蝉鸣优化:一种新的仿生进化算法

贺毅朝[1] 李 宁[1] 李文斌[2]
(石家庄经济学院信息工程学院 石家庄 050031)[1]
(石家庄经济学院网络信息安全实验室 石家庄 050031)[2]

摘 要 借鉴秋蝉鸣叫中表现出的某种同步化以及蝉的生活习性提出了一种新的仿生优化算法:蝉鸣优化(CSO),分析并指出了CSO除具有一般进化算法的特性外还具有两点独特的特性,并基于有限Markov链理论证明了CSO的渐近收敛性。利用CSO、PSO和DE对9个高维Benchmark函数的仿真计算比较表明:CSO是一种非常适于求解数值最优化问题的进化算法。

关键词 进化算法,蝉鸣方式,生存周期,渐近收敛性,Benchmark函数
中图法分类号 TP18 **文献标识码** A

Cicada Sing Optimization: A New Evolutionary Algorithm Based on Bionics

HE Yi-chao[1] LI Ning[1] LI Wen-bin[2]
(Information Engineering School, Shijiazhuang University of Economics, Shijiazhuang 050031, China)[1]
(Laboratory of Network & Information Security, Shijiazhuang University of Economics, Shijiazhuang 050031, China)[2]

Abstract Inspire of the synchronization of the cicada singing and the life habit of the cicada, this paper proposed a novel optimization algorithm based on bionics: Cicada Sing Optimization (CSO). Then analyzed and presented that besides the characters of the general evolutionary algorithms, it owns two more special characters, and proved its asymptotic convergence based on the Markov-chain theory. Through the comparision of simulating calculation results of 9 high dimension Benchmark functions by using CSO, PSO and DE, we can see: CSO is a kind of evolutionary algorithm very suitable to solve numerical optimization problems.

Keywords Evolutionary algorithm, Cicada sing mode, Survival period, Asymptotic convergence, Benchmark functions

近年来,人们基于仿生学提出了许多进化算法,其中具有较大影响的有粒子群优化(Particle Swarm Optimization, PSO)[1,2]、差分演化(Differential Evolution, DE)[3,4]、蚁群优化(Ant Colony Optimization, ACO)[5,6]、混合蛙跳算法(Shuffled Frog-Leaping Algorithm, SFLA)[7,8]、人工鱼群算法(Artificial Fish School Algorithm, AFSA)[9,10]和萤火虫算法(Firefly Algorithm, FA)[11,12],所有这些算法都是基于生物个体间的某种(部分)同步化,通过群体作用体现出的一致协同性来实现全局寻优的,在数值最优化、组合优化、神经网络优化和模糊控制等领域具有重要的应用[1-13]。虽然Wolpert和Macready[14]提出的"无免费午餐定理"(No Free Lunch Theorems, NFL)指出:在所有优化问题均以相同概率被求解的假设下,每个进化算法的平均表现度量是相同的。但是现实中的最优化问题往往是一类具有某种确定特征的特殊问题,所有的进化算法不可能都适于求解,因此对于不同的最优化问题设计更适于求解的进化算法是有积极意义的。

本文借鉴秋蝉鸣叫中表现出的某种同步化以及蝉的生活习性[15,16],提出了一种新的仿生优化算法——蝉鸣优化(Cicada Sing Optimization, CSO)。第1节介绍了CSO的原理,并给出了其算法伪代码描述;第2节分析了CSO的特性并证明了它的渐近收敛性;第3节利用9个高维Benchmark测试函数进行仿真计算,通过与PSO和DE的比较验证了CSO的可行性与高效性;最后总结全文并提出下一步的研究思路。

1 蝉鸣优化的原理与算法描述

众所周知,秋蝉的毫无计划的无休无止的鸣叫有时是很烦人的,然而人们发现秋蝉的鸣叫其实并非是无序的,它实质上是一种具有音乐节律的求偶歌,可以划分为5种不同的男生信号[15,16]。秋蝉的求偶歌将雄性和雌性都吸引到合唱团中,导致雄性的聚集,进而吸引更多的雌性并且达到局部同步化,从而产生聚优现象。每个秋蝉都有生存周期,在生存周期中秋蝉不断地鸣叫,直到吸引到异性为止。

本文从秋蝉的求偶歌使用5种信号构成一种音乐节律的现象得到启发,提出了一种基于5种不同进化方式的进化算子:五音进化算子(Five Timbre Evolution Operator, FTEO),通过利用FTEO产生新的个体,促进算法的不断进化。此外,借鉴秋蝉存在生存周期的自然现象,为算法中的每个个体设置一个生存因子(Survival factor)以评判个体的进化潜力

