文章编号: 1001-0920 (2015) 05-0839-09 DOI: 10.13195/j.kzyjc.2014.0750

Tent 混沌人工蜂群与粒子群混合算法

匡芳君[1,2], 金 忠[1], 徐蔚鸿[1,3], 张思扬[2]

(1. 南京理工大学 计算机科学与工程学院，南京 210094；2. 湖南安全技术职业学院 电气与信息工程系，长沙 410151；3. 长沙理工大学 计算机与通信工程学院，长沙 410114)

摘 要: 针对人工蜂群和粒子群算法的优势与缺陷, 提出一种 Tent 混沌人工蜂群粒子群混合算法. 首先利用 Tent 混沌反向学习策略初始化种群; 然后划分双子群, 利用 Tent 混沌人工蜂群算法和粒子群算法协同进化; 最后应用重组算子选择最优个体作为跟随蜂的邻域蜜源和粒子群的全局极值. 仿真结果表明, 该算法不仅能有效避免早熟收敛, 而且能有效跳出局部极值, 与其他最新人工蜂群和粒子群算法相比具有较强的全局搜索能力和局部搜索能力.

关键词: Tent 混沌搜索；人工蜂群算法；粒子群优化算法；混沌反向学习；重组算子

中图分类号: TP18 **文献标志码:** A

Hybridization algorithm of Tent chaos artificial bee colony and particle swarm optimization

KUANG Fang-jun[1,2], *JIN Zhong*[1], *XU Wei-hong*[1,3], *ZHANG Si-yang*[2]

(1. School of Computer Science and Engineering，Nanjing University of Science and Technology，Nanjing 210094, China；2. Department of Electronic and Information Engineering，Hu'nan Vocational Institute of Safety & Technology, Changsha 410151，China；3. College of Computer and Communications Engineering，Changsha University of Science and Technology，Changsha 410114，China. Correspondent：KUANG Fang-jun，E-mail：kfjztb@126.com)

Abstract: In view of the advantages and disadvantages of artificial bee colony(ABC) algorithm and particle swarm optimization(PSO) algorithm, a hybridization algorithm of Tent chaos artificial bee colony and particle swarm optimization (HTCAP) is proposed. In the HTCAP, an initialization strategy based on Tent chaotic opposition-based learning is applied. All individuals are divided into two sub-swarms by cooperative evolution with Tent chaos artificial bee colony(TCABC) algorithm and Tent chaos particle swarm optimization(TCPSO) algorithm. The best solution obtained by the recombination operator is as the neighbor food source for onlooker bees and the global best of particle swarm, respectively. Simulation results show that, the algorithm not only effectively avoids the premature convergence, but also gets rid of the local minimum. By comparison with the other latest algorithms based on the ABC algorithm and PSO algorithm, the proposed model has better global and local searching abilities.

Keywords: Tent chaos search；artificial bee colony；particle swarm optimization；chaotic opposition-based learning; recombination operator

0 引 言

Karaboga[1]于2005年提出的人工蜂群算法(ABC)是一种群智能优化算法. 它具有控制参数少、收敛速度快、全局寻优能力强、鲁棒性强等优点[2], 但当其接近全局最优解时, 收敛速度变慢, 种群多样性也减少, 甚至易陷入局部最优. ABC 算法可应用于求解复杂优化问题, 备受众多学者的关注, 衍生了很多改进算法和应用. 例如, Akay 等[3]提出了改进人工蜂群算法用于实参优化; 高卫峰等[4]在全局最优引导人工蜂群算法[5]的基础上, 引入反向学习初始化策略, 提出了一种新的改进人工蜂群算法; 许多学者利用混沌的随机性和遍历性优势, 提出了基于 Logistic 混沌的人工蜂群算法, 改善了算法后期易陷入局部最优的缺陷[6-7].

Kennedy 等[8]于 1995 年最初提出的粒子群算法(PSO)是一种基于随机搜索的群智能优化算法. PSO

收稿日期: 2014-05-14；**修回日期:** 2014-09-26.

基金项目: 国家自然科学基金项目(61373063, 61233011, 61402227)；湖南省科技计划项目(2013FJ4217).

作者简介: 匡芳君(1976−), 女, 副教授, 博士, 从事模式识别、智能优化、信息安全等研究；金忠(1961−), 男, 教授, 博士生导师, 从事模式识别、机器学习、计算机视觉等研究.

虽简单且易于实现, 但却易陷入局部最优, 求解精度较差, 随着问题维数或搜索空间的增加, 算法性能下降. 为了提高算法性能, 人们提出了各种改进算法. 例如, 陶新民等[9]提出了多尺度协同变异粒子群优化算法; 吴晓军等[10]提出了均匀搜索粒子群算法; 周新宇等[11]提出了精英反向学习粒子群优化算法.

研究表明: ABC具有很强的全局搜索能力, 但由于ABC在每一代不直接使用全局最优信息, 只是存储其最优信息, 局部搜索能力较弱[7]; 而PSO算法具有很强的局部搜索能力, 且每次迭代直接使用全局最优信息来决定新粒子的位置, 但种群易陷入局部极值, 而且不能很好地跳出局部极值, 故PSO算法的全局搜索能力较弱[12]. 为了提高算法的全局搜索能力, 迭代中需要保持种群的多样性, 但种群在迭代时又需要提高其局部搜索能力, 因此, 如何均衡算法的全局搜索能力和局部搜索能力是求解全局优化问题的关键. 有学者提出了人工蜂群和粒子群混合算法. 例如, Shi等[13]提出了一种混合方法IABAP, 该算法利用蜂群与粒子之间进行信息交互, 当蜂群算法中存在侦察蜂时, 一方面将粒子群算法中的最优粒子作为侦察蜂的蜜源位置, 另一方面粒子速度的更新在一定程序上又受蜂群的影响; 另一种混合方法是EI-ABD[14]提出的SPSO-ABC算法, 该算法是在PSO主循环后使用ABC更新规则, 即在PSO的所有粒子更新后, 对粒子的个体极值执行ABC更新规则.

本文针对ABC和PSO算法各自的优势和缺陷, 引入Tent混沌搜索、反向学习策略和重组算子, 提出一种新的Tent混沌人工蜂群与粒子群的混合算法(HTCAP). 首先利用Tent混沌反向学习策略初始化种群; 然后将种群等分成双子群, 分别利用Tent混沌人工蜂群算法(TCABC)和Tent混沌粒子群优化(TCPSO)算法协同进化. 与IABAP和SPSO-ABC不同的是, HTCAP是利用重组算子选择最优个体作为TCABC跟随蜂的邻域蜜源和TCPSO的全局极值, 即利用人工蜂群与粒子群之间的信息交互平衡算法的局部搜索能力和全局搜索能力, 进而提高解的质量.

