Journal of System Simulation

Volume 36 | Issue 4 Article 6

4-15-2024

Mixed-variable Particle Swarm Optimization Algorithm Based on Competitive Coevolution

Hu Zhang
Department of System Confrontation and Intelligent Information System, the Third Research Institute of CASIC, Beijing 100074, China, xzhanghu@126.com

Heng Zhang
School of Computer Science, Wuhan University, Wuhan 430072, China

Zilu Huang
School of Computer Science, Wuhan University, Wuhan 430072, China

Zhe Wang
School of Computer Science, Wuhan University, Wuhan 430072, China

See next page for additional authors

Mixed-variable Particle Swarm Optimization Algorithm Based on Competitive Coevolution

Abstract

Abstract: For the current algorithm, it is difficult to obtain the available solution due to the irregularity of problem decision space caused by the numerous mixed variable optimization problems during real industrial applications. The coevolution strategy is introduced and a mixed variable particle swarm optimization algorithm(CCPSO) based on competitive coevolution is proposed. The search direction adjustment mechanism based on tolerance is designed to judge the evolution state of particles, adaptively adjust the search direction of particles, avoid falling into local optimum, and balance the convergence and diversity of the population. The learning object generation mechanism is adopted for each particle to generate new learning objects when particle evolution stagnation is detected to promote the evolution of particles and improve the diversity of the population. The prediction strategy based on competitive learning is applied to select the appropriate learning objects for particles, which makes full use of the learning potential of new and old learning objects and ensures the convergence speed of the algorithm. Experimental results show that, CCPSO can obtain the better results compared with the other main mixed variable optimization algorithms.

Keywords

mixed variable optimization, coevolution strategy, evolutionary algorithm, particle swarm

Authors

Hu Zhang, Heng Zhang, Zilu Huang, Zhe Wang, Qingpo Fu, Jin Peng, and Feng Wang

Recommended Citation

Zhang Hu, Zhang Heng, Huang Zilu, et al. Mixed-variable Particle Swarm Optimization Algorithm Based on Competitive Coevolution[J]. Journal of System Simulation, 2024, 36(4): 844-858.

第 36 卷第 4 期 系统仿真学报© Vol. 36 No. 4
2024 年 4 月 Journal of System Simulation Apr. 2024

基于竞争式协同进化的混合变量粒子群优化算法

张虎[1]，张衡[2]，黄子路[2]，王喆[2]，付青坡[2]，彭瑾[3]，王峰[2*]
(1. 中国航天科工集团有限公司第三研究院 体系对抗与智能信息系统总体部，北京 100074；2. 武汉大学 计算机学院，武汉 430072；
3. 海军装备部北京局驻北京地区第三军事代表室，北京 100074)

摘要：现实工业生产应用中存在大量的混合变量优化问题，这类问题的决策变量既包含连续变量，又包含离散变量。由于决策变量为混合类型，导致问题的决策空间变得不规则，采用已有的方法很难进行有效求解。*引入协同进化策略，提出一种基于竞争式协同进化的混合变量粒子群优化算法*(competitive coevolution based PSO，CCPSO)。*设计基于容忍度的搜索方向调整机制来判断粒子的进化状态，从而自适应地调整粒子的搜索方向，避免陷入局部最优，平衡了种群的收敛性和多样性；引入基于竞争式协同进化的学习对象生成机制，在检测到粒子进化停滞时为每个粒子生成新的学习对象，从而推动粒子的进一步搜索，提高了种群的多样性；采用基于竞争学习的预测策略为粒子选择合适的学习对象，充分利用了新旧学习对象的学习潜力，保证了算法的收敛速度。*实验结果表明：相比其他主流的混合变量优化算法，CCPSO 可以获得更优的结果。

关键词：混合变量优化；协同策略；进化算法；粒子群
中图分类号：TP273 文献标志码：A 文章编号：1004-731X(2024)04-0844-15
DOI: 10.16182/j.issn1004731x.joss.22-1466
引用格式: 张虎, 张衡, 黄子路, 等. 基于竞争式协同进化的混合变量粒子群优化算法[J]. 系统仿真学报, 2024, 36(4): 844-858.
Reference format: Zhang Hu, Zhang Heng, Huang Zilu, et al. Mixed-variable Particle Swarm Optimization Algorithm Based on Competitive Coevolution[J]. Journal of System Simulation, 2024, 36(4): 844-858.

Mixed-variable Particle Swarm Optimization Algorithm Based on Competitive Coevolution

Zhang Hu[1], *Zhang Heng*[2], *Huang Zilu*[2], *Wang Zhe*[2], *Fu Qingpo*[2], *Peng Jin*[3], *Wang Feng*[2*]
(1. Department of System Confrontation and Intelligent Information System, the Third Research Institute of CASIC, Beijing 100074, China;
2. School of Computer Science, Wuhan University, Wuhan 430072, China; 3. The Third Military Representative Office of the Beijing Bureau of the Naval Armament Department in the Beijing Area, Beijing 100074, China)

Abstract: For the current algorithm, it is difficult to obtain the available solution due to the irregularity of problem decision space caused by the numerous mixed variable optimization problems during real industrial applications. The coevolution strategy is introduced and a mixed variable particle swarm optimization algorithm(CCPSO) based on competitive coevolution is proposed. *The search direction adjustment mechanism based on tolerance is designed to judge the evolution state of particles, adaptively adjust the search direction of particles, avoid falling into local optimum, and balance the convergence and diversity of the population. The learning object generation mechanism is adopted for each particle to generate new learning objects when particle evolution stagnation is detected to promote the evolution of*

收稿日期：2022-12-07 修回日期：2023-03-13
基金项目：国家自然科学基金(62173258，61773296)；国防基础科研项目(JCKY2019204A007)
第一作者：张虎(1986-)，男，研究员，博士，研究方向为作战概念设计仿真、体系需求开发、体系设计仿真。E-mail：jxzhanghu@126.com
通讯作者：王峰(1981-)，女，教授，博士，研究方向为智能优化、多目标优化、机器学习。E-mail：fengwang@whu.edu.cn

particles and improve the diversity of the population. The prediction strategy based on competitive learning is applied to select the appropriate learning objects for particles, which makes full use of the learning potential of new and old learning objects and ensures the convergence speed of the algorithm. Experimental results show that, CCPSO can obtain the better results compared with the other main mixed variable optimization algorithms.

Keywords: mixed variable optimization; coevolution strategy; evolutionary algorithm; particle swarm

0 引言

随着我国社会制造服务和物流运输等行业的快速发展，越来越多的设计、管理和调度等类型的优化问题涌现出来。在这些优化问题中，决策变量通常同时包含连续型变量和离散型变量，这种同时包含连续变量和离散变量的优化问题被称为混合变量优化问题(mixed-variable optimization problems, MVOPs)。

目前存在大量求解MVOPs的进化算法，根据变量处理方法，算法可分为3类：①基于松弛方法的算法。这类算法将离散变量松弛转化为连续变量[1]，在解的评价过程中，连续变量将被转化为离散值以适合函数评估。②基于离散化方法的算法。这类算法将连续变量离散化为离散变量，然后使用离散变量优化算子对问题求解[2]。然而，受限于二进制字符串的长度，这类算法求解问题时的准确度不能得到保证。③基于混合算子的算法。这类算法同时采用连续变量重组算子和离散变量重组算子生成包含混合变量的解[3]，因此，可以灵活地利用不同重组算子来生产高质量的解。然而，不同算子之间性能的不协调可能影响算法搜索效率[4]。

PSO是一种被广泛研究的进化算法，已被用于求解各类复杂优化问题。现有的PSO算法主要聚焦在连续变量优化PSO算法和离散变量优化PSO算法。

连续变量优化PSO算法主要有3类：①基于参数修改的PSO。文献[5]结合强化学习方法，将目标函数较好的粒子数量以及粒子的间距作为状态，将惯性与加速因子作为动作，自适应地调整参数，提出了QLPSO(Q-learning-based PSO)。②基于混合方法的PSO。文献[6]将PSO与细菌觅食算法(bacterial foraging optimization, BFO)进行混合，并应用于熔融沉积成形法中。文献[7]将粒子群算法与GWO进行混合，并应用于皮革嵌套问题中。③基于学习策略的PSO。文献[8]考虑多峰问题，提出了一种基于综合学习的粒子群算法CLPSO。文献[9]在CLPSO的基础上融合了局部搜索策略，取得了更好的结果。文献[10]也提出了一种利用社会学习的SL-PSO，提高了算法在解决大规模优化问题时的能力。