到稿日期:2013-09-01 返修日期:2013-10-21 本文受河北省教育厅自然科学基金项目(Z2013110),石家庄经济学院预研项目(2012-05)资助。
贺毅朝(1969—),男,硕士,教授,CCF高级会员,主要研究方向为智能计算、算法理论与计算复杂性,E-mail:heyichao119@163.com;李 宁(1977—),女,硕士,副教授,主要研究方向为人工智能;李文斌(1974—),男,博士,教授,硕士生导师,主要研究方向为机器学习与复杂网络。

和调节种群的多样性；当个体在每一代都不间断进化时，其生存因子保持不变，否则其生存因子将增大，当增大到超过预设的生存周期时，该个体将被一个随机产生的新个体所替代。

下面首先给出 FTEO、生存因子和贪心选择算子（Greedy Selection Operator，GSO）的定义，然后基于它们提出一种新的仿生优化算法——蝉鸣优化（CSO）。设 min $f(X)$ 为最小优化问题，$X\in \boldsymbol{S}$ 是问题的可行解，$\boldsymbol{S}$ 为解空间，于是有如下定义。

定义 1 设 $P(t)=\{X_1(t),X_2(t),\cdots,X_n(t)\}$ 为 CSO 的第 t 代种群，$Q(t)=\{Y_1(t),Y_2(t),\cdots,Y_n(t)\}$ 为第 t 代中间种群，其中 $X_i(t)=(x_{i1}(t),x_{i2}(t),\cdots,x_{id}(t))\in \boldsymbol{S}$，$Y_i(t)=(y_{i1}(t),y_{i2}(t),\cdots,y_{id}(t))\in \boldsymbol{S}$，$d$ 为问题的维数，n 为种群规模；又设 $X_{best}(t)=(x_{best,1}(t),x_{best,2}(t),\cdots,x_{best,d}(t))$ 为第 t 代的最优个体，$rand(5)$ 为 $\{1,2,3,4,5\}$ 中的一个随机数，则 FTEO 定义如下：

$$y_{ij}(t)=\begin{cases}A\times x_{best,j}(t)+(1-A)\times x_{p1,j}(t), & \text{if } rand(5)=1\\ A\times x_{ij}(t)+(1-A)\times x_{p1,j}(t), & \text{if } rand(5)=2\\ A\times x_{best,j}(t)+(1-A)\times (x_{p1,j}(t)-x_{p2,j}(t)), & \text{if } rand(5)=3\\ A\times x_{ij}(t)+(1-A)\times (x_{best,j}(t)-x_{p1,j}(t)), & \text{if } rand(5)=4\\ A\times x_{p1,j}(t)+(1-A)\times (x_{p2,j}(t)-x_{p3,j}(t)), & \text{if } rand(5)=5\end{cases}$$

其中，$1\leqslant i\leqslant n$，$1\leqslant j\leqslant d$；$p_1\neq p_2\neq p_3\neq i$，$A$ 为 $(0,1)$ 上的随机数。

定义 2 设阈值 *threshold* 为个体的生存周期，种群 $P(t)$ 中第 i 个个体的生存因子记为 $Sp_i(t)$，初始 $Sp_i(0)=0$，$1\leqslant i\leqslant n$。当产生新一代种群 $P(t+1)=\{X_1(t+1),X_2(t+1),\cdots,X_n(t+1)\}$ 后，如果 $f(X_i(t+1))\geqslant f(X_i(t))$，则 $Sp_i(t+1)=Sp_i(t)+1$，否则 $Sp_i(t+1)=Sp_i(t)$。

对于种群 $P(t+1)$ 中的个体 $X_i(t+1)$，如果其生存因子满足 $Sp_i(t+1)>threshold$，则用一个随机产生的新个体替换 $X_i(t+1)$。

定义 3 设 $P(t)=\{X_1(t),X_2(t),\cdots,X_n(t)\}$ 为第 t 代种群，$Q(t)=\{Y_1(t),Y_2(t),\cdots,Y_n(t)\}$ 为第 t 代中间种群，$P(t+1)=\{X_1(t+1),X_2(t+1),\cdots,X_n(t+1)\}$ 为第 $t+1$ 代新种群，则 GSO 的定义如下：

$$X_i(t+1)=\begin{cases}Y_i(t), & \text{if } f(Y_i(t))<f(X_i(t))\\ X_i(t), & \text{otherwise}\end{cases}$$

其中，$1\leqslant i\leqslant n$。

在 CSO 的进化过程中，当前种群 P 首先利用 FTEO 产生中间种群 Q，然后利用 GSO 通过对种群 P 和 Q 中相应个体的优劣比较，确定新一代种群 P_{new} 和当前最好个体 X_{best}；紧接着，检查 P_{new} 中每一个体的生存因子是否超过预设的生存周期，若超过则被一个随机产生的新个体替换，否则进入下一代的进化操作。

对于最小优化问题 min $f(X)$，令 Max$T$ 为 CSO 的最大迭代次数，$rand(0,1)$ 表示区间 $(0,1)$ 上的一个随机数，$rand(5)$ 表示随机选取的 $\{1,2,3,4,5\}$ 中的一个数，则 CSO 的算法伪代码描述如下：

