计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190400039

信息共享模型和组外贪心策略的郊狼优化算法

张新明[1,2]　李双倩[1]　刘　艳[1,2]　毛文涛[1,2]　刘尚旺[1]　刘国奇[1]
1 河南师范大学计算机与信息工程学院　河南 新乡 453007
2 智慧商务与物联网技术河南省工程实验室　河南 新乡 453007

摘　要　郊狼优化算法(Coyote Optimization Algorithm,COA)是最近提出的一种新颖群智能优化算法,具有较大的应用潜力,但存在运行时间长和搜索能力不足等问题。因此,文中提出了一种改进的COA,即基于信息共享和组外(静态)贪心的COA(COA based on Information sharing and Static greed selection,ISCOA)。首先,构建一种新型的信息共享模型,用于子群所有郊狼的成长,在郊狼成长前期,共享信息差异性大,以增加种群的多样性,在效狼成长后期,共享信息差异性小,以强化开采能力;其次,构建一种新的组内成长方式,即前期主要采用信息共享模型的成长方式,以郊狼的信息共享为主强化探索能力,后期主要采用原算法的成长方式,以***alpha***狼和文化趋势的引导为主强化开采能力;最后,将原算法的组内贪心算法改成组外贪心算法,即静态贪心算法,以便提高算法的稳定性和实现目标函数计算等的并行处理,提高运行速度。大量复杂的CEC2017函数优化实验结果表明,与COA相比,ISCOA在29个10维和30维函数上分别获得了23和24个函数的优势,其平均运行时间分别是COA的86.3%和85.7%,降低了运行时间;与7个最先进的算法相比,ISCOA在10维和30维函数上的平均排名分别是1.48和1.69,分别获得了17和18个第一,具有更好的优化效果。运用于实际工程问题的实验结果表明,ISCOA得到了最好的结果,证明了ISCOA有更强的搜索能力和竞争性以及更好的应用前景。

关键词:群智能优化算法;郊狼优化算法;贪心算法;探索能力;开采能力
中图法分类号　TP181

Coyote Optimization Algorithm Based on Information Sharing and Static Greed Selection

ZHANG Xin-ming[1,2],LI Shuang-qian[1],LIU Yan[1,2],MAO Wen-tao[1,2],LIU Shang-wang[1] and LIU Guo-qi[1]
1 College of Computer and Information Engineering,Henan Normal University,Xinxiang,Henan 453007,China
2 Engineering Lab of Intelligence Business and Internet of Things of Henan Province,Xinxiang,Henan 453007,China

Abstract　Coyote Optimization Algorithm (COA) is a novel intelligent optimization algorithm recently proposed and has great application potential,but it has some problems such as long running time and insufficient search ability. This paper proposes an improved COA,namely COA based on Information sharing and Static greed selection (ISCOA). Firstly,a new information sharing model is constructed and applied to the growth of all coyotes in the subgroup,the difference of the sharing information is larger in the early growth so as to increase the population diversity,and the one is smaller in the late growth to be beneficial to exploitation. Secondly,a new intra-group growth mode is constructed,that is to say,a new growth way is adopted in the early stage,mainly based on the information sharing model,to strengthen the growth process to improve the exploration ability,and the growth method of the original algorithm is kept in the later stage,mainly based on the guidance of the ***alpha*** wolf and the cultural trend,to strengthen the exploiting ability. Finally,the intragroup greedy algorithm of the original algorithm is changed into a static greedy algorithm to improve the stability of the algorithm,realize the parallel calculation of the objective function,and improve the running speed. A large number of experiment results on the complex functions from CEC2017 test set show that,compared with COA,ISCOA obtains the advantage of 23 and 24 of the 29 10-dimensional and 30-dimensional functions respectively,and its average running time is 86.3% and 85.7% of COA's on the 10-dimensional and 30-dimensional functions respectively,and its running time is decreased. Compared with the 7 state-of-the-art algorithms,the average ranking of ISCOA on the 10-dimensional and 30-dimensional functions are 1.48 and 1.69,ISCOA wins 17 and 18 times ranking the first,respectively,and obtains better optimization results. Experimental results on the practical engineering problem show that ISCOA has achieved the best results. These all proved that ISCOA has stronger search ability and more competitive,and that it has better application prospects.

到稿日期:2019-04-07　返修日期:2019-09-01　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(U1704158);河南省高等学校重点科研项目(19A520026)
This work was supported by the National Natural Science Foundation of China (U1704158) and Key Research Projects of Higher Education Institutions of Henan Province,China (19A520026).
通信作者:张新明(xinmingzhang@126.com)

Keywords Swarm intelligence optimization algorithm, Coyote optimization algorithm, Greedy algorithm, Exploration, Exploitation

1 引言

随着社会的进步与科学的发展,需要解决的优化问题越来越多也越来越复杂,这些问题的主要特征体现在非线性、非凸、约束以及多模态等方面,传统的优化方法在合理地计算成本范围内不能有效地找到这些问题最优或满意的解决方案。因此,越来越多的研究者开始寻求模拟自然界、生物行为的智能优化算法来处理这些优化问题。群体智能优化算法(Swarm Intelligence Optimization Algorithm, SIOA)作为现代智能计算的一类重要的优化方法,已经受到越来越多的研究者关注。SIOA 通过模拟自然界中的各种群体行为,利用群体中个体之间的信息交互和合作实现寻优。在过去的几十年里,研究者提出了大量 SIOA,如粒子群优化算法(Particle Swarm Optimization, PSO)[1]、和声搜索算法(Harmony Search, HS)[2]、混合蛙跳算法(Shuffle Frog Leaping Algorithm, SFLA)[3]、灰狼优化算法(Grey Wolf Optimizer, GWO)[4-5]等,这些 SIOA 已经被广泛地应用于各种工程领域、医学领域以及其他领域[6],并且取得了较大的成功,受到各个领域专业人员的青睐。但根据 No-Free-Lunch 定理[7],没有任何一个单一的优化算法能够解决所有的优化问题。因此,SIOA 被不断地改进。2018 年,Pierezan 等提出了一种新颖的 SIOA,即郊狼优化算法(COA)[8]。

COA 是模拟郊狼群居生活、成长、生死、被组驱离和接纳等现象的新型优化算法,该算法不仅具有独特的搜索模型、结构以及出色的优化能力,而且在解决全局优化问题中具有明显的优势[8]。但是,由于 COA 被提出的时间较短,有较多需要改进和完善的地方,如在处理复杂的优化问题时,存在运行时间长和搜索能力不足等问题。由此,本文提出了一种改进的 COA,即基于信息共享和组外(静态)贪心的 COA(COA based on Information sharing and Static greed selection, ISCOA)。大量的实验结果表明,与 COA 以及最新算法相比,ISCOA 的优化性能更好。

