Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **9**, No. 2, pp. 307-323

DOI: 10.4208/aamm.2016.m1481
April 2017

Hill-Climbing Algorithm with a Stick for Unconstrained Optimization Problems

Yunqing Huang* and Kai Jiang*

Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Key Laboratory of Intelligent Computing & Information Processing of Ministry of Education, School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan 411105, China

Received 4 March 2016; Accepted (in revised version) 17 June 2016

Abstract. Inspired by the behavior of the blind for hill-climbing using a stick to detect a higher place by drawing a circle, we propose a heuristic direct search method to solve the unconstrained optimization problems. Instead of searching a neighbourhood of the current point as done in the traditional hill-climbing, or along specified search directions in standard direct search methods, the new algorithm searches on a surface with radius determined by the motion of the stick. The significant feature of the proposed algorithm is that it only has one parameter, the search radius, which makes the algorithm convenient in practical implementation. The developed method can shrink the search space to a closed ball, or seek for the final optimal point by adjusting search radius. Furthermore our algorithm possesses multi-resolution feature to distinguish the local and global optimum points with different search radii. Therefore, it can be used by itself or integrated with other optimization methods flexibly as a mathematical optimization technique. A series of numerical tests, including high-dimensional problems, have been well designed to demonstrate its performance.

AMS subject classifications: 90C56, 90C30, 56K05

Key words: Direct search algorithm, stick hill-climbing algorithm, search radius.

1 Introduction

For simplicity, we limit our discussion to the following unconstrained optimization problem

$$\max_{x\in\Omega} f(x), \quad \Omega\subset\mathbb{R}^n,$$

*Corresponding author.
Email: huangyq@xtu.edu.cn (Y. Q. Huang), kaijiang@xtu.edu.cn (K. Jiang)

 ©2017 Global Science Press

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

where $f:\mathbb{R}^n \to \mathbb{R}$. The following statements are also suitable to find the minima of objective function with constraint conditions. There have been innumerable approaches developed to solve these optimization problems. These methods usually can be divided into two types, derivative-based methods and derivative-free methods, depending on whether they use derivative information or not. The derivative-based approaches appeal to the Taylor's series expansion of the objective function [1–3]. For example, the steepest descent method assumes the availability of first derivatives and uses the first-order Taylor polynomial to construct local linear approximations of f. Newton's method assumes the availability of first and second derivatives and uses the second-order Taylor polynomial to construct local quadratic approximations of f. However, for a variety of reasons there have always been many instances where (at least some) derivatives are unavailable or unreliable, or finite-difference derivative approximation is unavailable or available at a prohibitive cost. Some of the reasons contain increasing complexity in mathematical modeling, higher sophistication of scientific computing, an abundance of legacy codes, and data science [4]. Nevertheless, under such circumstances it may still be desirable to carry out optimization. It follows that the derivative-free optimization technique is required.

The derivative-free optimization is an area of long history and current rapid growth in the scientific and engineering communities. The derivative-free algorithms can mainly be classified as direct and model-based [5]. Direct algorithms usually determine search directions by evaluating the function f directly, whereas model-based algorithms construct and utilize a surrogate model of f to guide the search process. Recently developed methods based trust-region using interpolation model are in this category [6–9]. In practical implementation, heuristic algorithms, such as simulated annealing, genetic algorithm, neural networks, and deep learning [10, 11], have been also developed to solve derivative-free optimization. Here we focus our attention on the direct search algorithms.

The term "direct search", firstly coined by Hooke and Jeeves in 1961 year [12], is used to describe a sequential examination of trial generated by a certain strategy. From a modern viewpoint, the direct search methods neither compute nor approximate derivatives [13]. A popular direct search method is the Nelder-Mead simplex algorithm [14]. The algorithm starts with a set of points that form a simplex. In each iteration, the objective function values at the corner points of the simplex determine the worst corner point. The algorithm attempts to replace the worst point by introducing a new vertex by a reflection, an expansion, or a contraction operator that results in a new simplex. Another class of direct search methods are the directional direct-search methods, including the original Hooke-Jeeves algorithm [12], the coordinate-, or compass-search methods, pattern-search methods [4], generalized pattern search method [15], and generating set search method [16]. A multidirectional search algorithm, regarded as both a directional and a simplicial direct search method, has been also proposed by Dennis and Torczon [17]. The common ground of these directional direct-search methods is that they incorporate some mechanism to choose ascend directions and search along these directions with an appropriate step length from the current iterate for a new iterate with a higher function value.