1 Tent混沌搜索

1.1 Tent混沌映射

利用混沌变量的随机性、遍历性和规律性特征进行优化搜索, 可使算法跳出局部最优, 保持群体多样性, 并使全局搜索能力得到改善, 但不同的混沌映射对混沌优化过程的影响很大. 目前文献中引用较多的混沌映射是Logistic, 但它在$[0,0.1]$和$[0.9,1]$两个范围的取值概率较高, 且寻优速度受Logistic遍历不均匀性的影响, 算法效率会降低. 单梁等[15]指出Tent映射比Logistic映射具有更好的遍历均匀性和更快的迭代速度, 且在[0, 1]间产生的混沌序列分布均匀. Tent映射表达式如下:

$$x_{t+1}=\begin{cases}2x_t,\ 0\leqslant x_t\leqslant 1/2;\\ 2(1-x_t),\ 1/2\leqslant x_t\leqslant 1.\end{cases}\tag{1}$$

Tent映射经贝努利移位变换后表示为[15]

$$x_{t+1}=(2x_t)\ \bmod\ 1.\tag{2}$$

根据Tent映射的特性, 在可行域中产生Tent混沌序列的步骤描述如下.

Step 1: 随机产生$(0,1)$间的初值x_0(避免x_0落入小周期内$(0.2,0.4,0.6,0.8)$), 记为标志组z, $z(1)=x_0$, $i=j=1$.

Step 2: 按式(2)迭代产生一个x序列, $i=i+1$.

Step 3: 若达到最大迭代次数, 则转向Step 5.

Step 4: 若$x_i=\{0,0.25,0.5,0,75\}$, 或$x_i=x_{i-k}$, $k=\{0,1,2,3,4\}$, 则按式$x_i=z_{j+1}=z_j+\epsilon$改变迭代初值, ϵ是随机数, $j=j+1$, 否则转向Step 2.

Step 5: 终止运行, 保存产生的x序列.

1.2 Tent混沌搜索

本文采用的Tent混沌搜索是以当前搜索到的最优位置$X_k=(x_{k1},\cdots,x_{kD})$为基础, 利用Tent混沌搜索产生的最优解作为新蜜源位置或粒子群的全局极值, 使其跳出局部最优. 假设$[X^d_{\min},X^d_{\max}]$是搜索空间范围, $X^d_{\min}$、$X^d_{\max}$分别为第d维向量的最小值和最大值. 混沌搜索主要步骤如下.

Step 1: 利用$Z^d_k(0)=\dfrac{X^d_k-X^j_{\min}}{X^j_{\max}-X^j_{\min}}$将$X_k$映射到$(0,1)$. 其中: $k=1,2,\cdots,n,\ d=1,2,\cdots,D$.

Step 2: 将上式代入式(2)的Tent映射进行迭代, 产生混沌变量序列$Z^d_k(t)(t=1,2,\cdots,C_{\max})$, $C_{\max}$是混沌搜索的最大迭代次数.

Step 3: 利用式$V^d_k=X^d_k+\dfrac{X^d_{\max}-X^d_{\min}}{2}\times(2Z^d_k(t)-1)$将$Z^d_k$载波到原解空间邻域内以产生新解$V_k$.

Step 4: 计算V_k的适应度值$F(V_k)$, 并与原来解的适应度值$F(X_k)$比较, 保留最优解.

Step 5: 判断是否达到最大混沌搜索次数, 若达到, 则混沌搜索结束, 否则转向Step 2.

2 Tent混沌人工蜂群算法

利用Tent映射比Logistic映射具有更好的收敛速度和遍历均匀性, 本文提出了Tent混沌人工蜂群算法(TCABC). 该算法引入Tent混沌搜索和锦标赛策略, 并在引领蜂放弃蜜源成为侦察蜂时, 利用Tent混沌搜索产生新解, 以提高算法的收敛速度.

2.1 ABC算法

在ABC算法中, 蜂群分为引领蜂、跟随蜂和侦察蜂. 蜜源的丰富程度表示可行解的质量或适应度, 最

优适应度的蜜源表示问题最优解. 蜜源位置通过下式随机产生, 并被分派给引领蜂, 即

$$X_i^d(0) = X_{\min}^d + R \times (X_{\max}^d - X_{\min}^d). \tag{3}$$

其中: $X_i^d(0)$ 是初始化时第 i 个蜜源的第 d 维向量; $X_{\max}^d$、$X_{\min}^d$ 分别为第 d 维向量的上界和下界; R 为 [0, 1] 间的随机数. 蜜源、引领蜂和跟随蜂个数相等, 等于蜂群总数的一半.

所有蜜源的适应度通过下式计算:

$$F_i(t) = \begin{cases} \dfrac{1}{1+f_i(t)}, & f_i(t) \geqslant 0; \\ 1 + abs(f_i(t)), & \text{otherwise}. \end{cases} \tag{4}$$

其中: $F_i(t)$ 为 t 时刻第 i 个蜜源的适应度值, $f_i(t)$ 为具体优化问题的目标函数值.

为了能根据旧的蜜源位置 X_i^d 产生新的蜜源位置 V_i^d, ABC 算法采用如下表达式:

$$V_i^d(t) = X_i^d(t) + \Phi(X_i^d(t) - X_k^d(t)). \tag{5}$$

其中: Φ 是 $[-1, 1]$ 之间的随机数; $k = 1, 2, \cdots, N$ 是随机选择的下标, 且 $k \neq i$. 以上各式中 $i = 1, 2, \cdots, N$, $d = 1, 2, \cdots, D$, N 是蜜源或引领蜂的数量, D 是优化参数的个数或问题空间的维数.

另外, 如果一个蜜源位置经过限定次数 Limit 循环后仍不能得到改进, 则该蜜源处的引领蜂将成为侦察蜂, 并按式 (3) 在解空间中随机产生新蜜源位置以替换旧蜜源.

2.2　锦标赛选择策略

基本 ABC 算法采用比例选择策略选择蜜源, 使其在进化初期收敛速度较快, 但在算法后期易陷入局部最优. 而锦标赛选择策略[16]是基于局部竞争机制的选择策略. 本文采用锦标赛策略对跟随蜂蜜源进行选择, 即随机选取两个个体进行适应度比较, 对适应度大的个体加 1 分, 所有个体重复此过程, 最后得分最高者权重最大. 锦标赛选择策略只将适应度值的相对值作为选择标准, 从而避免了超级个体对算法的影响.

跟随蜂选择某个蜜源的概率 ρ_i 为

$$\rho_i(t) = c_i(t) \Big/ \sum_{i=1}^{N} c_i(t), \tag{6}$$

其中 c_i 为每个个体的得分.

2.3　Tent 混沌人工蜂群算法流程

Step 1: 初始化相关参数. 种群规模 SN、蜜源个数 $N = (\mathrm{SN}/2)$, 同一蜜源被限制次数 Limit 等参数.

Step 2: 初始迭代次数 $\mathrm{iter} = 0$, 利用 1.1 节 Tent 混沌映射产生 D 维 SN 个不同轨迹的混沌序列 $z_i^d(t)$, 并将其通过下式载波到解空间产生初始个体 X_i:

$$X_i^d(t) = X_{\min}^d + (X_{\max}^d - X_{\min}^d) z_i^d(t). \tag{7}$$

Step 3: 计算适应度 $F(X_i)$, 适应度值大的前 N 个向量作为蜜源位置, 对应 N 个引领蜂. 初始标志向量 $\mathrm{trial}(i) = 0$, 记录引领蜂停留在同一蜜源的次数.

Step 4: 引领蜂 i 按照式 (5) 在蜜源附近搜索产生新解 V_i, 并计算适应度值 $F(V_i)$.