2 郊狼优化算法

COA 是一种受郊狼社会生活以及其行为启发的基于群体的随机算法。在 COA 中,郊狼被随机分为 Np 个组,每组中含有 Nc 个郊狼,总的郊狼数量为 N,则 $N=Np\times Nc$。每一个郊狼代表问题的一个候选解 $\boldsymbol{soc}$,每个解向量由郊狼的社会状态因子构成,用郊狼的社会适应能力来评价每个候选解向量的质量。

2.1 初始化并随机分组

在 COA 中,初始化分组个数 Np、每组的郊狼个数 Nc 以及目标最大评价次数 FEs 等参数。依据式(1)随机初始化第 p 组第 c 个郊狼在第 j 维上的社会状态因子,并由式(2)评估郊狼的社会适应能力。

$$soc_{c,j}=lb_j+r\times(ub_j-lb_j) \tag{1}$$

$$fit_c=f(\boldsymbol{soc}_c) \tag{2}$$

其中,lb_j 和 ub_j 分别表示第 j 维社会状态因子的下限和上限,j 的取值范围为[1, D],D 为待优化问题的维数,r 是[0,1]范围内的均匀分布的随机数。

2.2 郊狼成长

在 COA 中,组内郊狼的成长受到最优狼 $\boldsymbol{alpha}$ 和组内文化趋势 $\boldsymbol{cult}$ 的共同影响。组内文化趋势就是:对组内的所有郊狼依据其社会状态因子进行升序排序,如果组内郊狼的个数为奇数,则取第 $(Nc+1)/2$ 个郊狼的各个社会状态因子作为该组郊狼的文化趋势;如果组内郊狼的个数为偶数,则取第 $Nc/2$ 个郊狼的各个社会状态因子和第 $(Nc/2+1)$ 个郊狼的各个社会状态因子之和除以 2 作为该组的文化趋势,如式(3)所示。另外,$\boldsymbol{\delta}_1$ 表示组内随机选取的一个郊狼(cr_1)与 $\boldsymbol{alpha}$ 的差值,$\boldsymbol{\delta}_2$ 表示组内随机选取的另一个郊狼(cr_2)与 $\boldsymbol{cult}$ 的差值,$\boldsymbol{\delta}_1$ 和 $\boldsymbol{\delta}_2$ 分别用式(4)和式(5)表示:

$$cult_j=\begin{cases}O_{(Nc+1)/2,j}, & Nc\text{ 是奇数}\\ (O_{Nc/2,j}+O_{Nc/2+1,j})/2, & Nc\text{ 是偶数}\end{cases} \tag{3}$$

$$\boldsymbol{\delta}_1=\boldsymbol{alpha}-\boldsymbol{soc}_{cr_1} \tag{4}$$

$$\boldsymbol{\delta}_2=\boldsymbol{cult}-\boldsymbol{soc}_{cr_2} \tag{5}$$

每个郊狼在 $\boldsymbol{\delta}_1$ 和 $\boldsymbol{\delta}_2$ 的共同作用下成长,如式(6)所示:

$$\boldsymbol{new\_soc}=\boldsymbol{soc}+r_1\times\boldsymbol{\delta}_1+r_2\times\boldsymbol{\delta}_2 \tag{6}$$

其中,r_1 和 r_2 分别是 $\boldsymbol{\delta}_1$ 和 $\boldsymbol{\delta}_2$ 的随机权重,r_1 和 r_2 为[0,1]范围内的均匀分布的随机数。在组内的一个郊狼成长后,就评估其社会适应能力,如式(7)所示:

$$new\_fit=f(\boldsymbol{new\_soc}) \tag{7}$$

如果成长后郊狼的社会适应能力比成长前郊狼的社会适应能力好,则用成长后的郊狼取代成长前的郊狼,反之则保持不变,如式(8)所示:

$$\boldsymbol{soc}=\begin{cases}\boldsymbol{new\_soc}, & new\_fit<fit\\ \boldsymbol{soc}, & \text{otherwise}\end{cases} \tag{8}$$

2.3 郊狼的生与死

出生和死亡也是郊狼群的两个自然事件,一般情况下,新生和老死并存。COA 计算郊狼的年龄(单位:年),用 $year$ 表示。新生郊狼来自于其父母郊狼的随机选择和环境因素的影响,如式(9)所示:

$$pup_j=\begin{cases}soc_{cr_1,j}, & r_j<P_s\text{ or }j=j_1\\ soc_{cr_2,j}, & r_j\geqslant P_s+P_a\text{ or }j=j_2\\ R_j, & 1-(P_s+P_a)\end{cases} \tag{9}$$

其中,cr_1 和 cr_2 是来自第 p 组的两个不相同的随机父郊狼标引号;j_1 和 j_2 是问题的两个随机维度,以确保幼狼一定遗传两个父郊狼的基因;P_s 是分散概率;P_a 是关联概率;R_j 是第 j 维社会状态因子在决策变量范围内的随机数;r_j 是均匀分布在[0,1]范围内的随机数。分散概率和关联概率决定幼小郊狼的遗传和变异情况。P_s 和 P_a 由式(10)表示:

$$P_s=1/D, P_a=(1-P_s/2) \tag{10}$$

其中,P_a 对父母双方的影响相同。

幼狼出生后,需要对其社会适应能力进行评估,然后将幼狼的社会适应能力与组内所有郊狼的社会适应能力进行比较,当组内只有一个郊狼的社会适应能力比幼狼差时,则社会适应能力差的郊狼死亡,幼狼存活;当组内有多个郊狼的社会适应能力比幼狼差时,则用幼狼取代这些适应能力差的郊狼中年龄最大的郊狼;当组内所有郊狼的社会适应能力都比幼

狼的社会适应能力强时，则幼狼死亡。

2.4 郊狼被驱离和接纳

在 COA 中，郊狼以 P_e 的概率被组驱离和接纳，其计算见式(11)，这种随机驱离和接纳保证了郊狼组的多样性。

$$P_e = 0.005 \times N_c^2 \tag{11}$$

COA 主要由 4 个步骤组成：初始化并随机组群、组内郊狼成长、郊狼生与死以及被组驱离与接纳。COA 的伪代码如算法 1 所示。

算法 1 COA 伪代码

```
1. 开始
2. 设置 Np 和 Nc 等参数，郊狼群随机初始化(式(1))并随机分组；
3. 计算每个郊狼的社会适应能力；
4. while 没有达到停止标准
5.    for p=1:Np
6.       确定组内 alpha
7.       由式(3)计算组文化趋势
8.          for c=1:Nc
9.             郊狼成长(式(6))，并归正其成长范围
10.            评估成长后郊狼的社会适应能力(式(7))
11.            贪心选择适应能力好的成长后郊狼(式(8))
12.         end for
13.         幼狼的出生和死亡(式(9))，如幼狼活，则其年龄为 0
14.   end for
15.   郊狼被组驱离和接纳(式(11))
16.   更新每个郊狼的年龄
17. end while
18. 输出最好的郊狼
19. 结束
```