算法 1 CSO Algorithm

1. 随机生成初始种群 $P(0)=\{X_i(0)\mid 1\leqslant i\leqslant n\}$；置 $t\leftarrow 0$；
2. 计算 $f(X_i(t))$；置 $Sp_i(t)\leftarrow 0$，$1\leqslant i\leqslant n$；确定 $X_{best}(t)$；
3. While $t\leqslant$MaxT Do
4. For i=1 to n Do
5. For j=1 to d Do
6. A←rand(0,1)；
7. If rand(5)=1 then $y_{ij}(t)\leftarrow A*x_{best,j}(t)+(1-A)*x_{p1,j}(t)$；
8. Else If rand(5)=2 then $y_{ij}(t)\leftarrow A*x_{ij}(t)+(1-A)*x_{p1,j}(t)$；
9. Else If rand(5)=3 then $y_{ij}(t)\leftarrow A*x_{best,j}(t)+(1-A)*(x_{p1,j}(t)-x_{p2,j}(t))$；
10. Else If rand(5)=4 then $y_{ij}(t)\leftarrow A*x_{ij}(t)+(1-A)*(x_{best,j}(t)-x_{p2,j}(t))$；
11. Else $y_{ij}(t)\leftarrow A*x_{p1,j}(t)+(1-A)*(x_{p2,j}(t)-x_{p3,j}(t))$；
12. EndFor
13. If $f(Y_i(t))<f(X_i(t))$ then $X_i(t+1)\leftarrow Y_i(t)$，$Sp_i(t+1)\leftarrow Sp_i(t)$；
14. Else $X_i(t+1)\leftarrow X_i(t)$，$Sp_i(t+1)\leftarrow Sp_i(t)+1$；
15. If $Sp_i(t+1)>$threshold then 重新随机生成 $X_i(t+1)$；
16. EndFor
17. 确定 $P(t+1)$ 中的最优个体 $X_{best}(t+1)$；
18. $t\leftarrow t+1$；
19. EndWhile
20. Return(X_{best}(MaxT)，$f(X_{best}$(MaxT))

在算法 1 中，*threshold* 的大小通常与维数成正比，一般可取 $threshold=2d$，d 为函数的维数。注意到算法 1 的时间复杂度主要取决于 Step3－Step19，因此算法时间复杂度为 $O(\text{Max}T*n*d)$。显然，CSO 所涉及到的参数较少，因此其运行速度非常快。

2 CSO 的特征与收敛性分析

2.1 蝉鸣优化的特征分析

由 FTEO 的定义以及算法描述易知：CSO 具有进化算法的两个基本特性，即隐含并行性和全局信息的有效利用[17]。除此以外，CSO 还有两点明显不同于其它算法的特性：

（1）CSO 的进化模式不同于其它进化算法，它是一种作用于更低层次上的多模式进化。

这是因为 CSO 的 FTEO 是一种对个体各维分量分别采用 5 种不同进化操作的进化模式，而 PSO、DE 等其它进化模式对个体的各维分量采用相同的进化操作。CSO 的这种进化模式体现了 CSO 在分量层次综合使用多种进化操作以达到低层次、多方向全面搜索的目的。此外，其它进化算法的进化模式最多涉及 2 种进化操作，并且是直接作用于个体层面，而 CSO 同时使用 5 种进化操作（见定义 1），并且直接作用于个体的各维分量层面。

（2）在 CSO 中，每个个体都具有生存因子 Sp，当其生存因子 Sp 超过生存周期 *threshold* 时，该个体将消亡，取而代之的是一个随机产生的新个体。在其它进化算法中生存周期的概念是不存在的。

CSO 中个体的生存因子 Sp 实质上是个体进化潜力的一种度量，个体的 Sp 值越大表示个体的进化潜力越差，而 Sp 值越小表示个体的进化潜力越强。如果某个体恒保持 $Sp=0$，表明它在经历过的所有进化迭代中每次进化均产生了比前一代更优的结果，所以有理由相信它在之后的进化迭代中将仍然保持这种良好的进化势头；而如果一个个体的 $Sp>threshold$，则表明此个体在经历过的 $threshold+1$ 代进化迭