离散变量优化PSO算法可以被分为3类：①基于二值化方法的PSO。②基于变量类型转换的PSO，这类算法不需要重新设计重组算子，算法采用连续版本PSO中的重组算子生成解。③基于集合理论的PSO。如文献[11]针对虚拟网络嵌入问题，采用PSO作为优化器，并基于历史信息进行学习。

目前的PSO算法主要适用于求解连续变量优化问题或离散变量优化问题，针对混合变量优化问题的求解算法还比较匮乏。由于混合变量使问题空间变得非常不规则，问题空间的复杂度随着规模不断提高，如何设计合适的算法来求解MVOPs成为一个挑战性的难题。

根据协同进化策略中个体或种群之间相互作用的类型，协同进化策略可以被分为合作式协同进化[12]和竞争式协同进化[13]。竞争式协同进化策略通常采用多个种群或多个个体模拟寄生者和宿主的竞争进化过程。目前，竞争式协同进化策略的研究通常包括适应度评估策略和个体竞争策略。

DOI: 10.16182/j.issn1004731x.joss.22-1466

例如，基于双种群进化策略使用信息共享机制实现协同进化[14]；通过将进化分为3个阶段来平衡多样性与收敛性[15]；将目标空间划分为多个子空间设计空间距离算子来评估其优化难度，并动态赋值目标向量，将计算资源引导到更难优化的子空间[16]。这些研究表明，竞争式协同进化策略能够有效提升进化算法的性能，帮助算法求解更加复杂的优化问题。

为更有效求解MVOPs，本文提出基于竞争式协同进化策略的混合变量粒子群优化算法(competitive coevolution based PSO, CCPSO)。

1 基于竞争式协同进化的混合变量粒子群优化算法

经典的PSO中的粒子主要通过学习全局最优粒子进行更新，这使得种群在全局最优粒子陷入局部最优时进化停滞，算法将过早收敛。尤其是当优化问题变得复杂时，PSO过早收敛的现象将越发频繁。混合变量优化问题通常具有复杂的搜索空间，包括多峰、不连续等特征，经典的PSO算法不能有效地解决这类问题。因此，为了提高算法求解混合变量优化问题的性能，引入协同进化策略提出了基于竞争式协同进化的混合变量粒子群优化算法CCPSO。

为了处理混合变量，CCPSO采用了松弛方法将离散变量松弛为连续变量。在对解评估时，算法将采用舍入法把连续值转化为离散值。因此，CCPSO将采用连续变量重组算子来处理混合变量。为了提高重组算子的性能，CCPSO采用基于竞争协同进化的学习对象生成机制为每一个粒子生成独特的高质量的学习对象。由于不同的粒子搜索方向不尽相同，采用上述机制可以有效提升算法的探索能力。为了保证粒子的学习对象的质量，CCPSO采用竞争学习的预测策略为粒子选择优秀的学习对象，有能力提升粒子的学习对象将被选择，因此，整个种群的勘探能力将得到提升。为了平衡算法的探索能力和勘探能力，CCPSO采用基于容忍度的搜索方向调整机制，粒子将自适应地判断自身进化状态，每个粒子在充分利用学习对象进化潜力的同时能够适时地调整学习对象从而避免陷入局部最优。

1.1 基于容忍度的搜索方向调整机制

在经典的PSO算法中，种群中的粒子通过学习全局最优粒子更新速度、位置向量，完成对问题空间的搜索。然而，当全局最优粒子陷入局部最优时，种群粒子也有可能陷入局部最优，导致算法过早收敛。图1在一个维度上展示了上述粒子的进化过程，对于一个最小化问题，红色的曲线表示了目标函数的变化趋势，星状的全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$目前位于一个局部最优点上。假设种群的学习对象为全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$，很明显，粒子$\boldsymbol{X}_1$和粒子$\boldsymbol{X}_2$将朝着全局最优粒子所在的方向搜索，并可能在几次搜索后与全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$一同陷入局部最优。在陷入局部最优之前，粒子$\boldsymbol{X}_1$的个体最优$\boldsymbol{P}_1$会被不断更新，即$\boldsymbol{P}_1$的适应度值$f(\boldsymbol{P}_1)$不断增加。然而，在粒子$\boldsymbol{X}_1$陷入局部最优之后，其个体最优$\boldsymbol{P}_1$将进化停滞。粒子的个体最优进化停滞可以用式(1)表示：

$$f(\boldsymbol{P}_i)^{t+1}-f(\boldsymbol{P}_i)^{t}=0 \tag{1}$$

式中：t为迭代次数。

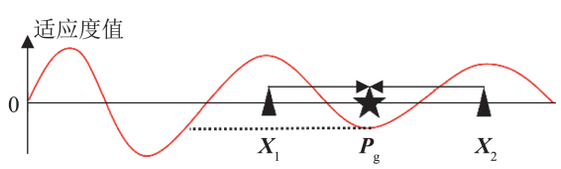


图1 全局最优粒子陷入局部最优

Fig. 1 Global optimal particle trapped in local optimal

当粒子的个体最优在之后的迭代中都不能得到提高时，粒子很有可能已经陷入了局部最优。

为了避免种群陷入局部最优，算法需要在式(1)满足时及时调整粒子的搜索方向。然而，式(1)满足只是粒子陷入局部最优的充分不必要条件，

算法不能仅通过式(1)是否成立而决定是否调整粒子的搜索方向，因为可能会出现如图2显示的情况。在第t次迭代时，种群的全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$位于靠近全局最优的位置，粒子$\boldsymbol{X}_1$和粒子$\boldsymbol{X}_2$将朝着全局最优粒子所在的方向搜索。虽然在第$t+1$次迭代时粒子的个体最优不会得到提高，即式(4)成立，但是在这之后的搜索中，全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$有潜力带领其他粒子进化，整个种群极有希望搜索到问题的全局最优点。

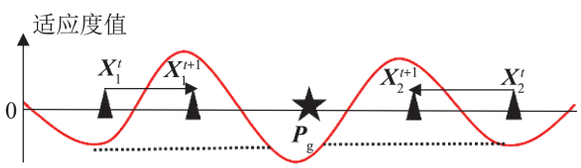


图2　粒子搜索到全局最优
Fig. 2　Particle search to global optimization

为了避免粒子陷入局部最优并且充分地利用粒子的学习对象(如全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$)的进化潜力，本文提出了基于容忍度的搜索方向调整机制(tolerance based search direction adjustment mechanism, TSDM)。TSDM能够帮助粒子判断当前的进化状态，并在合适的时机调整粒子的搜索方向以避免粒子陷入局部最优。

基于容忍度的搜索方向调整机制定义粒子的容忍度为粒子进化停滞的次数，表示为T。容忍度T增加1则代表粒子的个体最优在上一轮的更新中没有提高。显然，T值越大，粒子陷入局部最优的概率也将越大。为了避免粒子陷入局部最优，TSDM将按概率调整粒子搜索方向，对于第i个粒子，调整搜索方向的概率为

$$Q_i^{\mathrm{adjust}}=\frac{\exp(T_i)-1}{\exp(5)-1} \tag{2}$$

Q_i^{adjust}将在粒子每次更新后计算，Q_i^{adjust}随粒子X_i的容忍度T_i变化，如图3所示。

当Q_i^{adjust}大于均匀分布于[0,1]的随机数r时，粒子$\boldsymbol{X}_i$将停止向其当前的学习对象$\boldsymbol{L}_i$学习。TSDM的具体过程如算法1所示。

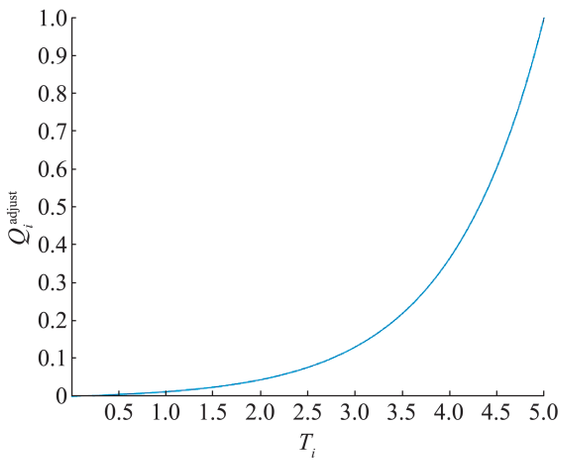


图3　搜索方向调整概率变化曲线
Fig. 3　Changing curve of search direction adjustment probability

算法1　基于容忍度的搜索方向调整机制(TSDM)