由算法 1 可知，COA 有如下的优势：1)与有类似结构的 SFLA 相比，郊狼的分组以及组内每个郊狼的成长使 COA 有更高的概率获得最优解；2)组内所有郊狼在组内最优郊狼和组文化趋势的引导下趋优成长，故 COA 具有较强的开采能力；3)与 PSO 等算法相比，郊狼在出生过程中随机选择父狼的基因并受环境因素的影响，使 COA 具有更好的探索能力。

3 改进的郊狼优化算法

与 PSO，HS 和 GWO 等算法相比，COA 虽然在解决一些优化问题上具有明显的优势[8]，但在解决复杂优化问题时仍存在寻优时间长以及搜索能力不足等问题：1)由算法 1 可知，每头郊狼在每次成长后都会进行归正成长范围和计算目标函数等操作，这种串行的操作方式及整个算法的多重迭代会导致寻优时间长；2)郊狼的成长是依靠组内最优郊狼和组文化趋势等的引导进行的，虽然该引导使得 COA 具有较好的开采能力，但组内郊狼的信息共享程度不高，导致前期种群多样性不足以及后期开采不足；3)在 COA 中，幼狼是由父狼基因和环境因素的共同影响决定的，使得算法具有一定的探索能力，但根据算法 1 可知，幼狼的诞生次数远远低于郊狼成长的次数，这使得 COA 的探索能力不足。针对以上问题，本文对 COA 进行了改进因素。

3.1 信息共享模型

企业信息化管理的首要因素是信息共享，能够实现信息共享才能在各个业务系统之间进行无障碍的信息交换。因此，一个成功的业务运营系统的关键在于构建一个完善的共享信息模型。

受企业信息化管理机制的启发，本文在郊狼的成长过程中引入信息共享模型。信息共享模型描述如下：在所有郊狼的社会状态因子的更新过程中，从当前组中随机选择两个不同的郊狼，然后计算两个郊狼的社会状态因子的差值，用于郊狼成长过程中社会状态因子的更新。根据算法 1 可知，每个郊狼在成长过程中都会随机选择两个郊狼，两个郊狼的社会状态因子的差值在不断变化。这种随机选择满足均匀分布，即在迭代 Nc 次之后，有可能组内所有的郊狼都已被选取，因此从总体情况看，这些社会状态因子的差值是组内所有郊狼贡献的信息。如果将此信息用于每个郊狼的成长，就能实现组内所有郊狼的信息共享，此模型被称为信息共享模型，用 $\boldsymbol{\delta}_3$ 表示，如式(12)所示：

$$\boldsymbol{\delta}_3 = \boldsymbol{soc}_{cr_4} - \boldsymbol{soc}_{cr_3} \tag{12}$$

其中，$\boldsymbol{\delta}_3$ 用于组内所有郊狼的社会状态因子的更新过程中。这种组内郊狼之间信息的随机交换，使得郊狼可以共享组内其他郊狼的信息。另外，在郊狼成长的前期，组内郊狼个体差异性较大，则由式(12)可知，$\boldsymbol{\delta}_3$ 值也较大，即共享信息值较大；反之在郊狼成长的后期，组内郊狼个体差异性较小，则 $\boldsymbol{\delta}_3$ 值较小，即共享信息值较小。如果在 COA 中利用此共享信息来引导组内每个郊狼的成长，则可以增强算法的搜索能力。

3.2 新型成长方式

如前所述，COA 中郊狼的成长方式使得组内郊狼在整个成长过程中一直朝着 ***alpha*** 和 ***cult*** 的方向移动，但此成长过程存在如下缺陷：1)***alpha*** 和 ***cult*** 对于同一个组来说是不变的，虽然在其引导下算法的开采能力强，但由其引导产生的新解前期多样性不足，使得后期开采的强度不够；2)如果 ***alpha*** 和 ***cult*** 是局部最优点，那么在其引导下算法很容易陷入局部最优，使得郊狼找到全局最优解的概率降低。

针对 COA 不能很好地平衡全局和局部搜索能力的问题，本文构建一种新型的成长方式，如式(13)所示。

$$\begin{aligned}\boldsymbol{new\_soc} = \boldsymbol{soc} + (gen/Maxgen) \times (r_1 \times \boldsymbol{\delta}_1 + r_2 \times \boldsymbol{\delta}_2) + \\ (1 - gen/Maxgen) \times r_3 \times \boldsymbol{\delta}_3\end{aligned} \tag{13}$$

其中，gen 是当前迭代数，$Maxgen$ 是最大迭代数，r_3 是信息共享模型的权重，且为[0,1]范围内的均匀分布的随机数。

比较式(13)和式(6)可知，式(13)是在式(6)的基础上增加了第 4 项：基于信息共享模型的扰动项，并采用归一化权重耦合原算法的第 2 项与第 3 项。在搜索的前期，由于 gen 的值较小，$gen/Maxgen$ 是一个较小的数，因此组内郊狼主要采用信息共享模型的成长方式，以郊狼的信息共享为主，以原 COA 成长方式为辅。由信息共享模型分析可知，前期 $\boldsymbol{\delta}_3$ 的值较大，即共享信息值较大，且 $\boldsymbol{\delta}_1$ 与 $\boldsymbol{\delta}2$ 的值都较大，以此增加种群的多样性，提高全局搜索能力。在搜索的后期，随着迭代次数的增加，$gen/Maxgen$ 逐渐增大，$(1-gen/Maxgen)$ 逐渐减小，此时郊狼的成长方式以 COA 的成长方式为主，以郊狼的信息共享为辅，在 ***alpha*** 和 ***cult*** 的共同引导下，算法具有较强的开采能力，另外，$\boldsymbol{\delta}_1$，$\boldsymbol{\delta}_2$ 以及 $\boldsymbol{\delta}_3$ 的值均较小，也强化了开采能力，使得局部搜索能力增强。信息共享模型的引入提高了 COA 的搜索能力，很好地平衡了全局搜索能力和局部搜索能力，使得算法的优化性大幅度提升。

3.3 组外贪心选择策略

COA 采用的是组内贪心选择策略，如算法 2 所示。可以看出，贪心选择位于组内成长过程中，在每个郊狼成长之后归正成长范围，并评估其社会适应能力，选择贪心算法优胜劣汰，即如果成长后的郊狼的社会适应能力较原来的好，则使其替代成长前的郊狼，反之，则保持不变。随后，成长后的社会适应能力好的郊狼会立即进入当前组，参与下一个郊狼的成长过程，使得以后组内郊狼在每一次成长过程中都朝着组内更优方向快速移动。这种组内贪心算法也称动态贪心算法，其优点是提高了局部搜索能力，加快了收敛速度，但易陷入局部最优，导致算法的稳定性较差。

算法 2 组内贪心算法

```
for p=1:Np
    确定组内 alpha
    由式(3)计算组文化趋势
    for c=1:Nc
      郊狼成长(式(6))，并归正成长范围
      评估成长后郊狼的社会适应能力(式(7))
      贪心选择适应能力好的成长后郊狼(式(8))
    end for
    幼狼的出生和死亡(式(9))，如幼狼活，则其年龄为 0
end for
```

