算法的奖励值设置如下：当 PSO 在当前环境最佳值发生变化，则奖励为 1，否则奖励为−1：

$$R_i = 1 \text{ (全局历史最优值发生变化)} \tag{16}$$

$$R_i = -1 \text{ (全局历史最优值未发生变化)} \tag{17}$$

这样设置是为了让网络寻找参数，使原有算法更快地对问题进行优化，提高算法的寻优速度。

1.4 算法步骤

1.4.1 训练步骤

1) 初始化训练所需参数：训练最大次数 TrainMax，当前训练次数 episode =1；

2) 随机初始化动作网络$\mu(s|\theta^{\mu})$和动作价值网络$Q(s,a|\theta^{Q})$的参数θ^{Q}和θ^{μ}，具体结构如表 1～表 2 所示；

3) 初始化目标网络Q'和μ'，其权重值从 Q 和 μ复制过来；

4) 初始化回放缓存 R；

5) 初始化环境 (粒子种群)$X_i(i=1,2,\cdots,Q)$，计算每个个体 X_i 的适应度值 F，得到初始最优位置 X^*，当前迭代次数 t=0，最大迭代次数 $T_{\max}$；

6) For episode=1, TrainMax do

7) 初始化一个随机过程来对动作进行探索；

8) 接收环境的初始化观测 s_1；

9) for t=1, $T_{\max}$ do

10) 根据当前的策略和探索噪声选择动作：

$$a_t = \mu(s_t|\theta^{\mu}) + N_t$$

11) 在环境中执行动作 a_t 并观察奖励回报 r_t 和新的状态 s_{t+1}；

12) 将 (s_t, a_t, r_t, s_{t+1}) 保存到缓存 R 中；

13) 从 R 中随机采样一个最小批次的 N 次状态转移，设置$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$；

14) 通过最小化损失函数$L = \frac{1}{N}\sum_{i=0}^{N}(y_i - Q(s_i,a_i|\theta^{Q}))^2$更新动作价值网络；

15) 通过采样的动作策略梯度来更新动作网络：

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{i=0}^{N} \nabla_a Q(s,a|\theta^{Q})|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_i}$$

16) 更新目标网络函数的权值：

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

17) end for

18) end for

19) 保存训练好的动作网络权重，用于之后的优化问题。

表 1 动作网络结构

层名称	输出维度	输入
输入层	6(状态向量)	
L0层	24	输入层
L1层	16	L0层
输出层	3(动作向量)	L1层

表 2 动作价值网络结构

层名称	输出维度	输入
输入层1	6(状态向量)	
输入层2	3(动作向量)	
拼接层	9	输入层1, 2
L0层	24	拼接层
L1层	16	L0层
输出层	1(动作价值)	L1层

1.4.2 优化步骤

1) 初始化粒子种群 $x_i(i=1,2,\cdots,Q)$，计算每个个体 x_i 的适应度值，得到初始的全局最优位置 x^*，最优解 F^*，当前迭代次数 t=0，最大迭代次数 $T_{\max}$；

2) while $t < T_{\max}$

3) 计算当前状态，更新全局最优值；

4) 根据动作网络$\mu(s|\theta^{\mu})$和当前状态，获取当前动作，由式 (15) 转化为群探索参数；

5) for $i = 1 : Q$

6) 应用式 (3) 更新 X_i 的速度 v_i；

7) 应用式 (2) 更新 X_i 的位置 x_i；

8) end for

9) end while

2 实验仿真测试

本文将DDPGPSO 算法同飞蛾扑火算法(moth flame optimization, MFO)、布谷鸟算法 (CUCKOO)、蝙蝠算法 (BAT)、乌贼算法 (cuttlefish algorithm, CFA)、多元宇宙算法 (multi verse optimizer, MVO)、鲸鱼算法 (WOA)、细菌觅食算法 (bacterial foraging optimization, BFA)8 种常用的优化算法，以及一种粒子群算法的变种混沌粒子群算法 (a chaotic pso

