Computer Science and Application 计算机科学与应用, 2017, 7(2), 134-145
Published Online February 2017 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2017.72017

Hans汉斯

An Improved Artificial Bee Colony Algorithm Based on Hooke-Jeeves Method

Yuehong Sun[1,2], Yaying Ding[2]

[1]Jiangsu Provincial Key Laboratory for Numerical Simulation Large Scale Complex Systems, Nanjing Jiangsu
[2]School of Mathematical Sciences, Nanjing Normal University, Nanjing Jiangsu
Email: 05234@njnu.edu.cn, yinhanwu@yeah.net

Received: Feb. 4th, 2017; accepted: Feb. 25th, 2017; published: Feb. 28th, 2017

Abstract

Artificial bee colony algorithm (ABC) is a relatively new swarm intelligence optimization method, which is superior to other population-based intelligent algorithms. However, ABC algorithm also has certain limitation because its updating formula is not good at exploitation. In order to enhance the exploitation capacity of ABC, this paper presents a new algorithm called an improved artificial bee colony algorithm based on Hooke-Jeeves method (IHABC). The altered formulas of employed bees and onlooker bees in IHABC not only keep exploration ability but also increase exploitation to a great extent. In addition, the algorithm optimizes initial base point selection in Hooke-Jeeves search phase by upper-middle individual and modifies step size formula of exploratory move, so that the whole population evolves spontaneously in the right direction. To test the effectiveness of the proposed algorithm, we compare IHABC with ABC and Hooke-Jeeves artificial bee colony algorithm (HABC). The numerical experimental results of 30 benchmark functions clearly indicate that IHABC gets higher approximate solution precision in solving unconstrained optimization problems.

Keywords

Swarm Intelligence, Artificial Bee Colony Algorithm, Hooke-Jeeves Search, Local Exploitation Capacity, Unconstrained Optimization Problems

基于Hooke-Jeeves的改进人工蜂群算法

孙越泓[1,2]，丁亚英[2]

[1]江苏省大规模复杂系统数值模拟重点实验室，江苏 南京
[2]南京师范大学数学科学学院，江苏 南京
Email: 05234@njnu.edu.cn, yinhanwu@yeah.net

收稿日期：2017年2月4日；录用日期：2017年2月25日；发布日期：2017年2月28日

文章引用：孙越泓，丁亚英. 基于 Hooke-Jeeves 的改进人工蜂群算法[J]. 计算机科学与应用, 2017, 7(2): 134-145.
https://doi.org/10.12677/csa.2017.72017

摘　要

人工蜂群算法(ABC)是新近提出的一种基于群智能的优化方法，它比其他基于种群的智能算法更优异，但蜂群的搜索更新公式在算法的局部寻优能力上存在缺陷。因此本文致力于将擅长局部寻优的搜索机制引入人工蜂群算法，提出了一种基于Hooke-Jeeves的改进人工蜂群算法(IHABC)。IHABC算法改进了采蜜蜂和跟随蜂的搜索公式，期望保留全局搜索能力的同时能更大限度地增加算法的局部寻优能力；采用质量中上的个体优化Hooke-Jeeves搜索法的初始基点以确保局部寻优的有效性；改进Hooke-Jeeves方法的探索移动的步长公式。为了检测新算法的性能，将其与人工蜂群算法、Hooke-Jeeves人工蜂群算法(HABC)进行比较分析，30个基准函数上的数值实验结果表明，IHABC算法在求解无约束优化问题时得到的近似解有更高的精度。

关键词

群智能，人工蜂群算法，Hooke-Jeeves搜索，局部寻优能力，无约束优化问题

1. 引言

人工蜂群算法(Artificial Bee Colony algorithm, ABC) [1] [2]属于智能优化方法之一，由 Karaboga 和 Basturk 受蜜蜂觅食行为启发而提出。它以结构简单、应用范围广、表现优异等特点，受到优化领域学者们的关注。但是 Zhu 和 Kwong [3]指出 ABC 的搜索公式精于全局搜索，疏于局部寻优。为了改善 ABC 算法的性能，memetic 算法[4]将 ABC 算法与局部寻优方法相结合，充分利用两种算法的优势，即将 ABC 算法的全局搜索能力贡献给 memetic 算法，使得它比单用局部寻优方法更不容易陷入局部最优；同时将局部寻优的能力贡献给 ABC 算法，使得它比单用全局搜索算法能更快地收敛。Bansal 等[5]在 ABC 算法中加入黄金分割搜索(Golden Section Search, GSS)，提出 MeABC 算法。Gao 等[6]将 Powell 方法与 ABC 结合提出 PABC 算法，充分利用 ABC 的全局搜索能力，将任意选出的个体往当前种群的适应度值最大的个体方向进行略微调整，在调整后的新个体上用 Powell 方法进行局部寻优。

2011 年 Kang 等[7]将 Rosenbrock 转轴法与原始 ABC 结合，得到了 RABC 算法。Kang 等[8] 2013 年进一步尝试将 Hooke-Jeeves 直接搜索方法与人工蜂群算法 ABC 结合，命名为 HABC，其全局搜索阶段与 ABC 类似，仅将适应度计算公式改为公式(7)，并对当前种群中目标函数值最小的个体用 Hooke-Jeeves 方法进行局部寻优。