算法 3 组外贪心算法

```
for p=1:Np
    确定组内 alpha
    由式(3)计算组文化趋势
    for c=1:Nc
      郊狼成长(式(6))
    end for
    并行归正每个郊狼的范围
    并行评估成长后郊狼的社会适应能力(式(7))
    for p=1:Np
      sp = { snew_sp,  new_fitnessp < fitnessp
           { sp,       otherwise
    end for
    幼狼的出生和死亡(式(9))，如幼狼活，则其年龄为 0
end for
```

为了提高 COA 的稳定性和探索能力，将 COA 的组内贪心算法改成组外贪心算法，如算法 3 所示。不同于组内贪心算法，组外贪心算法是等组内所有郊狼完成成长后，归正每个郊狼的成长范围，评估它们的社会适应能力，再进行贪心选择，这种贪心算法放置组外。由于成长后的郊狼不参与组内其他郊狼的成长过程，故这种组外贪心算法又称为静态贪心算法。虽然该算法不像动态贪心算法能够加快算法的收敛速度，但可以提高算法的稳定性，同时可以并行归正所有郊狼的成长范围以及并行计算所有郊狼社会适应能力，提高了运行速度。组外贪心选择策略不能加快算法的收敛速度，因此可以由 3.2 节引入的新的成长方式加以解决，以此提高算法的搜索能力，使得改进算法的总体性能较 COA 好。

3.4 ISCOA 总流程

ISCOA 由组内构建基于信息共享模型的新的成长方式和组外贪心算法构成，其伪代码如算法 4 所示。

算法 4 ISCOA 伪代码

```
1. 开始
2. 设置 Np 和 Nc 等参数，郊狼群随机初始化(式(1))并随机分组；
3. 计算每个郊狼的社会适应能力，令 year=0；
4. while 没有达到停止标准
5.    for p=1:Np
6.      确定组内 alpha
7.      由式(3)计算组文化趋势
8.        for c=1:Nc
9.          郊狼成长(式(13))
10.       end for
11.       并行归正成长范围和评估成长后郊狼的社会适应能力
12.       贪心选择适应能力好的成长后郊狼(式(8))
13.       幼狼的出生和死亡(式(9))，如幼狼活，则其年龄为 0
14.   end for
15. 郊狼被组驱离和接纳(式(11))
16. 更新每个郊狼的年龄
17. end while
18. 输出最好的郊狼
19. 结束
```

4 实验结果与分析

4.1 优化性能分析

为了测试 ISCOA 的优化性能，本文采用复杂函数集 CEC2017[9] 进行实验。CEC2017 函数集分为以下 4 类：单峰函数(F_1-F_3)、简单的多峰函数(F_4-F_{10})、混合函数($F_{11}-F_{20}$)和组合函数($F_{21}-F_{30}$)。为了公平比较，依据文献[9]的公共参数最佳推荐，本文设置最大目标函数评价次数为 $10000*D$(D 等于 10 或 30)，独立运行 51 次。本文采用均值(Mean)和方差(Std)作为评价各算法性能的标准，均值越小则算法的优化性能越高；方差越小，则算法的稳定性越好。

实验环境如下：Inter(R) Core(TM) i5-7400CPU，3.00GHz，内存 8GB，64 位操作系统 Windows10，编程语言 MATLAB2017A。

将 ISCOA 的结果与 COA 以及最先进的算法的结果进行比较，这些算法包括分层引力搜索算法(Hierarchical Gravitational Search Algorithm，HGSA)[10]、GWO、烟花算法(Fireworks Algorithm，FWA)[11]、“教与学”优化算法(Teaching-Learning-Based Optimization，TLBO)[12]、交叉搜索的粒子群优化算法(PSO using Crisscross Search，CSPSO)[13]以及萤火虫和粒子群的混合算法(Hybrid Firefly and Particle Swarm Optimization Algorithm，HFPSO)[14]。其中，GWO 与 COA 同属狼群算法；PSO 是当前最流行的群智能优化算法，本文选用 PSO 的最新改进版，即 CSPSO 和 HFPSO 作为对比算法；HGSA 是 GSA 的改进的优秀算法；FWA 和 TLBO 是目前比较主流的算法代表。这些对比算法具有可比性和代表性。为了公平起见，ISCOA 与 COA 的参数设置相同，依据文献[8]推荐的最佳参数，取郊狼种群个数为 100，Np 和 Nc 分别为 10。对比算法 HGSA，FWA，TLBO，GWO，CSPSO 以及 HFPSO 的参数设置分别参照其相应文献的最佳参数设置。

由于本文采用文献[10]的 HGSA 作为对比算法，并直接采用其数据，而此文献仅用了除 F_2 以外的 29 个基准函数，为了公平起见，本文也选取同样的 29 个 CEC2017 的基准函数。除 HGSA 外的其他对比算法的数据均来自本文的实验。因版面所限，本文在 4 种函数类型中各选取 2 个函数，各个算法在 10 维及 30 维 CEC2017 上的实验结果以及算法的排名情况分别如表 1 和表 2 所列，其余函数的均值和方差详见 OSID 码。其中，排名标准见文献[3]，在表 1 和表 2 中，最优的结果用黑体表示。