代中一直没有产生更好的结果，这样的个体再存在于当前种群显然是不适宜的，因此用一个随机产生的新个体取代它是一种明智的选择。

2.2 蝉鸣优化的渐近收敛性

不妨设 CSO 的计算精度为 $\varepsilon \leqslant 10^{-k}$，则问题的解空间 $\boldsymbol{S}=\prod_{i=1}^{d}[L_i,U_i]$，其中 $L_i<U_i$ 且 L_i 与 U_i 均为实数，于是 $|\boldsymbol{S}| \leqslant [10^k(U-L)]^d$，$U=\max\{\overline{U_i}\,|\,\overline{U_i}=\lceil U_i \rceil \wedge 1\leqslant i\leqslant d\}$，$L=\min\{\overline{L_i}\,|\,\overline{L_i}=\lfloor L_i \rfloor \wedge 1\leqslant i\leqslant d\}$。记 $F=\{f(X)\,|\,X\in\boldsymbol{S}\}$，则 $|F|\leqslant|\boldsymbol{S}|$。令 $F=\{F_1,\cdots,F_{|F|}\}$，其中 $F_1<F_2<\cdots<F_{|F|}$，则 $\{S_i\,|\,S_i=\{X\,|\,X\in\boldsymbol{S}\wedge f(X)=F_i\}\wedge 1\leqslant i\leqslant|F|\}$ 为 $\boldsymbol{S}$ 的一个划分，且 $\sum_{i=1}^{|F|}|S_i|=|\boldsymbol{S}|$，于是 F_1 为 $\min f(X)$ 的全局最优解，且 S_1 包含了适应度为 F_1 的所有个体。

记 CSO 的种群 $P=\{X_1,X_2,\cdots,X_n\}$ 的全体构成的集合为 $\boldsymbol{P}$，令 $f(P)=\min\{f(X_i)\,|\,X_i\in P\wedge 1\leqslant i\leqslant n\}$，则 $F_1\leqslant f(P)\leqslant F_{|F|}$，于是 $\boldsymbol{P}$ 有划分 $\{\boldsymbol{P}_i\,|\,\boldsymbol{P}_i=\{P\,|\,P\in\boldsymbol{P}\wedge f(P)=F_i\}\wedge 1\leqslant i\leqslant|F|\}$，$\sum_{i=1}^{|F|}|P_i|=|P|$，且 $\boldsymbol{P}_1$ 包含了所有满足 $f(X_{best})=F_1$ 的种群。令 P_{ij} 表示 $\boldsymbol{P}_i$ 中的第 j 个种群（$1\leqslant i\leqslant|F|$，$1\leqslant j\leqslant|P_i|$），从 P_{ij} 到 P_{kl}（$1\leqslant k\leqslant|F|$，$1\leqslant l\leqslant|P_k|$）的种群状态转移记为 $P_{ij}\to P_{kl}$；又令 $p_{ij,kl}$ 表示从 P_{ij} 到 P_{kl} 的转移概率，$p_{ij,k}$ 表示从 P_{ij} 到 $\boldsymbol{P}_k$ 中任一种群的转移概率，$p_{i,k}$ 表示从 $\boldsymbol{P}_i$ 中任一种群到 $\boldsymbol{P}_k$ 中任一种群的转移概率，则 $p_{ij,k}=\sum_{l=1}^{|P_k|}p_{ij,kl}$，$\sum_{k=1}^{|F|}p_{ij,k}=1$，$p_{i,k}\geqslant p_{ij,k}$。下面基于有限 Markov 链理论证明 CSO 是全局收敛的。

定义 4 设 $P(t)$ 是 CSO 的第 t 代种群，F^* 是全局最优适应度，如果

$$\lim_{t\to\infty}prob\{f(P(t))=F^*\}=1$$

成立，则称 CSO 是全局渐近收敛的。由于 $F^*=F_1$，因此式(3)又等价于 $\lim_{t\to\infty}prob\{P(t)\in\boldsymbol{P}_1\}=1$。

引理 1[18] 设 Q 是一个 n 阶可约随机矩阵，即可变换为 $Q=\begin{bmatrix}C & 0\\ R & T\end{bmatrix}$，$C$ 是一个 m 阶本原随机矩阵且 $R,T\neq 0$，则 $Q^\infty=\lim_{k\to\infty}Q^k=\lim_{k\to\infty}\begin{bmatrix}C^k & 0\\ \sum_{i=0}^{k-1}T^iRC^{k-i-1} & T^k\end{bmatrix}=\begin{bmatrix}C^\infty & 0\\ R^\infty & 0\end{bmatrix}$ 是一个稳定的随机矩阵且 $Q^\infty=1^{\mathrm{T}}q^\infty$，其中 $q^\infty=q^0\lim_{k\to\infty}Q^k=q^0Q^\infty$ 唯一确定且与初始分布 q^0 无关，而且 q^∞ 满足：当 $1\leqslant i\leqslant m$ 时，$q_i^\infty>0$，当 $m\leqslant i\leqslant n$ 时，$q_i^\infty=0$。

定理 1 在 CSO 中，对于 $\forall 1\leqslant j\leqslant|P_i|$，$1\leqslant i\leqslant|F|$，$1\leqslant k\leqslant|F|$ 有：当 $k\leqslant i$ 时，$p_{ij,k}>0$；当 $k>i$ 时，$p_{ij,k}=0$。