本文在 HABC 基础上，对采蜜蜂和跟随蜂的候选解产生公式做了改动，期望保留全局搜索能力的同时能更大程度地增加算法的局部寻优能力，从而加快算法的收敛速度。另外，对 Hooke-Jeeves 方法的初始基点进行优化，提高局部寻优的有效性并避免其陷入局部最优。数值实验表明提出的新算法在求解无约束优化问题时具有一定的优越性。

2. 人工蜂群算法和 Hooke-Jeeves 方法

为了便于描述新算法，首先对人工蜂群算法和改进的 Hooke-Jeeves 方法进行简要说明。

2.1. 人工蜂群算法

人工蜂群算法用于求解无约束优化问题

$$\min f(x) \tag{1}$$

其中自变量 $x=(x_1,x_2,\cdots,x_n)\in R^n$，$n$ 指维数。目标函数 $f:R^n\to R$。具体步骤如下：

步骤一：初始化。

初始化种群大小 NP，食物源数目 $NS(NS=NP/2)$，采蜜蜂数目 NS。接着随机产生 NS 个 n 维实向量(n 指优化问题的维数)，令 $x_i=(x_{i1},x_{i2},\cdots,x_{in})$ 表示产生的第 i 个向量，每维分量的生成公式如下：

$$x_{ij}=x_{\min,j}+\text{rand}(0,1)(x_{\max,j}-x_{\min,j}) \tag{2}$$

式中 $i=1,2,\cdots,NS$，$j=1,2,\cdots,n$，$\text{rand}(0,1)$ 是 $(0,1)$ 之间服从均匀分布的随机数，$x_{\max,j}$ 和 $x_{\min,j}$ 表示第 j 维的上下界。

将这些向量随机分配给 NS 只采蜜蜂，用公式(3)计算每只采蜜蜂对应食物源的适应度值，同时用变量 X_{best} 记录适应度值最大的食物源。

$$\text{fit}_i=\begin{cases}\dfrac{1}{1+f_i}, f_i\geq 0\\ 1+|f_i|, f_i<0\end{cases} \tag{3}$$

步骤二：重复地对食物源进行更新，直到终止条件满足。

① 采蜜蜂阶段

基于旧的食物源 x_i 产生一个候选解 v_i，公式如下：

$$v_{ij}=x_{ij}+\varphi_{ij}(x_{ij}-x_{kj}) \tag{4}$$

式中 $i=1,2,\cdots,NS$；k 是从集合 $\{1,2,\cdots,NS\}$ 中随机选择的一个整数，并且 k 必须不同于 i；j 是从集合 $\{1,2,\cdots,n\}$ 中随机选择的一个整数；φ_{ij} 是服从[−1,1]之间均匀分布的随机数。

计算并比较候选解 v_i 和食物源 x_i 两者的适应度值，用贪婪算子在 v_i 和 x_i 间做选择：如果 v_i 的适应度值不小于 x_i 的适应度值，那么用 v_i 替代 x_i；否则 x_i 保留。

② 跟随蜂阶段

跟随蜂根据每个食物源 x_i 的被选择概率 p_i，利用轮盘赌算子决定 x_i 是否被选择进行更新，即只有当随机数 $r_i<p_i$ 时，才用公式(4)产生 x_i 的一个候选食物源 v_i，计算并比较候选解和旧食物源的适应度值，用贪婪算子在两者之间做取舍。其中每个食物源的被选择概率 p_i 的公式如下：

$$p_i=\frac{\text{fit}_i}{\sum_{j=1}^{NS}\text{fit}_j},\quad i=1,2,\cdots,NS \tag{5}$$

③ 侦查蜂阶段

如果有食物源 x_i 在 Limit 循环次数之内一直没有被更新，那么 x_i 就会被抛弃，用公式(2)产生一个新的食物源来代替 x_i。Limit 是预先给定的一个值。

④ 更新 *gbest*

找出当前种群中的最大适应度值，与原来的 *Fitbest* 比较，适应度值更高的作为新的 *Fitbest*，同时将

Fitbest 对应的食物源存入 x_{best} 中。

步骤三：输出最终的 x_{best} 作为优化问题的近似解。

2.2. Hooke-Jeeves 方法

Hooke-Jeeves 方法包含两种类型的移动：探测移动(exploratory move)和模式移动(pattern move)。在确定了一个初始基点之后，探测移动依次沿 n 个坐标轴进行，用以确定新的基点和有利于函数数值下降的方向；模式移动沿相邻两个基点连线方向进行，试图顺着“山谷”使函数值更快地减小。这两种移动交替地进行直到终止条件满足，如图 1 所示，带箭头的实线为探测移动，不带箭头的实线为模式移动。