Step 5: 如果 $F(V_i) > F(X_i)$, 则 $X_i = V_i$, $\mathrm{trial}(i) = 0$; 否则原解 X_i 不变, $\mathrm{trial}(i) = \mathrm{trial}(i) + 1$.

Step 6: 跟随蜂根据式 (6) 计算的选择概率 $\rho_i(t)$ 选择蜜源, 并由式 (5) 进行邻域搜索以产生新解 V_i, 计算适应度值 $F(V_i)$.

Step 7: 按 Step 5 执行.

Step 8: 若 $\mathrm{trial}(i) > \mathrm{Limit}$, 则引领蜂 i 放弃当前蜜源而成为侦察蜂, 侦察蜂根据 1.2 节的 Tent 混沌搜索产生一个新解 V_i 替代蜜源.

Step 9: 记录当前所有蜜蜂找到的最优蜜源, 保存迄今为止的最优解.

Step 10: 更新 $\mathrm{iter} = \mathrm{iter} + 1$, 判断是否满足终止条件, 如满足, 则输出最优解, 否则返回 Step 4.

3　Tent 混沌粒子群算法

3.1　粒子群优化算法

假设在 D 维目标解空间中, $S = X_1, X_2, \cdots, X_M$ 表示 M 个潜在解组成的种群; 粒子 i 的位置记作 $X_i^d(t), d = 1, 2, \cdots, D$; 粒子 i 的飞行速度记作 $v_i^d(t)$; 个体极值 $X_{\mathrm{best}}^d(t)$ 表示粒子 i 自身找到的最好位置; 全局极值 $G_{\mathrm{best}}^d(t)$ 表示整个种群迄今为止找到的最优位置. 则粒子 i 的位置和速度更新方程如下:

$$\begin{cases} v_i^d(t+1) = W v_i^d(t) + C_1 R_1 (X_{\mathrm{best}}^d(t) - X_i^d(t)) + \\ \qquad\qquad C_2 R_2 (G_{\mathrm{best}}^d(t) - X_i^d(t)), \\ X_i^d(t+1) = X_i^d(t) + v_i^d(t+1). \end{cases} \tag{8}$$

其中: $X_i^d(t)$、$X_i^d(t+1)$、$v_i^d(t)$ 和 $v_i^d(t+1)$ 分别是粒子 i 的当前时刻、下一时刻的位置和速度; C_1 和 C_2 是学习因子, 一般取值在 1.5 ∼ 2.0 之间; R_1 和 R_2 是 [0, 1] 之间的随机数; W 是惯性权重.

3.2　Tent 粒子群算法流程

在 PSO 中引入 Tent 混沌搜索思想, 本文提出了 Tent 混沌粒子群算法 (TCPSO). 其算法流程描述如下.

Step 1: 初始化算法相关参数. 包括种群规模 M、粒子维数 D、各维的取值范围 $[X_{\min}^d, X_{\max}^d]$ 等.

Step 2: 随机产生 D 维每个分量在 $(0, 1)$ 之间的向量 $z_i^d(0)$. 根据 1.1 节的 Tent 混沌映射产生 L 个不同轨迹的混沌序列 $z_i^d(t)$, 并通过式 (7) 将其各个分量载波到对应变量的取值范围内.

Step 3: 按式 (4) 计算粒子适应度值, 从 L 个初始解中选取适应度较好的 M 个位置作为初始位置, 并

随机产生 M 个初始速度.

Step 4: 如果 $F(X_i^d) < F(X_{\text{best}}^d)$, 则更新 X_{best}^d.

Step 5: 比较更新后的 $F(X_{\text{best}}^d)$ 和 $F(G_{\text{best}}^d)$, 如果 $F(X_{\text{best}}^d) < F(G_{\text{best}}^d)$, 则更新 G_{best}^d.

Step 6: 根据式 (8) 更新粒子的速度和位置.

Step 7: 对当前最优位置 G_{best}^d, 利用 1.2 节 Tent 混沌搜索产生最优新解 G_{best}.

Step 8: 利用 G_{best} 替代种群任意一粒子的位置.

Step 9: 判断算法是否满足终止条件. 如果满足, 则输出全局最优位置和其适应度值; 否则返回 Step 4.

4 Tent 混沌人工蜂群与粒子群混合算法

为使 PSO 算法有效跳出局部最优值和提高 ABC 算法的局部搜索能力, 本文提出了 Tent 混沌人工蜂群与粒子群混合算法 (HTCAP). 该算法引入 Tent 混沌反向学习策略、混沌搜索思想和重组算子. 在每一次迭代时利用重组算子让两个种群互相跟踪对方的全局最优解, 并对两种算法每次进化得到的最优解通过选择概率选取对应算法的最优位置, 作为下一次迭代中 TCABC 算法中跟随蜂的邻域蜜源位置和 TCPSO 算法的全局最优位置, 从而实现算法间实时信息交流以达到同步进化, 使算法具有全局并行搜索和局部串行搜索的能力.

4.1 混沌反向学习策略

为保持种群多样性和使初始种群个体尽可能均匀分布, 本文结合 Tent 混沌初始化方法和基于反向学习初始化策略, 提出了 Tent 混沌反向学习初始化策略. 即利用 Tent 混沌序列产生初始解 X_i, 再根据 $\text{OP}_i = K(X_{\min}^d - X_{\max}^d) - X_i$, 求出每个初始解 X_i 所对应的反向解 OP_i, 其中 K 为 [0, 1] 间的随机数. 最后对产生的所有解排序, 将适应度值较优的前 N 个解作为初始种群的解, 这将有助于提高解的质量和求解效率.

4.2 重组算子

本文利用重组算子产生一个新的最优解 Best, 产生后的 Best 作为 TCABC 算法跟随蜂的邻域蜜源和 TCPSO 算法的全局最优位置. 为获取最优解 Best, TCABC 算法的最优解 A_{best} 和 TCPSO 算法的全局最优解 G_{best} 的适应度值都由式 (4) 计算, 最优解选择概率表达式如下:

$$P_{\text{best}} = \frac{F_{A_{\text{best}}}}{F_{G_{\text{best}}} + F_{A_{\text{best}}}}. \tag{9}$$

其中: P_{best} 是选择 TCABC 算法获取最优解 A_{best} 的概率, $F_{A_{\text{best}}}$ 是 TCABC 算法最优解 A_{best} 的适应度值, $F_{G_{\text{best}}}$ 是 TCPSO 算法全局最优解 G_{best} 的适应度值.

随机为问题的每一维产生一个 [0, 1] 间的随机数 r. 如果第 d 维产生的随机数 r 小于等于 P_{best}, 则该维的最优值使用 TCABC 算法获取的最优解 A_{best}^d 替代; 否则使用 TCPSO 算法获取的全局最优解 G_{best}^d 替代, 该维最优解的表达式定义如下:

$$\text{Best}_d = \begin{cases} A_{\text{best}}^d, & r < P_{\text{best}}; \\ G_{\text{best}}^d, & \text{otherwise}. \end{cases} \tag{10}$$

其中: Best_d 表示最优解 Best 的第 d 维最优解, $d = 1, 2, \cdots, D$; A_{best}^d 和 G_{best}^d 分别表示由 TCABC 算法和 TCPSO 算法所获取的第 d 维最优解.