类似文献[19]中的方法容易证定理 1 成立。利用定理 1 及 $p_{i,k}$ 与 $p_{ij,k}$ 的关系，显然有下述结论：

定理 2 在 CSO 中，对 $\forall 1\leqslant i,k\leqslant|F|$ 有：当 $k\leqslant i$ 时 $p_{i,k}>0$；当 $k>i$ 时 $p_{i,k}=0$。

定理 2 表明：在 CSO 中适应度高的种群可以转移到适应度相同或更低的种群，反之却不然。因此算法一旦进入到集合 $\boldsymbol{P}_1$ 就不可能再逃逸出来。

定理 3 CSO 是全局渐近收敛的。

证明：设每个 $\boldsymbol{P}_i$（$1\leqslant i\leqslant|F|$）为齐次有限 Markov 链上的一个状态，根据定理 2，从任一状态 $\boldsymbol{P}_i$ 转变到另一状态 $\boldsymbol{P}_k$ 的转移概率满足 $p_{i,k}>0$（$\forall k\leqslant i$）与 $p_{i,k}=0$（$\forall k>i$），于是状态转移矩阵为：

$$Q=\begin{bmatrix}p_{1,1} & 0 & \cdots & 0\\ p_{2,1} & p_{2,2} & \cdots & 0\\ \vdots & \vdots & & \vdots\\ p_{s,1} & p_{s,2} & \cdots & p_{s,s}\end{bmatrix}=\begin{bmatrix}C & 0\\ R & T\end{bmatrix}$$

则根据引理 1 可得：

$$Q^\infty=\lim_{k\to\infty}Q^k=\lim_{k\to\infty}\begin{bmatrix}C^k & 0\\ \sum_{i=0}^{k-1}T^iRC^{k-i-1} & T^k\end{bmatrix}=\begin{bmatrix}C^\infty & 0\\ R^\infty & 0\end{bmatrix}$$

其中，$R^\infty=[(p_{2,1},\ p_{3,1},\cdots,p_{s,1})^{\mathrm{T}}]^\infty=(1,1,\cdots,1)^{\mathrm{T}}$，$C^\infty=(p_{1,1})^\infty=1$，$s=|F|$，所以 Q^∞ 是一个稳定的随机矩阵，说明迭代次数 t 足够大时，CSO 的种群 $P(t)$ 必属于 $\boldsymbol{P}_1$，即

$$\lim_{t\to\infty}prob\{P(t)\in\boldsymbol{P}_1\}=1$$

所以，CSO 是全局渐近收敛的。

3 仿真计算与比较

为了验证 CSO 的可行性与有效性，下面利用 CSO 求解被广泛用于测试进化算法优劣的 9 个高维 Benchmark 测试函数[20,21]（见表 1），并与当前最有影响的进化算法 PSO 和 DE 的计算结果进行比较。仿真计算使用 Lenovo X201i 笔记本，硬件环境为 Intel(R) Pentium(R) CPU：U5400-1.20GHz，2.00GB内存，操作系统为 Windows 7，并利用 Visual C++ 6.0进行编程实现。

表 1 9 个高维 Benchmark 测试函数

Benchmark 函数	搜索区间	最值
$f_1(X)=\sum_{i=1}^{d}x_i^2$	$[-100,100]^d$	0
$f_2(X)=\sum_{i=1}^{d}\lvert x_i\rvert+\prod_{i=1}^{d}\lvert x_i\rvert$	$[-10,10]^d$	0
$f_3(X)=\sum_{i=1}^{d}(\sum_{j=1}^{i}x_j)^2$	$[-100,100]^d$	0
$f_4(X)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant d\}$	$[-100,100]^d$	0
$f_5(X)=\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]^d$	0
$f_6(X)=\sum_{i=1}^{d}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^d$	0
$f_7(X)=\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]^d$	0
$f_8(X)=-20\exp(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{d}x_i^2})-\exp(\frac{1}{30}\sum_{i=1}^{d}\cos 2\pi x_i)+20+e$	$[-32,32]^d$	0
$f_9(X)=\frac{1}{4000}\sum_{i=1}^{d}x_i^2-\prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^d$	0

由于 CSO、PSO 和 DE 的一次进化迭代所耗费的时间相差较大，因此对它们进行比较时应统一设定算法的运行时间，为此设定维数 $d=100$ 时，运行时间为 8 秒；当维数 $d=300$ 时，运行时间为 20 秒；当维数 $d=500$ 时，运行时间为 35 秒。此外，最优误差设定为 $\varepsilon=10^{-25}$（即 1.0E−25），3 个算法的有关参数设置如下：CSO 的种群规模为 20，$threshold=2d$；PSO 的种群规模为 100，$W=1.0$，$C_1=C_2=1.8$；DE 的种群规模为 50，采用 DE/r/1/bin 模式，$CR=0.3$ 且 $FS=0.5$。