表 1　8 种算法在 10 维函数上的实验结果对比

Table 1　Comparison results of 8 algorithms on 10-dimensional functions

Function	Value	HGSA	GWO	FWA	TLBO	CSPSO	HFPSO	COA	ISCOA
	Mean	4.7200×10^{2}	7.3584×10^{6}	3.2036×10^{5}	1.7673×10^{3}	3.1244×10^{2}	2.9673×10^{3}	3.5849×10^{2}	**2.4259**
F_1	Std	6.8500×10^{2}	4.8600×10^{7}	1.7102×10^{5}	2.1250×10^{3}	3.4549×10^{2}	3.5032×10^{3}	1.8467×10^{3}	**1.1542**
	Rank	4	8	7	5	2	6	3	**1**
	Mean	3.0000×10^{2}	6.5064×10^{2}	1.1533×10^{2}	5.2385×10^{-14}	4.3316×10	$\mathbf{3.5667\times10^{-14}}$	4.7382×10	2.4238×10^{-3}
F_3	Std	1.1800×10^{-8}	1.0266×10^{3}	1.2985×10^{2}	3.3808×10^{-14}	1.9936×10	$\mathbf{2.7756\times10^{-14}}$	3.1598×10	2.7337×10^{-3}
	Rank	7	8	6	2	4	**1**	5	3
	Mean	4.0000×10^{2}	1.1013×10	1.0912×10	1.0203×10^{-1}	4.6923	8.9402×10^{-1}	5.7436×10^{-3}	$\mathbf{7.7326\times10^{-4}}$
F_4	Std	4.3300×10^{-2}	1.0993×10	1.7893×10	6.6844×10^{-2}	1.3128	4.2472×10^{-1}	3.9789×10^{-3}	$\mathbf{5.8457\times10^{-4}}$
	Rank	8	7	6	3	5	4	2	**1**
	Mean	2.2600×10^{3}	5.2248×10^{2}	5.8638×10^{2}	5.0993×10^{2}	5.5813×10^{2}	3.8871×10^{2}	3.0531×10^{2}	$\mathbf{2.7712\times10^{2}}$
F_{10}	Std	2.0200×10^{2}	2.3269×10^{2}	2.5359×10^{2}	3.3028×10^{2}	$\mathbf{1.2961\times10^{2}}$	1.8130×10^{2}	1.5195×10^{2}	1.6262×10^{2}
	Rank	8	5	7	4	6	3	2	**1**
	Mean	1.1200×10^{3}	2.8334×10	1.9435×10	6.0699	7.1697	7.3091	**1.3212**	1.4423
F_{11}	Std	7.3800	2.5029×10	7.9806	4.4027	1.3788	5.4042	$\mathbf{8.1139\times10^{-1}}$	9.1466×10^{-1}
	Rank	8	7	6	3	4	5	**1**	2
	Mean	2.1500×10^{3}	5.0770×10	4.8741×10	1.6276×10	1.6101×10	5.3118×10	$\mathbf{2.8193\times10^{-1}}$	3.4773×10^{-1}
F_{20}	Std	4.4000	3.3568×10	5.7219×10	1.0266×10	6.9943	5.3728×10	$\mathbf{4.3232\times10^{-1}}$	5.1775×10^{-1}
	Rank	8	6	5	4	3	7	**1**	2
	Mean	2.3100×10^{3}	1.8305×10^{2}	1.4856×10^{2}	1.5169×10^{2}	$\mathbf{1.1497\times10^{2}}$	1.6165×10^{2}	1.5507×10^{2}	1.1609×10^{2}
F_{21}	Std	4.3300×10	4.9586×10	5.8301×10	5.3522×10	$\mathbf{2.3815\times10}$	5.8544×10	5.6683×10	3.8382×10
	Rank	8	7	3	4	**1**	6	5	2
	Mean	1.4200×10^{4}	4.5720×10^{5}	4.6536×10^{5}	8.4648×10^{4}	9.0978×10^{4}	2.5583×10^{5}	2.8790×10^{4}	$\mathbf{1.7257\times10^{3}}$
F_{30}	Std	7.1600×10^{3}	8.7799×10^{5}	5.7852×10^{5}	2.2375×10^{5}	4.5638×10^{4}	4.6752×10^{5}	1.2781×10^{5}	$\mathbf{1.3965\times10^{3}}$
	Rank	2	7	8	4	5	6	3	**1**
Count		0	0	0	1	3	2	6	17
Ave. Rank		7.28	6.14	6.52	3.48	4.00	4.72	2.38	1.48
Total. Rank		8	6	7	3	4	5	2	1

表 2　8 种算法在 30 维函数上的实验结果对比

Table 2　Comparison results of 8 algorithms on 30-dimensional functions

Function	Value	HGSA	GWO	FWA	TLBO	CSPSO	HFPSO	COA	ISCOA
	Mean	2.6800×10^{3}	7.3300×10^{8}	4.3987×10^{6}	2.9846×10^{3}	1.5356×10^{3}	3.9338×10^{3}	1.2099×10^{3}	$\mathbf{1.4621\times10^{2}}$
F_1	Std	2.5000×10^{3}	6.0900×10^{8}	1.4055×10^{6}	3.1471×10^{3}	1.1306×10^{3}	5.3689×10^{3}	1.2998×10^{3}	$\mathbf{1.6248\times10^{2}}$
	Rank	4	8	7	5	3	6	2	**1**
	Mean	4.3600×10^{4}	2.7200×10^{4}	2.4748×10^{4}	4.0488×10^{-4}	1.7371×10^{4}	$\mathbf{1.5595\times10^{-7}}$	6.0573×10^{4}	1.9681×10^{4}
F_3	Std	5.4900×10^{3}	8.3200×10^{3}	6.3467×10^{3}	1.6647×10^{-3}	2.2981×10^{3}	$\mathbf{2.3334\times10^{-7}}$	1.0177×10^{4}	3.7975×10^{3}
	Rank	7	6	5	2	3	**1**	8	4
	Mean	5.1900×10^{2}	5.4100×10^{2}	1.1370×10^{2}	5.9054×10	1.0689×10^{2}	6.9386×10	8.4041×10	**5.0569×10**
F_4	Std	**2.6300**	3.3400×10	1.7315×10	3.0429×10	1.3722×10	2.1364×10	8.5306	3.2243×10
	Rank	7	8	6	2	5	3	4	**1**
	Mean	4.2100×10^{3}	4.0700×10^{3}	3.7800×10^{3}	6.0667×10^{3}	5.3225×10^{3}	2.9908×10^{3}	2.7575×10^{3}	$\mathbf{2.4479\times10^{3}}$
F_{10}	Std	$\mathbf{2.9300\times10^{2}}$	9.8000×10^{2}	5.9660×10^{2}	1.0625×10^{3}	3.0892×10^{2}	5.9210×10^{2}	4.6685×10^{2}	5.8631×10^{2}
	Rank	6	5	4	8	7	3	2	**1**
	Mean	1.2000×10^{3}	1.420×10^{3}	1.6164×10^{2}	1.2672×10^{2}	1.1374×10^{2}	1.1553×10^{2}	4.1143×10	**2.9942×10**
F_{11}	Std	2.9800×10	3.7800×10^{2}	4.5263×10	4.5717×10	2.3721×10	3.9628×10	2.7367×10	**2.5193×10**
	Rank	7	8	6	5	3	4	2	**1**
	Mean	2.8600×10^{3}	2.3700×10^{3}	4.6345×10^{2}	2.4392×10^{2}	2.9224×10^{2}	2.6516×10^{2}	2.4290×10^{2}	1.9885×10^{2}
F_{20}	Std	2.2400×10^{2}	1.4000×10^{2}	1.7129×10^{2}	8.4432×10	4.6404×10	1.1737×10^{2}	1.4995×10^{2}	1.2411×10^{2}
	Rank	8	7	6	3	5	4	2	**1**
	Mean	2.4100×10^{3}	2.3700×10^{3}	3.7768×10^{2}	2.6988×10^{2}	3.7692×10^{2}	2.7446×10^{3}	2.5626×10^{2}	$\mathbf{2.4719\times10^{2}}$
F_{21}	Std	5.9000×10	1.9600×10	7.9462×10	1.9589×10	2.2070×10	1.9517×10^{3}	1.6800×10	**1.3038×10**
	Rank	7	6	5	3	4	8	2	**1**
	Mean	1.1000×10^{4}	4.2600×10^{6}	1.5965×10^{4}	$\mathbf{5.9572\times10^{3}}$	2.8412×10^{4}	1.8733×10^{4}	6.0618×10^{3}	7.0239×10^{3}
F_{30}	Std	2.6000×10^{3}	3.8900×10^{6}	8.7877×10^{3}	3.9139×10^{3}	1.9147×10^{4}	3.4470×10^{4}	4.7022×10^{3}	$\mathbf{1.2925\times10^{3}}$
	Rank	4	8	5	**1**	7	6	2	3
Count		0	0	0	4	3	2	2	18
Ave. Rank		6.38	7.34	5.86	3.79	4.03	4.03	2.86	1.69
Total. Rank		6	7	5	3	4	4	2	1

