Journal of System Simulation

Volume 36 | Issue 7 Article 11

7-15-2024

Adaptive Particle Swarm Optimization Algorithm Based on Trap Label and Lazy Ant

Wei Zhang
College of Electrical Engineering and Automation, Henan Polytechnic University, Jiaozuo 454003, China, zwei1563@126.com

Yuefeng Jiang
College of Electrical Engineering and Automation, Henan Polytechnic University, Jiaozuo 454003, China

Adaptive Particle Swarm Optimization Algorithm Based on Trap Label and Lazy Ant

Abstract

Abstract: Many existing strategies for improving particle swarm optimization (PSO) fall short in assisting particles trapped in local optima and experiencing premature convergence to recover optimization performance. In response, an adaptive particle swarm optimization algorithm based on trap label and lazy ant (TLLA-APSO) is proposed. Firstly, the trap label strategy dynamically adjusts particle velocities, enabling the particle swarm to escape from local optima. Secondly, the lazy ant optimization strategy is employed to diversify particle velocity and enhance population diversity. Finally, the inertia cognition strategy introduces historical position into velocity updates, promoting path diversity and particle exploration while effectively mitigating the risk of falling into new local optimum. The convergence of the particle swarm algorithm with the incorporation of historical positions has been empirically demonstrated. Simulation results validate the efficacy of TLLA-APSO, showcasing its ability to mitigate local optima and premature convergence while achieving faster convergence speed and higher optimization accuracy compared with other algorithms.

Keywords

particle swarm optimization algorithm (PSO), lazy ant, trap label, local optima, premature convergence

Recommended Citation

Zhang Wei, Jiang Yuefeng. Adaptive Particle Swarm Optimization Algorithm Based on Trap Label and Lazy Ant[J]. Journal of System Simulation, 2024, 36(7): 1631-1642.

第 36 卷第 7 期 系统仿真学报© Vol. 36 No. 7
2024 年 7 月 Journal of System Simulation Jul. 2024

陷阱标记联合懒蚂蚁的自适应粒子群优化算法

张伟，蒋岳峰
(河南理工大学 电气工程与自动化学院，河南 焦作 454003)

摘要：为解决现有粒子群改进策略无法帮助已陷入局部最优和过早收敛的粒子恢复寻优性能的问题，提出一种陷阱标记联合懒蚂蚁的自适应粒子群优化(adaptive particle swarm optimization based on trap label and lazy ant, TLLA-APSO)算法。*陷阱标记策略为粒子群提供动态速度增量，使其摆脱最优解的束缚。利用懒蚂蚁寻优策略多样化粒子速度，提升种群多样性。通过惯性认知策略在速度更新中引入历史位置，增加粒子的路径多样性和提升粒子的探索性能，使粒子更有效地避免陷入新的局部最优。*理论证明了引入历史位置的粒子群算法的收敛性。仿真实验结果表明，所提算法不仅能有效解决粒子群已陷入局部最优和过早收敛的问题，且与其他算法相比，具有较快的收敛速度和较高的寻优精度。

关键词：粒子群优化算法；懒蚂蚁；陷阱标记；局部最优；过早收敛
中图分类号：TP391.9 文献标志码：A 文章编号：1004-731X(2024)07-1631-12
DOI: 10.16182/j.issn1004731x.joss.23-0401
引用格式: 张伟, 蒋岳峰. 陷阱标记联合懒蚂蚁的自适应粒子群优化算法[J]. 系统仿真学报, 2024, 36(7): 1631-1642.
Reference format: Zhang Wei, Jiang Yuefeng. Adaptive Particle Swarm Optimization Algorithm Based on Trap Label and Lazy Ant[J]. Journal of System Simulation, 2024, 36(7): 1631-1642.

Adaptive Particle Swarm Optimization Algorithm Based on Trap Label and Lazy Ant

Zhang Wei, *Jiang Yuefeng*
(College of Electrical Engineering and Automation, Henan Polytechnic University, Jiaozuo 454003, China)

Abstract: Many existing strategies for improving particle swarm optimization (PSO) fall short in assisting particles trapped in local optima and experiencing premature convergence to recover optimization performance. In response, an adaptive particle swarm optimization algorithm based on trap label and lazy ant (TLLA-APSO) is proposed. *Firstly*, *the trap label strategy dynamically adjusts particle velocities, enabling the particle swarm to escape from local optima. Secondly*, *the lazy ant optimization strategy is employed to diversify particle velocity and enhance population diversity. Finally*, *the inertia cognition strategy introduces historical position into velocity updates, promoting path diversity and particle exploration while effectively mitigating the risk of falling into new local optimum.* The convergence of the particle swarm algorithm with the incorporation of historical positions has been empirically demonstrated. Simulation results validate the efficacy of TLLA-APSO, showcasing its ability to mitigate local optima and premature convergence while achieving faster convergence speed and higher optimization accuracy compared with other algorithms.

Keywords: particle swarm optimization algorithm (PSO); lazy ant; trap label; local optima; premature convergence

收稿日期：2023-04-08 修回日期：2023-06-12
基金项目：国家自然科学基金(61703145)；河南省科技攻关项目(222102210213)
第一作者：张伟(1978-)，女，教授，博士，研究方向为智能特征建模，神经网络结构优化，多目标优化等。E-mail：zwei1563@126.com

0 引言

元启发算法是解决优化问题的有效方法[1]，其中，粒子群优化(particle swarm optimization, PSO)算法因其简单易行且调整参数少的优点，成为目前应用最广泛的元启发算法之一[2]。但随着优化问题的复杂性增加，局部最优解的数量将指数级增长，导致粒子群很容易陷入局部最优[3]。同时，PSO 算法因收敛速度过快，容易出现过早收敛导致开发性不足。为解决上述问题，学者们从不同角度展开了研究，并提出了一些改进策略。

PSO 算法的寻优过程分为探索阶段和开发阶段，探索阶段侧重搜索解空间的最优区域，开发阶段侧重搜索最优区域中的最优解，二者的平衡能够改善粒子群局部最优和过早收敛问题。惯性权重作为影响算法平衡能力的参数之一[4]，其调整方法相继被提出[5]，通过调整算法的探索性来实现开发和探索的平衡，从而缓解局部最优和过早收敛问题。同时，一些学者提出利用认知系数来调整算法的开发性，以实现探索性和开发性的平衡。例如，文献[6]提出一种自组织分层粒子群算法，其认知系数随时间变化，使算法在不同阶段的开发性和探索性更加平衡，从而提升了整体寻优性能。尽管通过参数的改进在一定程度上平衡了算法的探索性和开发性，并缓解了粒子群算法容易陷入局部最优和过早收敛的问题，但参数策略的动态调整能力无法完全适应粒子寻优过程中多变的状态。因此，算法平衡能力仍有较大的提升空间。