探测移动的主要步骤描述在算法 1 中(见图 2)。假定 x_0 是基点，$\delta=(\delta_1,\delta_2,\cdots,\delta_n)$ 是 n 个方向上各自的步长，$f_{\min}$ 是当前目标函数的最小值。x_1 是过渡向量，用于存储探测移动后得到的点。如果探测移动成功，即两个解 x_0 和 x_1 满足关系 $f(x_0)<f(x_1)$，那么从 x_0 处以 (x_1-x_0) 为方向进行模式移动，模式移动后得到的点记为 $x_2=x_1+(x_1-x_0)$。Hooke-Jeeves 方法的主要步骤见算法 2 (图 3)。算法 2 中引入了辅助步长 s_a 用于终止算法；步长缩减率 $\rho=0.7$。

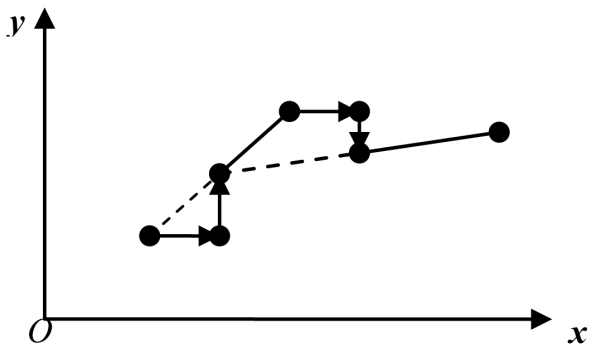

Figure 1. An example of Hooke-Jeeves method
图 1. Hooke-Jeeves 方法的一个例子

算法 1　探测移动

1. 初始化：$x_1=x_0$，$f_{\min}=f(x_0)$
2. for i= 1 : n
$x_1(i)=x_0(i)+\delta_i$; %进行第一次探测移动
if $f(x_1)<f_{\min}$　%第一次探测移动成功
$f_{\min}=f(x_1)$;
else
$x_1(i)=x_0(i)-\delta_i$;　%第一次探测移动失败，进行第二次探测移动
if $f(x_1)<f_{\min}$　%第二次探测移动成功
$f_{\min}=f(x_1)$;
else
$x_1(i)=x_0(i)$;　%第二次探测移动失败
end
end
end
3. 如果 $f_{min}<f(x_0)$，那么探测移动成功；否则失败。输出 x_1。

Figure 2. The main steps of exploratory move
图 2. 探测移动主要步骤

算法 2　Hooke-Jeeves 方法

步骤 1：初始化基点 x_0，步长 $\delta_i\left(i=1,2,\cdots,n\right)$，步长缩减率 $\rho<1$，终止参数 $\varepsilon_s>0$，迭代次数 $k=1$，辅助步长 $s_a=1.0$。

步骤 2：以 x_0 为基点进行探测移动(算法 1)得到点 x_1。如果探测移动成功，转步骤 3；否则，转步骤 7。

步骤 3：对于 $i=1,2,\cdots,n$，如果 $x_{1i}<x_{0i}$，$\delta_i=-\left|\delta_i\right|$；否则，$\delta_i=\left|\delta_i\right|$。

步骤 4：进行模式移动 $x_2=x_1+\left(x_1-x_0\right)$，值传递 $x_0=x_1$。

步骤 5：以 x_2 为基点进行探测移动(算法 1)得到点 x_1。

步骤 6：如果 $f\left(x_1\right)<f\left(x_0\right)$，转步骤 3；否则，转步骤 7。

步骤 7：如果 $s_a<\varepsilon_s$，终止；否则，迭代次数 $k=k+1$，$s_a=s_a\times\rho$，$\delta_i=\delta_i\times\rho\ \left(i=1,2,\cdots,n\right)$，转步骤 2。

Figure 3. The main steps of Hooke-Jeeves Method
图 3. Hooke-Jeeves 方法主要步骤

3. 基于 Hooke-Jeeves 的改进人工蜂群算法

为了加快基于Hooke-Jeeves的人工蜂群算法HABC [8]的收敛速度，并且防止算法陷入局部最优，我们提出基于 Hooke-Jeeves 的改进人工蜂群算法。本节首先描述三个改进策略，随后对新算法进行具体说明。

3.1. 采蜜蜂更新公式

在采蜜蜂阶段，根据文献[9]将候选解的产生公式改为

$$v_{ij}=x_{\text{best},j}+\varphi_{i,j}\left(x_{r1,j}-x_{r2,j}\right) \tag{6}$$

式中，指标 r_1 和 r_2 是从 $\{1,2,\cdots,NS\}$ 里随机选的两个不同的整数，且均不同于 i；x_{best} 是当前种群中最好的个体(指目标函数最小的个体)；j 是从 $\{1,2,\cdots,n\}$ 里随机取的一个整数；φ_{ij} 是[−1,1]之间服从均匀分布的随机数。原始的候选解公式(4)精于全局搜索、疏于局部寻优，导致算法的收敛速度减慢。而公式(6)是在前一次迭代后得到的最好个体附近产生候选解，这样充分利用了最好个体的信息，并且大大增加了算法的局部寻优能力，从而提升了收敛速度。

另外蜂群个体的适应度值计算公式如下：

$$\text{fit}_i=2-SP+\frac{2\left(SP-1\right)\left(r_i-1\right)}{NS-1} \tag{7}$$

式中，r_i 是指函数值按从大到小排序后第 i 个解在整个种群中的顺序号；SP 是选择压力，取值为[1.0, 2.0]，实验中取了中间值 1.5；NS 是食物源数目。