在表 2－表 4 中分别给出了 CSO、PSO 和 DE 求解 100 维、300 维和 500 维 Benchmark 函数的 30 次仿真计算所得最好结果(BEST)、平均结果(MEAN)和最差结果(WORST)；各种结果均以 fb/t 形式给出，其中 fb 为求解结果，t 为所用

时间(单位:秒)。对于 MEAN,fb 为平均求解结果,t 为平均时间。

从表 2—表 4 中的计算结果可以看出:除了 f_5 和 f_8 以外,CSO 能在设定的时间内极快地求得其它 7 个 Benchmark 测试函数的最优值;而对于 f_5 和 f_8,虽然 CSO 不能求得它们的最优值,但是其计算结果远远优于 PSO 和 DE。DE 在设定的时间内仅能求得 f_1,f_6 和 f_9 在维数 $d=100$ 时的最优值,对于 f_1,f_6 和 f_9 的维数 $d=300$ 和 $d=500$ 的情况以及其它 6 个 Benchmark 测试函数,DE 均不能求得最优值。而 PSO 在设定的时间内不能求出 100、300 和 500 维的任一个 Benchmark 测试函数的最优值,其求解效果在 3 种算法中明显是最差的。

表 2　100 维 Benchmark 测试函数的计算结果比较

测试函数	CSO			PSO			DE		
	BEST	MEAN	WORST	BEST	MEAN	WORST	BEST	MEAN	WORST
f_1	0.0/0.036	0.0/0.03758	0.0/0.04	5.598E2/8	8.304E2/8	1.43E3/8	0.0/3.763	0.0/3.9274	0.0/4.236
f_2	0.0/0.11	0.0/0.1219	0.0/0.192	42.57/8	65.8/8	90.12/8	1.61E−10/8	1.91E−10/8	2.87E−10/8
f_3	0.0/0.371	0.0/0.426	0.0/0.785	3.33E5/8	4.02E5/8	4.98E5/8	4.458E5/8	4.999E5/8	5.537E5/8
f_4	0.0/0.217	0.0/0.2418	0.0/0.312	59.31/8	61.964/8	65.19/8	10.57/8	14.3479/8	16.11/8
f_5	98.71/8	98.7893/8	98.853/8	3.17E6/8	4.47E6/8	6.13E6/8	70.29/8	87.2988/8	91.42/8
f_6	0.0/0.01	0.0/0.0135	0.0/0.07	27.0/8	41.25/8	52.0/8	0.0/1.837	0.0/1.9583	0.0/2.468
f_7	0.0/0.052	0.0/0.0628	0.0/0.069	2.214E3/8	2.355E3/8	3.74E3/8	578.92/8	587.176/8	626.06/8
f_8	3.99E−15/8	3.99E−15/8	3.99E−15/8	10.45/8	13.5225/8	17.70/8	2.74E−9/8	2.85E−9/8	3.33E−9/8
f_9	0.0/0.059	0.0/0.0634	0.0/0.067	14.66/8	18.6674/8	38.31/8	0.0/7.01	2.85E−13/7.9	8.28E−13/8

表 3　300 维 Benchmark 测试函数的计算结果比较

测试函数	CSO			PSO			DE		
	BEST	MEAN	WORST	BEST	MEAN	WORST	BEST	MEAN	WORST
f_1	0.0/0.12	0.0/0.1393	0.0/0.193	1.59E5/20	1.86E5/20	2.02E5/20	1.0488/20	1.44261/20	2.2235/20
f_2	0.0/0.349	0.0/0.3724	0.0/0.387	7.2E2/20	9.03E2/20	9.86E2/20	205.99/20	698.563/20	996.90/20
f_3	0.0/1.872	0.0/2.05388	0.0/2.659	3.15E6/20	3.84E6/20	4.25E6/20	4.72E6/20	5.53E6/20	6.98E6/20
f_4	0.0/0.701	0.0/1.9055	0.0/2.68	77.18/20	80.47/20	81.87/20	98.98/20	99.4384/20	99.74/20
f_5	298.69/20	298.75/20	298.8/20	3.08E8/20	4.06E8/20	4.49E8/20	1.09E7/20	1.39E7/20	2.88E7/20
f_6	0.0/0.361	0.0/0.3715	0.0/0.433	1.49E5/20	1.68E5/20	1.80E5/20	1.24E4/20	1.34E4/20	1.41E4/20
f_7	0.0/0.167	0.0/0.2015	0.0/0.321	1.47E5/20	1.65E5/20	2.08E5/20	3.28E3/20	3.35E3/20	3.53E3/20
f_8	3.99E−15/20	3.99E−15/20	3.99E−15/20	19.304/20	19.5791/20	19.686/20	13.157/20	13.4342/20	13.883/20
f_9	0.0/0.191	0.0/0.2035	0.0/0.259	1.28E3/20	1.56E3/20	1.93E3/20	232.77/20	259.665/20	307.99/20