应用重组算子可加深粒子与蜜蜂之间的联系和信息交流, 这样 TCABC 算法通过直接使用全局最优信息, 使其局部搜索能力得到提高, 而且 TCPSO 算法的全局搜索能力得到改善, 能有效地跳出局部最优, 从而均衡算法的全局搜索和局部探索能力.

4.3 HTCAP 算法流程

HTCAP 算法流程如图 1 所示, 主要步骤如下.

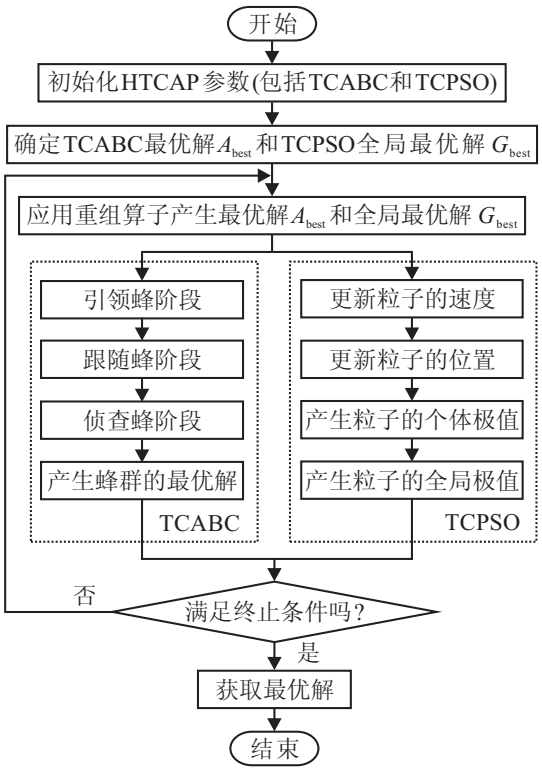

图 1 HTCAP 的流程图

Step 1: 初始化 HTCAP 算法相关参数. 包括群体规模 N、最大迭代次数 $G_{\max}$、求解精度等.

Step 2: 利用 4.1 节中的混沌反向学习策略产生初始种群的解 $X_i(i = 1, 2, \cdots, N)$, 计算其适应度值, 并将种群等分为双种群 A 群和 P 群.

Step 3: 利用 2.3 节的 TCABC 算法对 A 群中所有个体优化.

Step 4: 利用 3.2 节的 TCPSO 算法对 P 群中所有个体进行速度、位置更新等操作.

Step 5: 求出 A 群最佳个体 A_{best} 及相应适应度值.

Step 6: 求出P群最佳个体G_{best}及相应适应度值.

Step 7: 利用4.2节中的重组算子产生一个新的最优解Best, 并将Best作为下一次迭代TCABC算法跟随蜂的邻域蜜源和TCPSO算法的全局极值.

Step 8: 记录迄今为止最好的解.

Step 9: 判断算法是否达到最大迭代次数或满足求解精度要求. 如果满足, 则输出最优位置和其适应度值, 否则转Step 3.

5　实验与结果分析

为了验证本文HTCAP算法的寻优精度和收敛速度, 选取CEC2010中12个Benchmark标准函数进行寻优测试. 这12个函数具有不同的特性, 可以充分考察算法的寻优能力. 函数表达式、取值范围及全局最小值、特性等如表1所示. 其中: 函数f_1、f_2、$f_7 \sim f_{12}$是单峰函数, 主要用于测试算法的寻优精度和收敛速度, 另外由于Rosenbrock函数是非凸、病态单峰函数, 有局部极小值, 主要用于测试算法的收敛速度和执行效率; 函数$f_3 \sim f_6$是复杂的非线性多峰函数, 有许多局部极值点, 主要用于测试算法的全局搜索能力、跳出局部极值并避免早熟的收敛能力. 实验在主频2.5 G, 内存3 G, Windows 7操作系统, Matlab7.10的计算机上实现. 实验中较好的算法结果在表2～表5中用粗体表示.

表 1　12 个标准测试函数

函数名称	表达式及取值范围	全局最小值	特性
Sphere	$f_1=\sum_{i=1}^{D}x_i^2,\ x_i\in[-100,100]$	0	单峰
Rosenbrock	$f_2=\sum_{i=1}^{D-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2),\ x_i\in[-30,30]$	0	单峰
Rastrigin	$f_3=\sum_{i=1}^{D}(100(x_i^2-10\cos(2\pi x_i)+10)),\ x_i\in[-5.12,5.12]$	0	多峰
Griewank	$f_4=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,\ x_i\in[-600,600]$	0	多峰
Ackley	$f_5=-20\exp\Big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D}x_i^2}\Big)-\exp\Big(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)+20+e\Big),\ x_i\in[-32,32]$	0	多峰
Schwefel2.26	$f_6=-\sum_{i=1}^{D}(x_i\sin(\sqrt{\lvert x_i\rvert})),\ x_i\in[-500,500]$	$-418.982\,9D$	多峰
Schwefel2.22	$f_7=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert,\ x_i\in[-10,10]$	0	单峰
Schwefel1.2	$f_8=\sum_{i=1}^{D}\Big(\sum_{j=1}^{i}x_j\Big)^2,\ x_i\in[-100,100]$	0	单峰
Zakharow	$f_9=\sum_{i=1}^{D}x_i^2+\Big(\sum_{i=1}^{D}0.5ix_i\Big)^2+\Big(\sum_{i=1}^{D}0.5ix_i\Big)^4,\ x_i\in[-5,5]$	0	单峰
Step	$f_{10}=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor,\ x_i\in[-100,100]$	0	单峰
Schwefel2.21	$f_{11}=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant D\},\ x_i\in[-100,100]$	0	单峰
Quartic	$f_{12}=\sum_{i=1}^{D}ix_i^4+\text{rand}[0,1),\ x_i\in[-1.28,1.28]$	0	单峰

5.1　Tent 混沌反向学习初始化策略的性能分析

为了评价Tent混沌反向学习策略的性能, 并与随机初始化和反向学习初始化策略进行比较, 分别对表1中标准函数$f_1 \sim f_6$进行初始化种群测试, 并对适应度值和种群多样性进行检验. 种群多样性定义如下:

$$\text{Diversity}=\frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{D}\sum_{d=1}^{D}(x_{i,d}-\bar{x}_d)^2}\ . \qquad (11)$$

其中: N表示蜜源的个数; D表示函数变量个数; $\bar{x}_d$表示所有蜂群第d维的均值, 即蜂群中心位置.

对测试函数分别取50维, 使用ABC算法, 种群规模为100, Limit = 100, 最大迭代次数$G_{\max}=3\,000$, 独立运行30次. 由于所测试函数都求最小值, 适应度值越小越好, 种群的多样性值越大越好. 由式(11)可知, Diversity越小, 表示种群越集中; 反之, 表示种群越分散, 即种群的多样性越好. 各种群初始化策略的性能比较如表2所示.