3.2. 跟随蜂更新公式

在跟随蜂阶段，根据文献[10]将候选解的产生公式改为

$$v_{id}=x_{\text{best}\,j}+\varphi_d*f_{\text{best}}*\left(x_{ij}-x_{\text{best}\,j}\right) \tag{8}$$

式中，x_i 是第 i 个食物源；v_i 是第 i 个食物源的候选解，d 是从 $\{1,2,\cdots,n\}$ 里依次取的整数；x_{best} 是当前种群中的最好个体，f_{best} 是将 x_{best} 带入公式(3)得到的值；φ_d 是[−1,1]之间服从均匀分布的随机数；j 是从 $\{1,2,\cdots,n\}$ 里随机取的一个整数。公式(8)将 f_{best} 作为候选解产生公式的误差校正项乘子，根据参考文献[10]所述，此举能够加快算法的收敛速度。

3.3. 局部寻优初始基点的选择

在 Hooke-Jeeves 方法中，每 Nc 次循环后以最好个体 x_{best} 为基点进行一次 Hooke-Jeeves 搜索，如果此时 x_{best} 为靠近局部最优解的点，那算法很容易就陷入了局部最优。PABC [6]算法是采用从任意解往最好解方向移动后的新个体上进行搜索，虽然能一定程度上避免陷入局部最优，但是对于任取的个体，即使往最好个体方向上移动之后仍无法确定会不会很偏离最优值。

本文对 Hooke-Jeeves 方法进行的局部搜索做了改动，一个很自然的想法就是折中处理，即取目标函数值处于中间的个体，再往最好个体方向稍做移动得到新的个体 U，以 U 为基点进行 Hooke-Jeeves 搜索。U 的生成公式如下：

$$U_j = x_{\text{media},j} + \varphi_j \left(x_{\text{best},j} - x_{\text{media},j} \right) \tag{9}$$

式中，$j = 1,2,3,\cdots,n$；x_{media} 表示目标函数值位于中间的个体；x_{best} 表示目标函数值最小的个体；φ_j 是[0,1]之间服从均匀分布的随机数。本文还将探测移动的步长公式更改为：

$$\delta_j = 0.1 \times \frac{\sum_{i=1}^{m} \left(x'_{ij} - U_j \right)}{m} \tag{10}$$

式中，δ_j 指第 j 维的探测移动步长值；$m = NP \times 10\%$ 是用于计算步长的个体数；x'_i 是按目标函数值从大到小排序后序号为 i 的个体；U_j 是用公式(9)得到的新个体。该步长公式取的是前 m 个最好解与新解 U 之间的平均距离，也就是以 U 为基点，朝着前 m 个最好解的方向进行搜索。迭代初期，由于种群的多样性，δ_j 会偏大；随着种群渐渐收敛，种群间的差距会渐渐缩小，从而步长也会随之减小。

U 与 δ 的选取使得 Hooke-Jeeves 搜索在一个相对不错的个体上沿着指向最好解的方向进行，既避免了在较差解上 Hooke-Jeeves 搜索的浪费，又避免了在最好解上 Hooke-Jeeves 搜索的停滞。Hooke-Jeeves 搜索方法的终止参数是 ε_s，当辅助步长 $s_a < \varepsilon_s$ 时，算法会自动跳出局部搜索循环。选取 Hooke-Jeeves 搜索得到的个体与 x_{media} 中函数值较小的那个个体，从整体上把握种群的发展方向。

3.4. 改进算法

我们提出的新算法称为基于 Hooke-Jeeves 的改进人工蜂群算法，简记为 IHABC。算法的大体框架同 HABC 一致，主要有两个阶段：

第一阶段是全局搜索阶段，用人工蜂群算法 ABC 搜索较优质的解，采蜜蜂阶段改用公式(6)，跟随蜂阶段改用公式(8)，其它地方与 ABC 算法相同。

第二阶段是局部寻优阶段，每 N_C 次循环调用一次 Hooke-Jeeves 方法对公式(9)得到的新解 U 进行局部寻优，对应的步长用公式(10)计算。接着进行如 1.2 节所述的两种移动：探测移动和模式移动，探测移动一次只考虑一个分量用以确定合适的寻优方向，而下一步的模式移动是为了在探测移动得到的方向上进行加速寻优。这两种移动交替进行不断重复直到开采阶段的终止条件满足，得到的结果记为 U'。比较 U' 与 x_{media} 两者的目标函数值，目标函数值更小者替代原来的 x_{media}。

搜索阶段和开采阶段重复进行直到终止条件满足，终止条件可以是最大函数计算次数、最大循环次数或者与理论最优值之间的差异很小等。为了表述更清晰，图 4 给出了 IHABC 算法(算法 3)的整个过程。

4. 数值实验

本实验是在 Windows XP 系统，Intel Core13(2.4GHz CPU/2G)的环境下运行的，使用 Matlab 2012b 软件编写的程序。本节将 ABC 算法、HABC 算法以及 IHABC 算法在 30 个基准函数[11]上做了测试，比较了三个算法独立运行 50 次得到的基准函数最小值的平均值和标准差。