输入：第i个粒子的容忍度值T_i和个体最优$\boldsymbol{P}_i$

1：if $f(\boldsymbol{P}_i^{t+1})-f(\boldsymbol{P}_i^t)==0$ then

2：　　$T_i \leftarrow T_i+1$

3：end if

4：生成在[0 , 1]区间内随机分布的随机数r

5：根据式(2)更新学习对象调整概率Q_i^{adjust}

6：if $Q_i^{\mathrm{adjust}}>r$ then

7：　第i个粒子停止向其学习对象$\boldsymbol{L}_i$学习

8：end if

很明显，粒子停止进化的次数增加，算法调整其搜索方向的概率也将增加。在粒子$\boldsymbol{X}_i$前几次进化停止时，粒子$\boldsymbol{X}_i$调整搜索方向的概率Q_i^{adjust}很小，粒子$\boldsymbol{X}_i$将很有可能继续沿着其学习对象进行搜索，学习对象L_i的进化潜力将被充分利用。当粒子$\boldsymbol{X}_i$的容忍度不断增加，粒子$\boldsymbol{X}_i$调整搜索方向的概率Q_i^{adjust}将指数地增加。此时，粒子$\boldsymbol{X}_i$将很有可能调整自己的学习对象，改变搜索方向，避免自身陷入局部最优。总之，TSDM在保证粒子不陷入局部最优的前提下充分利用了粒子的学习对象的进化潜力，平衡了收敛性和多样性。

1.2 基于竞争协同进化的学习对象生成机制

在经典的PSO中，每一个粒子都通过学习全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$更新自己的速度向量和位置向量。然而，当全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$陷入局部最优时，整个种群粒子也将陷入局部最优从而导致算法过早收敛。为了提高种群的多样性，避免算法过早收敛，CCPSO分别为每一个粒子设置了适合其进化的学习对象。因此，粒子的速度更新规则修改为

$$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+cr(\boldsymbol{L}_i(t)-\boldsymbol{X}_i(t))+cr(\boldsymbol{P}_{\mathrm{g}}(t)-\boldsymbol{X}_i(t)) \tag{3}$$

式中：$\boldsymbol{v}_i(t)$和$\boldsymbol{X}_i(t)$为粒子$\boldsymbol{X}_i$在第t次迭代时的速度向量和位置向量；r为在[0,1]内分布的随机数；c为加速因子；w为惯性权重；$\boldsymbol{L}_i(t)$为第t次迭代时粒子的学习对象；$\boldsymbol{P}_{\mathrm{g}}(t)$为全局最优粒子。学习对象$\boldsymbol{L}_i$对粒子的搜索效率有着至关重要的影响，当TSDM机制生效时，即旧的学习对象$\boldsymbol{L}_i$不能带领粒子进化时，将生成新的学习对象$\boldsymbol{L}_i^{\mathrm{new}}$，避免粒子陷入局部最优。

图4在一个维度上展示了粒子的进化过程，对于一个最小化问题，红色的曲线表示了目标函数的变化趋势，星状的全局最优粒子$\boldsymbol{P}_{\mathrm{g}}$目前位于一个局部最优点上。

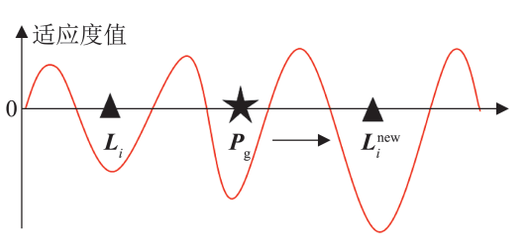


图4 学习对象生成机制示例

Fig. 4 Example of learning object generation mechanism

学习对象的生成有多种方法。一种是可以在搜索空间中随机地生成。这种方式可以很轻易地带领粒子跳出当前的局部最优。然而，这种随机生成的学习对象的质量无法保证，尤其是在包含连续变量和离散变量优化问题的搜索空间中，新生成的学习对象很有可能带领粒子陷入到另一个局部最优。另一种是当前种群的搜索信息可以被用来生成新的学习对象。因为每个粒子都会将搜索到的最优信息保存在其个体最优中，个体最优在多个维度的结构是较为出色的。

为了生成出色的学习对象，本文提出了基于竞争式协同进化的学习对象生成机制(competitive coevolution based learning object generation mechanism, CCLOG)。种群当前的搜索信息可以被借鉴用来生成适合当前粒子的学习对象，为了保证粒子与种群协同进化，CCLOG利用种群当前的搜索信息构成学习对象。为了提高生成的学习对象的质量，CCLOG以竞争的方式选择种群中的搜索信息并生成新的学习对象。CCLOG的过程如算法2所示。

算法2 基于竞争进化的学习对象生成机制(CCLOG)

输入：算法当前迭代次数t，第i个粒子的学习对象$\boldsymbol{L}_i$

输出：新生成的学习对象$\boldsymbol{L}_i^{\mathrm{new}}$

1：if $t==0$ then

2：　$\boldsymbol{L}_i^{\mathrm{new}} \leftarrow \boldsymbol{P}_i$

3：else

4：　for 粒子的每个维度j，$1 \leqslant j \leqslant D$，do

5：　　if $Q_i^{\mathrm{comp}} > r$ then

6：　　随机选取2个粒子的个体最优$\boldsymbol{P}_{\mathrm{k}}$和$\boldsymbol{P}_{\mathrm{h}}$

7：　　if $f(\boldsymbol{P}_{\mathrm{k}}) < f(\boldsymbol{P}_{\mathrm{h}})$ then

8：　　$L_i^{\mathrm{new},j} \leftarrow P_{\mathrm{k}}^j + G(\sigma^j)$

9：　　else

10：　　$L_i^{\mathrm{new},j} \leftarrow P_{\mathrm{h}}^j + G(\sigma^j)$

11：　　end if

12：　else

13：　　$L_i^{\mathrm{new},j} \leftarrow P_i^j$

14：　end if

15：　end for

16：end if

从算法的第1~3行表示算法初始化时，粒子

的学习对象$\boldsymbol{L}_i$被初始化为粒子的个体最优粒子$\boldsymbol{P}_i$。随着算法迭代，当TSDM决定调整粒子的搜索方向时，新的学习对象被生成并替换旧的学习对象。算法的第4~15行表示整个种群的搜索信息将被随机地选取用来生成新的学习对象$\boldsymbol{L}_i^{\text{new}}$。其中，算法的第5~11行表示当竞争概率$Q_i^{\text{comp}}$大于一个在[0,1]内随机分布的随机数$r$时，算法将按照二元锦标赛的规则竞争式地选择其他粒子的个体最优来构成学习对象$\boldsymbol{L}_i$。在本文算法中Q_i^{comp}设置为0.09。$G(\sigma^j)$为按照σ^j为标准差的高斯分布生成的随机偏置，σ^j为描述当前维度j中所有个体分布状况的标准差。

$$\sigma^j=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(P_i^j-m^j)} \tag{4}$$

式中：m^j为所有个体最优在第j维度的均值；P_i^j为第i个粒子的第j维度的值。为保证$\boldsymbol{L}_i^{\text{new}}$的质量，$\boldsymbol{L}_i^{\text{new}}$也保留了一部分粒子$\boldsymbol{X}_i$的个体最优$\boldsymbol{P}_i$的结构，如算法的12行到第14行所示。

1.3 基于竞争学习的预测策略

CCLOG机制生成的新的学习对象能够帮助粒子跳出局部最优。然而，由于生成的新的学习对象的过程存在随机性，新的学习对象领导粒子进化的能力并不能保证。武断地将CCLOG生成的学习对象替换旧的学习对象可能会导致粒子质量下降。因此，为了提高算法的勘探能力并充分地利用新旧学习对象的领导潜力，本文提出了基于竞争学习的预测策略(competitive learning based prediction strategy, CLP)帮助粒子选择合适的学习对象。

