控制与决策
Control and Decision

带有策略自适应的状态转移算法

董颖超，张宏立，王聪

引用本文:
董颖超, 张宏立, 王聪. 带有策略自适应的状态转移算法[J]. 控制与决策, 2022, 37(3): 574−582.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2020.1558

带有策略自适应的状态转移算法

董颖超, 张宏立†, 王 聪

(新疆大学 电气工程学院，乌鲁木齐 830047)

摘 要: 针对基本状态转移算法(state transition algorithm, STA)搜索效率低和后期收敛速度慢的不足,对不同算子求解特定优化问题的效果差异性展开统计研究,提出一种带有策略自适应的状态转移算法(SaSTA). 首先,定义成功率和下降率两个指标,并在3个测试函数上进行统计研究,以证明不同算子对算法搜索能力的影响,设计一种综合成功率和下降率的评价指标对最优算子进行自适应选择;然后,采用一种非线性控制参数策略平衡算法的探索和开发能力;最后,将所提出算法应用于15个基准测试函数(100 维、300维和500维). 仿真结果表明,所提出算法在求解精度、收敛速度和稳定性方面均明显优于其他对比算法.

关键词: 状态转移算法；元启发式；策略自适应；统计研究；全局优化

中图分类号: TP301 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2020.1558

引用格式: 董颖超,张宏立,王聪. 带有策略自适应的状态转移算法[J]. 控制与决策, 2022, 37(3): 574-582.

State transition algorithm with strategy adaptation

DONG Ying-chao, ZHANG Hong-li†, WANG Cong

(College of Electrical Engineering，Xinjiang University，Urumqi 830047，China)

Abstract: In view of the shortcomings of basic state transition algorithm (STA) such as slow search efficiency and low convergence accuracy in the later search stage, based on the statistical study of the difference of the effects of different operators in solving specific optimization problems, a state transition algorithm with strategy adaptation (SaSTA) is proposed. Firstly, two indexes of success rate and descent rate are defined, and statistical studies are conducted on three test functions to prove the influence of different operators on the search capability of the algorithm, and an evaluation index of comprehensive success rate and descent rate is designed to adaptively select the optimal operator. Then, a nonlinear control parameter strategy is adopted to balance the exploration and exploitation ability of the algorithm. Finally, the proposed algorithm is applied to 15 benchmark functions (100, 300 and 500 dimension). The simulation results show that the proposed algorithm is superior to other comparative algorithms in terms of solution precise, convergence speed and stability.

Keywords: state transition algorithm； metaheuristic； strategy adaptation； statistical study； global optimization

0 引 言

许多实际工程优化问题通常具有离散性、约束性、非线性和非凸性的特点,传统的优化方法如共轭梯度法、最速下降法和拟牛顿法等很难解决此类问题. 而智能优化算法对待优化问题的形式没有严格的要求,使得受群体智能行为启发的元启发式算法得到了更广泛的应用. 在过去的几十年里,元启发式优化技术已经成功地解决了许多工程优化问题[1-3].

状态转移算法(state transition algorithm, STA)是Zhou等[4]于2012年提出的一种元启发式全局优化方法. 不同于粒子群算法[5]、蚁群算法[6]、人工蜂群算法[7]等基于群体的进化算法,状态转移算法是一种基于个体的随机搜索技术,它将特定优化问题的解视为一个状态,将解的更新过程比作状态转移的过程,并利用现代控制理论中的状态空间表达式作为产生候选解的统一框架. 在求解连续优化问题时,基本状态转移算法以交替轮换的方式使用旋转算子、平移算子、伸缩算子和坐标算子,通过抽样技术生成候选解集,采用贪婪准则更新当前最优解,直到满足迭代的终止条件. 由于状态转移算法的状态变换算子具有可控性,使得每种算子都可以产生大小可控的规则邻域,满足了局部搜索、全局搜索和启发式搜索的功

收稿日期: 2020-11-11；**录用日期:** 2021-01-19.
基金项目: 国家自然科学基金项目(51767022, 51967019).
责任编委: 王凌.
†通讯作者. E-mail: zhlxju@163.com.

能需要,使其能够概率性地获得优化问题的全局最优解. 由于状态转移算法具有原理简单、易于实现、寻优能力强等特点,已成功应用于风电功率预测[8-10]、非线性系统辨识[11]、图像分割[12]、动态优化[13]等领域. 然而,基本状态转移算法不能有效地求解大规模优化问题;另一方面,固定的变换因子参数降低了算法的寻优速度,在搜索过程中难以协调全局探索和局部开发能力,尤其在算法搜索的后期,过多的无效搜索浪费了搜索时间,且不能满足精度的要求. 针对变换因子的选择问题,Zhou等[14]设计了一个参数集合,在迭代过程中选择具有最佳适应度函数值的对应参数值,并在该参数值下保持一段时间,从而提出一种具有最优参数自适应选择策略的状态转移算法. Huang等[15]根据适应度函数值之间的相对变化对旋转因子和伸缩因子进行自适应调整. 虽然这些改进在一定程度上可以加速状态转移算法搜索的过程,但均未考虑针对特定优化问题的不同阶段每种算子作用效果的差异性. 基本状态转移算法中采用交替轮换的机制使用各种变换算子,并不能最大限度地发挥每种算子的功能. 如在算法的某一搜索阶段,若某个算子作用效果不大,则对该算子的参数进行选择是毫无意义的.

针对上述基本状态转移算法存在的不足,本文提出一种带有策略自适应的状态转移算法(SaSTA). 通过在算法搜索过程中对算子成功率和下降率进行统计研究,选择作用效果最大的算子,并在该算子作用下保持一段时间. 此外,采用一种非线性控制参数策略平衡算法的探索和开发能力,以增强算法的搜索效率. 最后,通过在15个大规模基准测试函数上的仿真实验验证了所提出SaSTA算法的有效性和优越性.

1　基本状态转移算法

状态转移算法是一种基于状态空间和状态转移的随机优化算法. 基于现代控制理论中状态空间的概念,状态转移算法生成候选解的统一形式定义为

$$\begin{cases} s_{k+1} = A_k s_k + B_k u_k, \\ y_{k+1} = f(s_{k+1}). \end{cases} \tag{1}$$

其中: $s_k, s_{k+1} \in \boldsymbol{R}^n$ 为当前状态和候选状态; $A_k, B_k \in \boldsymbol{R}^{n\times n}$ 为随机矩阵,对应于状态转移算法中的状态变换算子; y_k 为 s_k 的适应度函数值; u_k 为关于当前状态和历史状态的表达式; $f(\cdot)$ 为适应度函数.