算法 3：IHABC 算法

1. 初始化种群：用公式(2)产生 $x_i, i=1,2,\cdots,NS$，迭代次数 $iter = 1$。
2. 计算每个个体的目标函数值，记录目标函数值最小的个体为 x_{best}。
3. Repeat

(全局搜索阶段)

采蜜蜂阶段：用公式(6)产生每个个体 x_i 的候选解 v_i，若 v_i 的目标函数值小于 x_i 的目标函数值，则替换 x_i。

对种群中所有个体按目标函数值从大到小排序，序号为 r；

用公式(7)计算适应度值 fit_i，用公式(5)计算每个 x_i 的被选择概率 p_i。

跟随蜂阶段：根据 p_i 用轮盘赌选出解 x_i，用公式(8)产生候选解 v_i，若 v_i 的目标函数值小于 x_i 的目标函数值，则替换 x_i。

侦查蜂阶段：如果有需要放弃的解 x_i，用公式(2)产生新解来取代 x_i。

记录到目前为止的最好解 x_{best}（目标函数值最小的个体）。

(局部寻优阶段)

If 迭代次数 $iter$ 是 N_C 的倍数

按目标函数值对种群进行从大到小排序，位于中间位置的个体记为 x_{media}，用公式(9)计算初始基点 U，用公式(10)计算局部寻优步长 δ。以 U 为基点，δ 为初始步长进行 Hooke-Jeeves 搜索直到 $s_a < \varepsilon_s$，得到的点记为 U'。若 U' 的目标函数值小于 x_{media} 的目标函数值，则替换 x_{media}。

End

$iter = iter + 1$

Until 终止条件满足

4. 输出最好解 x_{best}（目前为止目标函数值最小的个体）。

Figure 4. The main steps of IHABC
图 4. IHABC 主要步骤

4.1. 基准函数和参数设置

基准函数是从检验全局优化算法表现的函数库中选出的有代表性的函数，涵盖了不同的维数、不同的特点(单峰 *U*/多峰 *M*、可分 *S*/不可分 *N*)，基本信息见表 1，具体的表达式详见参考文献[7]和[11]。由于多峰函数和不可分函数能较好地检测智能优化算法的优劣，我们选择了 16 个多峰不可分函数，10 个单峰不可分函数，1 个多峰可分函数，2 个单峰可分函数。

为了便于比较，三个算法中 ABC 阶段的参数设置均取相同的值：种群数 $NP = 50$；食物源数目 $NS = NP/2 = 25$；$limit = NS \times n$，n 指优化问题的维数；最大函数计算次数 300000。HABC 中参数设置见文献[8]，IHABC 中其它参数设置如下：$\text{SP} = 1.5$，$N_C = 5 \times n$，$m = NS \times 10\%$，$\rho = 0.7$，$s_a = 1.0$，$\varepsilon_s = 1e-3$。

4.2. 实验结果

每个算法在基准函数 C01-C30 上独立运行 50 次得到函数最小值的平均值和标准差总结在表 2 中，较好的实验结果用黑色粗体标出。

在单峰可分 Sphere (SP)函数 C27 上，新算法 IHABC 算法在 50 次实验中均取得了全局最小值 0，标准差也为零，求解精度和算法的稳定性都超过了 ABC 和 HABC 算法；在单峰可分离 Quartic (QU)函数

Table 1. The information of benchmark functions
表 1. 函数基本信息

Fun	Name	Dim	Search range	Min	feature
C01	Beale (BE)	2	[−4.5,4.5]	0	UN
C02	Goldstein and Price (GP)	2	[−2,2]	3	MN
C03	Matyas(MA)	2	[−10,10]	0	UN
C04	Shekel's Foxholes (SF)	2	[−65.536,65.536]	0.998004	MN
C05	Shubert (SH)	2	[−10,10]	−186.7309	MN
C06	Hartman 3 $\left(H_{3,4}\right)$	3	[0,1]	−3.86278	MN
C07	Helical valley problem (HV)	3	[−10,10]	0	UN
C08	Colville (CO)	4	[−10,10]	0	UN
C09	Perm (PE)	4	[−4,4]	0	MN
C10	Power Sum (PS)	4	[0,4]	0	MN
C11	Shekel's Problem Family $\left(S_{4,10}\right)$	4	[0,10]	−10.53641	MN
C12	Fletcher-Powell (FP)	2	$[-\pi,\pi]$	0	MN
C13	Modified langerman (ML)	5	[0,10]	−0.965	MN
C14	Modified langerman (ML)	2	[0,10]	−1.08094	MN
C15	Hartman 6 $\left(H_{6,4}\right)$	6	[0,1]	−3.32237	MN
C16	Modified langerman (ML)	10	[0,10]	−0.965	MN
C17	Michalewicz (MI)	5	$[0,\pi]$	−4.687658	MS
C18	Whitley (WI)	10	[−100,100]	0	MN
C19	Powell (PO)	24	[−4,5]	0	UN
C20	Quartic function (QU)	30	[−1.28,1.28]	0	US
C21	Schwefel's Problem 2.21 (S21)	30	[−100,100]	0	UN
C22	Weierstrass (WE)	60	[−0.5,0.5]	0	MN
C23	Ackley (AC)	30	[−32,32]	0	MN
C24	Griewank (GR)	30	[−600,600]	0	MN
C25	Rosenbrock (RO)	30	[−30,30]	0	UN
C26	Schwefel's Ridge or Schwefel's problem 1.2 (SR)	30	[−100,100]	0	UN
C27	Sphere (SP)	30	[−100,100]	0	US
C28	Schwefel's problem 2.22 (S22)	30	[−10,10]	0	UN
C29	Zakharov (ZA)	30	[−5,10]	0	UN
C30	Ackley (AC)	100	[−32,32]	0	MN