为进一步提升算法的平衡能力，一些学者提出将种群分为不同分工的子群[7]。不仅大大降低对动态调整的需求，也相对平衡地保证了粒子群的探索性和开发性。文献[8]采用多种群协作并引入了差分变异策略，既提升了探索性，又保持了与开发性的平衡，局部最优问题得到改善。文献[9]提出将种群划分为具有不同学习机制的主群和悬停群。前者负责强化算法的探索能力，后者负责在子群层次上平衡探索和开发。探索能力的提升也使算法能有效地避免陷入局部最优。尽管上述改进措施提升了算法的平衡能力，但对种群的划分削弱了整体的探索和开发强度，因此局部最优问题和过早收敛问题仍未得到有效解决。

为增强粒子群算法的探索性和开发性，避免陷入局部最优和过早收敛，文献[10]在速度更新中引入随机项来增加粒子群的多样性，虽提升了算法的探索性，但导致了其开发能力的下降。文献[11]通过随机选取粒子指导更新的方式提高种群多样性，进一步改善了过早收敛和局部最优问题。文献[12]通过改变拓扑结构来调整群成员间的信息流动速率，改善了算法的探索能力，但开发能力的下降使其在处理单峰问题时表现不佳。经过上述策略改进后的粒子群算法能一定程度上避免局部最优和过早收敛问题，但由于开发能力的下降，算法的整体性能并未得到明显提升。

为解决探索性与开发性提升间的冲突问题，学者们提出将新的学习策略融入到粒子群算法中。文献[13]引入正交学习策略，使粒子群算法的开发效率得到显著提升。将其他算法中的学习策略融入到 PSO 的方式也得到了广泛应用。文献[14]将遗传算法中选择、交叉和突变算子引入到 PSO 更新过程，使得算法探索性和开发性得到提升。文献[15]提出一种粒子群算法和遗传算法的混合算法，在粒子完全停滞时，用遗传算法中的变异算子替换 PSO 中与速度相关的部分，以解决局部最优或过早收敛问题。然而，由于变异算子的随机性作用，粒子无法实现有效搜索。文献[16]引入混沌理论，并结合 Logistic 映射，使种群具有更好的遍历性，以避免陷入局部最优。同时，通过混沌理论自适应控制搜索范围，增加了算法跳出局部最优的概率。虽然融合了新学习策略的粒子群算法的探索性和开发性在互不影响的前提下得到了提升，并能有效避免局部最优和过早收敛的问题，但在复杂环境中，当粒子群已陷入局部最优或开发精度不满足要求而导致过早收敛时，以上改进

DOI: 10.16182/j.issn1004731x.joss.23-0401

策略无法通过指导粒子群有效搜索来帮助其摆脱困境。因此，在提升粒子群算法探索性的同时，还要考虑当粒子群已陷入局部最优或过早收敛时，如何恢复粒子群的探索和开发能力，并确保改进策略能指导粒子进行有效搜索。

为进一步改进粒子群算法，帮助已陷入局部最优和过早收敛的粒子群摆脱困境，本文提出一种陷阱标记联合懒蚂蚁的自适应粒子群算法(adaptive particle swarm optimization based on trap label and lazy ant, TLLA-APSO)。主要创新工作如下：①与目前大多数改进粒子群算法仅考虑如何避免陷入局部最优和过早收敛不同，所提算法能够解决粒子群已陷入局部最优和过早收敛的优化问题，同时指导粒子群有效搜索并保持良好的探索性和开发性。并且所提算法的探索性得到进一步增强，更加有效地避免陷入新的局部最优。②提出一种帮助粒子群摆脱束缚的陷阱标记联合懒蚂蚁策略，通过标记陷阱中最优粒子的位置，为粒子群提供动态速度增量。利用懒蚂蚁的寻优策略对陷阱中的粒子进行速度多样化处理，提升种群在摆脱陷阱束缚时搜索方向的多样性。③提出一种增强算法探索性的惯性认知策略，在速度更新过程中引入历史位置使认知部分具有惯性，增强粒子的路径多样性，从而提升算法的探索性能，对引入历史位置的PSO算法进行收敛性证明，以验证其可行性。

1 PSO算法及其问题分析

在PSO算法中，一群粒子在搜索空间飞行以寻找最优解，根据个体认知和社会认知调整粒子速度。设t时刻第i个粒子的位置表示为$\boldsymbol{X}_i(t)$，t时刻第i个粒子的速度表示为$\boldsymbol{V}_i(t)$，第i个粒子的个体最优粒子位置$\boldsymbol{X}_{\mathrm{p}i}$，全局最优粒子位置$\boldsymbol{X}_{\mathrm{g}}$。第$i$个粒子在$t$+1时刻的速度以及位置更新公式为

$$\begin{aligned}\boldsymbol{V}_i(t+1)=&w\boldsymbol{V}_i(t)+c_1 rand(1)(\boldsymbol{X}_{\mathrm{p}i}-\boldsymbol{X}_i(t))+\\&c_2 rand(1)(\boldsymbol{X}_{\mathrm{g}}-\boldsymbol{X}_i(t))\end{aligned}\tag{1}$$

$$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)+\boldsymbol{V}_i(t+1)\tag{2}$$

式中：w为惯性权重；c_1、c_2为学习因子；$rand(1)$为[0,1]区间上均匀分布的随机数。

在PSO算法中，粒子的寻优路径始终在一个相对固定的区间内等幅振荡，路径多样性不足，开发性能较弱。定义个体最优粒子和群体最优粒子为引导粒子。在引导粒子未更新时，粒子群迅速向其周围聚集，并因束缚而停滞在固定区域。

因此，引导粒子虽然能够指导粒子群向最优区域搜索，但是当粒子群探索性和开发性不足时，引导粒子容易束缚粒子群，导致其陷入局部最优和过早收敛。

2 TLLA-APSO算法

TLLA-APSO算法整体结构如图1所示。

根据停滞评价指标和陷阱评价指标评估PSO算法的寻优环境，当种群处于局部最优或过早收敛的陷阱状态时，采用陷阱标记联合懒蚂蚁的粒子群算法进行后续寻优。陷阱标记联合懒蚂蚁策略帮助粒子群摆脱陷阱中引导粒子的束缚，并避免粒子群再次陷入该区域。陷阱增量评价指标确保算法搜索的有效性。惯性认知策略改善粒子群算法的探索性能，降低陷入新的局部最优的概率。

2.1 陷阱状态粒子群的判断

在评估粒子群的寻优状态方面，采用停滞评价指标和陷阱评价指标进行判断。停滞评价指标利用粒子群在陷入局部最优或过早收敛时最优解更新幅度小的特点，判断粒子群所处状态是否良好。