1.1　状态变换算子

在基本状态转移算法中,通过参考不同类型的状态空间变换,共设计4种状态变换算子,它们对应于算法求解连续优化问题时的4种寻优策略.

1) 旋转变换(rotation transformation). 有

$$s_{k+1} = s_k + \alpha \frac{1}{n\|s_k\|_2} R_r s_k. \tag{2}$$

其中: $\alpha > 0$ 为旋转因子; $R_r \in \boldsymbol{R}^{n\times n}$ 为一个元素在区间 $[-1, 1]$ 上均匀分布的随机矩阵; $\|\cdot\|_2$ 为向量的欧几里德范数或2-范数. 旋转变换保证了生成的候选解在最大半径为 α 的超球体内具有局部搜索功能,可用于控制解的精度.

2) 平移变换(translation transformation). 有

$$s_{k+1} = s_k + \beta R_t \frac{s_k - s_{k-1}}{\|s_k - s_{k-1}\|_2}. \tag{3}$$

其中: $\beta > 0$ 为平移因子, $R_t \in \boldsymbol{R}$ 为分布在区间[0,1]上的随机变量. 显然,平移变换具有线搜索的功能,它以 s_k 为起点,沿着 s_{k-1} 指向 s_k 的直线搜索, β 为最大搜索长度.

3) 伸缩变换(expansion transformation). 有

$$s_{k+1} = s_k + \gamma R_e s_k. \tag{4}$$

其中: $\gamma > 0$ 为伸缩因子, $R_e \in \boldsymbol{R}^{n\times n}$ 为元素服从高斯分布(标准正态分布)的随机对角矩阵. 伸缩变换可以概率性地在整个空间中进行搜索,具有全局搜索的功能.

4) 坐标变换(axesion transformation). 有

$$s_{k+1} = s_k + \delta R_a s_k. \tag{5}$$

其中: $\delta > 0$ 为坐标因子; $R_a \in \boldsymbol{R}^{n\times n}$ 为元素服从高斯分布的稀疏随机对角矩阵,且只有一个随机位置是非零值. 坐标变换用于沿轴向搜索,旨在增强算法的单维搜索能力.

1.2　规则邻域与采样技术

不难理解,对于某个给定的当前状态,利用某种特定的状态变换算子,理论上可以生成无限多个候选解,其可以形成一个规则邻域. 显然,将所有的候选解都列举出来是不切实际的,考虑到状态变换矩阵的随机性,生成的候选解也不是唯一的,且任意两个解都存在相互独立的关系. 综上所述,为了节省搜索时间,可以将生成的候选解视为样本,并使用具有代表性的样本反映整体邻域的特征. 以伸缩变换为例,独立随机均匀采样SE次的伪代码如下:

1: for $i \leftarrow 1$ to SE do

2: state$(i,:) \leftarrow \mathrm{best}_k + \gamma R_e \mathrm{best}_k$

3: end for

在上述伪代码中: SE为样本数,称为搜索力度; best_k 为当前最好解; state为伸缩变换下生成的状态集合.

1.3 更新策略

如上所述,基于当前最好解best_k,由某个状态变换算子可以生成一个状态集合state. 因此,可以计算出状态集合中每个候选解的适应度函数值,从中挑选出一个新的最好解,记为best_{k+1}. 最后,采用贪婪准则更新当前最好解best_k,如下所示:

$$\text{best}_k = \begin{cases} \text{best}_{k+1},\ f(\text{best}_{k+1}) < f(\text{best}_k); \\ \text{best}_k,\ \text{otherwise}. \end{cases} \tag{6}$$

1.4 基本状态转移算法的实现

基于对状态变换算子、邻域与采样以及更新策略的描述,为了快速获得优化问题的全局最优解,基本状态转移算法以交替轮换的方式使用每个算子进行采样,其求解连续优化问题的伪代码如下:

1: $\text{best}_0 \leftarrow \text{randominitialize}(\cdot)$

2. repeat

3: if $\alpha < \alpha_{\min}$ then

4: $\alpha \leftarrow \alpha_{\max}$

5: end if

6: $\text{best}_k \leftarrow \text{expansion}\,(\text{fun}, \text{best}_k, \text{SE}, \beta, \gamma)$

7: $\text{best}_k \leftarrow \text{rotation}\,(\text{fun}, \text{best}_k, \text{SE}, \alpha, \beta)$

8: $\text{best}_k \leftarrow \text{axesion}\,(\text{fun}, \text{best}_k, \text{SE}, \beta, \delta)$

9: $\alpha \leftarrow \alpha/\text{fc}$

10: until迭代的终止条件

11: $\text{best}^* \leftarrow \text{best}_k$

在上述伪代码中,$\text{expansion}(\cdot)$、$\text{rotation}(\cdot)$以及$\text{axesion}(\cdot)$分别对应于伸缩变换,旋转变换和坐标变换并包含了更新策略;best_0为初始最好解;fc为衰减系数;旋转因子α以fc为基底,呈指数形式周期性地从最大值$a_{\max}$降到最小值$a_{\min}$. 需要指出的是,在基本状态转移算法中,只有旋转变换、伸缩变换或坐标变换找到比当前最优解更好的解时,才执行平移变换操作. 基本状态转移算法的参数设置如下:$\text{fc} = 2$, $\text{SE} = 30, a_{\max} = 1, a_{\min} = 1\text{e-}4, \beta = \gamma = \delta = 1$.

2 状态变换算子的统计研究

基本状态转移算法以交替轮换的方式使用不同算子生成候选解,但针对优化问题的不同阶段,每种算子的作用效果是不同的. 因此,为了充分利用每种算子的功能,选取表1中3个基准测试函数(Sphere, Rosenbrock, Griewank)进行统计研究,分析不同算子对算法寻优性能的影响. 以5维测试问题为例,给定2组初始解$\text{best}_0 = (10,10,10,10,10), (0.01, 0.01, 0.01, 0.01, 0.01)$,设总样本数$\text{SE} = 1\text{e}6$,状态变换因子参数$\alpha = \gamma = \delta = 1$、0.8、0.5、0.3、0.1,对于每组给定的当前解best_0,分别采用3种状态变换算子(旋转算子、伸缩算子和坐标算子)在每个基准函数上独立执行SE次.