如算法3所示，对于第i个粒子，当新的学习对象$\boldsymbol{L}_i^{\text{new}}$生成后，CLP策略将通过分别预测新旧学习对象对粒子的领导能力，选择更加适合粒子的学习对象。预测的方式：新旧学习对象$\boldsymbol{L}_i^{\text{new}}$和$\boldsymbol{L}_i^{\text{old}}$将分别带领粒子进化一个周期$T$，其中，能够帮助粒子进化的学习对象将被选择作为粒子在未来的学习对象。

算法3　基于竞争学习的预测机制(CLP)

输入：当前的迭代次数I，预测周期T，第i个粒子的旧的学习对象$\boldsymbol{L}_i^{\text{old}}$和新的学习对象$\boldsymbol{L}_i^{\text{new}}$

输出：选择的学习对象$\boldsymbol{L}_i$

1：for 每个预测周期t，$1\leqslant t\leqslant T$，do

2：　　$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+cr(\boldsymbol{L}_i^{\text{old}}-\boldsymbol{X}_i(t))$

3：　　$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)+\boldsymbol{v}_i(t+1)$

4：end for

5：根据式(7)计算旧学习对象的竞争力$C^{\text{old}}=f(\boldsymbol{X}_i)^I-f(\boldsymbol{X}_i)^{I+T}$

6：for 每个预测周期t，$1\leqslant t\leqslant T$，do

7：　　$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+cr(\boldsymbol{L}_i^{\text{new}}-\boldsymbol{X}_i(t))$

8：　　$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)+\boldsymbol{v}_i(t+1)$

9：end for

10：根据式(7)计算新学习对象的竞争力$C^{\text{new}}=f(\boldsymbol{X}_i)^I-f(\boldsymbol{X}_i)^{I+T}$

11：if $C^{\text{new}}>C^{\text{old}}$ then

12：　　$\boldsymbol{L}_i\leftarrow\boldsymbol{L}_i^{\text{new}}$

13：else

14：　　$\boldsymbol{L}_i\leftarrow\boldsymbol{L}_i^{\text{old}}$

15：end if

图5在一个维度上展示了上述粒子的进化过程，对于一个最小化问题，红色的曲线表示了目标函数的变化趋势，星状的全局最优粒子$\boldsymbol{P}_{\text{g}}$目前位于一个局部最优点上，新的学习对象$\boldsymbol{L}_i^{\text{new}}$更加适合带领粒子进化，那么新的学习对象将会取代旧的学习对象$\boldsymbol{L}_i^{\text{old}}$。

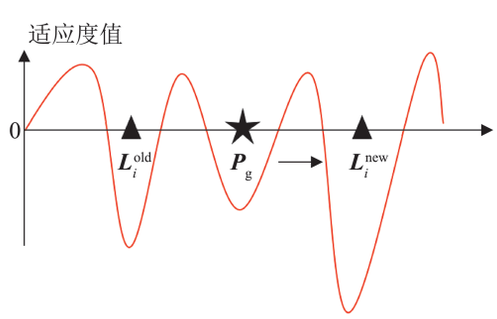


图5　基于竞争学习的预测策略示例

Fig. 5　Example of predictive strategy based on competitive learning

当粒子向新的对象学习时，速度向量为

$$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+cr(\boldsymbol{L}_i{}^{\text{new}}-\boldsymbol{X}_i(t)) \tag{5}$$

当粒子向旧的对象学习时，速度向量为

$$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+cr(\boldsymbol{L}_i{}^{\text{old}}-\boldsymbol{X}_i(t)) \tag{6}$$

由于新旧学习对象之间存在竞争关系，为了从中选择更合适的学习对象，定义竞争力为学习对象带领粒子进化的程度，表达式如公式(7)下：

$$C=f(\boldsymbol{X}_i)^{I}-f(\boldsymbol{X}_i)^{I+T} \tag{7}$$

T表示预测周期。如果新的学习对象的竞争力大于旧的学习对象的竞争力，这意味着新的学习对象更加适合带领粒子进化，那么新的学习对象将会取代旧的学习对象，反之亦然。

1.4 算法流程

CCPSO的整个流程如算法4所示。

算法4　算法CCPSO框架

输入：种群大小N，包含D维变量的优化问题f，算法最大迭代次数$t_{\max}$

输出：全局最优粒子$\boldsymbol{P}_{\text{g}}$

1：for 每个粒子i，$1\leqslant i\leqslant N$，do

2：　初始化$\boldsymbol{X}_i$为搜索区间内的随机位置，$\boldsymbol{v}_i$为0向量。

3：　$\boldsymbol{P}_i\leftarrow\boldsymbol{X}_i$，$\boldsymbol{L}_i\leftarrow\boldsymbol{X}_i$，$T_i\leftarrow 0$，评估粒子$i$

4：end for

5：从初始化的N个粒子中选取适应值最优的粒子作为全局最优粒子$\boldsymbol{P}_{\text{g}}$，算法当前迭代次数$t\leftarrow 0$

6：while $t\leqslant t_{\max}$ do　//算法迭代结束条件

7：　for 每个粒子i，$1\leqslant i\leqslant N$，do

8：　根据式(2)更新粒子i的调整概率Q_i^{adjust}

9：　if $Q_i^{\text{adjust}}>\text{rand}()$ do　// TSDM

10：　根据CCLOG策略生成新的学习对象$\boldsymbol{L}_i$

11：　根据CLP策略更新学习对象$\boldsymbol{L}_i$

12：　$t_i\leftarrow 0$

13：　end if

14：　$\boldsymbol{v}_i(t+1)=w\boldsymbol{v}_i(t)+c(\boldsymbol{L}_i-\boldsymbol{X}_i(t))$ //更新粒子速度

15：　$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)+\boldsymbol{v}_i(t+1)$　//更新粒子位置

16：　评估粒子i，$\boldsymbol{P}_i\leftarrow\boldsymbol{X}_i(t+1)$，

17：　end for

18：　更新$\boldsymbol{P}_{\text{g}}$

19：　$t\leftarrow t+1$

20：end while

CCPSO的流程图如图6所示。

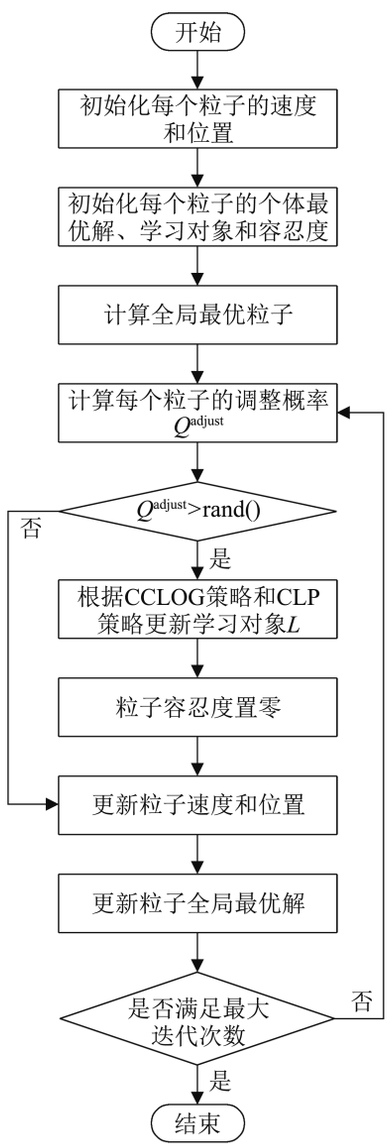


图6　CCPSO的流程图

Fig. 6　Flowchart of CCPSO

2 实验结果与分析

2.1 参数设置

由于CCPSO采用了连续变量重组算子，并基于松弛方法同时处理连续变量和离散变量。为了保证实验结果的公平性，本文选取的对比算法同样基于松弛方法处理混合变量。各个算法参数设置如表1所示。

表1 算法参数设置
Table 1 Algorithm parameter settings

算法名称	算法参数设置
GPSO	$N=20$，$w=0.4$，$c_1=c_2=2$，$I_{max}=10\,000$
CLPSO[8]	$N=20$，$w\in[0.4,0.9]$，$c=1.494\,45$，$m=7$，$I_{max}=10\,000$
AEDA[17]	$N=20$，$NP_{best}=0.45$，$g_p=0.5$，$I_{max}=10\,000$
SLPSO[10]	$N=20$，$\alpha=0.5$，$\beta=0.01$，$I_{max}=10\,000$
DEmv[18]	$N=20$，$F=R=0.2$，$I_{max}=10\,000$
CCPSO	$N=20$，$w=0.4$，$c=2$，$T=5$，$I_{max}=10\,000$