表 2　种群初始化策略的性能比较

函数	随机初始化		反向学习初始化		Tent 混沌反向学习	
	Fitness	Diversity	Fitness	Diversity	Fitness	Diversity
f_1	1.294 7e+05	58.876 5	1.187 0e+05	65.507 9	**8.005 0e+04**	**118.452 4**
f_2	8.699 4e+08	17.364 7	6.707 5e+08	17.380 8	**3.708 3e+08**	**35.219 0**
f_3	750.570 9	2.982 27	886.795 4	3.448 44	**715.309 8**	**5.763 84**
f_4	2.104 9e+03	344.612 4	1.704 9e+03	349.472 7	**1.012 1e+03**	**681.109 2**
f_5	21.151 6	18.752 8	21.056 8	19.513 2	**20.439 3**	**39.196 9**
f_6	-3.563 1e+03	289.492 4	-3.101 2e+03	290.798 9	**-2.815 1e+03**	**617.504 3**

从表2可以看出, 与其他两种初始化策略相比, 混沌反向学习初始化策略的适应度和种群多样性都好. 总之, 通过结合Tent混浊映射的全局遍历性和分布的均匀性特性及反向学习方法的优势, 使得Tent混沌反向初始化策略能提高种群的多样性, 从而获得好的初始解.

5.2 HTCAP算法的性能分析

为了验证HTCAP算法性能的有效性, 分别与基本的ABC、PSO进行寻优测试. 在实验中, $[X^d_{\min}, X^d_{\max}]$是函数输入变量的取值范围, 所有算法的种群规模均为100. 其中: HTCAP包含50个粒子, 25个引领蜂, 25个跟随蜂; ABC包含50个引领蜂, 50个跟随蜂; PSO包含100个粒子; 算法最大迭代次数$G_{\max} = (D \times 10\,000)/N$. TCPSO和PSO算法的惯性因子$W = (G_{\max} - t)/G_{\max}$, 学习因子$C_1 = C_2 = 1.496\,18$, 最大速度$v_{\max}$为搜索空间的一半; ABC、TCABC的跟随蜂采用锦标赛策略选择蜜源, Limit $= (\mathrm{NE} \times D)/2$, NE为蜜蜂总个数; 对于涉及到混沌搜索的算法, 其混沌搜索次数$C_{\max} = 300$.

通过对每个测试函数运行30次所得适应度的最优值(Best)、最差值(Worst)、均值(Mean)和标准方差(Std)来考察算法性能, 当求解精度达到10^{-20}时就假定结果为0. PSO、ABC和HTCAP算法性能比较如表3所示.

表 3 测试函数的优化性能比较

函数	算法	维数	Best	Worst	Mean	Std	维数	Best	Worst	Mean	Std
	PSO	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
f_1	ABC	30	2.605 51e-016	5.603 92e-016	4.744 03e-016	9.219 69e-017	60	9.474 35e-016	1.232 03e-015	1.110 64e-015	1.069 26e-016
	HTCAP	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	PSO	30	3.455 71e+000	8.110 53e+001	2.856 74e+001	2.363 89e+001	60	3.044 19e+001	9.015 71e+004	9.208 72e+003	2.673 14e+004
f_2	ABC	30	6.797 79e-005	0.160 14e-001	0.019 9e-002	3.370 12e-002	60	2.564 73e-004	0.432 51e-001	0.036 4e-002	7.909 21e-002
	HTCAP	30	**2.165 2e-006**	**9.390 16e-003**	**1.457 34e-006**	**2.454 31e-007**	60	**4.533 0e-005**	**1.783 25e-003**	**2.813 72e-004**	**4.101 09e-005**
	PSO	30	1.791 43e+001	6.772 81e+001	3.648 42e+001	1.153 05e+001	60	7.850 17e+001	1.523 85e+002	1.148 23e+002	1.938 91e+001
f_3	ABC	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	HTCAP	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	PSO	30	**0**	1.176 52e-001	2.016 33e-002	2.332 06e-002	60	**0**	3.920 21e-002	5.978 31e-003	1.104 51e-002
f_4	ABC	30	**0**	1.109 82e-016	9.250 99e-017	4.136 96e-017	60	**0**	1.110 21e-016	8.510 97e-017	4.695 67e-017
	HTCAP	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	PSO	30	**7.993 62e-016**	**1.509 95e-015**	**8.590 57e-017**	**1.853 61e-017**	60	1.510 99e-014	3.268 63e-014	2.392 18e-014	4.565 74e-015
f_5	ABC	30	2.931 10e-014	3.995 68e-014	3.273 94e-014	2.510 04e-015	60	6.483 67e-014	8.614 53e-014	7.559 89e-014	5.306 80e-015
	HTCAP	30	**7.993 62e-016**	**1.509 95e-015**	1.130 98e-016	3.544 81e-017	60	**1.509 94e-015**	**3.286 31e-015**	**2.386 27e-015**	**5.406 12e-017**
	PSO	30	−1.254 53e+004	−1.257 41e+004	−1.256 75e+004	5.369 36e-002	60	−2.513 89e+004	−2.513 87e+004	−2.513 89e+004	8.893 49e-002
f_6	ABC	30	**−1.256 95e+004**	−1.256 45e+004	**−1.256 95e+004**	1.235 27e-012	60	**−2.513 90e+004**	−2.513 56e+004	**−2.513 90e+004**	5.326 94e-011
	HTCAP	30	**−1.256 95e+004**	**−1.256 95e+004**	**−1.256 95e+004**	**6.947 21e-016**	60	**−2.513 90e+004**	**−2.513 90e+004**	**−2.513 90e+004**	**1.876 23e-014**
	PSO	30	1.363 46e-011	1.532 18e-009	1.574 36e-011	2.165 42e-010	60	3.983 32e-009	5.973 21e-009	4.028 38e-009	3.897 34e-010
f_7	ABC	30	1.674 69e-006	7.327 75e-006	2.798 64e-006	1.256 81e-006	60	4.897 42e-006	9.834 75e-005	5.643 28e-006	1.867 43e-006
	HTCAP	30	**0**	**0**	**0**	**0**	60	**3.654 83e-015**	**6.836 49e-014**	**4.214 73e-014**	**1.163 41e-014**
	PSO	30	1.325 31e-001	8.532 43e-001	3.675 36e-001	1.786 85e-001	60	4.832 67e-001	9.387 46e-001	5.982 34e-001	3.189 35e-001
f_8	ABC	30	6.136 72e-001	8.396 48e-001	6.265 46e-001	1.986 78e-001	60	8.893 49e-001	9.894 35e-001	8.973 82e-001	5.759 45e-001
	HTCAP	30	**3.798 35e-007**	**5.893 40e-005**	**2.485 74e-005**	**8.237 38e-008**	60	**4.931 42e-006**	**8.839 17e-005**	**4.829 31e-005**	**2.832 97e-007**
	PSO	30	4.054 65e-003	3.786 53e-002	6.656 45e-003	1.457 93e-003	60	8.384 19e-003	6.384 72e-002	9.384 72e-003	4.983 41e-003
f_9	ABC	30	1.643 85e-001	2.756 32e-001	1.896 57e-001	3.674 82e-003	60	4.895 23e-001	3.834 58e-001	2.987 45e-001	4.839 79e-003
	HTCAP	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	PSO	30	**0**	**0**	**0**	**0**	60	1.384 79e-009	9.389 47e-009	4.893 42e-009	1.253 49e-010
f_{10}	ABC	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	HTCAP	30	**0**	**0**	**0**	**0**	60	**0**	**0**	**0**	**0**
	PSO	30	1.567 56e-002	3.758 35e-002	2.868 43e-002	4.879 84e-002	60	5.384 25e-002	8.370 21e-002	6.892 03e-002	8.378 49e-002
f_{11}	ABC	30	1.138 46e-001	2.654 72e-001	1.845 35e-001	3.267 33e-002	60	3.983 74e-001	8.823 91e-001	5.983 74e-001	7.834 79e-002
	HTCAP	30	**8.893 21e-004**	**6.347 92e-003**	**5.786 43e-003**	**8.645 64e-004**	60	**4.238 92e-002**	**3.843 39e-001**	**1.658 42e-001**	**5.987 41e-002**
	PSO	30	8.3789 3e-003	1.283 90e-001	2.783 29e-002	2.832 92e-003	60	6.028 93e-003	9.435 78e-002	1.892 89e-002	4.238 71e-002
f_{12}	ABC	30	7.456 84e-003	1.537 43e-002	1.145 36e-002	1.855 46e-003	60	1.398 47e-002	3.974 59e-002	2.980 12e-002	1.987 34e-002
	HTCAP	30	**5.983 49e-006**	**4.989 02e-005**	**1.864 22e-005**	**3.264 21e-006**	60	**3.893 75e-003**	**9.985 47e-003**	**8.943 62e-003**	**6.532 16e-003**