algorithm based on sequential algorithm, CPSOS)[20] 进行实验对比，参数设置如表 3 所示。由于本文主要验证 DDPGPSO 的参数生成能力，为了保持一致，CPSOS 的一些复杂操作不予考虑。选用了 8 个常用的基准测试函数[21](F1～F8) 如表 4 所示，最优值均为 0。对于所有的算法和测试函数，算法的参数设置如下：最大迭代次数设为 100，种群数量设为 40，重复计算 10 次，采用平均值 (mean)、标准差 (std) 和迭代中的最优值 (best) 来评价算法的性能，F1～F8 的迭代图分别如图 1～图 8 所示。在这些测试中，所有的优化算法的初始值均随机生成，在解空间中符合均匀分布。由于画图中采用对数坐标作为纵坐标，为了防止在算法搜索到最优值 0 时产生无意义数据，所有绘图的 y 数据均增加了一个极小值。图 2 和图 4 中的 DDPGPSO 由于已经搜索到最小值 0，对数图中会产生无穷小，因此删去部分末尾数据。

表 3　算法参数设置

算法	参数设置
DDPGPSO	无需参数设置，惯性因子搜索范围0.1～0.9；加速因子搜索范围0.5～2.5
CPSOS	惯性因子初值0.9，末值0.4；全局加速因子初值0.5，末值2.5；自身加速因子初值2.5，末值0.5；
PSO	惯性因子w = 0.5；加速因子c_1 = 1, c_2 = 2
WOA	定义螺旋形状的常数b = 1
MFO	定义螺旋形状的常数b = 1
CUCKOO	布谷鸟蛋发现概率pa = 0.25, 步长控制量α = 1
BAT	响度A=0.5, 脉冲发射率r=0.5, 最小频率Qmin= 0, 最大频率Qmax= 2
BFA	探索步长1
MVO	虫洞存在率最大值WEPmax= 1和最小值WEPmin= 0.2, 开发准确率p= 0.6
CFA	伸展度常数: $r1$ = 2,$r2$ =−1, 可见度常数：$v1$ =−1.5,$v2$ = 1.5

表 4　基准测试函数

名称	测试函数	公式(目标均为最小化)	初始范围	维数
F1	Ackley	$-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+\exp(1)$	[−100, 100]	100
F2	Griewank	$\sum_{i=1}^{D}\frac{x_i^2}{4\,000}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[−600, 600]	100
F3	Quartic	$\sum_{i=1}^{D}ix_i^4+\text{random}[0,1)$	[−1.28,1.28]	100
F4	Rastrigrin	$\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	[−5.12,5.12]	100
F5	RosenBrock	$\sum_{i=1}^{D}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	[−30, 30]	100
F6	Schwefel	$\min(\lvert x_i\rvert, 1\leqslant i\leqslant D)$	[−10, 10]	100
F7	Sphere	$\sum_{i=1}^{D}x_i^2$	[−100, 100]	100
F8	Step	$\sum_{i=1}^{D}(x_i+0.5)^2$	[−100, 100]	100