从表 1 可知，在 29 个 10 维的基准函数上，ISCOA 获得了 17 次排名第一；COA 获得了 6 次排名第一；CSPSO，HFPSO 和 TLBO 分别获得了 3 次、2 次及 1 次排名第一；HGSA，GWO 以及 FWA 没有获得排名第一。SCOA 的平均排名值为 1.48，总排名第一。各算法在 10 维基准函数上实验结果的排名是：ISCOA，COA，TLBO，CSPSO，HFPSO，GWO，FWA 和 HGSA。因此，与 COA 以及对比算法相比，ISCOA 获得了更好的优化性能，值得注意的是，COA 的优化性能优于其他对比算法，这说明 COA 是一种优秀的 SIOA，对其进行改进具有较大意义。

另外，与 COA 相比，ISCOA 在 20 个测试函数上有更小的方差值，表明 ISCOA 有更好的稳定性。综上所述，与 COA 相比，ISCOA 有更好的优化性能和更好的稳定性。由表 2 可知，在 29 个 30 维的基准函数上，ISCOA 获得了 18 次排名第一，6 次排名第二，2 次排名第三，2 次排名第四，1 次排名第五，没有排名第六、第七和第八的情况。而 COA，HGSA，GWO，FWA，TLBO，CSPSO 及 HFPSO 分别获得 2 次、0 次、0 次、0 次、4 次、3 次及 2 次排名第一。在 23 个函数上 ISCOA 的方差都小于 COA 的方差。表 2 的结果再次证明，与 COA 相比，ISCOA 的搜索能力更强、稳定性更好。

ISCOA 有出色的表现是因为其在 COA 中引入了信息共享模型，在搜索前期增加了种群的多样性，提高了 COA 的全局搜索能力；在后期以原算法为主，信息共享模型为辅，使其具有更强的开采能力。实验结果表明，以上的改进较好地解决了 COA 的搜索能力不足的问题；另外，ISCOA 的稳定性更强是因为 ISCOA 将 COA 中的组内贪心算法替换为组外贪心算法等，证明了本文的改进是有效的。

此外，为了验证 ISCOA 的运行速度，将 ISCOA 与 COA 在 10 维和 30 维 CEC2017 函数上的运行时间进行比较，51 次独立运行后的平均运行时间如图 1 所示。

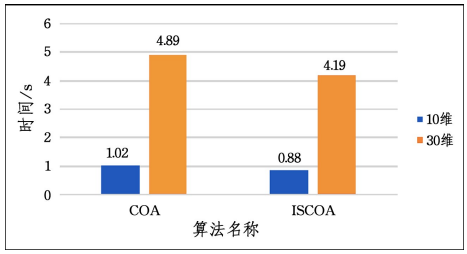

图 1 两种算法在 10 维和 30 维函数上的平均运行时间对比

Fig. 1 Comparison of average running time of two algorithms on 10D and 30D functions

由图 1 可知，ISCOA 与 COA 在 10 维函数上的平均耗时分别为 0.88 s 和 1.02 s，前者是后者的 86.3%；在 30 维函数上的平均耗时分别为 4.19 s 和 4.89 s，前者是后者的 85.7%。在 10 维和 30 维上，ISCOA 的平均耗时均小于 COA 的平均耗时，这表明 ISCOA 较好地解决了 COA 运行时间长的问题，这归功于 ISCOA 把 COA 中的组内贪心算法改成组外贪心算法，使得并行处理成为可能，即目标函数的计算和成长范围的归正操作能够并行处理，从而减少了算法的运行时间。

4.2 收敛性分析

为了验证 ISCOA 的收敛性能，本文采用 28 个 10 维 CEC2017 基准函数进行对比实验。因版权所限，选取部分函数做收敛性分析，选取的函数与 4.1 节相同。代表性函数的收敛如图 2 所示。其余函数的收敛图详见 OSID 码。图 2 展示了 ISCOA 与 COA，GWO，FWA，TLBO 以及 HFPSO 的收敛曲线对比图。

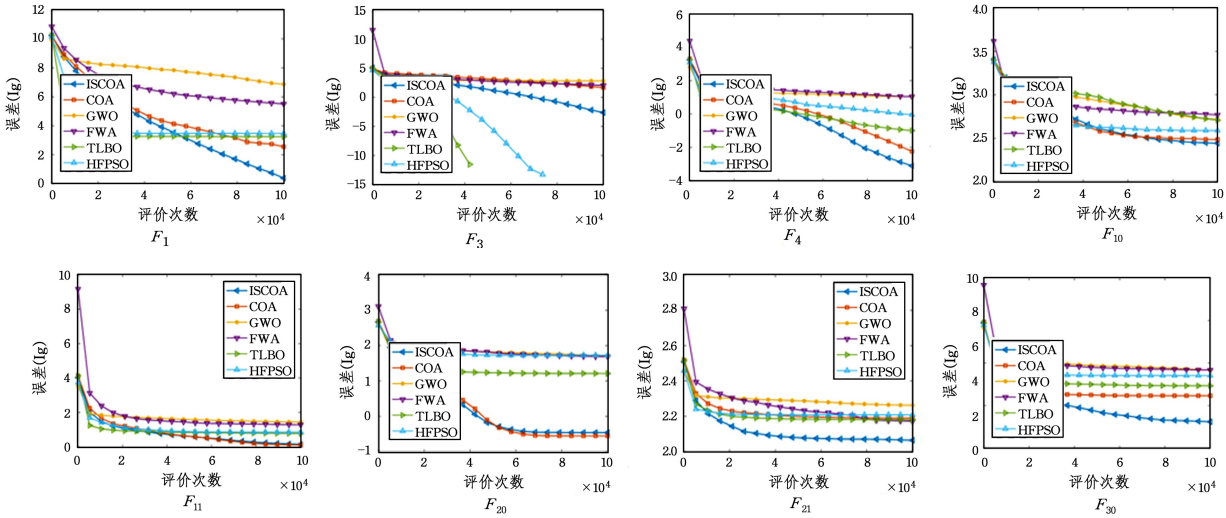

图 2 不同算法在 10 维函数上的收敛图对比

Fig. 2 Comparison convergence curves of different algorithms on 10-dimensional functions