为了评价旋转变换、伸缩变换和坐标变换3种搜索策略对算法寻优性能的影响,引入成功率ρ_s和下降率ρ_d两个指标对实验结果进行统计评价:

$$\rho_s = \frac{N_s}{\text{SE}}, \tag{7}$$

$$\rho_d = \frac{|f_{\text{ave}} - f_{\text{best}_0}|}{f_{\text{best}_0}} \tag{8}$$

其中:f_{best_0}为best_0的适应度函数值,N_s为样本中适应度函数值小于f_{best_0}的个数,f_{ave}为N_s个样本的平均适应度值. 3种算子在Sphere、 Rosenbrock和Griewank测试函数上的统计结果如表2~表4所示. 由表2~表4可以得出如下结论:

1) 在初始解best_0和变换因子α、γ、δ取值固定时,旋转变换、伸缩变换和坐标变换得到的成功率和下降率值差别较大. 这表明在算法搜索的某一特定阶段,3种状态变换的作用效果不同.

2) 当初始解best_0取值不同时,旋转变换得到的成功率和下降率值差异明显. 具体而言,对于Sphere函数,当$\text{best}_0 = (10,10,10,10,10)$时,旋转变换具有较大的成功率和较小的下降率;相反,当$\text{best}_0 = (0.01,0.01,0.01,0.01,0.01)$时,旋转变换具有较大的下降率和较小的成功率. 这表明在求解某一特定问题的不同阶段,3种搜索策略的作用效果也存在差异.

3) 就总体趋势而言,在初始解best_0相同的条件下,伸缩变换、旋转变换和坐标变换的成功率随着变换因子的减小而增大,而下降率则随着变换因子的减小而减小. 因此,在算法搜索前期变换因子应取较大值,以快速获得一个较好解;在算法搜索的后期,为了保证能够获得一个比当前解更优的解,变换因子应取较小值.

3 带有策略自适应的状态转移算法

统计研究结果表明,固定的变换因子以及交替轮换地使用各种算子都不是最优机制. 因此,本文引入一种非线性控制参数策略对变换因子进行调整,并提出一种寻优策略自适应机制,以加速基本状态转移算法的搜索过程.

3.1 非线性控制参数策略

为了使状态转移算法搜索前期获得一个相对较好解,后期获得一个精确解,变换因子前期应取较大值以获得较大的下降率,后期应取较小值以获得较大的成功率. 因此,引入一种非线性控制参数策略[16],并对其进行改进以调整变换因子的取值,其表达式如下:

表 1　基准测试函数

函数名	函数表达式	搜索区间	最优点	最优值	收敛精度
Sphere	$f_1=\sum_{i=1}^{n}x_i^2$	$[-100,100]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Schwefel 2.22	$f_2=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Cigar	$f_3=x_1^2+10^6\sum_{i=2}^{n}x_i^6$	$[-100,100]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Rosenbrock	$f_4=\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$[-30,30]$	$[1,1,\ldots,1]$	0	1×10^{0}
Elliptic	$f_5=\sum_{i=2}^{n}(10^6)^{(i-1)/(n-1)}\cdot x_i^2$	$[-100,100]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Sumsquare	$f_6=\sum_{i=1}^{n}ix_i^2$	$[-10,10]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Sumpower	$f_7=\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	$[-1,1]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Csendes	$f_8=\sum_{i=1}^{n}x_i^6\left(2+\sin\frac{1}{x_i}\right)$	$[-1,1]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Rastrigin	$f_9=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5,12]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Griewank	$f_{10}=\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Ackley	$f_{11}=20+\mathrm{e}-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	$[-32,32]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Weierstrass	$f_{12}=\sum_{i=1}^{n}\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i+0.5))]-n\sum_{k=0}^{k_{\max}}a^k\cos(\pi b^k x_i)$ $k_{\max}=20,a=0.5,b=3,-0.5\leqslant x_i\leqslant0.5,i=1,2,\ldots,n$	$[-0.5,0.5]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Alpine	$f_{13}=\sum_{i=1}^{n}\lvert x_i\sin(x_i)+0.1x_i\rvert$	$[-10,10]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Pathological	$f_{14}=\sum_{i=1}^{n-1}\left(0.5+\frac{(\sin\sqrt{100x_i^2+x_{i+1}^2})^2-0.5}{1+0.001(x_i^2-2x_ix_{i+1}+x_{i+1}^2)^2}\right)$	$[-10,10]$	$[0,0,\ldots,0]$	0	1×10^{-8}
Penalized 1	$f_{15}=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m,\ x_i>a;\\0,\ -a<x_i<a;\\k(-x_i-a)^m,\ x_i<a\end{cases}$ $y_i=1+\frac{x_i+1}{4}$	$[-50,50]$	$[-1,-1,\ldots,-1]$	0	1×10^{-2}

表 2　3种状态变换在Sphere函数上的成功率和下降率统计

状态变换	指标	$\mathrm{best}_0=(10,10,10,10,10)$					$\mathrm{best}_0=(0.01,0.01,0.01,0.01,0.01)$				
		1	0.8	0.5	0.3	0.1	1	0.8	0.5	0.3	0.1
伸缩变换	成功率	0.179 0	0.232 7	0.326 2	0.393 7	0.464 5	0.178 4	0.232 9	0.325 3	0.394 1	0.463 6
	下降率	0.3168	0.3025	0.244 4	0.172 2	0.066 6	0.316 4	0.302 4	0.244 5	0.172 0	0.066 4
旋转变换	成功率	0.495 8	0.496 7	0.498 3	0.499 1	0.498 9	1.70e-05	3.50e-05	3.30e-04	3.81e-03	0.140 4
	下降率	8.20e-03	6.60e-03	4.10e-03	2.47e-03	8.25e-04	0.320 1	0.300 0	0.289 5	0.293 2	0.321 9
坐标变换	成功率	0.477 0	0.493 1	0.499 7	0.499 6	0.499 9	0.477 2	0.493 7	0.498 9	0.498 9	0.500 7
	下降率	0.134 4	0.130 6	0.109 7	0.077 7	0.029 9	0.134 4	0.130 5	0.109 6	0.077 7	0.029 9