Table 2. The results of ABC, HABC and IHABC on function C01-C30
表 2. ABC、HABC 和 IHABC 算法在函数 C01-C30 上的结果

Fun	Dim		ABC	HABC	IHABC
C01	2	**MeanVal** **StdVal**	5.02262e−13 2.33684e−12	3.31089e−17 8.56149e−17	**0** **0**
C02	2	**MeanVal** **StdVal**	3.00001 4.50637e−05	**3** **2.54873e−15**	**3** 2.29381e−14
C03	2	**MeanVal** **StdVal**	1.88342e−14 9.19506e−14	1.85964e−16 9.49337e−16	**0** **0**
C04	2	**MeanVal** **StdVal**	**0.998004** **0**	**0.998004** 1.05206e−16	**0.998004** 1.18688e−16
C05	2	**MeanVal** **StdVal**	**−186.731** **1.05556e−14**	**−186.731** 1.46394e−14	**−186.731** 3.2482e−14
C06	3	**MeanVal** **StdVal**	**−3.86278** **2.40284e−15**	**−3.86278** 3.09825e−15	**−3.86278** 4.90882e−15
C07	3	**MeanVal** **StdVal**	0.002556 0.00243314	1.33466e−07 4.87096e−07	**1.08073e−13** **2.87356e−13**
C08	4	**MeanVal** **StdVal**	0.169419 0.118691	1.4811e−06 3.50147e−06	**0** **0**
C09	4	**MeanVal** **StdVal**	0.0417135 0.0395745	0.0050728 0.018772	**8.2582e−06** **1.26267e−05**
C10	4	**MeanVal** **StdVal**	0.00565621 0.00456532	0.000670047 0.00139635	**4.95624e−07** **6.84974e−07**
C11	4	**MeanVal** **StdVal**	**−10.5364** 9.32393e−15	**−10.5364** **5.663e−15**	**−10.5364** 1.31248e−14
C12	2	**MeanVal** **StdVal**	2.54012e−19 2.20575e−19	**0** **0**	**0** **0**
C13	5	**MeanVal** **StdVal**	−0.964908 4.69291e−04	**−0.965** **3.22144e−13**	−0.950218 0.0185058
C14	2	**MeanVal** **StdVal**	**−1.08094** 1.64121e−06	**−1.08094** 1.56413e−15	**−1.08094** **1.34579e−15**
C15	6	**MeanVal** **StdVal**	**−3.322** 1.35504e−15	**−3.322** 2.2349e−16	**−3.322** **8.97196e−17**
C16	10	**MeanVal** **StdVal**	−0.533924 0.084979	−0.569837 0.143236	**−0.6077** 0.221634
C17	5	**MeanVal** **StdVal**	**−4.68766** 2.75367e−15	**−4.68766** 2.75367e−15	**−4.68766** **2.54399e−15**
C18	10	**MeanVal** **StdVal**	0.374182 1.07672	**0.0394587** **0.19527**	0.138105 0.345766
C19	24	**MeanVal** **StdVal**	0.00096315 0.00012072	6.18214e−08 9.73955e−08	**2.29692e−64** **1.62417e−63**
C20	30	**MeanVal** **StdVal**	0.0370203 0.00797128	0.0351516 0.00822433	**1.76447e−05** **1.42544e−05**
C21	30	**MeanVal** **StdVal**	0.0972994 0.0398698	3.39613e−05 1.48887e−05	**7.06268e−29** **4.9921e−28**
C22	60	**MeanVal** **StdVal**	1.42109e−14 8.3449e−15	1.33582e−14 8.80708e−15	**0** **0**
C23	30	**MeanVal** **StdVal**	3.49942e−14 4.8119e−15	3.48166e−14 4.76106e−15	**0** **0**

Continued

C24	30	**MeanVal** **StdVal**	1.1168e−12 6.1165e−12	**0** **0**	**0** **0**
C25	30	**MeanVal** **StdVal**	0.213231 0.472143	5.04318e−06 1.44344e−05	**0** **0**
C26	30	**MeanVal** **StdVal**	2180.6 1076.14	1.85287e−08 1.31017e−07	**0** **0**
C27	30	**MeanVal** **StdVal**	4.64308e−16 6.48251e−17	6.13319e−97 1.48311e−96	**0** **0**
C28	30	**MeanVal** **StdVal**	1.16608e−15 1.40688e−16	1.3805e−35 9.76164e−35	**0** **0**
C29	30	**MeanVal** **StdVal**	172.588 30.6762	1.81761e−15 4.69437e−15	**0** **0**
C30	100	**MeanVal** **StdVal**	2.39054e−11 9.57251e−12	1.79625e−13 2.66457e−14	**0** **0**