当t+1时刻最优解大于t时刻最优解的1/10时，表明种群最优解更新幅度较小，则停滞次数指标sg更新为

$$sg=sg+1,\text{if } f_{\mathrm{g}}(t+1)>0.1f_{\mathrm{g}}(t)\tag{3}$$

式中：$f_{\mathrm{g}}(t)$为t时刻最优解$\boldsymbol{X}_{\mathrm{g}}$的适应度。

当陷阱评价指标中的停滞次数sg大于设定阈值时，判定粒子群陷入局部最优或过早收敛。停滞阈值的设定对于算法的性能有一定影响，若太

小则容易使算法发散，太大则降低了摆脱陷阱的效率。设定停滞阈值为10左右时算法性能最优。

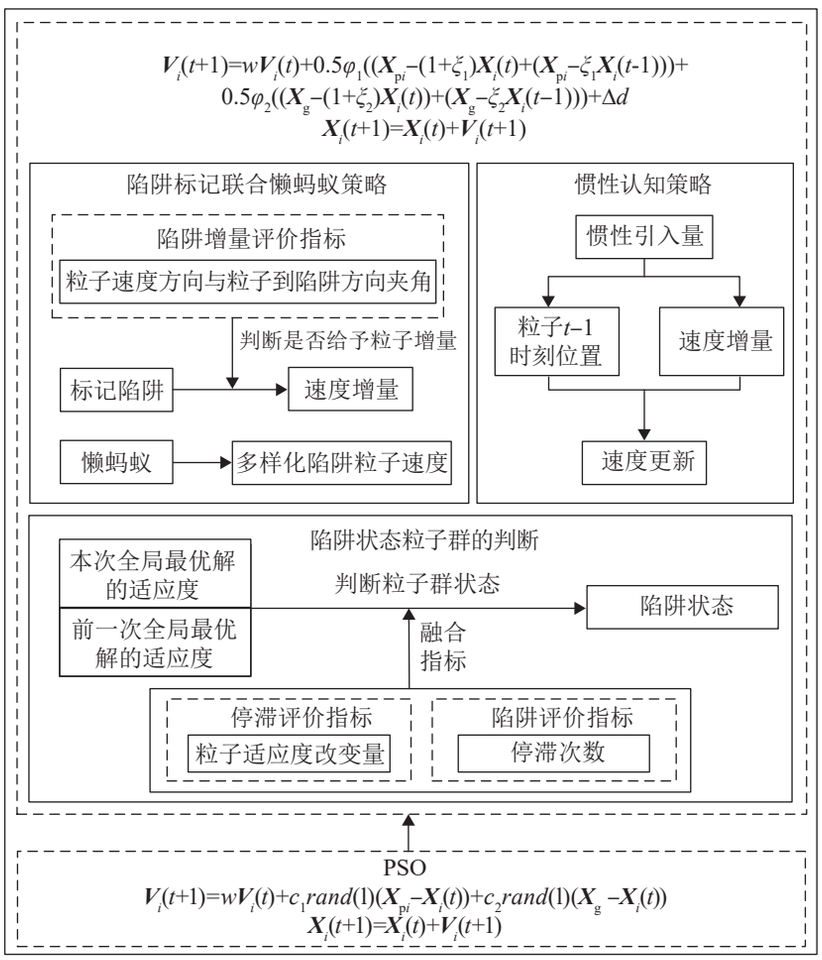


图1 TLLA-APSO算法整体框架
Fig. 1 TLLA-APSO Algorithm framework

2.2 陷阱标记联合懒蚂蚁策略

陷阱标记联合懒蚂蚁策略分为陷阱标记和速度多样化两部分。前者帮助粒子群摆脱困境，恢复探索性和开发性，后者提升种群多样性。

2.2.1 陷阱标记

陷阱标记的对象是粒子群的最优解。陷阱增量评价指标通过粒子的速度方向与粒子到被标记位置方向的夹角，判断粒子的运动与被标记位置的关系，如图2所示。

定义粒子到被标记位置的状态判断量为

$$S=\boldsymbol{V}_i^{\mathrm{T}}(\boldsymbol{trap}_k-\boldsymbol{X}_i) \tag{4}$$

式中：$\boldsymbol{trap}_k$为第k个被标记的位置。

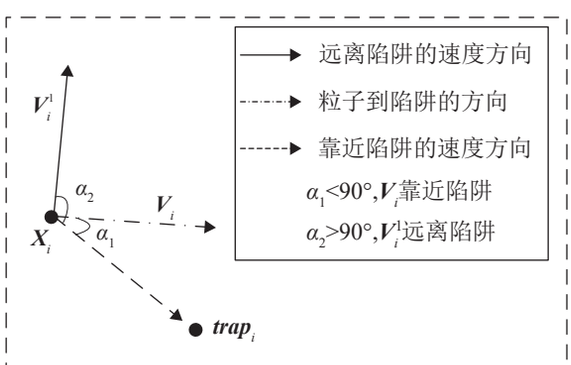


图2 粒子的运动与被标记位置的关系
Fig. 2 Relationship between particle motion and marked position

当粒子速度方向与粒子到被标记位置方向的夹角大于90°时，式(4)中$S<0$，粒子远离被标记

DOI: 10.16182/j.issn1004731x.joss.23-0401

位置；反之，$S>0$，粒子靠近被标记位置。

当粒子远离被标记位置时，给予粒子速度增量。第k个被标记的位置对第i个粒子产生的速度增量为

$$\Delta d_k(i)=\mathrm{e}^{-\left\|\boldsymbol{X}_i-\boldsymbol{trap}_k\right\|}\left\|\boldsymbol{V}_i\right\|(\boldsymbol{X}_i-\boldsymbol{trap}_k) \tag{5}$$

通过陷阱增量评价指标来决定是否给予其他粒子速度增量。这既能帮助粒子摆脱束缚，防止粒子群再次陷入该区域，同时也不影响粒子群对该区域附近的正常开发，保证了群体搜索的有效性。当提供的速度增量无法帮助粒子群摆脱束缚时，陷阱评价指标会使该最优解被反复标记，速度增量不断叠加，直到粒子群摆脱束缚。

2.2.2 粒子速度多样化

为了避免粒子群在陷阱标记的作用下在远离陷阱的区域聚集，影响后续的寻优效率，因此在提供速度增量前，恢复种群一定程度的多样性是必要的。在生物学背景下，蚁群中的懒蚂蚁因不受群体信息的约束，具有良好的多样性，其在寻优过程中大部分时间保持直线运动，有时左转或右转，少数情况下反向运动[17]，这与粒子群的不足形成一种互补[18]。当陷阱评价指标判定粒子群陷入局部最优或过早收敛时，利用懒蚂蚁寻优机制来多样化粒子群的速度方向，提升种群的多样性，使其在摆脱困境后拥有更大的探索区域。这不仅提高算法后续的探索性能，还在不影响粒子群有效搜索的前提下，最大程度地利用了懒蚂蚁寻优机制。

2.3 惯性认知策略

摆脱陷阱束缚的粒子群仍然有可能陷入新的局部最优。为解决此问题，提出惯性认知策略进一步增强算法探索性。惯性认知策略在粒子的速度更新中引入t–1时刻历史位置$\boldsymbol{X}_i(t-1)$，使粒子速度更新的认知部分具有惯性，增加速度更新的决策多样性。懒蚂蚁寻优机制使得引入的处于陷阱中的历史位置具有较好的多样性，从而提升算法的探索性能。TLLA-APSO算法的速度、位置更新公式为

$$\begin{aligned}\boldsymbol{V}_i(t+1)=&w\boldsymbol{V}_i(t)+\\&0.5\varphi_1(2\boldsymbol{X}_{pi}-(1+\xi_1)\boldsymbol{X}_i(t)-\xi_1\boldsymbol{X}_i(t-1))+\\&0.5\varphi_2(2\boldsymbol{X}_g-(1+\xi_2)\boldsymbol{X}_i(t)-\xi_2\boldsymbol{X}_i(t-1))\end{aligned} \tag{6}$$

$$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)+\boldsymbol{V}_i(t+1) \tag{7}$$