表 3 3种状态变换在Rosenbrock函数上的成功率和下降率统计

状态变换	指标	$\text{best}_0=(10,10,10,10,10)$					$\text{best}_0=(0.01,0.01,0.01,0.01,0.01)$				
		1	0.8	0.5	0.3	0.1	1	0.8	0.5	0.3	0.1
伸缩变换	成功率	0.1855	0.2233	0.2708	0.2675	0.2203	0.2249	0.2830	0.3724	0.3572	0.4756
	下降率	0.1245	0.1231	0.1230	0.3119	0.0821	0.3359	0.3276	0.2769	1.38e-03	0.0787
旋转变换	成功率	0.4461	0.4569	0.4731	0.4964	0.4948	1.60e-05	5.80e-05	5.43e-04	0.0178	0.1794
	下降率	8.62e-03	6.81e-03	4.20e-03	5.89e-03	8.22e-04	0.2049	0.2794	0.3124	0.0101	0.3369
坐标变换	成功率	0.2496	0.2616	0.2774	0.4999	0.4495	0.4772	0.4942	0.4992	0.2316	0.5001
	下降率	0.0394	0.0396	0.0402	0.1149	0.0208	0.1345	0.1305	0.1095	9.47e-04	0.0299

表 4 3种状态变换在Griewank函数上的成功率和下降率统计

状态变换	指标	$\text{best}_0=(10,10,10,10,10)$					$\text{best}_0=(0.01,0.01,0.01,0.01,0.01)$				
		1	0.8	0.5	0.3	0.1	1	0.8	0.5	0.3	0.1
伸缩变换	成功率	0.1869	0.2226	0.2707	0.1864	0.2201	0.2240	0.2837	0.3719	0.4257	0.4767
	下降率	0.1249	0.1242	0.1223	0.1251	0.0823	0.3362	0.3267	0.2767	0.1998	0.0785
旋转变换	成功率	0.4454	0.4565	0.4721	0.4457	0.4936	1.90e-05	5.30e-05	5.34e-04	5.56e-03	0.1794
	下降率	8.63e-03	6.82e-03	4.20e-03	8.63e-03	8.22e-04	0.3068	0.2828	0.2858	0.2952	0.3368
坐标变换	成功率	0.2496	0.2607	0.2773	0.2509	0.4494	0.4778	0.4944	0.5006	0.5002	0.5002
	下降率	0.0395	0.0395	0.0400	0.0393	0.0209	0.1343	0.1304	0.1095	0.0777	0.0298

$$\alpha(\beta,\gamma,\delta)=(C_a-C_b)\times\left(1-\sin\left(\frac{\pi}{2}\cdot\left(\frac{t}{T_{\max}}\right)^{\mu}\right)\right). \tag{9}$$

其中: C_a 和 C_b 分别为变换因子的初始值和最终值, μ 为参数控制因子, t 为当前迭代次数, $T_{\max}$ 为最大迭代次数.

为了直观地看出变换因子的控制调整效果, 图1展示了 $C_a=1, C_b=0, T_{\max}=1\,000$ 以及 μ 分别取1、2和4时变换因子的变化过程. 通过研究发现, 在求解函数优化问题中, 当 $\mu=2$ 时, 算法具有最好的寻优性能, 因此设置 $\mu=2$. 由图1可见, 当 $\mu=2$ 时, 变换因子从1非线性减小到0, 算法迭代前期变换因子取值较大, 以增强算法的全局探索能力, 使其快速得到一个较好解. 在算法迭代后期变换因子取值较小, 以实现对某个区域进行局部开发来获得更加精确的解. 因此, 引入的非线性控制参数策略可以有效地协调算法的全局和局部搜索能力.

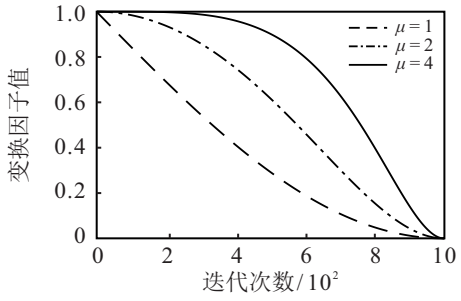

图 1 变换因子随迭代次数的变化曲线

3.2 自适应搜索策略

在经典的数值优化迭代方法中, 通常采用以下迭代公式搜索最优解:

$$x_{k+1}=x_k+a_kd_k. \tag{10}$$

其中: a_k 为搜索步长, d_k 为搜索方向. 状态转移算法本质上也是一种迭代算法, 只是搜索的方向和步长是不确定的. 通过类比发现, 状态转移算法的搜索步长和方向可以描述如下:

$$\left.\begin{array}{l} R_r\dfrac{1}{n\|s_k\|_2}s_k \\ R_t\dfrac{s_k-s_{k-1}}{\|s_k-s_{k-1}\|_2} \\ R_es_k \\ R_as_k \end{array}\right\}\widetilde{d}_k,\quad \left.\begin{array}{l}\alpha\\ \beta\\ \gamma\\ \delta\end{array}\right\}\widetilde{a}_k. \tag{11}$$

由于改进的状态转移算法中变换因子均按式(9)变化, 在算法搜索的任意时刻, 搜索步长是确定的, 但搜索方向是随机的. 受到统计研究结果的启发, 提出一种综合成功率和下降率的评价指标对旋转变换、伸缩变换和坐标变换进行自适应选择. 设 ρ_{s1}、ρ_{s2}、ρ_{s3} 分别为算法某一搜索阶段中伸缩变换、旋转变换和坐标变换的成功率指标, ρ_{d1}、ρ_{d2}、ρ_{d3} 分别对应其下降率指标. 以伸缩变换为例, 其综合评价指标定义为

$$G_s=\frac{\rho_{s1}}{\rho_{s1}+\rho_{s2}+\rho_{s3}}\times 100\,\%, \tag{12}$$

$$G_d=\frac{\rho_{d1}}{\rho_{d1}+\rho_{d2}+\rho_{d3}}\times 100\,\%, \tag{13}$$

$$\rho_c=a_1G_s+a_2G_d. \tag{14}$$

其中: G_s 和 G_d 分别为伸缩变换的成功率和下降率百分占比, $a_1 + a_2 = 1, a_1$、a_2 分别为成功率和下降率指标的重要性系数. 通常成功率和下降率指标具有同等重要性,所以一般设定 $a_1 = a_2 = 0.5$.