从表3可以看出: 对于单峰函数Sphere和30维的Step, PSO和HTCAP算法性能相同; 而多峰函数Ackley在30维时, PSO算法性能要比HTCAP算法的性能好, 但当其维数为60时, HTCAP的性能要比PSO的性能稍好. 对于其他函数, HTCAP的性能相对要比PSO的性能好. 尤其当函数存在很多局部最优值时, 算法全局搜索能力比算法局部探索能力更重要. HTCAP算法的全局搜索能力源于ABC算法, 所以

在优化多峰函数 Rastrigin 和 Griewank 时性能比 PSO 算法好. 尽管 Rosenbrock 不是多峰函数, 但因为它是非凸、病态单峰函数, 其全局搜索能力更重要, 所以 HTCAP 算法优化性能比 PSO 算法性能好. 对于多峰函数 Rastrigin 和单峰函数 Step, ABC 和 HTCAP 算法的性能相同; 对于单峰函数 Sphere 和 Rosenbrock, HTCAP 算法性能明显优于 ABC 算法, 这表明 ABC 算法全局搜索能力明显比 PSO 和 HTCAP 算法局部搜索能力要弱. 而且对于单峰函数, PSO 有较好的均值与方差; 对于多峰函数, ABC 有较好的均值与方差; HTCAP 对所有函数都有较好的均值与方差. 综上可知, HTCAP 算法整体性能比 PSO 和 ABC 都好, 主要是因为 HTCAP 使用 Tent 混沌反向学习策略和重组算子, 在增加种群多样性的同时, 提高了收敛速度.

为了进一步测试 HTCAP 算法的性能, 将其与 CGPSO[12]、MAEPSO[9]、EOPSO[11]、IABC[17]、HABC[18]、LRABC[19]等最近较新改进算法在各标准函数 30 维的均值 (Mean)、标准方差 (Std) 和平均运行时间 (AvgTime) 的优化结果进行比较, 测试比较结果如表 4 所示. 所有基于 PSO 的比较算法的参数设置相同于上一个实验设置. 另外, MAEPSO 中的 $M = 5$, 初始方差 σ_0 为优化变量的范围, $T_d = 0.5$; EOPSO 中的 JR $= 0.3$; 所有基于 ABC 的比较算法的参数设置相同于上一个实验的参数设置.

表 4　算法对 12 个标准函数在 30 维的优化性能比较

函数	性能	CGPSO	MAEPSO	EOPSO	IABC	HABC	LRABC	SATC-ABC
	Mean	4.179 53e-018	**0**	**0**	**0**	**0**	**0**	**0**
f_1	Std	6.153 29e-018	**0**	**0**	**0**	**0**	**0**	**0**
	AvgTime/s	24.692 7	23.598 2	22.876 1	23.983 9	23.894 6	20.387 2	**19.607 2**
	Mean	1.532 37e-002	2.843 62e-006	3.316 35e-002	1.564 79e+002	2.816 35e+001	1.364 18e-001	**1.457 34e-006**
f_2	Std	3.542 72e-003	1.648 95e-006	2.564 38e-002	8.193 57e+001	5.442 79e-001	8.353 17e-002	**2.454 31e-007**
	AvgTime/s	26.928 4	27.342 9	26.354 1	24.018 2	26.253 8	23.184 3	**20.356 1**
	Mean	4.212 46e-005	2.041 43e-008	**0**	**0**	**0**	**0**	**0**
f_3	Std	8.426 25e-006	7.852 36e-008	**0**	**0**	**0**	**0**	**0**
	AvgTime/s	27.439 8	25.593 7	23.891 4	24.291 9	26.885 3	20.512 7	**20.5018**
	Mean	3.818 09e-006	1.624 79e-010	**0**	**0**	**0**	**0**	**0**
f_4	Std	4.276 43e-006	6.178 36e-010	**0**	**0**	**0**	**0**	**0**
	AvgTime/s	28.439 7	29.348 1	28.932 6	30.012 8	29.742 9	**26.893 4**	29.234 7
	Mean	1.652 24e-011	2.879 64e-015	3.424 51e-015	3.875 32e-014	8.455 31e-016	9.645 75e-015	**1.130 98e-016**
f_5	Std	1.623 18e-011	3.162 85e-015	1.453 52e-015	2.534 89e-015	5.537 22e-017	8.325 27e-016	**3.544 81e-017**
	AvgTime/s	27.871 4	26.348 9	25.034 8	27.791 4	26.967 5	23.849 1	22.598 3
	Mean	−1.259 86e+004	−1.842 39e+004	−1.568 35e+004	**−1.256 95e+004**	**−1.256 95e+004**	**−1.256 95e+004**	**−1.256 95e+004**
f_6	Std	1.856 23e-005	2.546 94e-004	1.316 57e-002	3.956 26e-012	3.643 87e-011	1.965 38e-011	**6.947 21e-016**
	AvgTime/s	28.967 6	26.943 2	24.030 5	26.982 3	22.709 2	26.673 2	**22.671 5**
	Mean	2.535 41e-011	1.834 69e-011	3.837 49e-014	**0**	**0**	**0**	**0**
f_7	Std	1.642 25e-011	1.193 49e-011	8.838 21e-015	**0**	**0**	**0**	**0**
	AvgTime/s	20.475 4	19.348 9	18.489 2	19.325 9	18.214 2	20.875 8	**18.0154**
	Mean	3.6742 2e-002	8.381 73e-002	4.897 13e-003	2.837 53e-003	8.152 47e-004	1.975 25e-003	**2.485 74e-005**
f_8	Std	2.538 48e-002	3.938 42e-002	6.932 87e-004	6.235 68e-003	5.196 65e-003	6.564 15e-005	**8.237 38e-008**
	AvgTime/s	22.942 5	23.284 3	22.592 6	23.436 1	19.349 7	21.054 7	**19.108 2**
	Mean	3.134 74e-005	2.983 47e-005	1.896 29e-005	4.135 37e-005	3.147 53e-005	1.674 18e-010	**0**
f_9	Std	4.647 52e-005	2.895 21e-005	1.987 31e-006	4.128 32e-006	2.538 29e-006	2.375 32e-013	**0**
	AvgTime/s	30.481 5	29.965 8	29.892 5	31.451 8	30.880 26	29.118 3	**26.656 8**
	Mean	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f_{10}	Std	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	AvgTime/s	21.396 1	20.340 6	19.193 2	19.258 4	17.892 1	17.456 3	**17.101 2**
	Mean	3.452 72e-003	2.834 78e-003	2.123 12e-003	6.786 36e-003	3.186 59e-003	**0**	5.786 43e-003
f_{11}	Std	6.053 27e-003	1.734 82e-003	1.092 34e-003	1.653 18e-002	8.013 58e-003	**0**	8.645 64e-004
	AvgTime/s	19.023 7	19.548 1	18.348 7	19.031 7	**16.948 3**	17.457 8	17.537 5
	Mean	4.075 35e-004	2.143 53e-007	**0**	9.453 16e-003	**0**	2.234 71e-004	1.864 22e-005
f_{12}	Std	8.963 36e-005	1.635 34e-006	**0**	2.169 64e-003	**0**	2.216 54e-004	3.264 21e-006
	AvgTime/s	18.847 2	19.384 7	18.673 9	18.239 8	18.325 4	19.031 5	**18.125 9**