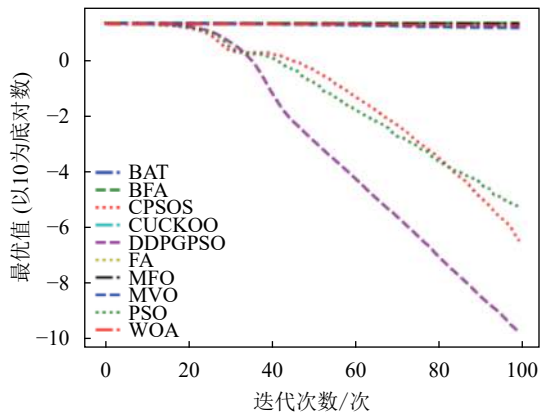

图 1　F1 迭代图

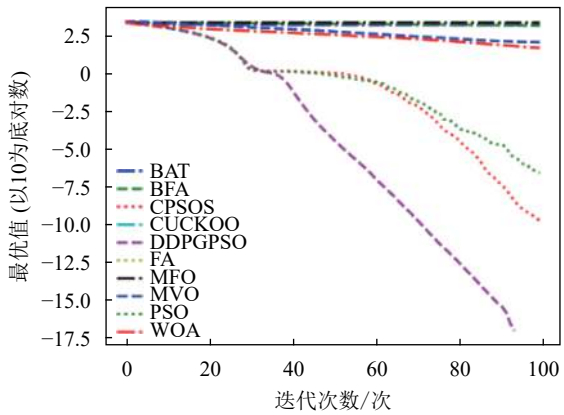

图 2　F2 迭代图

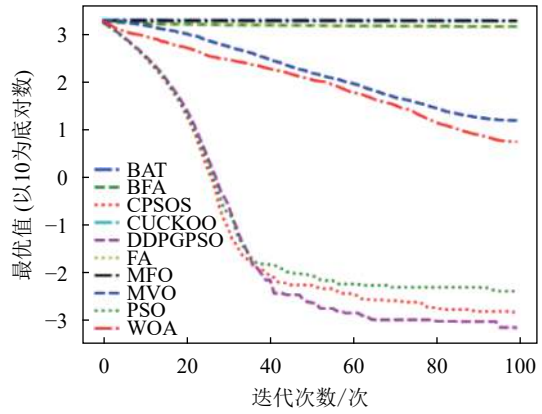

图 3 F3 迭代图

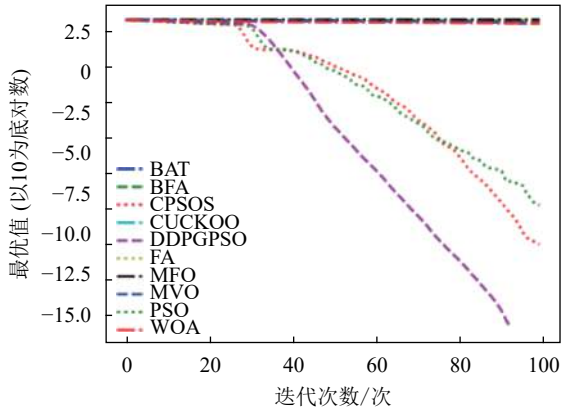

图 4 F4 迭代图

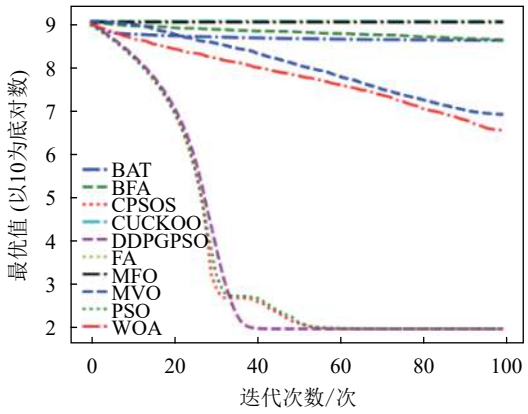

图 5 F5 迭代图

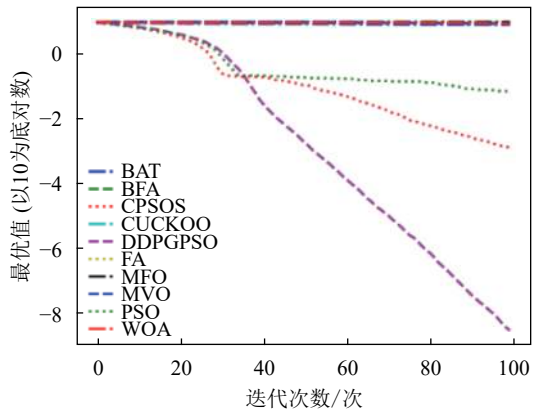

图 6 F6 迭代图

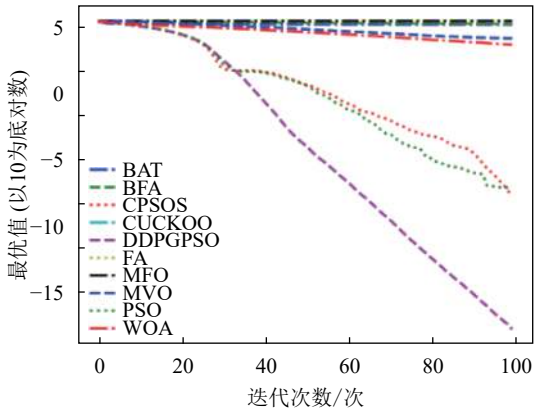

图 7 F7 迭代图

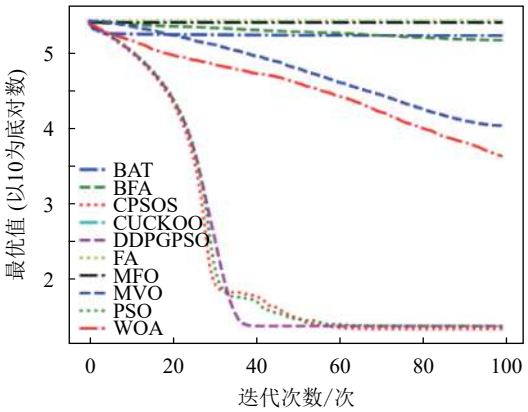

图 8 F8 迭代图

表 5 为 8 种算法在 8 个测试函数的实验结果对比，可以看出，DDPGPSO 在 F2、F4 函数上直接搜索到了最优值，而在其他函数上虽然没有搜索到最优值，但是与其他算法相比，在寻优精度上也有很大提高，另外，在 F5 和 F8 中，虽然算法不是第一名，但是所搜索出的最优值与第一名相比较，差距也在 10% 以内，说明 DDPGPSO 在各种情况下的性能都处于较好水平。另外可以看到，CPSOS 在多种情况下也优于标准粒子群网络，说明采用人工方式对参数进行一些设定也可以优化算法性能。而 DDPGPSO 在大多数情况下优于 CPSOS 也证明了机器学习在参数生成方面相较于人工方式的优势。

从图 1～图 8 可以看出，DDPGPSO 算法相较其他算法在收敛速度上提升明显，另外在多个测试函数中也取得了更好的测试结果。这种结果与强化学习中设定的更快收敛的奖励是一致的。这说明通过强化学习对 PSO 进行训练可以显著改善算法的各项性能，同时在多个函数测试项目上也取得了更好的收敛精度。