为了证明综合评价指标的有效性,分别采用带成功率指标的状态转移算法(STA1)、带下降率指标的状态转移算法(STA2)、不带任何指标的状态转移算法(STA3)以及带综合评价指标的状态转移算法(SaSTA)在Rastrigin函数上测试. 为了充分利用选出的最优算子,STA1、STA2和SaSTA对最优算子均保持执行 T 次. 实验中,测试函数的维度为200,变换因子根据式(9)变化. SE $= 50, C_a = 0.8, C_b = 0, T = 6$, $a_1 = a_2 = 0.5$,最大函数评价次数FEs为1.5 e-5. 图2为STA1、STA2、STA3和SaSTA在Rastrigin函数上运行30次的平均迭代曲线.

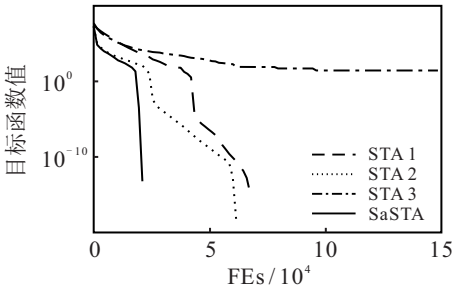

图 2　4种算法在Rastrigin函数(200维)上的收敛曲线

由图2可见,仅STA3在Rastrigin测试函数上未能得到全局最优解. 虽然STA1、STA2和SaSTA算法均能获得全局最优解,但SaSTA的收敛速度要快于STA1和STA2,这表明综合成功率与下降率的评价指标对最优算子选择的有效性.

3.3　算法步骤

基于对基本状态转移算法中变换因子和自适应寻优策略的描述,本节提出一种带有策略自适应的状态转移算法(SaSTA),所提出算法的具体步骤如下.

step 1: 设置算法相关参数.

step 2: 由式(9)对变换因子进行更新.

step 3: 对当前解进行伸缩变换、旋转变换和坐标变换,并计算每种变换的综合评价指标 ρ_c.

step 4: 执行最好的 ρ_c 指标对应的算子,若执行后的结果好于当前解,再进行平移变换操作.

step 5: 重复执行 T 次step 4.

step 6: 判断是否满足迭代的终止条件,若是,则终止搜索,输出最终优化结果,否则返回step 2.

4　仿真实验及分析

为了验证SaSTA算法的有效性,采用表1中的15个基准测试函数(100维、300维和500维)对所提出算法进行测试,并与几种流行的元启发式算法进行实验对比,包括人工蜂群算法(artificial bee colony algorithm, ABC)[7]、带有策略自适应的差分进化算法(differential evolution algorithm with strategy adaptation, SaDE)[17]、综合学习粒子群算法(comprehensive learning particle swarm optimizer, CLPSO)[18]、基本状态转移算法(STA)和带有最优算子参数选择的状态转移算法(POSTA). 其中ABC、SaDE和CLPSO算法的种群大小均设置为50, STA、POSTA和SaSTA算法的搜索力度SE设置为50,最大适应度函数评价次数设置为1.5e5. 为了保证实验的公平性,对比算法的其他参数设置与原始文献相同. SaSTA算法的其他参数设置如下: $a_1 = a_2 = 0.5$, $C_{\max} = 0.8, C_{\min} = 0, T = 6$. 所有程序在Matlab (R 2018 a版本)平台中编写,且在Windows10环境下, Intel(R) Core (TM) i5-9500F CPU@3.00 GHz的台式机上独立运行30次. 本文使用精度AC和到达率 r 两个性能指标对实验结果进行评价, AC和 r 定义如下:

$$\mathrm{AC} = |f_{\mathrm{best}} - \mathrm{opt}|, \tag{15}$$

$$r = \frac{z'}{z} \times 100\,/\,\%. \tag{16}$$

其中: f_{best} 为算法获得的最优解; opt为测试问题的实际最优解, z' 为算法收敛到实际最优解的次数, z 为总的实验次数.

4.1　STA、POSTA和SaSTA算法的实验对比

表5列出了STA、POSTA和SaSTA算法在15个基准测试函数上的实验对比结果. 由表5可见, SaSTA算法除了在函数 f_3(500维)和函数 f_4 上未能得到全局最优解以外,在其他13个测试函数上均能达到全局最优解. 与STA和POSTA算法相比, SaSTA获得了更优异的结果.

图3展示了3种算法在Weierstrass函数上的平均迭代曲线. 结果表明,相较于STA和POSTA算法, SaSTA具有更快的寻优速度和更高的精度. 具体而言,对于函数 f_2 和 f_7,虽然STA和SaSTA算法的到达率均为100 %,但STA的收敛精度明显差于SaSTA. 由函数 f_9、f_{10}、f_{11}、f_{14} 和 f_{15} 的结果可知,随着测试函数维度的增加,STA和POSTA算法的到达率均急剧下降. 相反SaSTA算法除了在函数 f_{14} 上表现略差外,在其他函数上的结果都比较稳定. 虽然SaSTA算法在函数 f_4 上未能达到全局最优解,但其结果也明显优于STA算法,且在函数 f_4 维度等于100时,结果优于POSTA算法. 通过以上实验结果分析可知,与STA和POSTA算法相比, SaSTA算法具有以下4个优势: 1)收敛速度快; 2)收敛精度高; 3)鲁棒性强; 4)求解高维优化问题优势明显.