由表 4 可知: 对于函数 f_1 和 f_{10}, 所有算法都能获得很好的均值和标准方差; 对于函数 f_{11}, LRABC 表现最优; 对于函数 f_{12}, EOPSO 和 HABC 表现最优; 对于函数 f_3, CGPSO 和 MAEPSO 表现较差; 对于函数 f_7, IABC、HABC、LRABC 和 HTCAP 表现最优; 除函数 f_{11} 和 f_{12} 外, HTCAP 算法的均值和标准方差都较

好; 特别是对于函数 f_6, HTCAP 算法取得了很好的标准方差, 并获得了理论最优值, 这表明 HTCAP 算法整体具有较好的稳定性和鲁棒性.

另外, 除了 LRABC 算法对函数 f_4 和 HABC 算法对函数 f_{11} 的执行时间稍小于 HTCAP 算法外, HTCAP 算法的执行时间相对而言用时稍短, 主要原因是 HTCAP 算法引入的 Tent 映射是通过贝努利移位变换得到, 即将 Tent 小数部分的二进制数进行无符号左移, 它能充分地利用计算机的特性, 能更快、更有效地处理大数量级数据序列的运算.

为了更进一步测试 HTCPA 算法的性能, 将其与 IABAP[13]和SPSO-ABC[14]两种混合算法进行寻优测试, HTCAP 算法的参数设置同上, IABAP 和 SPSO-ABC 算法的参数设置分别参见文献 [13] 和文献 [14]. 对表 1 中的前 6 个标准函数在 10、30、50、100、200 维上的优化结果比较如表 5 所示.

由表 5 可知: 对于 50 维的 f_4 和 10、30、50 维的 f_1 函数, HTCAP 与 SPSO-ABC 算法的性能相同; 对于 10 维的 f_3, HTCAP 与 IABAP 算法性能相同; 对于 10 维的 f_4, HTCAP 算法表现出最差的均值和方差, 主要是因为函数 f_4 虽是多峰特性, 但函数在 30 维及以上时却表现出单峰特性[3]; 对于其他函数, HTCAP 算法在均值和标准方差上均优于其他算法, 这主要是因为 HTCAP 算法是利用重组算子选择最优个体作为 TCABC 跟随蜂的邻域蜜源和 TCPSO 的全局极值, 即利用人工蜂群与粒子群之间的信息交互平衡算法的局部搜索能力和全局搜索能力, 进而提高解的质量.

综上分析, HTCAP 算法在计算精度上有明显提高, 不仅具有较强的全局搜索能力, 而且具有较强的局部搜索能力, 能有效地克服 ABC 和 PSO 算法后期收敛速度慢和易陷入局部最优等缺陷, 并且随着目标维数和增加, 能够保持较好的有效性和鲁棒性.

表 5 算法对 6 个标准函数在不同维数的优化性能比较

函数	算法	性能	维数				
			10	30	50	100	200
f_1	IABAP	Mean	4.874 53e-017	5.386 58e-016	1.293 51e-015	4.425 79e-015	2.254 68e-014
		Std	1.154 79e-017	6.764 83e-017	1.729 83e-016	8.856 73e-016	9.053 61e-015
	SPSO-ABC	Mean	**0**	**0**	**0**	5.786 22e-018	7.567 31e-016
		Std	**0**	**0**	**0**	8.657 34e-018	3.531 82e-016
	HTCAP	Mean	**0**	**0**	**0**	**0**	**6.185 64e-19**
		Std	**0**	**0**	**0**	**0**	**5.0543 6e-019**
f_2	IABAP	Mean	8.054 26e-002	1.154 68e-001	1.163 28e-001	1.634 51e-001	4.108 76e-001
		Std	7.967 82e-002	2.176 34e-001	1.989 36e-001	2.679 64e-001	4.967 45e-001
	SPSO-ABC	Mean	2.834 92e-002	6.082 39e-002	9.576 48e-002	3.983 74e-001	1.384 92e+001
		Std	5.238 78e-002	4.893 41e-002	8.874 29e-002	5.349 17e-001	2.582 89e-001
	HTCAP	Mean	**1.643 68e-003**	**1.457 34e-006**	**2.713 72e-003**	**7.034 57e-003**	**5.536 31e-002**
		Std	**2.453 82e-002**	**2.454 31e-007**	**3.101 12e-004**	**8.754 21e-003**	**1.576 43e-001**
f_3	IABAP	Mean	**0**	1.463 56e-016	3.432 61e-014	3.965 32e-010	1.537 65e-004
		Std	**0**	4.889 53e-016	4.659 82e-014	1.698 42e-009	1.023 68e-003
	SPSO-ABC	Mean	6.643 49e-013	4.789 76e-015	8.987 46e-015	4.892 34e-012	9.217 45e-009
		Std	2.786 94e-012	2.860 35e-015	3.256 84e-016	8.982 91e-013	3.839 47e-010
	HTCAP	Mean	**0**	**0**	**0**	**3.413 67e-015**	**4.678 49e-014**
		Std	**0**	**0**	**0**	**5.216 54e-016**	**1.357 64e-015**
f_4	IABAP	Mean	4.534 15e-017	4.442 16e-017	7.364 24e-017	2.643 17e-015	1.579 73e-014
		Std	5.012 36e-017	1.875 34e-017	3.543 255e-017	3.135 76e-016	8.506 46e-015
	SPSO-ABC	Mean	3.987 98e-013	8.239 82e-017	**0**	9.328 14e-010	8.237 49e-014
		Std	2.974 35e-012	2.340 99e-017	**0**	3.923 16e-011	3.839 45e-014
	HTCAP	Mean	3.183 45e-004	**0**	**0**	**0**	**3.053 72e-017**
		Std	4.238 91e-004	**0**	**0**	**0**	**1.794 58e-018**
f_5	IABAP	Mean	4.875 18e-015	2.932 35e-014	2.956 17e-014	3.422 86e-014	1.059 48e-013
		Std	3.486 34e-016	2.065 99e-015	4.198 32e-015	6.623 56e-015	1.037 81e-014
	SPSO-ABC	Mean	2.256 73e-015	3.204 52e-014	3.692 86e-014	5.542 65e-013	2.234 26e-013
		Std	8.326 42e-017	2.186 32e-015	5.052 85e-015	7.556 02e-015	1.653 21e-014
	HTCAP	Mean	**6.534 16e-017**	**1.130 98e-016**	**2.059 25e-016**	**2.349 18e-015**	**1.543 45e-014**
		Std	**1.214 57e-017**	**3.544 81e-017**	**3.457 92e-018**	**1.542 62e-016**	**2.467 28e-015**
f_6	IABAP	Mean	**−4.189 83e+003**	**−1.256 95e+004**	**−2.094 91e+004**	**−4.189 83e+004**	**−8.379 66e+004**
		Std	2.775 13e-012	2.866 14e-012	3.637 98e-012	1.560 98e-007	3.894 01e-005
	SPSO-ABC	Mean	**−4.189 83e+003**	**−1.256 95e+004**	**−2.094 91e+004**	**−4.189 83e+004**	**−8.377 22e+004**
		Std	4.394 78e-012	3.289 39e-012	4.387 49e-012	2.894 31e-005	4.790 23e-003
	HTCAP	Mean	**−4.189 83e+003**	**−1.256 95e+004**	**−2.094 91e+004**	**−4.189 83e+004**	**−8.379 66e+004**
		Std	**1.398 76e-018**	**6.947 21e-016**	**2.489 52e-014**	**8.364 19e-009**	**5.734 25e-006**