式中：ξ_1、ξ_2为[0,1]上均匀分布的随机数。

图3为粒子速度更新中社会认知增量即$\boldsymbol{\alpha}=(2\boldsymbol{X}_g-(1+\xi_2)\boldsymbol{X}_i(t)-\xi_2\boldsymbol{X}_i(t-1))$的示意图。该增量可分解为

$$(\boldsymbol{X}_g-(1+\xi_2)\boldsymbol{X}_i(t))+(\boldsymbol{X}_g-\xi_2\boldsymbol{X}_i(t-1)) \tag{8}$$

令$\boldsymbol{\alpha}_1=\boldsymbol{X}_g-(1+\xi_2)\boldsymbol{X}_i(t)$，$\boldsymbol{\alpha}_2=\boldsymbol{X}_g-\xi_2\boldsymbol{X}_i(t-1)$，$\boldsymbol{b}=\boldsymbol{X}_g-\boldsymbol{X}_i(t)$。

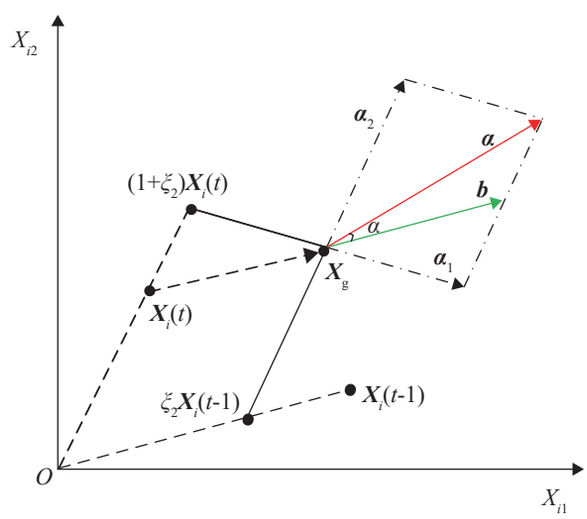


图3 社会认知增量
Fig. 3 Social cognitive increment

从图3可以看出，引入历史位置后的粒子社会认知增量得到提升，加速粒子对最优区域的开发且不易被最优解束缚。与直接指向最优解的原始社会认知增量方向$\boldsymbol{b}$的小角度偏差α能防止粒子深陷局部最优中。这些特性使得经过惯性认知策略改进后的粒子群算法能更加有效地避免陷入新的局部最优中。

3 实验研究与分析

为验证TLLA-APSO算法的有效性，设计3个实验：陷阱实验、路径多样性实验和标准测试函数下的寻优实验。陷阱实验和路径多样性实验测

试与分析TLLA-APSO算法解决粒子群已陷入局部最优和过早收敛问题的能力，寻优实验测试TLLA-APSO算法的收敛速度和寻优精度。所有的实验算法均以寻找测试函数的最小值为目标。

3.1 测试函数

为进行上述实验，选择9个常用基准测试函数进行对比实验。各测试函数信息见表1。其中，f_1~f_3是单峰函数，f_4~f_9是典型的多峰函数，有大量的局部最优解。

表1 测试函数
Table 1 Test function

函数	含义	搜索范围
f_1	Schwefel	[−10,10]
f_2	Quadric	[−100,100]
f_3	Rosenbrock	[−30,30]
f_4	Rastrigin	[−5.12,5.12]
f_5	Ackley	[−32,32]
f_6	Griewank	[−30,30]
f_7	Levy	[−30,30]
f_8	Happy Cat	[−100,100]
f_9	Expanded Schaffer's F6	[−100,100]

3.2 参数设置

实验选取GPSO[4]、HPSO-TVAC[6]、FIPS[11]、CLPSO[11]、OLPSO[13]、GAPSO[15]、GLPSO[14]、GEPSO[10]和CAPSO[16]算法作为对比算法，对比算法与TLLA-APSO算法参数设置如表2。各对比算法中的参数设置均与原文献一致。

粒子数为n，最大迭代次数$T_{\max}$，粒子维度为30，为了减少实验中的随机性，各个算法在每个测试函数上均随机独立运行30次。实验均在Inter Core i5-9300HF 2.5 GHz CPU，16 GB内存，Windows 10操作系统环境下采用Matlab 2020b进行。

3.3 陷阱实验

实验采用PSO算法对9个测试函数进行寻优，选择使其陷入局部最优和过早收敛的测试函数作为陷阱函数。利用陷阱函数中的信息测试各算法解决粒子群已陷入局部最优和过早收敛问题的能力。当算法最优解的适应度小于陷阱值的1/10时，表明引导粒子已经不在陷阱区域，即视为成功解决局部最优或过早收敛问题。

表2 参数配置
Table 2 Optimum configuration

算法	参数配置
GPSO	w=0.9~0.4, c_1=c_2=2
HPSO-TVAC	c_1=0.5, c_2=0.5~2.5
FIPS	w=0.729 8, $\sum_{i=1}^{3} c_i = 4.1$
CLPSO	w=0.729 8, c=1.496 18, m=7, Pc=0.05~0.5
OLPSO	w=0.9~0.4, c=2, G=5
GAPSO	w=0.9~0.4, c_1=c_2=2, P_c=0.5, P_m=0.05, ds=7
GLPSO	w=0.729 8, c=1.496 18, P_m=0.01, sg=7
CAPSO	$w_{\max}=0.9$, $w_{\min}=0.4$, $c_{\max}=2.5$, $c_{\min}=0.5$, $\xi=0.2$
GEPSO	$c_1=c_2=c_3=c_4=\alpha_1=3$, $\alpha_2=\alpha_3=2$, $w_2=w_3=w_4=w_5=0.5$
TLLA-APSO	$w=0.9\sim0.4$, $c_1=c_2=2$, $t=10$

3.3.1 陷阱函数选取

实验发现，PSO算法在Levy、Rastrigin、Quadric、Rosenbrock和Ackley函数中陷入局部最优，引导粒子不在最优区域，且存在明显的停滞现象。在Schwefel、Griewank函数中过早收敛，引导粒子在最优区域，其适应度值与真实最优适应度值的误差大于10^{-6} s，且出现了明显的停滞现象。当算法陷入局部最优和过早收敛时，由于引导粒子接近停止更新状态，速度增量小，且小于1的惯性权重导致速度持续衰减，使得粒子群停滞在引导粒子附近，失去了有效的探索性和开发性。选择以上7个函数作为陷阱函数。表3给出了标准粒子群在7个陷阱函数中的最优解的适应度值，即陷阱值。

3.3.2 陷阱实验结果与分析

为验证所提算法解决粒子群已陷入局部最优和过早收敛问题的能力，利用7个陷阱函数对其进行测试，选择实验结果最好的GLPSO、CLPSO和OLPSO算法作为对比算法。表4给出了GLPSO