N和I_{max}分别代表种群规模和最大迭代次数。GPSO算法中，惯性权重w固定为0.4，学习因子c_1、c_2均设置为2.0；CLPSO的惯性权重w从0.9线性地递减到0.4，学习因子c设置为1.494 45，允许粒子经过m次无效迭代后刷新学习对象；AEDA算法中，用来更新概率模型的最优个体数量的比重NP_{best}等于0.45，选择高斯分布采样新种群的概率g_p设置为0.5；SLPSO中没有惯性权重，α用于平衡历史搜索信息和当前搜索信息，β设为较小值以避免过早收敛；DEmv算法中，F代表变异操作过程中的缩放因子，R表示交叉概率。上述算法都是按照原论文的参数设置的。本文提出的算法CCPSO中，惯性权重w和学习因子c均与GPSO算法保持一致，对于参数预测周期T，T值较大时可以更加准确地选择出适合当前粒子的学习对象，但是CLP将消耗过多的计算资源。为了在准确度和预测效率之间保持平衡，本文的预测周期T设置为5。

2.2 测试函数实验

2.2.1 人工测试函数

本文选取了CEC2013[18]中的28个测试函数来验证CCPSO算法在解决混合变量优化问题时的有效性。由于CEC2013中的测试函数只包含连续变量，为了验证算法在求解混合变量优化问题的性能，这些测试函数被扩展为了包含连续变量和离散变量的测试函数。扩展方式：在离散变量部分对应的每一个维度中，搜索区间$[U,L]^{disc}$与连续变量的搜索区间$[U,L]^{cont}$保持相同，离散变量的可选值被规定为搜索区间$[U,L]^{disc}$内的整数，全局最优点在整数0处。

扩展后的28个测试函数的名称和特点如表2所示，各测试函数的搜索区间为$[-100,100]$。

为了保证实验结果的公平性和可靠性，所有算法将独立地在每一个测试函数上运行30次，实验结果是运行30次结果的平均值。每个算法的种群大小N为20，测试函数的维度D为50，其中，包括25维的连续变量和25维的离散变量。算法的最大迭代次数I_{max}为10 000，即总的目标函数评价次数为200 000。

2.2.2 实验结果与分析

实验的数值统计结果如表3所示。

表2 CEC2013测试函数
Table 2 Test function of CEC2013

函数类型	编号	函数名称	最优解
单峰	f1	Sphere Function	−1 400
	f2	Rotated High Conditioned Elliptic Function	−1 300
	f3	Rotated Bent Cigar Function	−1 200
	f4	Rotated Discus Function	−1 100
	f5	Different Powers Function	−1 000

续表

函数类型	编号	函数名称	最优解
多峰	f6	Rotated Rosenbrocks Function	−900
	f7	Rotated Schaffers f7 Function	−800
	f8	Rotated Ackleys Function	−700
	f9	Rotated Weierstrass Function	−600
	f10	Rotated Griewanks Function	−500
	f11	Rastrigins Function	−400
	f12	Rotated Rastrigins Function	−300
	f13	Non-Continuous Rotated Rastrigins Function	−200
	f14	Schwefel’s Function	−100
	f15	Rotated Schwefel’s Function	100
	f16	Rotated Katsuura Function	200
	f17	Lunacek-Bi-Rastrigin Function	300
	f18	Rotated Lunacek Bi-Rastrigin Function	400
	f19	Expanded Griewanks plus Rosenbrocks Function	500
	f20	Expanded Scaffers f6 Function	600
复合	f21	Composition Function 1 (n=5, Rotated)	700
	f22	Composition Function 2 (n=3, Unrotated)	800
	f23	Composition Function 3 (n=3, Rotated)	900
	f24	Composition Function 4 (n=3, Rotated)	1 000
	f25	Composition Function 5 (n=3, Rotated)	1 100
	f26	Composition Function 6 (n=5, Rotated)	1 200
	f27	Composition Function 7 (n=5, Rotated)	1 300
	f28	Composition Function 8 (n=5, Rotated)	1 400

表3 实验数值统计结果

Table 3 Experimental results of numerical statistics

函数类型	函数		GPSO	CLPSO	AEDA	SLPSO	DEmv	CCPSO
单峰	f1	误差均值	1.87E+01	2.57E-01	3.89E-02	3.60E+02	4.31E+03	**2.36E-07**
		误差方差	1.15E+02	2.42E+00	9.83E-01	1.44E+03	3.62E+03	**2.16E-06**
	f2	误差均值	6.25E+06	5.78E+06	8.60E+07	2.18E+07	1.64E+08	**5.38E+06**
		误差方差	1.57E+07	1.06E+07	2.99E+08	4.30E+07	1.34E+08	**6.61E+06**
	f3	误差均值	1.25E+09	**1.16E+08**	6.33E+08	1.20E+10	7.34E+10	3.26E+08
		误差方差	6.69E+09	6.46E+08	2.36E+09	2.23E+10	7.92E+10	**6.04E+08**
	f4	误差均值	**1.12E+03**	6.13E+03	4.80E+04	4.41E+04	9.18E+04	3.86E+03
		误差方差	4.26E+03	7.02E+03	3.08E+04	4.81E+04	4.71E+04	**1.37E+03**
	f5	误差均值	1.35E+01	1.10E+00	3.69E-01	2.10E+02	7.43E+02	**3.68E-06**
		误差方差	1.28E+02	7.74E+00	2.33E+00	7.73E+02	4.51E+02	**1.98E-05**
多峰	f6	误差均值	1.29E+02	1.09E+02	2.10E+02	2.56E+02	6.73E+02	**7.31E+01**
		误差方差	4.10E+02	1.67E+02	2.12E+02	2.81E+02	4.59E+02	**1.07E+02**
	f7	误差均值	5.69E+01	**2.47E+01**	2.55E+01	6.92E+01	1.88E+02	4.32E+01
		误差方差	8.91E+01	4.97E+01	3.21E+01	7.38E+01	1.18E+02	**3.08E+01**