表 6 为 DDPGPSO 在不同迭代次数情况下的结果。可以看出，在迭代次数增加到 500、1 000 后，

DDPGPSO 仍是几种对比算法中最有竞争力的算法，在平均排名和排名第一的百分比两项参数上都有很大优势，也说明了该算法的优越性和强化学习应用于群智能算法的潜力。

表 5　实验结果对比

函数	数据名	DDPGPSO	BAT	BFA	CPSOS	CUCKOO	FA	MFO	MVO	PSO	WOA
	mean	2.35×10^{-10}	2.00×10	2.11×10	4.81×10^{-5}	2.07×10	2.12×10	2.03×10	1.45×10	2.34×10^{-2}	1.77×10
F1	best	$\mathbf{1.78\times10^{-10}}$	2.00×10	2.06×10	3.28×10^{-7}	2.01×10	2.08×10	2.00×10	1.44×10	5.87×10^{-6}	1.73×10
	std	3.84×10^{-11}	1.45×10^{-4}	1.77×10^{-1}	7.74×10^{-5}	2.20×10^{-1}	1.16×10^{-1}	9.87×10^{-2}	2.00×10^{-2}	4.53×10^{-2}	4.57×10^{-1}
	mean	5.55×10^{-19}	1.52×10^{3}	2.33×10^{3}	1.24×10^{-6}	6.49×10^{3}	2.99×10^{3}	8.09×10^{3}	1.12×10^{2}	1.21×10^{-1}	4.89×10
F2	best	**0.00**	1.52×10^{3}	1.37×10^{3}	1.97×10^{-10}	2.36×10^{3}	2.45×10^{3}	2.44×10^{3}	1.11×10^{2}	2.67×10^{-7}	4.73×10
	std	2.42×10^{-18}	2.13	6.70×10^{2}	6.43×10^{-6}	1.08×10^{3}	2.66×10^{2}	2.29×10^{2}	2.11×10^{-1}	2.79×10^{-1}	1.35
	mean	5.17×10^{-1}	2.38×10^{3}	2.42×10^{3}	5.15×10^{-1}	7.03×10^{3}	2.95×10^{3}	1.19×10^{4}	1.53×10	5.29×10^{-1}	5.78
F3	best	$\mathbf{6.79\times10^{-4}}$	1.80×10^{3}	1.37×10^{3}	1.23×10^{-3}	1.78×10^{3}	1.33×10^{3}	1.82×10^{3}	1.48×10	3.93×10^{-3}	5.26
	std	2.85×10^{-1}	2.38×10^{2}	4.17×10^{2}	2.81×10^{-1}	1.52×10^{3}	4.76×10^{2}	4.91×10^{2}	2.87×10^{-1}	2.75×10^{-1}	3.36×10^{-1}
	mean	0.00	1.58×10^{3}	1.75×10^{3}	1.93×10^{-7}	2.37×10^{3}	1.85×10^{3}	2.74×10^{3}	9.72×10^{2}	1.29×10^{-1}	9.80×10^{2}
F4	best	**0.00**	1.47×10^{3}	1.42×10^{3}	2.31×10^{-11}	1.59×10^{3}	1.52×10^{3}	1.66×10^{3}	9.67×10^{2}	6.28×10^{-9}	9.63×10^{2}
	std	0.00	4.45×10	1.27×10^{2}	6.00×10^{-7}	1.96×10^{2}	1.04×10^{2}	5.47×10	2.05	4.55×10^{-1}	2.35×10
	mean	1.00×10^{2}	4.54×10^{8}	1.11×10^{9}	9.99×10	5.17×10^{9}	1.62×10^{9}	7.17×10^{9}	8.55×10^{6}	1.32×10^{2}	3.77×10^{6}
F5	best	9.99×10	4.27×10^{8}	4.38×10^{8}	$\mathbf{9.98\times10}$	1.18×10^{9}	1.14×10^{9}	1.16×10^{9}	8.50×10^{6}	9.99×10	3.65×10^{6}