6 结　论

针对ABC和PSO各自的优缺点, 本文提出了Tent混沌人工蜂群与粒子群混合算法(HTCAP). 该算法具有如下主要特点: 1) 引入Tent混沌反向学习策略初始化种群, 以获得分布均匀的初始个体, 从而提高种群多样性; 2) 引入Tent混沌搜索算法在局部最优解的附近产生许多邻域, 不但使其能有效地跳出局部最优点, 而且能加快其收敛速度; 3) 应用重组算子为每次迭代产生最优解, 并将其作为跟随蜂的邻域蜜源和粒子的全局极值, 利用TCABC和TCPSO的实时信息交互来平衡HTCAP算法的全局搜索能力和局部搜索能力.

通过对12个标准函数寻优测试, 仿真结果表明, HTCAP算法对函数类型不太敏感, 对有局部极值或易陷入局部最优的非线性函数也具有较高的寻优精度, 并具有较好的稳定性和鲁棒性, 能确保整个种群的全局优化能力. 下一步工作将考虑把该算法应用到约束优化、多目标优化、网络组播优化等实际组合优化问题, 并结合其他智能优化算法, 提出性能更好的全局优化算法.

参考文献(References)

[1] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Kayseri: Erciyes University, 2005.

[2] Akay B. Performance analysis of artificial bee colony algorithm on numerical optimization problems[D]. Kayseri: Graduate School of Natural and Applied Sciences, Erciyes University, 2009.

[3] Akay B, Karaboga D. A modified artificial bee colony algorithm for real-parameter optimization[J]. Information Sciences, 2012, 192(6): 120-142.

[4] 高卫峰, 刘三阳, 黄玲玲. 受启发的人工蜂群算法在全局优化问题中的应用[J]. 电子学报, 2012, 40(12): 2396-2403.
(Gao W F, Liu S Y, Huang L L. Inspired artificial bee colony algorithm for global optimization problems[J]. Acta Electronica Sinica, 2012, 40(12): 2396-2403.)

[5] Zhu G P, Sam K. Gbestguided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation, 2010, 217(7): 3166-3173.

[6] 暴励, 曾建潮. 自适应搜索空间的混沌蜂群算法[J]. 计算机应用研究, 2010, 27(4): 1330-1334.
(Bao L, Zeng J C. Self-adapting search space chaos artificial bee colony algorithm[J]. Application Research of Computers, 2010, 27(4): 1330-1334.)

[7] Alatas B. Chaotic bee colony algorithms for global numerical optimization[J]. Expert Systems with Applications, 2010, 37(8): 5682-5687.

[8] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Piscataway, 1995: 1942-1948.

[9] 陶新民, 刘福荣, 刘玉, 等. 一种多尺度协同变异的粒子群优化算法[J]. 软件学报, 2012, 23(7): 1805-1815.
(Tao X M, Liu F R, Liu Y, et al. Multi-scale cooperative mutation particle swarm optimization algorithm[J]. J of Software, 2012, 23(7): 1805-1815.)

[10] 吴晓军, 杨战中, 赵明. 均匀搜索粒子群算法[J]. 电子学报, 2011, 39(6): 1261-1266.
(Wu X J, Yang Z Z, Zhao M. A uniform searching particle swarm optimization algorithm[J]. Acta Electronica Sinica, 2011, 39(6): 1261-1266.)

[11] 周新宇, 吴志健, 王晖, 等. 一种精英反向学习的粒子群优化算法[J]. 电子学报, 2013, 41(8): 1647-1652.
(Zhou X Y, Wu Z J, Wang H, et al. Elite opposition-based particle swarm optimization[J]. Acta Electronica Sinica, 2013, 41(8): 1647-1652.)

[12] Jia D L, Zheng G X, Qu B Y, et al. A hybrid particle swarm optimization algorithm for high-dimensional problems[J]. Computers and Industrial Engineering, 2011, 61(4): 1117-1122.

[13] Shi X H, Li Y W, Li H J, et al. An integrated algorithm based on artificial bee colony and particle swarm optimization[C]. Int Conf on Natural Computation. Yantai: IEEE Press, 2010: 2586-2590.

[14] El-Abd M. A hybrid ABC-SPSO algorithm for continuous function optimization[C]. IEEE Symposium on Swarm Intelligence(SIS). Paris: IEEE Press, 2011: 1-6.

[15] 单梁, 强浩, 李军, 等. 基于Tent映射的混沌优化算法[J]. 控制与决策, 2005, 20(2): 179-182.
(Shan L, Qiang H, Li J, et al. Chaotic optimization algorithm based on Tent map[J]. Control and Decision, 2005, 20(2): 179-182.)

[16] Bao L, Zeng J C. Comparison and analysis of the selection mechanism in the artificial bee colony algorithm[C]. Int Conf on Hybrid Intelligent Systems. Shenyang: IEEE Press, 2009: 411-416.

[17] Gao W F, Liu S Y. Improved artificial bee colony for global optimization[J]. Information Processing Letters, 2011, 111(17): 871-882.

[18] Yan X H, Zhu Y L, Zou W P, et al. A new approach for data clustering using hybrid artificial bee colony algorithm[J]. Neurocomputing, 2012, 97(11): 241-250.

[19] 刘三阳, 张平, 朱明敏. 基于局部搜索的人工蜂群算法[J]. 控制与决策, 2014, 29(1): 123-128.
(Liu S Y, Zhang P, Zhu M M. Artificial bee colony algorithm based on local search[J]. Control and Decision, 2014, 29(1): 123-128.)

(责任编辑: 李君玲)