C20 上，ABC 和 HABC 的性能相当，其平均最优值为 0.0370203 和 0.0351516，标准差为 0.00797128 和 0.00822433，而 IHABC 的平均最优值为 1.76447e−05，标准差为 1.42544e−05，比 ABC 和 HABC 算法提高了 3 个数量级。多峰可分 Michalewicz (MI)函数 C17，取参数 $m = 10$ 和维数 $n = 5$ 时，它有 5!个局部最优点，三种算法的平均最优值均达到全局最小值−4.68766，HABC 和 ABC 算法的标准差均为 2.75367e−15，IHABC 的标准差为 2.54399e−15，IHABC 算法标准差最小。

对于单峰不可分 Beale (BE)函数 C01、Matyas (MA)函数 C03、Colville (CO)函数 C08、Rosenbrock (RO)函数 C25、Schwefel's Ridge (SR)函数 C26、Schwefel's problem 2.22 (S22)函数 C28 和 Zakharov (ZA)函数 C29，均无局部最小值，只有一个全局最小值，IHABC 算法在 50 次试验中均找到了全局最小值 0，标准差是 0，寻优性能高于 ABC 和 HABC 算法。其中 Colville 函数在唯一最小值点 $x^* = (1,1,1,1)$ 附近有一个鞍点，所以 HABC 算法的平均最优值为 1.4811e−06，而 ABC 算法在 50 次试验中未能找到全局最小值，其平均最优函数值为 0.169416，最好的一次寻优结果为 $f(\tilde{x}) = 0.0148$，其中 $\tilde{x} = (1.0756, 1.1641, 0.8801, 0.7787)$；Rosenbrock 函数是单模态的高维函数，变量间相互关联，其内部是一个长而狭窄，形如抛物线的平坦山谷地带，最小值点 $x^* = (1,1,\cdots,1)$ 位于其中，ABC 算法很难收敛于全局最优值 $f(x^*) = 0$，ABC 和 HABC 算法的 50 次实验的平均最优值分别为 0.213231 和 5.04318e−06，ABC 算法的求解精度比 IHABC 算法低。

对于单峰不可分函数 Helical valley problem (HV) C07、Powell (PO)函数 C19 和 Schwefel's Problem 2.21 (S_{21})函数 C21，多峰不可分函数 Perm (PE) C09 和 Power Sum (PS) C10，IHABC 算法的寻优精度和稳定性要优于 HABC 和 ABC 算法，HABC 算法则优于 ABC 算法。

对于多峰不可分函数 Shekel's Foxholes (SF)函数 C04、Shubert (SH)函数 C05、Hartman 3 ($H_{3,4}$)函数 C06，Shekel's Problem Family ($S_{4,10}$)函数 C11 和 Hartman 6 ($H_{6,4}$)函数 C15，三种算法的平均最优值均能达到全局最小值，分别为 0.998004、−186.731、−3.86278、−10.5364 和−3.322；Goldstein and Price (GP)函数 C02、Fletcher-Powell (FP)函数 C12 和 Griewank (GR)函数 C24，IHABC 算法和 HABC 算法性能相当，平均最优值都取到了全局最小值，寻优性能和稳定性都超过了 ABC 算法；对于 Weierstrass (WE)函数 C22、Ackley (AC)函数 C23 ($n = 30$ 维)和 Ackley (AC)函数 C30 ($n = 100$ 维)，HABC 算法和 ABC 算法性能相当，IHABC 算法优于 IHABC 和 ABC，能够得到全局最小值 0，标准差也为 0，对于 Ackley 函数，30 维和 100 维时 IHABC 算法的寻优结果一样，表明了该算法的稳健性。

对于多峰不可分函数 Whitley (WI) C18，维数 $n = 10$ 时，全局最小值点为 $x^* = (1,1,\cdots,1)$，全局最小

值为 $f\left(x^*\right)=0$，ABC 算法在 50 次实验中均找不到全局最小值；IHABC 算法有 43 次找到全局最小值 0，有 7 次找到局部最小值 0.9865；HABC 算法有 47 次找到全局最小值 0，有 2 次寻优得到 0.9865，还有 1 次找到的最优值为 6.6613e−16，HABC 在函数 C18 上寻优精度最高。