	std	2.24×10^{-3}	3.27×10^{7}	5.28×10^{8}	4.16×10^{-2}	1.04×10^{9}	2.25×10^{8}	2.55×10^{8}	2.55×10^{4}	1.73×10^{2}	7.84×10^{4}
	mean	5.19×10^{-9}	8.21	9.76	3.00×10^{-3}	9.97	9.93	1.00×10	8.32	3.15×10^{-1}	8.72
F6	best	$\mathbf{3.09\times10^{-9}}$	7.89	8.88	1.35×10^{-3}	9.52	9.41	9.62	8.30	7.09×10^{-2}	8.59
	std	1.66×10^{-9}	1.89×10^{-1}	1.90×10^{-1}	1.41×10^{-3}	1.06×10^{-1}	9.70×10^{-2}	0.00	8.67×10^{-3}	1.57×10^{-1}	1.62×10^{-1}
	mean	2.66×10^{-18}	1.51×10^{5}	2.60×10^{5}	1.95×10^{-6}	7.08×10^{5}	3.34×10^{5}	9.02×10^{5}	1.32×10^{4}	1.84×10^{-1}	4.56×10^{3}
F7	best	$\mathbf{1.28\times10^{-18}}$	1.49×10^{5}	1.56×10^{5}	1.66×10^{-8}	2.74×10^{5}	2.66×10^{5}	2.70×10^{5}	1.32×10^{4}	6.14×10^{-8}	4.45×10^{3}
	std	1.16×10^{-18}	1.54×10^{3}	7.01×10^{4}	2.66×10^{-6}	1.26×10^{5}	3.09×10^{4}	2.39×10^{4}	2.39×10	8.43×10^{-1}	9.98×10
	mean	2.44×10	1.74×10^{5}	2.60×10^{5}	2.33×10	7.13×10^{5}	3.30×10^{5}	8.99×10^{5}	1.10×10^{4}	4.91×10	4.53×10^{3}
F8	best	2.36×10	1.73×10^{5}	1.50×10^{5}	$\mathbf{2.16\times10}$	2.66×10^{5}	2.72×10^{5}	2.65×10^{5}	1.09×10^{4}	2.34×10	4.29×10^{3}
	std	4.36×10^{-2}	1.53×10^{3}	7.25×10^{4}	4.57×10^{-1}	1.08×10^{5}	2.98×10^{4}	3.27×10^{4}	2.17×10	2.69×10	2.49×10^{2}

表 6　不同迭代次数下 DDPGPSO 排名

迭代次数	平均排名	排名第一百分比/%
100	1.375	75
500	1.625	62.5
1 000	1.75	62.5

综上所述，基于深度确定性策略梯度算法的粒子群优化算法 (DDPGPSO) 相较于其他同类别算法具有更高的寻优精度，更快的收敛速度，能够有效解决高维度的复杂数值优化问题。基于强化学习的群优化算法相较于手工设定参数的群优化算法拥有更好的寻优能力与探索能力。

3　结 束 语

粒子群优化算法是模拟鸟群和鱼群的觅食和聚集行为而提出的一种群智能优化算法。本文提出了一种基于深度确定性策略梯度的粒子群算法 (DDPGPSO)，利用强化学习的手段来自动控制参数的变化，提高了原有 PSO 算法的收敛精度和收敛速度。通过测试算法在基准测试函数上的效果，验证了该算法的有效性和可行性，为以后强化学习与群智能算法的结合提供了一种方案。结果表明，同其他 9 种优化算法相比，DDPGPSO 算法求解质量取得了很好的效果。今后的研究工作一方面会通过更多的实验来验证并改进算法的性能，另一方面将探索 DDPG 与其他算法结合是否能产生更好的效果。

参 考 文 献

[1] KENNEDY J, EBERHART R. Particle swarm optimization[C]//International Conference on Neural Networks. Perth, Australia: IEEE, 1995: 1942-1948.