DOI: 10.16182/j.issn1004731x.joss.23-0401

和TLLA-APSO算法跳出陷阱的判定结果、所需迭代次数以及跳出时的最优适应度值。因未能满足跳出陷阱的条件，所以未展示CLPSO和OLPSO算法的实验结果。由表4可知，GLPSO算法在7个陷阱中成功跳出5个，而所提算法在7个陷阱中均成功跳出，且迭代次数明显少于GLPSO算法。在过早收敛问题中，GLPSO算法只解决了Schwefel函数中的过早收敛，而所提算法在2个过早收敛问题中均表现良好。

表3 7个陷阱函数中的陷阱值
Table 3 Trap values in 7 trap functions

函数	陷阱值
Schwefel	1.42e−06
Rosenbrock	2.02e+01
Rastrigin	9.65e+01
Quadric	1.38e+00
Levy	8.90e+00
Griewank	3.45e−02
Ackley	1.77e+00

表4 陷阱实验测试结果
Table 4 Test results from trap experiments

函数	GLPSO			TLLA-APSO		
	跳出陷阱	迭代次数	最优解	跳出陷阱	迭代次数	最优解
Schwefel	√	183	3.26e−08	√	13	2.32e−08
Rosenbrock	×	—	1.71e+01	√	853	1.87e+00
Rastrigin	√	20	1.89e+01	√	16	4.17e+00
Quadric	√	387	1.32e−01	√	12	5.10e−03
Levy	√	11	3.98e−01	√	51	5.48e−01
Griewank	×	—	3.45e−02	√	15	1.80e−03
Ackley	√	183	1.18e−01	√	12	1.18e−01

3.4 路径多样性实验

陷阱实验已证明所提算法能够解决局部最优和过早收敛问题，为验证多样性对粒子群跳出陷阱的关键作用，设计面对局部最优和过早收敛的路径多样性实验。通过路径多样性指标来分析种群在局部最优和过早收敛情况下的寻优状态。选择陷阱实验中的陷阱函数作为测试函数，设定粒子数n为50。为减少不同引导粒子对路径多样性指标的影响，设置最大迭代次数$T_{\max}$为20。

3.4.1 实验设计步骤

step 1：在引导粒子不变的情况下，从粒子群中随机选择一个粒子作为star粒子，统计star粒子迭代20次的位置信息$X_{\text{star}}(t)$，$t=1,2,\cdots,20$，求出平均位置$\overline{X_{\text{star}}}=\frac{1}{T_{\max}}\sum_{t=1}^{T_{\max}}X_{\text{star}}(t)$。

step 2：求出star粒子的各次迭代位置$X_{\text{star}}(t)$与平均位置$\overline{X_{\text{star}}}$之间的距离$Dis(t)$，为减小偶然性的影响，去掉距离$\overline{X_{\text{star}}}$最远的位置$X_{\text{far}}$和最近的位置$X_{\text{near}}$，求出中位数距离$Dis(t)$所对应的位置$X_{\text{med}}$。

step 3：求出其余18个位置$X_{\text{star}}(t)$与X_{med}之间差值的均方差，即star粒子的多样性值。通过star粒子的多样性值反映粒子在处于局部最优和过早收敛状态时的探索性和开发性。

3.4.2 实验结果及分析

表5给出了4种算法在7个陷阱测试函数下的路径多样性值以及排名。多样性测试结果表明，TLLA-APSO算法在7个陷阱函数中均取得最佳的路径多样性值，且领先CLPSO算法和OLPSO算法10个数量级以上。有外部元素引入的GLPSO算法在多样性值排名中，取得了7次第2，但其多性值较TLLA-APSO算法相差至少一个数量级。由于OLPSO和CLPSO算法没有外部元素的引入，它们的多样性明显不足。

为了直观展示粒子在陷阱中的寻优路径，图4给出所提算法与3种对比算法在Ackley陷阱函数下，star粒子的一维路径图。

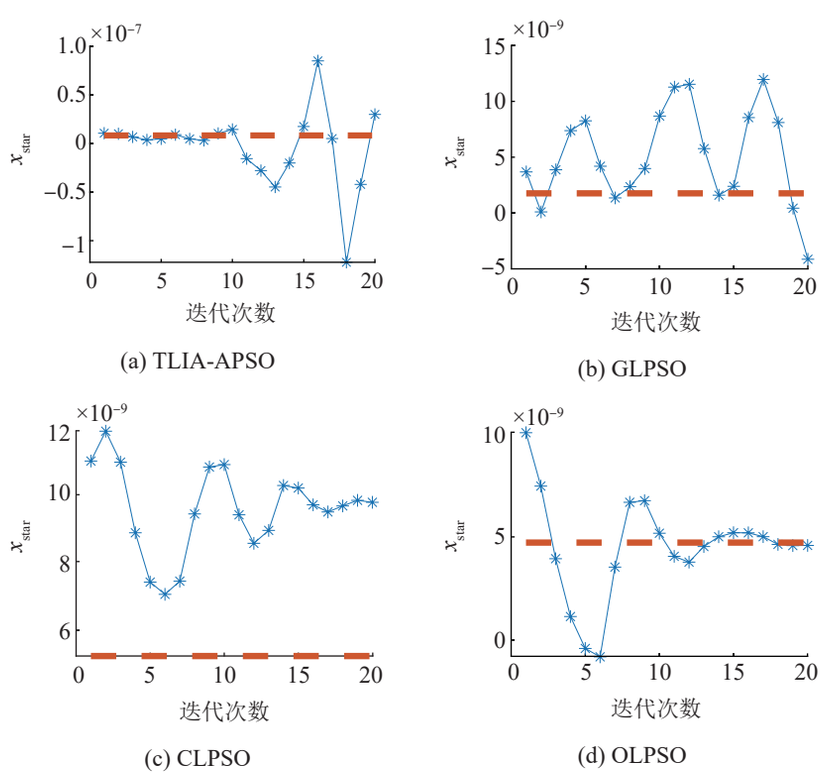
由图4可以看出，GLPSO、CLPSO和OLPSO算法的路径变化范围小，仅为10^{-9}，而多样值仅为10^{-18}，与TLLA-APSO的路径变化范围相差2个数量级。根本原因在于这3种算法无外部元素的引入，群体多样性较差，导致引导粒子对其他粒子形成束缚。而TLLA-APSO算法在面对陷阱时，路径变化范围的数量级从10^{-7}逐渐增加，使粒子重新恢复探索和开发的能力。一方面，陷阱标记引入了外部元素；另一方面，懒蚂蚁多样化粒子速度策略和惯性认知策略提高了群体的多样性。

所提算法利用每个粒子引入外部元素，而不是像GLPSO算法通过一定概率下的突变算子引入外部元素，大大提升了种群的多样性，即使在局部最优或过早收敛状态，粒子群仍然具有良好的寻优能力。实验结果表明，GLPSO、CLPSO和OLPSO算法在面对过早收敛问题时，粒子因速度严重减小几乎停滞，而TLLA-APSO算法依靠陷阱标记联合懒蚂蚁策略，重新激活粒子的速度，使其摆脱了过早收敛状态。