Modified Langerman 函数 C13，C14，C16 是非对称的复杂多模态函数，其局部最优解个数未知且随机分布，随着维数的增加，搜索难度增大。当维数 n = 2 时，三种算法在 50 次实验中均取得了全局最小值−1.08094，IHABC 算法的标准差最小，算法最稳定；当维数 n = 5 时，50 次实验中 ABC、HABC、IHABC 算法的平均最优值分别为−0.964959，−0.963356 和−0.954508，标准差分别为 0.000142，0.008745 和 0.016397，50 次实验中三种算法分别 42 次、48 次和 33 次得到全局最小值−0.965，IHABC 算法 10 次得到局部最小值−0.9398，ABC 算法在 n = 5 时，性能最好；当维数 n = 10 时，ABC、HABC、IHABC 算法的平均最优值分别为−0.533924、−0.569837 和−0.6077，标准差分别为 0.084979、0.143236 和 0.221634，50 次实验中，ABC、HABC 和 IHABC 算法分别有 0、3 和 4 次达到全局最小值−0.9605，分别有 47、43 和 24 次寻优得到局部最小值−0.5132，意味着该函数在多个点处取得局部极小值−0.5132，ABC 算法另外三次的寻优结果为−0.8057，−0.8059 和−0.9649，HABC 算法还得到局部最优值−0.9080 三次，−0.8060 一次，IHABC 算法得到局部最优值−0.2749 四次，−0.4286 三次，−0.4829 和−0.9645 各两次，−0.5164、−0.5168、−0.5170、−0.8019、−0.80306、−0.9585、−0.9614、−0.9637、−0.9640、−0.9641 和−0.9644 各一次。与 ABC、HABC 算法相比，IHABC 算法的通用性和局部寻优能力更强，在得到全局最小值的同时，能搜索到更多的局部最优值。

总体来说，新算法 IHABC 的性能在 23 个函数上优于 HABC 算法，在 5 个函数上与 HABC 相当，在 2 个函数上略逊于 HABC 算法。实验结果证明了 IHABC 算法比 HABC 以及 ABC 算法在寻优精度上有了相当程度的提高。

5. 结论

基于 Hooke-Jeeves 的人工蜂群算法 HABC，其全局搜索能力与局部寻优能力不均衡，局部寻优阶段初始解的选择易导致算法陷入局部最优，为解决这些问题本文提出基于 Hooke-Jeeves 的改进人工蜂群算法 IHABC。新算法通过三个改进策略分别对蜂群的采蜜蜂更新公式、跟随蜂更新公式和局部寻优阶段的初始基点的选择以及局部寻优的步长进行了调整，最后将算法在 30 个测试函数上进行了数值实验。实验结果表明，新算法与 HABC 和 ABC 算法相比，在大部分测试函数上得到的全局最优解的精度更高，算法更稳定。

基金项目

教育部人文社会科学青年基金项目(12YJCZH179)，江苏省大规模复杂系统数值模拟重点实验室开放基金项目(201601)。

参考文献 (References)

[1] Karaboga, D. and Basturk, B. (2008) On the Performance of Artificial Bee Colony (ABC) Algorithm. *Applied Soft Computing*, **8**, 687-697. https://doi.org/10.1016/j.asoc.2007.05.007

[2] Karaboga, D. and Akay, B. (2009) A Comparative Study of Artificial Bee Colony Algorithm. *Applied Mathematics and Computation*, **214**, 108-132. https://doi.org/10.1016/j.amc.2009.03.090

[3] Zhu, G.P. and Kwong, S. (2010) Gbest-Guided Artificial Bee Colony Algorithm for Numerical Function Optimization. *Applied Mathematics and Computation*, **217**, 3166-3173. https://doi.org/10.1016/j.amc.2010.08.049

[4] Urselmann, M., Barkmann, S., Sand, G., *et al.* (2011) A Memetic Algorithm for Global Optimization in Chemical Process Synthesis Problems. *IEEE Transactions on Evolutionary Computation*, **15**, 659-683.

https://doi.org/10.1109/TEVC.2011.2150753

[5] Bansal, J.C., Sharma, H., Aryak, V., *et al*. (2013) Memetic Search in Artificial Bee Colony Algorithm. *Soft Computing*, **17**, 1911-1928. https://doi.org/10.1007/s00500-013-1032-8

[6] Gao, W.F., Liu, S.Y. and Huangl, L. (2013) A Novel Artificial Bee Colony Algorithm with Powell's Method. *Applied Soft Computing*, **13**, 3763-3775. https://doi.org/10.1016/j.asoc.2013.05.012

[7] Kang, F., Li, J. and Ma, Z. (2011) Rosenbrock Artificial Bee Colony Algorithm for Accurate Global Optimization of Numerical Functions. *Information Science*, **181**, 3508-3531. https://doi.org/10.1016/j.ins.2011.04.024

[8] Kang, F., Li, J. and Li, H. (2013) Artificial Bee Colony Algorithm and Pattern Search Hybridized for Global Optimization. *Applied Soft Computing*, **13**, 1781-1791. https://doi.org/10.1016/j.asoc.2012.12.025

[9] Gao, W.F. and Liu, A.Y. (2012) A Modified Artificial Bee Colony Algorithm. *Computers & Operations Research*, **39**, 687-697. https://doi.org/10.1016/j.cor.2011.06.007

[10] Banharnsakun, A. and Achalakul, T. (2011) The Best-So-Far Selection in Artificial Bee Colony Algorithm. *Applied Soft Computing*, **11**, 2888-2901. https://doi.org/10.1016/j.asoc.2010.11.025

[11] Chelouah, R. and Siarry, P. (2000) A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions. *Journal of Heuristics*, **6**, 191-213. https://doi.org/10.1023/A:1009626110229