[2] MIRJALILI S, LEWIS A. The whale optimization algorithm[J]. Advances in Engineering Software, 2016, 95: 51-67.

[3] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimizer[J]. Advances in Engineering Software, 2014, 69: 46-61.

[4] YANG X S. A new metaheuristic bat-inspired algorithm[J]. Computer Knowledge & Technology, 2010, 284: 65-74.

[5] ZHAO Fu-qing, QIN Shuo, YANG Guo-qiang, et al. A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop

scheduling problem with the makespan objective[J]. Expert Systems with Application, 2019, 126: 41-53.
[6] SANG Jin-guo, CUI Huan-qiang. Energy saving schedule of mine drainage system based on particle swarm optimization[J]. Journal of Physics: Conference Series, 2019, DOI: 10.1088/1742-6596/1168/2/022001.
[7] CUI Zhi-hua, ZHANG Jiang-jiang, WU Di, et al. Hybrid many-objective particle swarm optimization algorithm for green coal production problem[J]. Information Sciences, 2020, 518: 256-271.
[8] WU Yu-qiang, WU Yong-gang, LIU Xing-long. Couple-based particle swarm optimization for short-term hydrothermal scheduling[J]. Applied Soft Computing, 2019, 74: 440-450.
[9] WANG Yu, LI Bin, THOMAS W, et al. Self-adaptive learning based particle swarm optimization[J]. Information Sciences, 2011, 181(20): 4515-4538.
[10] SHI Y. A modified particle swarm optimizer[C]//1998 IEEE World Congress on Computational Intelligence. [S.l.]: IEEE, 1998: 69-73.
[11] CLERC M, KENNEDY J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(1): 58-73.
[12] RATNAWEERA A, HALGAMUGE S K, WATSON H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 240-255.
[13] ZHAN Zhi-hui, ZHANG Jun, LI Yun, et al. Adaptive particle swarm optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(6): 1362-1381.
[14] XU Gui-ping, CUI Quan-long, SHI Xiao-hu, et al. Particle swarm optimization based on dimensional learning strategy[J]. Swarm and Evolutionary Computation, 2019, 45: 33-51.
[15] ANG K M, LIM W H, ISA N A M, et al. A constrained multi-swarm particle swarm optimization without velocity for constrained optimization problems[J]. Expert Systems with Applications, 2020, DOI: 10.1016/j.eswa.2019.112882.
[16] ZHANG Xu-wei, LIU Hao, ZHANG Tong, et al. Terminal crossover and steering-based particle swarm optimization algorithm with disturbance[J]. Applied Soft Computing, 2019, DOI: 10.1016/j.asoc.2019.105841.
[17] WANG Sheng-liang, LIU Gen-you, GAO Ming, et al. Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators[J]. Information Sciences, 2020, DOI: 10.1016/j.ins.2020.06.027.
[18] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[J]. Computer Science, 2016, 8(6): A187.
[19] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms[C]//Proceedings of the International Conference on Machine Learning. [S.l.]: ACM, 2014: 387-395.
[20] TIAN Dong-qiang, ZHAO Xiao-fei, SHI Zhong-zhi. Chaotic particle swarm optimization with sigmoid-based acceleration coefficients for numerical function optimization[J]. Swarm and Evolutionary Computation, 2019, 51: 100573.
[21] 张强, 郭玉洁, 王颖, 等. 一种离散鲸鱼算法及其应用[J]. 电子科技大学学报, 2020, 49(4): 622-630.
ZHANG Qiang, GUO Yu-jie, WANG Ying, et al. A discrete whale optimization algorithm and application[J]. Journal of the University of Electronic Science and Technology of China, 2020, 49(4): 622-630.

编 辑 叶 芳