综上所述，结合陷阱实验结果和路径多样性分析，所提算法能更好地维持路径多样性，更有效地跳出陷阱。陷阱实验和路径多样性实验已验证所提算法能够通过自适应的方式较好地解决粒子群已陷入局部最优和过早收敛的问题。

表5 Table 5 路径多样性测试结果
Path diversity test results

函数	CLPSO		OLPSO		GLPSO		TLLA-APSO	
	多样性	排名	多样性	排名	多样性	排名	多样性	排名
Schwefel	4.71e−52	4	5.04e−47	3	3.72e−14	2	5.49e−12	1
Rosenbrock	2.47e−07	3	4.73e−09	4	1.98e+01	2	2.87e+02	1
Rastrigin	2.20e−19	3	5.02e−19	4	1.96e+00	2	1.81e+01	1
Quadric	1.05e−05	3	2.73e−05	4	4.40e+00	2	2.82e+01	1
Levy	3.75e−17	4	1.18e−16	3	1.15e+01	2	2.73e+03	1
Griewank	2.24e−16	4	1.92e−16	3	3.21e+00	2	2.03e+01	1
Ackley	1.00e−18	3	2.75e−18	4	1.06e+00	2	1.37e+01	1



(a) TLIA-APSO (b) GLPSO

(c) CLPSO (d) OLPSO

图4 star粒子的单维路径图
Fig. 4 One-dimension paths of star particles

DOI: 10.16182/j.issn1004731x.joss.23-0401

3.5 标准测试函数的寻优实验结果与分析

为验证所提算法的收敛速度和寻优精度，将TLLA-APSO算法与GPSO[4]、HPSO-TVAC[6]、FIPS[11]、CLPSO[11]、OLPSO[13]、GAPSO[15]、GLPSO[14]、GEPSO[10]和CAPSO[16]算法在9个测试函数中进行寻优结果对比。各算法的参数设置同陷阱实验。图5给出各算法在测试函数上的收敛曲线。可以看出，所提算法的陷阱标记联合懒蚂蚁策略使处于陷阱中的粒子依旧保持较好的探索性，并帮助粒子群更快地跳出局部最优。惯性认知策略使群体探索性能进一步增强，更有效地避免后续的局部最优，提高了算法的寻优效率。因此，在局部最优较多的测试函数f_4~f_8中，与寻优结果较好的对比算法相比，算法的收敛速度更快。陷阱评价指标的引入避免了粒子探索性增强对开发效率的影响。在单峰函数上，所提TLLA-APSO算法的收敛速度也明显优于大部分对比算法。

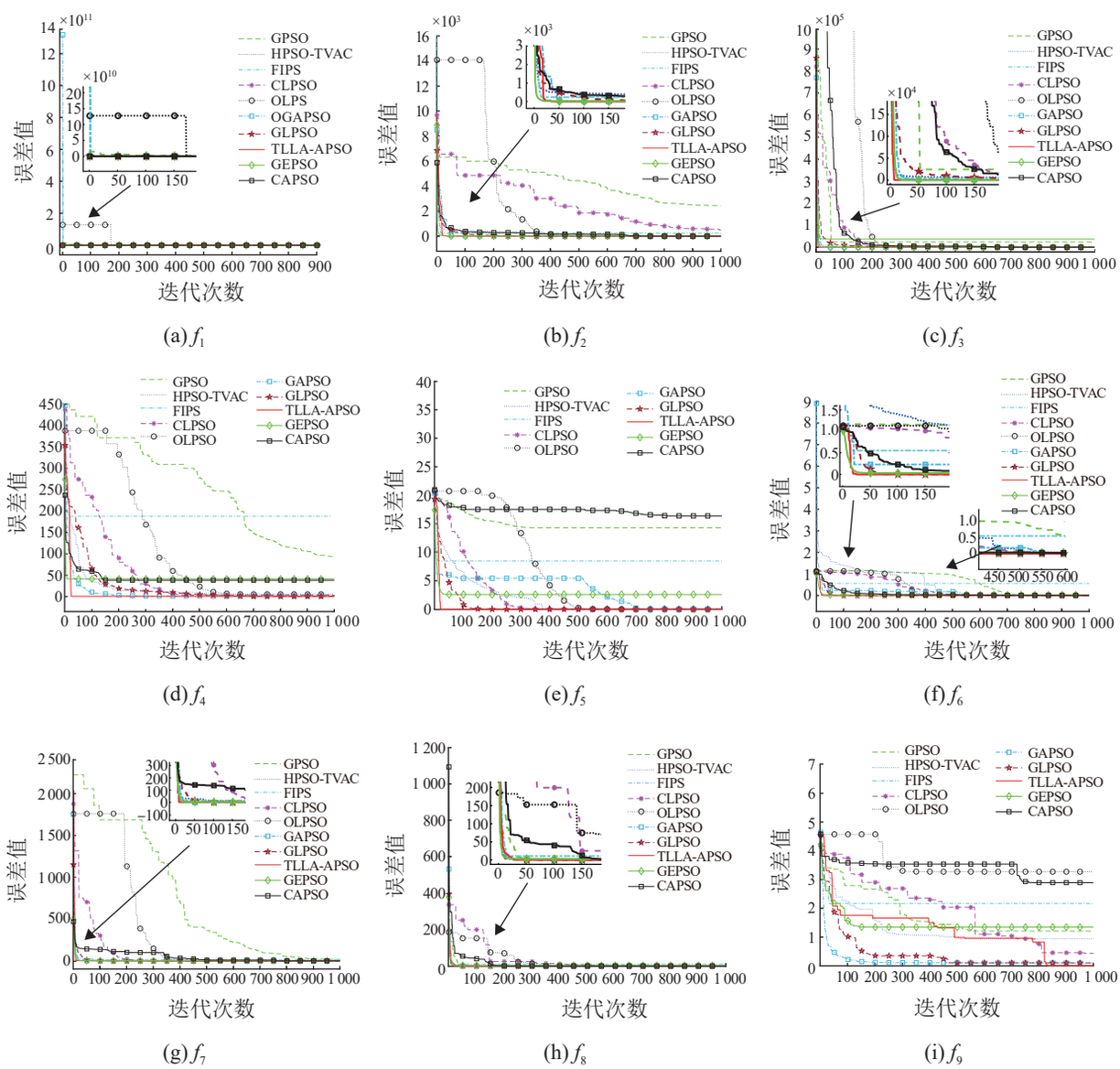
TLLA-APSO算法与对比算法实验结果由表6给出，包括30次实验结果的平均值、最优值、标准差以及精度排名。每个测试函数在各算法中的最优结果用粗体标记。