表 4　500 维 Benchmark 测试函数的计算结果比较

测试函数	CSO			PSO			DE		
	BEST	MEAN	WORST	BEST	MEAN	WORST	BEST	MEAN	WORST
f_1	0.0/0.201	0.0/0.2176	0.0/0.247	4.87E5/35	4.99E5/35	5.23E5/35	4.57E4/35	5.0564/35	6.07E4/35
f_2	0.0/0.601	0.0/0.62341	0.0/0.687	1.88E3/35	1.95E3/35	2.27E3/5	2.26E3/35	2.38E3/35	2.42E3/35
f_3	0.0/3.899	0.0/4.5284	0.0/7.451	1.47E7/35	1.59E7/35	1.98E7/35	1.48E7/35	1.53E7/35	1.87E7/35
f_4	0.0/2.868	0.0/6.56245	0.0/13.652	81.92/35	84.57/35	86.97/35	99.499/35	99.9267/35	99.951/35
f_5	498.754/35	498.761/35	498.796/35	1.36E9/35	1.51E9/35	1.62E9/35	7.21E9/35	7.58 E9/35	8.94E9/35
f_6	0.0/0.71	0.0/0.8725	0.0/0.951	4.63E5/35	4.99E5/35	5.27E5/35	5.78E5/35	6.04E5/35	7.54E5/35
f_7	0.0/0.301	0.0/0.5512	0.0/0.744	4.65E5/35	5.13E5/35	5.45E5/35	7.57E3/35	7.62E3/35	8.15E3/35
f_8	3.99E−15/35	6.72E−15/35	7.55E−15/35	19.94/35	20.0412/35	20.084/35	20.459/35	20.5812/35	20.822/35
f_9	0.0/0.356	0.0/0.5945	0.0/0.773	4.72E3/35	4.85E3/35	5.17E3/35	7.89E3/35	9.76E3/35	1.03E4/35

通过以上对 CSO、PSO 和 DE 的仿真计算结果比较表明:对于所给出的 9 个高维 Benchmark 测试函数,CSO 的求解效果远远优于 PSO 和 DE,说明 CSO 是一种比 PSO 和 DE 更适于求解上述 Benchmark 测试函数的进化算法。

结束语　本文借鉴秋蝉鸣叫中表现出的某种同步化以及蝉的生活习性提出了一种新的仿生优化算法 CSO,既分析了 CSO 的独特性质及其全局渐近收敛性,又利用 9 个高维 Benchmark 函数进行了仿真计算。通过与 PSO 和 DE 的计算结果比较可以看出:CSO 是一种更适于求解数值优化问题的进化算法,其求解效果不仅比 PSO 和 DE 更优,而且求解速度也更快。虽然 CSO 非常适于求解数值优化问题,但是如何将其用于求解组合优化问题(如背包问题、可满足性问题等)还是一个有待研究的方向,今后将在此方面做进一步的探讨。

参 考 文 献

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]// Proceedings of the IEEE International Conference on Neural Networks (Perth). IEEE Service Center, Piscataway, NJ, IV, 1995, 1942-1948

[2] 贺毅朝,王彦祺,刘建芹. 一种适于求解离散问题的二进制粒子群优化算法[J]. 计算机应用与软件,2007,24(1):157-159

[3] Storn R, Price K. Differential Evolution—A simple and efficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization, 1997, 11: 341-359

[4] 贺毅朝,王熙照,等. 差分演化的收敛性分析与算法改进[J]. 软件学报,2010,21(5):875-885

[5] Dorigo M, Caro G. The Ant Colony Optimization meta-heuristic [C]// Corne D, Dorigo M, Glover F. New Ideas in Optimization. London: McGraw Hill, 1999: 11-32

[6] Dorigo M, Gambardella L M. Ant colony system: a coorperative learning approach to the traveling salesman problem[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(2): 53-66

(下转第 249 页)

Combinatorial Pattern Matching, CPM 96. Berlin, Germany: Springer-Verlag, 1996: 50-63

[19] Ma B, Tromp J, Li M, et al. PatternHunter: faster and more sensitive homology search[J]. Bioinformatics, 2002, 18(3): 440-445

[20] Egidi L, Manzini G. Better spaced seeds using Quadratic Residues[J]. Journal of Computer and System Sciences, 2013, 79 (7): 1144-1155

[21] Baeza-Yates R, Navarro G. Faster approximate string matching [J]. Algorithmica, 1999, 23(2): 127-158

[22] Myers E W. A sublinear algorithm for approximate keyword searching[J]. Algorithmica, 1994, 12(4/5): 345-374

[23] Sutinen E, Szpankowski W. On the collapse of the q-gram filtration[C]// Proceedings of the International Conference on FUN with Algorithms. Waterloo, Canada: Carleton Scientific, 1998: 178-193