续表

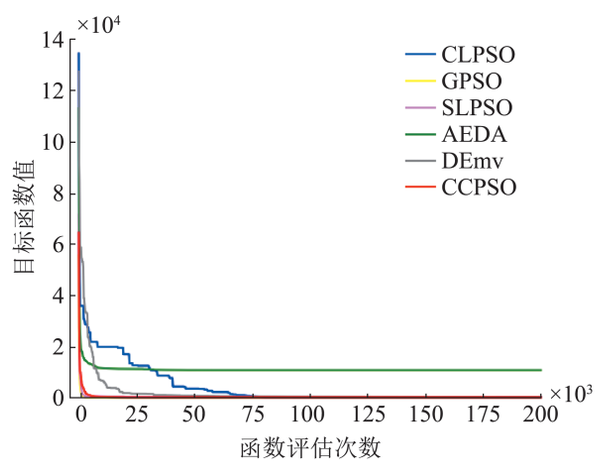
函数类型	函数		GPSO	CLPSO	AEDA	SLPSO	DEmv	CCPSO
多峰	f8	误差均值	2.11E+01	2.11E+01	2.13E+01	2.11E+01	2.13E+01	**2.11E+01**
		误差方差	2.22E-01	2.39E-01	1.71E-01	3.28E-01	2.47E-01	**1.62E-01**
	f9	误差均值	4.30E+01	**2.08E+01**	6.40E+01	3.54E+01	6.58E+01	3.27E+01
		误差方差	4.67E+01	2.73E+01	1.11E+02	2.43E+01	**9.55E+00**	3.11E+01
	f10	误差均值	3.20E+01	4.65E+00	2.77E+01	2.18E+02	1.46E+03	**2.75E+00**
		误差方差	1.37E+02	1.92E+01	7.31E+01	3.53E+02	8.86E+02	**1.24E+00**
	f11	误差均值	8.50E+01	4.90E+01	1.57E+02	1.38E+02	3.82E+02	**4.40E+01**
		误差方差	**1.09E+02**	1.12E+02	1.11E+02	1.34E+02	1.15E+02	**3.64E+01**
	f12	误差均值	2.16E+02	2.64E+02	5.70E+02	2.05E+02	6.23E+02	**1.66E+02**
		误差方差	2.17E+02	2.87E+02	**1.01E+02**	1.94E+02	1.49E+02	1.60E+02
	f13	误差均值	**2.95E+02**	3.37E+02	3.71E+02	3.22E+02	6.34E+02	3.12E+02
		误差方差	3.31E+02	2.37E+02	**7.72E+01**	2.71E+02	1.61E+02	1.04E+02
	f14	误差均值	2.41E+03	4.54E+03	6.47E+03	4.43E+03	6.24E+03	**1.27E+03**
		误差方差	4.14E+03	1.20E+04	1.68E+03	5.02E+03	1.44E+03	**1.28E+03**
	f15	误差均值	8.59E+03	1.31E+04	1.36E+04	7.02E+03	1.31E+04	**6.97E+03**
		误差方差	8.30E+03	3.68E+03	2.04E+03	4.58E+03	**1.94E+03**	3.55E+03
	f16	误差均值	3.27E+00	3.51E+00	3.73E+00	**1.50E+00**	3.75E+00	3.15E+00
		误差方差	2.24E+00	1.38E+00	1.27E+00	2.95E+00	2.08E+00	**1.03E+00**
	f17	误差均值	1.19E+02	9.64E+01	1.69E+02	2.17E+02	6.28E+02	**4.72E+01**
		误差方差	1.81E+02	2.55E+02	7.76E+01	3.18E+02	2.23E+02	**2.70E+01**
	f18	误差均值	1.07E+02	4.86E+01	1.76E+02	2.16E+02	6.40E+02	**4.65E+01**
		误差方差	1.62E+02	1.88E+02	3.62E+02	2.93E+02	1.91E+02	**3.65E+01**
	f19	误差均值	1.21E+01	2.11E+01	1.62E+01	6.14E+01	3.71E+01	**7.42E+00**
		误差方差	3.98E+01	1.75E+01	1.15E+01	1.90E+02	6.69E+02	**2.93E+00**
	f20	误差均值	1.96E+01	2.04E+01	2.16E+01	1.87E+01	2.18E+01	**1.85E+01**
		误差方差	5.31E+00	2.46E+00	1.32E+00	4.52E+00	**1.22E+00**	4.54E+00
复合	f21	误差均值	2.01E+02	2.00E+02	2.01E+02	2.37E+02	4.78E+02	**2.00E+02**
		误差方差	1.11E+01	3.36E-01	1.71E-01	1.02E+02	1.84E+02	**6.64E-09**
	f22	误差均值	2.56E+03	5.31E+03	6.61E+03	4.72E+03	6.39E+03	**1.42E+03**
		误差方差	2.95E+03	1.14E+04	1.68E+03	4.39E+03	**1.43E+03**	1.70E+03
	f23	误差均值	1.29E+04	1.54E+04	1.64E+04	1.21E+04	1.62E+04	**1.20E+04**
		误差方差	5.56E+03	2.70E+03	**9.06E+02**	3.43E+03	1.50E+03	4.90E+03
	f24	误差均值	1.05E+03	1.19E+03	1.43E+03	9.93E+02	1.74E+03	**9.14E+02**
		误差方差	4.93E+02	7.32E+02	2.96E+02	4.93E+02	**2.75E+02**	3.94E+02
	f25	误差均值	4.72E+02	5.00E+02	5.36E+02	4.65E+02	6.14E+02	**4.47E+02**
		误差方差	8.24E+01	1.24E+02	6.83E+01	1.15E+02	**5.44E+01**	7.89E+01
	f26	误差均值	1.20E+03	7.94E+02	**7.70E+02**	1.65E+03	2.05E+03	1.05E+03
		误差方差	1.15E+03	8.89E+02	8.39E+02	1.60E+03	2.33E+03	**5.63E+02**
	f27	误差均值	5.35E+03	**4.91E+03**	5.08E+03	5.49E+03	5.34E+03	5.01E+03
		误差方差	8.66E+02	5.59E+02	5.83E+02	7.93E+02	**2.19E+02**	3.04E+02
	f28	误差均值	8.56E+03	9.94E+03	1.15E+04	7.68E+03	1.05E+04	**7.56E+03**
		误差方差	4.27E+03	7.47E+03	**1.18E+03**	3.56E+03	1.45E+03	2.97E+03

DOI: 10.16182/j.issn1004731x.joss.22-1466

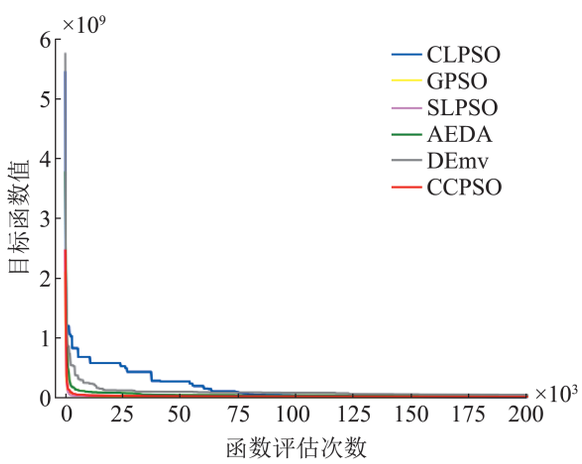
如表 3 所示，CCPSO 在大部分测试函数上取得了最优的结果。CLPSO 和 SLPSO 表现不错的原因是算法采用了不同的方法维持种群的多样性，避免种群陷入局部最优。GPSO 中的种群仅向全局最优粒子学习，在求解多峰函数时，整个种群会因为全局最优粒子陷入局部最优而进化停滞，导致算法过早收敛。CCPSO 采用的 TSDM 机制能够自适应地判断粒子的进化状态，避免粒子陷入局部最优，此外，CCPSO 采用的 CCLOG 机制为种群中的每一个粒子生成不同的学习对象，使整个种群的多样性得到提高，而 CLP 策略能够为粒子选择合适的学习对象，保证了种群多样性的同时提高了每一个粒子的局部勘探能力。

本文进一步对比了各个算法的收敛速度，部分测试函数上算法的收敛结果如图 7 所示。从图 7 可以看出，CCPSO 可以较快地收敛到最优解，这是由于 CCPSO 采用了基于容忍度的搜索方向调整机制，该机制可以帮助粒子调整搜索方向以避免粒子陷入局部最优。

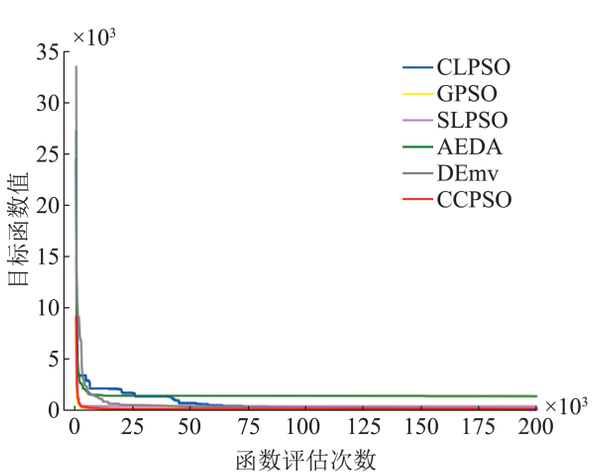
为验证不同策略的性能，本文将 CCPSO 与不采用对应策略的 CCPSO 进行了实验对比，实验结果如表 4 所示。其中，CCPSO-CLP 表示不采用基于竞争学习的预测机制，直接替换原有学习对象。CCPSO-CCLOG 表示不采用基于竞争进化的学习对象生成机制，而是随机生成学习对象。