对于单峰函数f_1~f_3，求解精度是衡量算法性能最重要的标准，FIPS算法中邻域为5的粒子在单峰测试函数中，求解精度较GPSO算法并没有明显改善，这表明局部版的粒子群因信息传递缓慢而不适用于单峰函数。而GLPSO、GAPSO、CAPSO、GEPSO和TLLA-APSO算法较GPSO算法有不同程度的进步。其中，GLPSO算法中的选择算子避免了引导粒子的无效突变，在函数f_1上寻优误差保持在10^{-6}以内，精度较高。但对比算法中只有HPSO-TVAC、OLPSO和CAPSO算法在后续2个相对复杂的单峰函数中保持了前5的排名。其中，OLPSO算法中的OED机制在单峰函数中优势明显，均保持了前3的排名。而其他对比算法则出现了不同程度的波动，例如，GLPSO算法在f_2中排名第2，在f_3中却排名第7；GAPSO算法在f_3中排名第4，在f_2中却排名第6；GEPSO算法和CLPSO算法在f_2和f_3中均排名第6以后。这些算法引入了不同程度的随机性，虽然增加了种群的多样性，但对开发性造成了负面影响，导致算法的稳定性不足。所提算法在3个单峰函数中均取得最高求解精度，并且在其中2个单峰函数中稳定收敛到函数最优解0。函数f_3的全局最优解在抛物线形山谷中，解的优劣与粒子到全局最优解的距离并非正相关，寻优难度较大。CAPSO算法中的Logistic映射使初始种群具有良好的遍历性，根据图5的观察，该算法在初始阶段拥有更好的学习样本，因此在测试函数f_3中，最佳寻优误差控制在10^1以内。而TLLA-APSO算法在该函数上的寻优精度远超各对比算法，最佳的寻优误差达到了10^{-6}以内，展现出较好的寻优精度和稳定性。

在多峰函数f_4~f_9中，所提算法在对6个多峰函数的寻优中取得了4次最佳，在对函数f_8的寻优中排名第2，但与第1名的寻优结果1.02e–01仍处于同一数量级。虽然对函数f_7的寻优仅排名第3，但寻优误差依然控制在10^{-6}以内。这是因为TLLA-APSO算法增强了种群的探索能力和开发能力，但在一定程度上减缓了粒子群的收敛速度。当迭代结束时，粒子群的全局最优解仍处于持续更新的状态，因此未能展现出TLLA-APSO算法在该函数上的最佳寻优结果。在单峰函数中表现较好的HPSO-TVAC、OLPSO和CAPSO算法在多峰函数中都有至少3次排名在第5以后，并且都不同程度地出现了陷入局部最优和过早收敛的问题。而GEPSO算法在多峰函数中并未表现出引入随机项后的多样性优势，却因为随机性的引入导致粒子发散，搜索有效性降低。因此，寻优效果较差。综合平均寻优结果及排名分析，CLPSO、GAPSO和GLPSO算法在多峰函数上的寻优结果均优于它们在单峰函数上的寻优结果，体现引入随机性后的探索优势。对比GLPSO和GAPSO算法在函数f_5的结果，对外部元素开放性更强的GAPSO算法的寻优结果理论上应该优于GLPSO算法，然而，

由于GAPSO算法的收敛性不足，导致其整体性能弱于GLPSO算法，两者的结果相差11个数量级。因此，对引入随机性的适度约束能够很好地控制其所带来的负面影响，GLPSO算法中的选择算子很好地发挥了该作用，避免了粒子发散对开发性造成的负面影响。在算法复杂度方面，所提算法相比PSO算法增加了根据所标记的陷阱给予粒子速度增量的步骤，因此复杂度方面略有增加。但是，算法能够自适应地根据需要分配运行资源，在获得更优解的情况下，总体运行时间不会明显增加。

综上所述，相比调整参数和结构以及引入新学习策略的对比算法，在多峰函数中无法解决粒子群陷入局部最优和过早收敛的问题，以及引入随机性的各对比算法因其负面影响在单峰函数和多峰函数中所展现出的性能差异，TLLA-APSO算法较好地解决了粒子群陷入局部最优和过早收敛的问题，并且在单峰和多峰函数上均获得了较高的寻优精度和较快的收敛速度，体现出所提算法性能的优越性和稳定性。



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

(g) f_7 (h) f_8 (i) f_9

图5 各测试函数误差迭代曲线

Fig. 5 Error iterative curve of optimal fitness for each test function

https://dc-china-simulation.researchcommons.org/journal/vol36/iss7/11
DOI: 10.16182/j.issn1004731x.joss.23-0401