表5 3种算法在15个基准测试函数上的寻优性能比较

函数	维度	STA算法			POSTA算法			SaSTA算法		
		平均值	标准差	到达率/%	平均值	标准差	到达率/%	平均值	标准差	到达率/%
f_1	100	9.05e-160	4.05e-159	**100**	5.88e-043	1.50e-042	**100**	**0**	**0**	**100**
	300	8.09e-004	3.64e-003	0	4.30e-021	4.09e-021	**100**	**0**	**0**	**100**
	500	1.97e-003	1.15e-003	0	8.94e-012	5.89e-012	**100**	**0**	**0**	**100**
f_2	100	1.86e-134	4.81e-134	**100**	1.87e-146	6.47e-146	**100**	**7.07e-298**	**0**	**100**
	300	4.28e-096	9.46e-096	**100**	1.40e-108	3.35e-108	**100**	**6.42e-241**	**0**	**100**
	500	1.62e-087	2.22e-087	**100**	1.68e-097	7.44e-097	**100**	**5.52e-227**	**0**	**100**
f_3	100	2.40e-021	2.70e-021	**100**	4.85e-067	1.38e-066	**100**	**4.59e-072**	**9.17e-072**	**100**
	300	1.53e+005	1.63e+005	0	**5.63e-013**	**1.78e-012**	**100**	7.32e-012	8.35e-012	**100**
	500	2.31e+013	1.18e+013	0	5.18e+004	9.90e+004	0	**2.16e+002**	**7.19e+001**	0
f_4	100	3.20e+002	5.01e+002	0	1.20e+002	4.28e+001	0	**1.13e+002**	**3.39e+001**	0
	300	7.53e+002	1.43e+002	0	**3.41e+002**	**5.50e+001**	0	4.59e+002	9.57e+001	0
	500	1.84e+003	3.03e+002	0	**5.48e+002**	**7.80e+001**	0	8.02e+002	2.19e+002	0
f_5	100	5.62e-222	0	**100**	5.38e-042	1.34e-041	**100**	**0**	**0**	**100**
	300	7.50e-004	5.81e-004	0	3.04e-018	1.14e-017	**100**	**0**	**0**	**100**
	500	7.65e-001	1.68e-001	0	1.37e-011	3.21e-011	**100**	**0**	**0**	**100**
f_6	100	1.71e-184	0	**100**	8.44e-043	2.27e-042	**100**	**0**	**0**	**100**
	300	2.17e-005	4.70e-006	0	1.07e-020	7.64e-021	**100**	**0**	**0**	**100**
	500	9.11e-002	2.20e-001	0	3.56e-014	1.72e-014	**100**	**0**	**0**	**100**
f_7	100	2.10e-129	4.17e-129	**100**	**0**	**0**	**100**	2.61e-250	0	**100**
	300	7.53e-074	8.95e-074	**100**	**0**	**0**	**100**	2.28e-184	0	**100**
	500	1.26e-045	2.51e-045	**100**	8.77e-079	3.92e-078	**100**	**5.58e-126**	**1.25e-125**	**100**
f_8	100	1.87e-035	2.95e-035	**100**	1.33e-040	4.57e-040	**100**	**3.37e-067**	**8.09e-067**	**100**
	300	4.30e-014	2.93e-014	**100**	1.28e-017	1.05e-017	**100**	**6.04e-023**	**3.06e-023**	**100**
	500	5.63e-009	4.18e-009	90	6.63e-011	6.99e-011	**100**	**3.04e-012**	**1.70e-012**	**100**
f_9	100	**0**	**0**	**100**	1.03e-012	4.01e-013	**100**	**0**	**0**	**100**
	300	2.07e-004	2.79e-004	0	7.10e-008	2.43e-007	80	**0**	**0**	**100**
	500	9.00e-001	1.68e+000	0	7.37e+000	3.07e+000	0	**0**	**0**	**100**
f_{10}	100	**0**	**0**	**100**	2.78e-017	1.01e-016	**100**	**0**	**0**	**100**
	300	8.08e-009	2.43e-009	80	4.93e-004	4.13e-003	95	**0**	**0**	**100**
	500	1.66e+000	5.11e+000	0	4.07e-004	1.75e-003	95	**0**	**0**	**100**
f_{11}	100	4.97e-015	1.30e-015	**100**	1.10e-013	1.32e-014	**100**	**8.88e-016**	**0**	**100**
	300	5.48e-005	3.02e-005	0	1.86e-007	8.33e-007	95	**8.88e-016**	**0**	**100**
	500	1.34e-003	1.47e-004	0	2.32e-007	5.27e-007	60	**8.88e-016**	**0**	**100**
f_{12}	100	**0**	**0**	**100**	3.13e-014	8.14e-014	**100**	**0**	**0**	**100**
	300	**0**	**0**	**100**	7.16e-013	1.13e-012	**100**	**0**	**0**	**100**
	500	**0**	**0**	**100**	1.74e-003	7.26e-003	85	**0**	**0**	**100**
f_{13}	100	7.11e-004	6.19e-004	0	1.86e-008	1.21e-008	30	**6.54e-296**	**0**	**100**
	300	2.61e-002	2.59e-002	0	5.14e-003	1.26e-003	0	**2.36e-005**	**5.03e-005**	**80**
	500	1.37e-001	8.30e-002	0	8.39e-002	8.40e-003	0	**3.82e-004**	**1.13e-003**	**80**
f_{14}	100	7.29e-001	1.18e+000	30	7.47e-001	1.07e+000	45	**9.70e-010**	**1.99e-009**	**100**
	300	1.28e+001	2.73e+000	0	1.12e+001	3.65e+000	0	**3.99e+000**	**3.01e+000**	**25**
	500	3.92e+001	1.21e+001	0	3.38e+001	1.03e+001	0	**2.01e+001**	**1.70e+001**	**30**
f_{15}	100	5.09e-009	1.96e-009	**100**	**1.83e-020**	**5.16e-020**	**100**	2.56e-011	6.84e-012	**100**
	300	6.17e-005	2.70e-005	**100**	5.18e-004	2.36e-003	95	**3.59e-010**	**5.29e-011**	**100**
	500	1.13e+000	1.53e-001	0	3.52e-002	3.86e-002	35	**1.32e-003**	**2.23e-003**	**100**

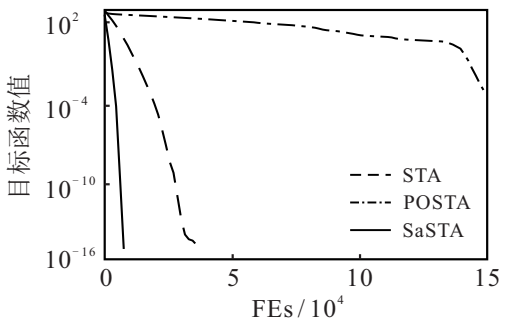

图3 3种算法在Weierstrass函数(500维)上的收敛曲线

4.2 CLPSO、SaDE、ABC和SaSTA算法的实验对比

为了进一步验证SaSTA算法的性能,将SaSTA算法的结果与CLPSO、SaDE和ABC算法进行实验对比. 表6列出了4种算法在15个测试函数上独立运行30次的平均值和标准差. 由表6可见,除了在函数f_4(100维)上SaSTA得到的平均值和标准差仅差于ABC算法外,在剩余14个测试函数上的结果均是最好的,这表明SaSTA算法具有更强的搜索能力. 图4给出了4种算法在Ackley函数上的寻优过程. 由图4