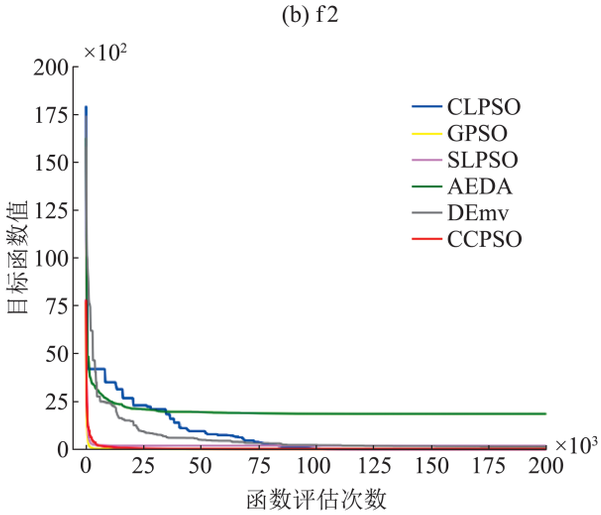
(a) f1



(b) f2



(c) f6



(d) f10

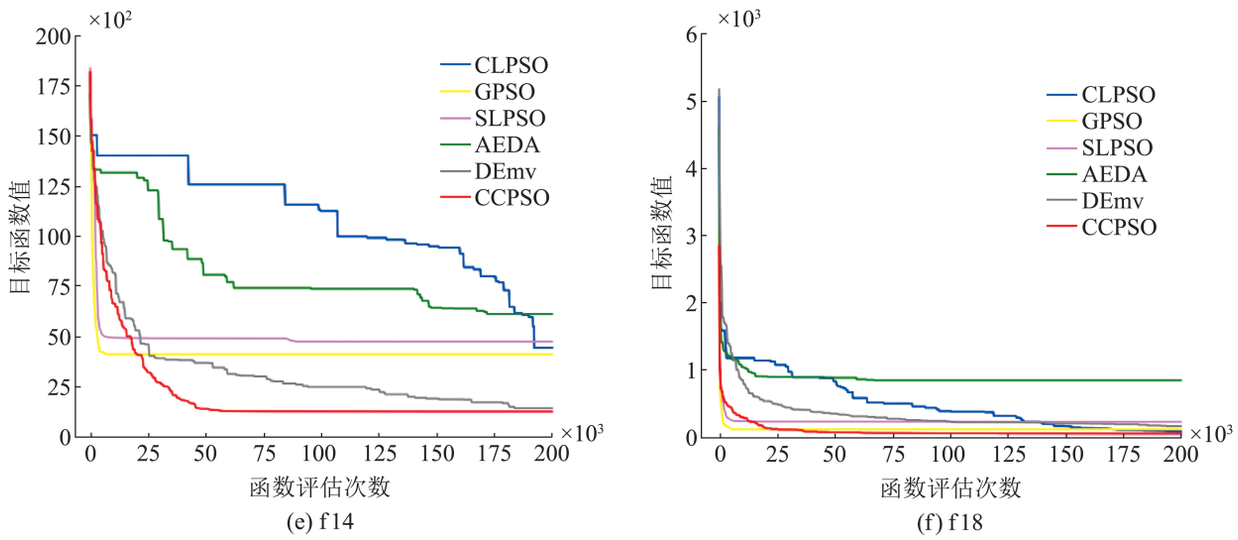


图 7　CCPSO 的收敛曲线
Fig. 7　Convergence curve of CCPSO

表 4　策略对比实验结果
Table 4　Experimental results of strategy comparison

函数类型	函数		CCPSO-CLP	CCPSO-CCLOG	CCPSO
单峰	f1	误差均值	8.19E+01	1.04E+02	2.36E-07
		误差方差	1.04E+03	1.14E+03	**2.16E-06**
	f2	误差均值	7.28E+06	1.22E+08	**5.38E+06**
		误差方差	2.22E+07	3.84E+08	**6.61E+06**
	f3	误差均值	1.06E+09	2.61E+10	**3.26E+08**
		误差方差	6.80E+09	4.39E+10	**6.04E+08**
	f4	误差均值	2.65E+04	3.32E+04	**3.86E+03**
		误差方差	2.56E+04	1.51E+04	**1.37E+03**
	f5	误差均值	1.63E+01	1.81E+01	**3.68E-06**
		误差方差	1.31E+02	1.18E+02	**1.98E-05**
多峰	f6	误差均值	2.34E+02	3.01E+02	**7.31E+01**
		误差方差	3.84E+02	5.74E+02	**1.07E+02**
	f7	误差均值	**2.97E+01**	1.11E+02	4.32E+01
		误差方差	8.70E+01	6.27E+01	**3.08E+01**
	f8	误差均值	2.12E+01	2.12E+01	**2.11E+01**
		误差方差	2.44E-01	1.82E-01	**1.62E-01**
	f9	误差均值	4.07E+01	5.62E+01	**3.27E+01**
		误差方差	4.21E+01	**2.31E+01**	3.11E+01
	f10	误差均值	5.20E+01	2.35E+02	**2.75E+00**
		误差方差	2.68E+02	1.22E+03	**1.24E+00**
	f11	误差均值	1.56E+02	2.24E+02	**4.40E+01**
		误差方差	2.22E+02	4.58E+02	**3.64E+01**
	f12	误差均值	3.25E+02	5.09E+02	**1.66E+02**
		误差方差	5.04E+02	2.14E+02	**1.60E+02**
	f13	误差均值	3.99E+02	5.28E+02	**3.12E+02**
		误差方差	1.13E+02	**9.92E+01**	1.04E+02

https://dc-china-simulation.researchcommons.org/journal/vol36/iss4/6
DOI: 10.16182/j.issn1004731x.joss.22-1466

续表

函数类型	函数		CCPSO-CLP	CCPSO-CCLOG	CCPSO
	f14	误差均值	3.99E+03	9.68E+03	**1.27E+03**
		误差方差	4.44E+03	1.59E+04	**1.28E+03**
	f15	误差均值	1.27E+04	1.31E+04	**6.97E+03**
		误差方差	3.40E+03	**2.03E+03**	3.55E+03
	f16	误差均值	3.49E+00	3.63E+00	**3.15E+00**
		误差方差	2.27E+00	2.04E+00	**1.03E+00**
多峰	f17	误差均值	2.04E+02	3.44E+02	**4.72E+01**
		误差方差	4.70E+02	7.52E+02	**2.70E+01**
	f18	误差均值	2.45E+02	3.23E+02	**4.65E+01**
		误差方差	6.17E+02	8.31E+02	**3.65E+01**
	f19	误差均值	1.57E+01	7.46E+01	**7.42E+00**
		误差方差	3.04E+01	6.62E+02	**2.93E+00**
	f20	误差均值	2.07E+01	2.09E+01	**1.85E+01**
		误差方差	1.79E+00	**1.44E+00**	4.54E+00
	f21	误差均值	2.06E+02	2.04E+02	**2.00E+02**
		误差方差	6.15E+01	5.79E+01	**6.64E-09**
	f22	误差均值	4.07E+03	8.68E+03	**1.42E+03**
		误差方差	4.51E+03	1.57E+04	**1.70E+03**
	f23	误差均值	1.57E+04	1.59E+04	**1.20E+04**
		误差方差	2.40E+03	**9.87E+02**	4.90E+03
	f24	误差均值	1.45E+03	1.56E+03	**9.14E+02**
复合		误差方差	3.55E+02	**2.27E+02**	3.94E+02
	f25	误差均值	5.33E+02	5.66E+02	**4.47E+02**
		误差方差	**5.40E+01**	6.21E+01	7.89E+01
	f26	误差均值	1.57E+03	1.91E+03	**1.05E+03**
		误差方差	1.99E+03	2.91E+03	**5.63E+02**
	f27	误差均值	5.31E+03	5.64E+03	**5.01E+03**
		误差方差	7.39E+02	8.60E+02	**3.04E+02**
	f28	误差均值	1.08E+04	1.14E+04	**7.56E+03**
		误差方差	3.86E+03	**1.22E+03**	2.97E+03

从表4可以看出，若不采用CCLOG或CLP机制，算法的性能会有明显的下降。这说明CCLOG机制有助于找到更好的学习对象，CLP机制则可以从旧的学习对象和新的学习对象中选择更好的进行学习。

2.3 实际案例实验

CCPSO在人工测试函数上的表现验证了本文提出的协同进化策略的有效性。为了进一步扩展CCPSO的应用场景，选择了真实场景中的混合变量优化问题做实验。

2.3.1 螺旋弹簧设计问题

螺旋弹簧设计问题[19] (coil spring design problem, CSD)旨在设计出能够承受横向载荷的压缩弹簧，优化目标是最小化制造弹簧所需要的金属丝消耗。CSD问题中包含3个决策变量：N表示弹簧圈数的整数变量，D表示弹簧直径的连续变量，d表示导线直径的离散变量，可选值为0.009、0.009 5、0.010 4、0.011 8、0.012 8、0.013 2、