表6 各算法在9个测试函数上的寻优性能

Table 6 Optimization performance of each algorithm on 9 test functions

函数		PSO	HPSO-TVAC	FIPS	CLPSO	OLPSO	GEPSO	GAPSO	GLPSO	CAPSO	TLLA-APSO
f_1	Mean	5.66e+01	2.09e−08	1.66e+01	3.75e−10	2.18e−12	4.16e−01	3.97e−07	8.00e−20	1.98e−02	**0**
	Best	2.00e+01	5.93e−10	1.40e+01	1.45e−10	9.96e−13	1.87e−02	6.36e−10	1.56e−20	1.50e−03	**0**
	Std	2.41e+01	2.64e−08	1.87e+01	1.63e−10	8.84e−13	5.17e−01	8.99e−07	4.35e−20	2.33e−02	**0**
	Rank	10	5	9	4	3	8	6	2	7	**1**
f_2	Mean	3.35e+03	8.16e−01	4.22e+02	3.41e+02	5.36e−02	3.01e+00	1.70e+00	3.72e−04	6.28e−02	**0**
	Best	2.01e+03	1.18e−01	2.29e+02	1.83e+02	1.60e−03	7.97e−01	3.23e−01	3.61e−05	1.60e−03	**0**
	Std	8.20e+02	8.42e−01	1.53e+02	1.21e+02	1.22e-01	2.63e+00	6.21e−01	2.87e−04	6.11e−02	**0**
	Rank	10	5	9	8	3	7	6	2	4	**1**
f_3	Mean	1.72e+05	8.78e+01	4.61e+03	5.01e+02	1.56e+01	9.28e+01	4.59e+01	3.11e+02	2.94e+01	**9.76e+00**
	Best	2.43e+04	1.75e+01	2.79e+03	5.10e+01	1.50e+00	2.02e+01	1.04e+01	2.97e+01	1.65e+01	**9.71e−07**
	Std	8.86e+04	8.89e+01	1.27e+03	2.89e+02	3.15e+01	4.21e+01	2.24e+01	2.41e+02	2.05e+01	**1.30e+01**
	Rank	10	5	9	8	2	6	4	7	3	**1**
f_4	Mean	1.33e+02	3.92e+00	2.00e+02	3.78e+00	7.71e+00	2.40e+01	4.52e−12	2.71e-01	4.81e+01	**0**
	Best	8.29e+01	5.32e−15	1.73e+02	9.95e−01	2.98e+00	1.99e+01	3.55e−15	**0**	1.98e+01	**0**
	Std	2.93e+01	3.13e+00	1.54e+01	1.69e+00	3.59e+00	5.53e+00	1.47e-11	4.53e-01	1.85e+01	**0**
	Rank	9	5	10	4	6	7	2	3	8	**1**
f_5	Mean	1.03e+01	2.56e−11	9.15e+00	5.67e−08	4.96e−10	2.54e+00	1.96e−04	6.92e−15	7.73e−04	**0**
	Best	1.34e+00	1.35e−13	7.96e+00	2.34e−08	1.94e−10	1.65e+00	5.02e−07	3.55e−15	6.52e−06	**0**
	Std	8.02e+00	5.12e−11	1.04e+00	2.65e−08	2.16e−10	6.01e−01	7.82e−04	7.94e−16	1.10e−03	**0**
	Rank	10	3	9	5	4	8	6	2	7	**1**
f_6	Mean	3.73e−02	2.83e−02	4.75e−01	4.58e−02	7.30e−03	7.01e−03	1.54e−12	3.70e−03	2.38e−02	**0**
	Best	1.69e−07	1.11e−16	3.70e−01	2.51e−05	**0**	6.21e−04	2.24e−14	**0**	8.56e−09	**0**
	Std	4.68e−02	3.60e−02	7.54e−02	2.02e−01	6.90e−03	3.90e−03	3.75e−12	5.60e−03	1.54e−02	**0**
	Rank	8	7	10	9	5	4	2	3	6	**1**
f_7	Mean	4.98e+01	2.15e−01	3.39e+01	4.54e−02	2.08e−18	3.23e+00	1.25e−04	**1.49e−32**	5.16e+00	2.06e−07
	Best	4.22e−01	6.56e−16	1.64e+01	1.33e−08	3.21e−19	1.88e+00	2.14e−15	**1.49e−32**	2.80e−03	7.83e−09
	Std	4.83e+01	2.69e−01	7.42e+00	1.39e−01	2.00e−18	1.88e+00	4.15e-04	**5.60e−48**	4.39e+00	1.66e−07
	Rank	10	6	9	5	2	7	4	**1**	8	3
f_8	Mean	2.21e−01	4.55e−01	7.82e+00	3.94e−01	1.41e+01	1.22e+00	3.25e−01	**1.02e−01**	5.56e−01	1.77e−01
	Best	1.76e−01	2.03e−01	6.01e+00	3.02e−01	1.20e−01	3.19e−01	1.16e−01	**4.11e−02**	1.42e−01	1.57e−01
	Std	3.65e−02	1.69e−01	1.90e+00	7.90e−02	3.65e+01	1.22e+00	1.42e−01	4.38e−02	1.85e−01	**1.62e−02**
	Rank	3	6	9	5	10	8	4	**1**	7	2
f_9	Mean	1.36e+00	8.39e−01	2.70e+00	5.76e−01	4.55e+00	2.00e+00	1.27e−01	2.78e−01	2.19e+00	**1.58e−14**
	Best	2.07e−01	1.52e−01	2.34e+00	3.90e−01	4.24e+00	7.02e−01	3.89e−02	9.72e−02	1.73e+00	**0**
	Std	8.60e−01	5.82e−01	2.11e−01	1.71e−01	1.53e−01	6.41e-01	1.17e−01	3.52e−01	5.64e−01	**5.49e−14**
	Rank	6	5	9	4	10	7	2	3	8	**1**

4 结论

本文提出了一种陷阱标记联合懒蚂蚁的自适应粒子群优化算法。算法中的陷阱标记联合懒蚂蚁策略有效地解决了粒子群已陷入局部最优和过早收敛的问题，使种群恢复了探索性和开发性。惯性认知策略引入历史位置使粒子群的探索性得到进一步提升，更加有效地避免了新的局部最优。

陷阱实验和路径多样性实验验证了所提算法帮助粒子群摆脱困境和指导粒子搜索的有效性。标准测试函数的仿真实验结果表明所提算法具有良好的收敛速度和寻优精度。后续拟将所提算法应用于城市给水管网的优化设计中，以进一步验证算法在实际工程优化中的适用性。

参考文献：

[1] Fatma A Hashim, Essam H Houssein, Kashif Hussain, et al. Honey Badger Algorithm: New Metaheuristic Algorithm for Solving Optimization Problems[J]. Mathematics and Computers in Simulation, 2022, 192: 84-110.

[2] Wang Shengliang, Liu Genyou, Gao Ming, et al. Heterogeneous Comprehensive Learning and Dynamic Multi-swarm Particle Swarm Optimizer with Two Mutation Operators[J]. Information Sciences, 2020, 540: 175-201.

[3] Shami T M, Ayman A El-Saleh, Alswaitti M, et al. Particle Swarm Optimization: A Comprehensive Survey [J]. IEEE Access, 2022, 10: 10031-10061.

[4] Sengupta S, Basak S, Peters R A II. Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives[J]. Machine Learning and Knowledge Extraction, 2019, 1 (1): 157-191.

[5] Vikash Yadav, Indresh Kumar Gupta. Modified Adaptive Inertia Weight Particle Swarm Optimisation for Data Clustering[J]. International Journal of Innovative Computing and Applications, 2022, 13(1): 34-40.

[6] 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.

[7] Adam P Piotrowski, Jaroslaw J Napiorkowski, Agnieszka E Piotrowska. Population Size in Particle Swarm Optimization[J]. Swarm and Evolutionary Computation, 2020, 58: 100718.

[8] Li Wei, Meng Xiang, Huang Ying, et al. Multipopulation Cooperative Particle Swarm Optimization with a Mixed Mutation Strategy[J]. Information Sciences, 2020, 529: 179-196.

[9] Aasam Abdul Karim, Nor Ashidi Mat Isa, Wei Hong Lim. Hovering Swarm Particle Swarm Optimization[J]. IEEE Access, 2021, 9: 115719-115749.

[10] Davoud Sedighizadeh, Masehian E, Mostafa Sedighizadeh, et al. GEPSO: A New Generalized Particle Swarm Optimization Algorithm[J]. Mathematics and Computers in Simulation, 2021, 179: 194-212.

[11] 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.

[12] Mendes R, Kennedy J, Neves J. The Fully Informed Particle Swarm: Simpler, Maybe Better[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 204-210.

[13] Zhan Zhihui, Zhang Jun, Li Yun, et al. Orthogonal Learning Particle Swarm Optimization[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(6): 832-847.

[14] Gong Yuejiao, Li Jingjing, Zhou Yicong, et al. Genetic Learning Particle Swarm Optimization[J]. IEEE Transactions on Cybernetics, 2016, 46(10): 2277-2290.

[15] Premalatha K, Natarajan A M. Hybrid PSO and GA for Global Maximization[J]. International Journal of Open Problems in Computer Science and Mathematics, 2009, 2 (4): 597-608.

[16] Duan Youxiang, Chen Ning, Chang Lunjie, et al. CAPSO: Chaos Adaptive Particle Swarm Optimization Algorithm[J]. IEEE Access, 2022, 10: 29393-29405.

[17] Imirzian N, Zhang Yizhe, Kurze C, et al. Automated Tracking and Analysis of Ant Trajectories Shows Variation in Forager Exploration[J]. Scientific Reports, 2019, 9(1): 13246.

[18] Tseng H Y, Chu P H, Lu Haochun, et al. Easy Particle Swarm Optimization for Nonlinear Constrained Optimization Problems[J]. IEEE Access, 2021, 9: 124757-124767.

DOI: 10.16182/j.issn1004731x.joss.23-0401