从图 2 可知，随着目标函数评价次数的增加，在 F_1 和 F_3 单峰函数上，ISCOA 的收敛速度明显比 COA 的收敛速度快，表明在局部搜索能力方面 ISCOA 比 COA 好。与 GWO，FWA，TLBO 以及 HFPSO 相比，在 F_1 上，随着目标函数评价次数的增加，ISCOA 的收敛速度是最快的；在 F_3 上，ISCOA 比 TLBO 与 HFPSO 的收敛速度稍差，但比 GWO 和 FWA 的收敛速度快；在多峰函数上，除在 F_5，F_7 和 F_9 上的收敛速度不明显外，在 F_4，F_6，F_9 及 F_{10} 上，ISCOA 的收敛速度优于

COA。故与 COA 相比，ISCOA 在多峰函数上的收敛性能更好。与 GWO，FWA，TLBO 和 HFPSO 相比，在 F_7 上 ISCOA，TLBO 和 HFPSO 三者的收敛速度相近，但 ISCOA 明显比 GWO 和 FWA 的收敛速度快；在 F_4，F_5，F_6 及 F_{10} 上，ISCOA 的收敛速度优于 GWO，FWA，TLBO 和 HFPSO 的收敛速度。故与 GWO，FWA，TLBO 和 HFPSO 相比，在多峰函数上，ISCOA 的收敛性具有更强的竞争性。在 $F_{11}-F_{30}$ 的混合函数和复合函数上，除了在 F_{20} 和 F_{29} 上 ISCOA 收敛性能稍差外，ISCOA 的收敛性能都优于 COA 的收敛性能，其虽在 F_{11}，$F_{13}-F_{15}$，F_{19}，F_{22}，F_{25} 及 F_{27} 上优势不明显，但在其他函数上优势很明显，证明了 ISCOA 在处理复杂优化问题的能力上大体优于 COA。与 GWO，FWA，TLBO 及 HFPSO 相比，除了在 F_{24} 上 ISCOA 较 FWA 的收敛性能稍差外，ISCOA 的收敛性能都优于这 4 个算法的收敛性能，其虽在 F_{15}，F_{22}，F_{25} 及 F_{29} 上优势不明显，但在其他函数上优势很明显，证明了 ISCOA 在处理复杂优化问题的能力上大体优于 GWO，FWA，TLBO 及 HFPSO。从总体上，不管是单峰和多峰函数，还是混合和复合函数，与 COA，GWO，FWA，TLBO 及 HFPSO 相比，ISCOA 均有更好的收敛性能。这也说明本文提出的信息共享模型和新型成长方式不仅提高了全局搜索能力，也强化了局部搜索能力。

4.3 Wilcoxon 符号秩检验

Wilcoxon 符号秩检验方法[15]是一种非参数统计性检验方法，本文采用该方法检验 ISCOA 与 COA，HFPSO，CSPSO，TLBO，FWA，GWO 和 HGSA 的显著性差异。Wilcoxon 符号秩检验数据取自表 1、表 2 以及 OSID 中的表，当 ISCOA 优于对比算法时，其秩为正；当 ISCOA 劣于对比算法时，其秩为负。表 3 列出了 ISCOA 和对比算法在不同维度上的 Wilcoxon 检验的结果。其中，p 值代表置信度；R^+ 表示正秩的总和，R^- 表示负秩的总和，当 ISCOA 与对比算法的性能相同时，对应的秩平均分给 R^+ 和 R^-；$n/w/t/l$ 分别表示在 n 个函数上 ISCOA 取得了 w 次优，t 次性能相同，l 次差的结果。

表 3 Wilcoxon 符号秩检验结果

Table 3 Wilcoxon sign rank test results

	在 CEC2017 测试集 10 维函数上的结果			
	p-value	R^+	R^-	$n/w/t/l$
ISCOA vs COA	1.0532×10^{-3}	369	66	29/23/0/6
ISCOA vs HFPSO	3.5150×10^{-6}	432	3	29/27/0/2
ISCOA vs CSPSO	9.5012×10^{-5}	378	37	29/26/0/3
ISCOA vs TLBO	2.5631×10^{-6}	427	8	29/27/0/2
ISCOA vs FWA	3.1652×10^{-6}	433	2	29/28/0/1
ISCOA vs GWO	2.5631×10^{-6}	435	0	29/29/0/0
ISCOA vs HGSA	2.5631×10^{-6}	435	0	29/29/0/0
	在 CEC2017 测试集 30 维函数上的结果			
	p-value	R^+	R^-	$n/w/t/l$
ISCOA vs COA	9.7536×10^{-4}	370	65	29/24/0/5
ISCOA vs HFPSO	1.6542×10^{-3}	363	72	29/25/0/4
ISCOA vs CSPSO	4.7915×10^{-4}	379	56	29/25/0/4
ISCOA vs TLBO	2.8181×10^{-2}	319	116	29/23/0/6
ISCOA vs FWA	2.5631×10^{-6}	435	0	29/29/0/0
ISCOA vs GWO	2.5631×10^{-6}	435	0	29/29/0/0
ISCOA vs HGSA	4.5822×10^{-5}	406	29	29/28/0/1

从表 3 可知，与 COA 相比，ISCOA 在 10 维和 30 维上的 p 值分别为 1.0532×10^{-3}，9.7536×10^{-4}，均小于 0.05，这表明在优化性能上 ISCOA 在置信水平为 0.05 上显著优于 COA。与其他对比算法相比，ISCOA 的 p 值也均小于 0.05，表明 ISCOA 显著优于对比算法。其中，与 GWO 相比，ISCOA 无论在 10 维还是 30 维函数上 p 值均小于 0.00001，说明 ISCOA 显著优于 GWO，其他情况不做赘述。

5 ISCOA 用于拉伸/压缩弹簧设计问题

为了验证 ISCOA 解决时间工程问题的能力，本文将其应用到经典的拉伸/压缩弹簧设计问题中。

5.1 拉伸/压缩弹簧设计问题

拉伸/压缩弹簧设计问题是一种受约束的工程设计问题，目的是在受到最小偏差、剪切应力、冲击频率、外径限制以及设计变量的限制下，最小化拉伸/压缩弹簧的重量。拉伸/压缩弦设计问题可以抽象为图 3，它有 3 个设计变量[16]，分别是线径 $d(x_1)$、平均线圈直径 $D(x_2)$及有效线圈数 $P(x_3)$。数学模型由式(14)表示，式(15)为不等式约束条件。

$$f(x)=(x_3+2)x_2x_1^2 \tag{14}$$

$$\begin{cases} g_1(x)=1-\dfrac{x_2^3x_3}{71785x_1^4}\leqslant 0 \\ g_2(x)=1-\dfrac{4x_2^2-x_1x_2}{12566(x_2x_1^3-x_1^4)}+\dfrac{1}{5108x_1^4}\leqslant 0 \\ g_3(x)=1-\dfrac{140.45x_1}{x_2^2x_3}\leqslant 0 \\ g_4(x)=\dfrac{(x_1+x_2)}{1.5}-1\leqslant 0 \end{cases} \tag{15}$$

其中，$0.05\leqslant x_1\leqslant 2.00$，$0.25\leqslant x_2\leqslant 1.30$，$2.00\leqslant x_3\leqslant 15.0$。