表 6　4种算法在15个基准测试函数上的寻优性能比较

函数	维度	CLPSO算法		SaDE算法		ABC算法		SaSTA算法	
		平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
f_1	100	5.52e+000	8.09e-001	3.31e-011	4.42e-011	1.77e-007	2.31e-007	**0**	**0**
	300	6.24e+003	5.59e+002	1.22e+002	1.33e+002	6.06e-002	5.55e-002	**0**	**0**
	500	4.02e+004	1.90e+003	4.29e+003	1.46e+003	9.37e+003	7.15e+003	**0**	**0**
f_2	100	1.25e+000	1.15e-001	6.46e-006	1.29e-005	9.02e-005	3.17e-005	**7.07e-298**	**0**
	300	1.15e+002	5.25e+000	1.34e+001	3.26e+000	3.06e+000	2.00e+000	**6.42e-241**	**0**
	500	2.38e+002	8.39e+000	8.49e+001	1.10e+001	1.88e+000	1.07e+000	**5.52e-227**	**0**
f_3	100	7.10e+008	2.58e+008	4.18e-001	6.00e-001	2.82e-007	4.51e-007	**4.59e-072**	**9.17e-072**
	300	6.59e+014	1.42e+014	2.62e+012	3.34e+012	2.09e+003	5.49e+003	**7.32e-012**	**8.35e-012**
	500	1.36e+016	2.07e+015	1.50e+014	7.38e+013	1.04e+008	3.20e+008	**2.16e+002**	**7.19e+001**
f_4	100	2.43e+003	3.86e+002	2.19e+002	5.99e+001	**1.20e+001**	**9.92e+000**	1.13e+002	3.39e+001
	300	1.40e+006	1.81e+005	1.50e+004	4.72e+003	6.81e+002	3.47e+002	**4.59e+002**	**9.57e+001**
	500	1.51e+007	1.60e+006	4.48e+005	1.16e+005	1.73e+004	3.19e+004	**8.02e+002**	**2.19e+002**
f_5	100	2.04e+004	4.12e+003	3.34e-008	5.77e-008	3.82e-002	4.32e-002	**0**	**0**
	300	4.76e+007	6.47e+006	1.71e+005	2.08e+005	7.81e+003	9.61e+003	**0**	**0**
	500	4.88e+008	5.99e+007	8.32e+006	5.31e+006	1.22e+007	3.68e+007	**0**	**0**
f_6	100	2.44e+000	3.48e-001	8.46e-012	8.44e-012	6.69e-008	7.17e-008	**0**	**0**
	300	8.07e+003	4.53e+002	1.70e+002	1.06e+002	7.70e-002	6.11e-002	**0**	**0**
	500	8.56e+004	4.55e+003	7.80e+003	2.07e+003	2.11e+004	1.38e+004	**0**	**0**
f_7	100	3.76e-025	6.74e-025	9.96e-034	2.43e-033	6.66e-008	5.93e-008	**2.61e-250**	**0**
	300	3.57e-019	4.26e-019	2.89e-023	1.19e-022	7.13e-003	6.02e-003	**2.28e-184**	**0**
	500	1.07e-016	2.04e-016	2.85e-022	6.26e-022	8.41e-002	4.94e-002	**5.58e-126**	**1.25e-125**
f_8	100	1.18e-009	4.24e-010	2.96e-006	8.78e-007	3.86e-015	3.13e-015	**3.37e-067**	**8.09e-067**
	300	1.32e-003	1.63e-004	1.98e-005	3.69e-006	4.29e-008	1.13e-007	**6.04e-023**	**3.06e-023**
	500	2.99e-002	3.34e-003	3.05e-004	1.19e-004	7.16e-003	7.36e-003	**3.04e-012**	**1.70e-012**
f_9	100	1.90e+002	1.13e+001	3.40e+001	8.25e+000	7.50e+000	1.30e+000	**0**	**0**
	300	1.75e+003	5.24e+001	4.15e+002	4.41e+001	3.19e+002	2.80e+001	**0**	**0**
	500	3.67e+003	7.29e+001	1.01e+003	1.00e+002	1.24e+003	4.42e+001	**0**	**0**
f_{10}	100	1.03e+000	2.49e-002	2.46e-002	5.30e-002	4.73e-006	1.46e-005	**0**	**0**
	300	5.86e+001	4.14e+000	2.35e+000	1.12e+000	1.38e-001	1.35e-001	**0**	**0**
	500	3.55e+002	2.24e+001	3.85e+001	8.43e+000	1.03e+002	6.08e+001	**0**	**0**
f_{11}	100	1.49e+000	1.57e-001	2.88e+000	4.31e-001	4.39e-004	2.75e-004	**8.88e-016**	**0**
	300	8.08e+000	2.03e-001	7.28e+000	5.09e-001	4.53e+000	3.44e-001	**8.88e-016**	**0**
	500	1.14e+001	1.98e-001	9.23e+000	5.24e-001	1.17e+001	3.57e-001	**8.88e-016**	**0**
f_{12}	100	3.74e+000	2.48e-001	9.19e+000	2.76e+000	8.13e-003	1.49e-003	**0**	**0**
	300	1.05e+002	3.73e+000	1.02e+002	4.64e+000	3.18e+001	3.01e+000	**0**	**0**
	500	2.88e+002	8.40e+000	2.29e+002	1.44e+001	1.78e+002	7.95e+000	**0**	**0**
f_{13}	100	8.33e+000	1.63e+000	4.54e-007	1.31e-006	4.02e-002	3.50e-002	**6.54e-296**	**0**
	300	1.45e+002	7.24e+000	1.91e+000	8.67e-001	9.58e+000	1.63e+000	**2.36e-005**	**5.03e-005**
	500	3.38e+002	7.99e+000	2.07e+001	4.85e+000	8.13e+001	6.64e+000	**3.82e-004**	**1.13e-003**
f_{14}	100	2.91e+001	6.60e-001	3.35e+001	3.73e-001	8.61e+000	6.62e-001	**9.70e-010**	**1.99e-009**
	300	1.17e+002	7.55e-001	1.25e+002	4.32e-001	6.89e+001	1.36e+000	**3.99e+000**	**3.01e+000**
	500	2.11e+002	1.23e+000	2.18e+002	6.68e-001	1.46e+002	1.57e+000	**2.01e+001**	**1.70e+001**
f_{15}	100	1.36e-001	4.06e-002	9.66e-002	1.11e-001	2.15e-009	2.82e-009	**2.56e-011**	**6.84e-012**
	300	6.62e+002	5.64e+002	5.06e+000	1.18e+000	2.34e-004	3.14e-004	**3.59e-010**	**5.29e-011**
	500	8.27e+005	2.90e+005	1.40e+001	6.55e+000	5.73e-003	4.54e-003	**1.32e-003**	**2.23e-003**

可见,相较于CLPSO、SaDE和ABC算法,SaSTA算法具有更快的收敛速度.