[24] Puglisil S J, Smyth W F, Turpin A. Inverted files versus suffix arrays for locating patterns in primary memory[C]// Proceedings of the 13th Symposium on String Processing and Information Retrieval. Berlin, Germany: Springer Verlag, 2006: 122-133

[25] Karp R M, Rabin M O. Efficient randomized pattern-matching algorithms[J]. IBM Journal of Research and Development, 1987, 31(2): 249-260

[26] Smith T F, Waterman M S. Identification of common molecular subsequences[J]. Journal of Molecular Biology, 1981, 147(1): 195-197

[27] NCBI. UniGene Build #223, Homo sapiens[DB/OL]. ftp. ncbi. nih. gov//repository/ UniGene/Homo_sapiens/Hs. seq. uniq. gz, 2010-04-28

(上接第 238 页)

[7] Eusuff M M, Lansey K E. Optimization of water distribution network design using the shuffled frog-leaping algorithm[J]. Journal of Water Resources Planning and Management, 2003, 129(3): 210-225

[8] 贺毅朝, 曲文龙, 许冀伟. 一种改进的混合蛙跳算法及其收敛性分析[J]. 计算机工程与应用, 2011, 47(22): 37-40

[9] 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002(11): 32-38

[10] 李晓磊, 冯少辉, 钱积新, 等. 基于人工鱼群算法的鲁棒 PID 控制器参数整定方法研究[J]. 信息与控制, 2004, 33(1): 112-115

[11] Yang X S. Biology-derived algorithms in engineering optimization[M]// Olarius S, Zomaya A Y, eds. Handbook of Bioinspired Algorithms and Applications. Chapman & Hall /CRC, 2005

[12] Yang X S. Nature-Inspired Metaheuristic Algorithms[M]. UK: Luniver Press, 2008

[13] Engelbrecht A P. Computational intelligence: an introduction [M]. Wiley Publishing. Inc. , 2009

[14] Wolpert D H, Macready W G. No Free Lunch Theorems for Optimization [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67-82

[15] Pringle J W S. A physiological analysis of cicada song[J]. J. Exp. Biol. , 1954, 32: 525-560

[16] Simmons P J. Periodical cicada: sound production and hearing [J]. Science, 1971, 171: 212-213

[17] 刘勇, 康立山, 陈毓屏. 非数值并行算法——遗传算法[M]. 北京: 科学出版社, 2003: 22-86

[18] Iosifescu M. Finite Markov processes and their applications [M]. Chichester: Wiley, 1980

[19] 江瑞, 罗予频, 胡东成, 等. 一种协调勘探和开采的遗传算法: 收敛性及性能分析[J]. 计算机学报, 2001, 24(12): 1233-1241

[20] Yao Xin, Liu Yong, Lin Guang-ming. Evolutionary programming made faster[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82-102

[21] He S, Wu Q H, Saunders J R. Group search optimizer: An optimization algorithm inspired by animal searching behavior[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 973-990

(上接第 242 页)

[6] Rosenstein M T, Barto A G. Supervised Learning Combined with an Actor-Critic Architecture[J]. CMPSCI Technical Report 02-41. October 2002

[7] Peters J, Schaal S. Natural actor-critic [J]. Neurocomputing, 2008, 71(7-9): 1180-1190

[8] Bathnagar S, Sutton R S, Ghavamzadeh M, et al. Natural actor-critic algorithms[J]. Automatica, 2009, 45(11): 2471-2482

[9] Vamvoudakis K G, Lewis F L. Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem[J]. Automatica, 2010, 46(5): 878-888

[10] Grondman I, Vaandrager M, Busoniu L, et al. Efficient Model Learning Methods for Actor-Critic Control[J]. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics, 2012, 42(3): 591-602

[11] Grondman I, Vaandrager M, Busoniu L, et al. Actor-Critic Control with Reference Model Learning[C]// Proceedings of the 18th IFAC World Congress. Milan, Italy, 2011: 14723-14728

[12] Kuvayev L, Sutton R. Model-Based Reinforcement Learning with an Approximate, Learned Model[C]// Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems. 1996: 101-105

[13] Goschin W S, Littman M. Integrating sample-based planning and model-based reinforcement learning[C]// Proc. Assoc. Adv. Artif. Intell. . Atlanta, GA, 2010: 612-617

[14] Santamaria J, Sutton R, Ram A. Experiments with reinforcement learning in problems with continuous state and action spaces [J]. Adaptive Bechavior, 1998, 6: 163-138

[15] Sherstov A A, Stone P. Function Approximation via Tile Coding: Automating parameter choice[C]// Zucker J-D, Saitta L, eds. SARA, volume 3607 of Lecture Notes in Computer Science. Springer, 2005: 194-205

[16] Lanzi P L, Loiacono D, Wilson S W, et al. Classifier Prediction based on Tile Coding[C]// Proceedings of the 2006 Geneticand Evolutionary Computation Conference Workshop Program (GECCO 2006). Seattle, Washington, 2006: 1497-1504