图 3 拉伸/压缩弹簧设计问题

Fig. 3 Tension/compression string design problem

5.2 将 ISCOA 用于拉伸/压缩弹簧设计问题

为了验证 ISCOA 用于工程应用问题上的有效性，本文将 ISCOA 与 COA，GWO，TLBO，HPSO，CPSO 以及 CDE 进行了比较，其中 ISCOA，COA，GWO 以及 TLBO 的参数设置如下：$N=50$，$Num=30$，$MaxDT=800$，HPSO，CPSO 以及 CDE 的数据直接取自文献[17]，需要注意的是，NA 表示没有相应的数据。

对获得最佳解决方案的不同算法进行比较，结果如表 4 所列，包括 3 个变量组成的最佳解向量、目标函数值最佳值(Best)、最差值(Worst)、平均值(Mean)、标准方差(Std)和以及各个算法的平均运行时间(Time)。从表 4 可知，与 COA，GWO，TLBO，HPSO，CPSO 以及 CDE 相比，在 Best，Worst，Mean 和 Std 上，ISCOA 均获得更好结果且运行时间更少，特别是与 HPSO，CPSO 以及 CDE 相比，在最大目标函数评价次数 $MaxFEs$ 较少的情况下，ISCOA 获得了更好的结果，这说明 ISCOA 能更好地解决拉伸/压缩弹簧设计问题。

表 4 用于拉伸/压缩弹簧设计问题的比较结果

Table 4 Comparison results for tension/compression string design problem

	ISCOA	COA	GWO	TLBO	HPSO	CPSO	CDE
x_1	0.051702	0.051682	0.051584	0.051691	0.051706	0.051728	0.051611
x_2	0.357018	0.356539	0.354166	0.356756	0.357126	0.357644	0.354714
x_3	11.2714	11.2995	11.4446	11.2868	11.26508	11.24454	11.4108
Best	0.012665	0.012665	0.01267	0.012665	0.012665	0.012675	0.012670
Worst	0.012665	0.012719	0.013134	0.012735	0.012719	0.012924	NA
Mean	0.012665	0.012669	0.012735	0.012685	0.012707	0.012730	0.012703
Std	1.5×10^{-8}	1.4×10^{-5}	9.7×10^{-5}	1.8×10^{-5}	1.6×10^{-5}	5.2×10^{-4}	NA
Time/s	1.0515	1.496	1.1269	1.6097	NA	NA	NA
MaxFEs	4.0×10^{4}	4.0×10^{4}	4.0×10^{4}	4.0×10^{4}	8.1×10^{4}	2.4×10^{5}	2.4×10^{5}

结束语 针对 COA 中存在运行时间长和搜索能力不足等问题，本文提出一种改进的郊狼优化算法，即 ISCOA。首先，构建一种新型的信息共享模型，并用于组内郊狼的成长，使得郊狼的成长不仅受到 ***alpha*** 狼和文化环境等的影响，还受到组内共享信息的影响；其次，构建一种新型的搜索方式，前期以郊狼的信息共享为主来强化成长过程和探索能力，后期以原算法的成长方式，即以 ***alpha*** 狼和文化趋势的引导为主强化开采能力；最后，将原算法的组内贪心算法替换成组外贪心算法，以提高算法的稳定性以及目标函数计算等操作的并行处理，从而提高运行速度。在 10 维和 30 维 CEC2017 复杂函数上的实验结果表明，与 COA 以及其他先进算法相比，ISCOA 具有更好的优化性能和更强的稳定性，能很好地解决 COA 存在的一些问题。运用于拉伸/压缩弹簧设计问题中的实验结果表明，ISCOA 同样获得了最好的结果。由于 COA 是新提出的算法，许多方面还需要进一步研究，如参数、拓扑结构、混合算子以及混合算法等，因此我们在未来将进一步改进和完善 COA，并扩展其应用领域。

参 考 文 献

[1] ZAHNG X M, WANG X, TU Q, et al. Particle swarm optimization algorithm combining example learning and opposition-based learning[J]. Journal of Henan Normal University (Natural Science Edition), 2017, 45(6): 97-105.

[2] YANG J S, ZENG B Q, HU P P. Spectrum allocation and power control based on harmony search algorithm in cognitive radio network[J]. Computer Science, 2015, 42(S2): 257-262.

[3] ZHANG X M, CHENG J F, WANG X, et al. Improved Shuffled Frog Leaping Algorithm and Its in Multi-threshold Image Segmentation [J]. Computer Science, 2018, 45(8): 54-62.

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

[5] ZAHNG X M, WANG X, KANG Q, et al. Hybrid grey wolf optimizer with artificial bee colony and itsapplication to clustering optimization[J]. Acta Electronica Sinca, 2018, 46(10): 2430-2442.

[6] WANG G G, TAN Y. Improving metaheuristic algorithms with information feedback models[J]. IEEE Transactions on Cybernetics, 2017, 99: 1-14.

[7] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67-82.

[8] PIEREZAN J, COELHO L D S. Coyote optimization algorithm: a new metaheuristic for global optimization problems[C]// 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2018: 1-8.

[9] AWAD N H, ALI M Z, LIANG J J, et al. Problem definitions and evaluation criteria for the CEC2017 specialsession and competition on single objective bound constrainedreal-parameter numerical optimization[R]. Nanyang Technological University, Singapore, Technical Report, 2016.

[10] WANG Y R, YU Y, GAO S C, et al. A hierarchical gravitational search algorithm with an effective gravitational constant [J]. Swarm and Evolutionary Computation, 2019, 46: 118-139.

[11] TAN Y, ZHU Y C. Fireworks Algorithm for Optimization [C] // International Conference in Swarm Intelligence. 2010: 355-364.

[12] RAO R V, SAVSANI V J, VAKHARIA D P. Teaching-learning-based optimization: A novel method for constrainedmechanical design optimization problems[J]. Computer-Aided Design, 2011, 43(3): 303-315.

[13] MENG A, LI Z, YIN H, et al. Accelerating particle swarm optimization using crisscross search[J]. Information Sciences, 2016, 329(SI): 52-72.

[14] AVDILEK I B. A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems [J]. Applied Soft Computing, 2018, 66: 232-249.

[15] DERRAC J, GARCIA S, MOLINA D, et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms [J]. Swarm and Evolutionary Computation, 2011, 1(1): 3-18.

[16] BAYKASOGLU A, AKPINAR Ş. Weighted superposition attraction (WSA): a swarm intelligence algorithm for optimization problems — part 2: constrained optimization [J]. Applied Soft Computing, 2015, 37: 396-415.

[17] LU C, GAO L, YI J. Grey wolf optimizer with cellular topological structure[J]. Expert Systems with Applications, 2018, 107: 89-114.

ZHANG Xin-ming, born in 1963, professor, master's supervisor, is a member of China Computer Federation. His main research interests include intelligent optimization algorithm and image segmentation.