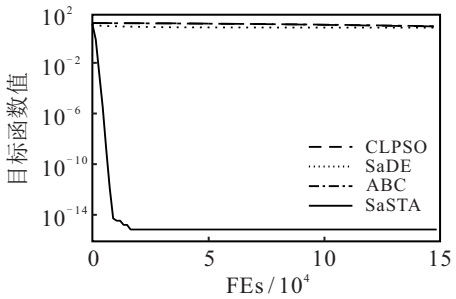

图 4　4种算法在Ackley函数(500维)上的收敛曲线

4.3　算法的时间复杂度分析

本文所提出SaSTA算法与基本状态转移算法相比,在每次迭代的过程中需要对最优算子进行自适应选择,从而会增加SaSTA算法的时间复杂度. 因此,为了实验对比的公平性,每种算法的最大评价函数运行次数均设置为1.5e5次. 表7为SaSTA和基本STA算法在15个测试函数(100维)上独立运行30次的平均运行时间. 由表7可见,在相同的评价函数次数下,SaSTA比STA寻优时间更短,表明SaSTA在提高寻优精度的同时也缩短了算法的收敛时间.

表7 时间复杂度对比

函数	平均运行时间/s	
	STA	SaSTA
Sphere	5.11	3.62
Schwefel 2.22	5.04	3.86
Cigar	7.83	7.51
Rosenbrock	5.15	4.55
Elliptic	8.36	6.92
Sumsquare	4.93	3.57
Sumpower	7.46	7.12
Csendes	8.23	7.21
Rastrigin	4.94	2.94
Griewank	5.10	2.93
Ackley	5.15	3.09
Weierstrass	12.30	9.65
Alpine	5.34	3.20
Pathological	6.11	3.75
Penalized 1	11.35	10.71

5 结 论

本文通过对状态转移算法的算子进行统计研究,提出了一种带有策略自适应的状态转移算法. 所提出算法采用一种综合评价指标对算子进行自适应选择,克服了基本状态转移算法交替轮换地使用各种算子带来的搜索效率低的缺陷. 此外,为了协调算法的探索和开发能力,引入了一种非线性控制参数策略对变换因子进行调整. 对15个基准测试函数进行仿真实验的结果表明,与基本状态转移算法以及其他元启发式算法相比,所提出算法在求解高维复杂函数问题上具有速度快、精度高和鲁棒性强的优势.

参考文献(References)

[1] Wang L, Zheng J, Wang J J. A hybrid discrete fruit fly optimization algorithm for distributed permutation flowshop scheduling with interval data[J]. Control and Decision, 2020, 35(4): 930-936.

[2] Wang L, Wang J J, Wu C G. Advances in green shop scheduling and optimization[J]. Control and Decision, 2018, 33(3): 385-391.

[3] Yang X, Gong W Y, Wang L. Comparative study on parameter extraction of photovoltaic models via differential evolution[J]. Energy Conversion and Management, 2019, 201: 112113.

[4] Zhou X J, Yang C H, Gui W H. State transition algorithm[J]. Journal of Industrial & Management Optimization, 2012, 8(4): 1039-1056.

[5] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proceedings ICNN95 Interationol Conference on Neural Networks. Piscataway: IEEE, 1995: 1942-1948.

[6] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, 1996, 26(1): 29-41.

[7] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm[J]. Journal of Global Optimization, 2007, 39(3): 459-471.

[8] Wang C, Zhang H L, Fan W H, et al. A new chaotic time series hybrid prediction method of wind power based on EEMD-SE and full-parameters continued fraction[J]. Energy, 2017, 138: 977-990.

[9] Wang C, Zhang H L, Fan W H, et al. A new wind power prediction method based on chaotic theory and bernstein neural network[J]. Energy, 2016, 117: 259-271.

[10] Wang C, Zhang H L, Ma P. Wind power forecasting based on singular spectrum analysis and a new hybrid Laguerre neural network[J]. Applied Energy, 2020, 259: 114139.

[11] Zhou X J, Yang C H, Gui W H. Nonlinear system identification and control using state transition algorithm[J]. Applied Mathematics and Computation, 2014, 226: 169-179.

[12] Han J, Yang C H, Zhou X J, et al. A new multi-threshold image segmentation approach using state transition algorithm[J]. Applied Mathematical Modelling, 2017, 44: 588-601.

[13] Han J, Yang C H, Zhou X J, et al. Dynamic multi-objective optimization arising in iron precipitation of zinc hydrometallurgy[J]. Hydrometallurgy, 2017, 173: 134-148.

[14] Zhou X J, Yang C H, Gui W H. A statistical study on parameter selection of operators in continuous state transition algorithm[J]. IEEE Transactions on Cybernetics, 2019, 49(10): 3722-3730.

[15] Huang M, Zhou X J, Huang T W, et al. Dynamic optimization based on state transition algorithm for copper removal process[J]. Neural Computing and Applications, 2019, 31(7): 2827-2839.

[16] 黄晨晨, 魏霞, 黄德启, 等. 求解高维复杂函数的混合蛙跳-灰狼优化算法[J]. 控制理论与应用, 2020, 37(7): 1655-1666.
(Huang C C, Wei X, Huang D Q, et al. Shuffled frog leaping grey wolf algorithm for solving high dimensional complex functions[J]. Control Theory & Applications, 2020, 37(7): 1655-1666.)

[17] Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398-417.

[18] 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.

作者简介

董颖超(1993−), 男, 博士生, 从事综合能源调度、群智能优化算法的研究, E-mail: 625408917@qq.com;

张宏立(1972−), 男, 教授, 博士, 从事复杂生产过程优化与调度的研究, E-mail: zhlxju@163.com;

王聪(1989−), 女, 副教授, 博士, 从事智能控制、群智能优化算法等研究, E-mail: 641087385@qq.com.

(责任编辑: 郑晓蕾)