0.014、0.015、0.016 2、0.017 3、0.018、0.02、0.023、0.025、0.028、0.032、0.035、0.041、0.047、0.054、0.063、0.072、0.08、0.092、0.105、0.12、0.135、0.148、0.162、0.177、0.192、0.207、0.225、0.244、0.263、0.283、0.307、0.331、0.362、0.394、0.437 5、0.5。

CSD 问题的目标函数和约束条件：

$$\min f(N,D,d)=\frac{\pi^2 Dd^2(N+2)}{4}$$

$$g_1:\ \frac{8C_f F_{max} D}{\pi d^3}-S\leqslant 0$$

$$g_2:\ l_f-l_{max}\leqslant 0$$

$$g_3:\ d_{min}-d\leqslant 0$$

$$g_4:\ D-D_{max}\leqslant 0$$

$$g_5:\ 3.0-D/d\leqslant 0$$

$$g_6:\ \sigma_p-\sigma_{pm}\leqslant 0$$

$$g_7:\ \sigma_p+(F_{max}-F_p)/K+1.05(N+2)d-l_f\leqslant 0$$

$$g_8:\ \sigma_w-(F_{max}-F_p)/K\leqslant 0$$

$$C_f=\frac{4\frac{D}{d}-1}{4\frac{D}{d}-4}+\frac{0.061\,5d}{D}$$

$$K=\frac{Gd^4}{8ND^3}$$

$$\sigma_p=F_p/K$$

$$l_f=F_{max}/K+1.05(N+2)d$$

2.3.2 实验结果与分析

目前已知的 CSD 问题的最优结果为 2.628 52，表 5 展示了各个算法在 CSD 问题上运行 30 次的结果。

表 5 各个算法求解 CSD 的数值结果

Table 5 Numerical results of CSD obtained by each algorithm

算法	最优	最差	平均	方差
GPSO	2.658 55	3.487 27	3.128 16	2.2492E+00
CLPSO	**2.628 52**	2.628 56	2.628 54	4.8647E-15
AEDA	**2.628 52**	11.357 50	4.686 63	1.0079E+01
DEmv	2.658 55	2.800 16	2.673 12	2.3224E-01
SLPSO	2.658 55	3.487 27	3.321 53	1.8156E+01
CCPSO	**2.628 52**	**2.628 52**	**2.628 52**	**0**

从表 5 可以看出 CCPSO 在每次运行后都能求得 CSD 问题目前已知的最优解。GPSO、DEmv 和 SLPSO 算法在求解 CSD 问题时表现较差，无法求得最优解。CLPSO、AEDA 和其他算法相比，虽可以求得 CSD 问题的最优解，但均值和方差较大。综上，本文提出的 CCPSO 算法不仅能求得最优解且稳定性更优。

3 结论

为了求解混合变量优化问题，本文基于竞争式协同进化策略提出了竞争式协同进化粒子群算法 CCPSO。该算法采用基于容忍度的搜索方向调整机制自适应地调整粒子的搜索方向，避免粒子陷入局部最优。同时采用基于竞争协同进化的学习对象生成机制为每一个粒子生成适合粒子进化的学习对象，提高种群多样性的同时保证了粒子的进化效率。采用基于竞争学习的预测策略为粒子选择最合适的学习对象，保证了粒子局部搜索的能力。

通过在多个测试函数上的实验表明，相较于其他混合变量的粒子群算法，CCPSO 在单峰测试函数、多峰测试函数、复合函数上都表现出色，验证了 CCPSO 采用的竞争式的协同进化策略不仅可以提高算法多样性，同时也能保证算法收敛性。CCLOG 和 CLP 策略虽然提升了最终目标函数的均值，但方差上的表现却不是很好，说明算法在稳定性方面还存在一定的改进空间，如何提高算法的稳定性是未来工作的研究重点。

参考文献：

[1] 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.

[2] Wang Feng, Zhang Heng, Zhou Aimin. A Particle Swarm Optimization Algorithm for Mixed-variable Optimization Problems[J]. Swarm and Evolutionary Computation, 2021, 60: 100808.

[3] Wang Feng, Li Yixuan, Zhou Aimin, et al. An Estimation of Distribution Algorithm for Mixed-variable

DOI: 10.16182/j.issn1004731x.joss.22-1466

Newsvendor Problems[J]. IEEE Transactions on Evolutionary Computation, 2020, 24(3): 479-493.

[4] Lin Ying, Liu Yu, Chen Weineng, et al. A Hybrid Differential Evolution Algorithm for Mixed-variable Optimization Problems[J]. Information Sciences, 2018, 466: 170-188.

[5] Liu Yaxian, Lu Hui, Cheng Shi, et al. An Adaptive Online Parameter Control Algorithm for Particle Swarm Optimization Based on Reinforcement Learning[C]// 2019 IEEE Congress on Evolutionary Computation (CEC). Piscataway, NJ, USA: IEEE, 2019: 815-822.

[6] Maraboina Raju, Munish Kumar Gupta, Neeraj Bhanot, et al. A Hybrid PSO-BFO Evolutionary Algorithm for Optimization of Fused Deposition Modelling Process Parameters[J]. Journal of Intelligent Manufacturing, 2019, 30(7): 2743-2758.

[7] Fatih Ahmet Şenel, Fatih Gökçe, Asım Sinan Yüksel, et al. A Novel Hybrid PSO-GWO Algorithm for Optimization Problems[J]. Engineering with Computers, 2019, 35(4): 1359-1373.

[8] 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.

[9] Cao Yulian, Zhang Han, Li Wenfeng, et al. Comprehensive Learning Particle Swarm Optimization Algorithm with Local Search for Multimodal Functions[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(4): 718-731.

[10] Cheng Ran, Jin Yaochu. A Social Learning Particle Swarm Optimization Algorithm for Scalable Optimization[J]. Information Sciences, 2015, 291: 43-60.

[11] Song An, Chen Weineng, Gu Tianlong, et al. Distributed Virtual Network Embedding System with Historical Archives and Set-based Particle Swarm Optimization[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, 51(2): 927-942.

[12] Zheng Jie, Wang Ling, Wang Jingjing. A Cooperative Coevolution Algorithm for Multi-objective Fuzzy Distributed Hybrid Flow Shop[J]. Knowledge-Based Systems, 2020, 194: 105536.

[13] Daniel H Stolfi, Matthias R Brust, Grégoire Danoy, et al. Competitive Evolution of a UAV Swarm for Improving Intruder Detection Rates[C]//2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Piscataway, NJ, USA: IEEE, 2020: 528-535.

[14] 段绍米, 罗会龙, 刘海鹏. 人群搜索和樽海鞘群的混合算法优化PID参数[J]. 系统仿真学报, 2022, 34(6): 1230-1246.
Duan Shaomi, Luo Huilong, Liu Haipeng. A Hybrid Algorithm Based on Seeker Optimization Algorithm and Salp Swarm Algorithm for PID Parameters Optimization[J]. Journal of System Simulation, 2022, 34(6): 1230-1246.

[15] Huang Chen, Zhou Xiangbing, Ran Xiaojuan, et al. Co-evolutionary Competitive Swarm Optimizer with Three-phase for Large-scale Complex Optimization Problem[J]. Information Sciences, 2023, 619: 2-18.

[16] Qiu Qicang, Yu Wei, Wang Liping, et al. Preference-inspired Coevolutionary Algorithm Based on Differentiated Resource Allocation Strategy[J]. IEEE Access, 2020, 8: 205798-205813.

[17] Shi Wen, Chen Weineng, Lin Ying, et al. An Adaptive Estimation of Distribution Algorithm for Multipolicy Insurance Investment Planning[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(1): 1-14.

[18] Liang J J, Qu B Y, Suganthan P N, et al. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-parameter Optimization[EB/OL]. [2022-12-03]. https://alroomi. org/multimedia/CEC_Database/CEC2013/RealParameterOptimization/CEC2013_ParameterOptimization_TechnicalReport.pdf.

[19] Sandgren E. Nonlinear Integer and Discrete Programming in Mechanical Design Optimization[J]. Journal of Mechanical Design, 1990, 112(2): 223-229.