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

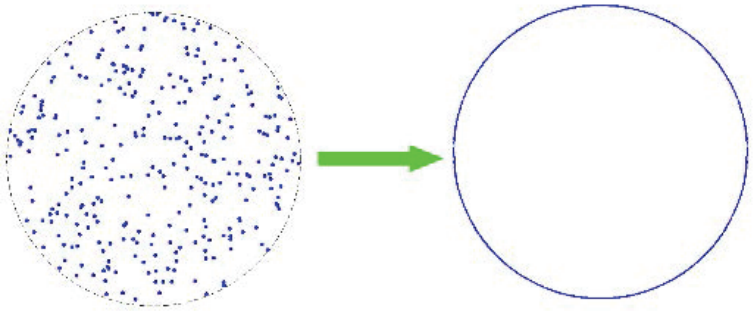


Figure 1: The schematic image demonstrates how to choose the search set from the traditional hill-climbing method to our proposed algorithm in two dimensions. In the traditional hill-climbing method, the search set is a neighbourhood of current iterate, while in our method, that is a closed surface, here a circle.

The same is done by derivative-based algorithms to enforce global convergence [1, 3]. However the predetermined search direction set may contain none of ascend or descent direction. Moreover, due to the absence of derivative information, the search directions could not be adjusted flexibly according to f [4]. Under such circumstances, the directional direct search methods fail to converge on this function. Recently, Gratton et al. [19] analyzed the convergent properties of direct-search algorithms when the search direction are probabilistic descent.

Our method is inspired by the behavior of the blind for hill-climbing that they find a high position with the help of a stick using circular motions, usually without a priori information. Based on the observation, we propose a new search strategy that the search set is a closed manifold in each iteration, for example, a circle in two dimensions, a spherical surface in higher dimensions, rather than a neighbourhood of current iterate as done in traditional hill-climbing methods [20]. An example of the search set in two dimensions is presented in Fig. 1, Then we compare function values on the closed manifold with the current iterate. It follows that the iterate is replaced once a higher function value on the manifold has been found. The new formed optimization method dose not predetermine the search directions. More importantly, it has a unique parameter, the radius of the circle or the spherical surface, which makes the algorithm convenient in practical implementation.

In the following, we will give a concrete and feasible numerical algorithm in Section 2. The length of the radius is determined by the motion of the stick, therefore, it can be adjusted during the iteration procedure. Correspondingly, a regulated search radius method is also presented in this section. In contrast to the above-mentioned directional direct search algorithms, our algorithm does not incorporate the search directions. A large number of numerical tests are showcased in Section 3. Numerical results demonstrate that the new algorithm has finite-step convergence, and is able to distinguish the local and global optimum points with different search radii. Finally the conclusion and discussions are given in Section 4.

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

2 Algorithm description

For convenience, we introduce some notations before the algorithm description. $f(x)$ is the objective function. x^* is one of the maximizers of $f(x)$ which may be local or global. x_0 is the initial position when starting the optimization method, and x_k is the k-th iterate. ρ is the search radius determined by the stick. $O(x_k,\rho)=\{x:|x-x_k|=\rho\}$ is the search surface in the k-th iteration with radius ρ. $\Omega(x_k,\rho)$ is the neighbourhood of x_k with radius of ρ. In what follows, we will propose the heuristic algorithm and point out that it can capture the area of maximizers.

Algorithm 1 Stick Hill-Climbing Algorithm.

Initialization: Choose x_0 and ρ.
For $k=0,1,2,\cdots$,
Try to find $\bar{x}\in O(x_k,\rho)$ such that $f(\bar{x})>f(x_k)$.
If such a point is found, then set $x_{k+1}=\bar{x}$.
Otherwise, terminate the iteration.

Algorithm 1 is the proposed method, the hill-climbing algorithm with a stick. For simplicity, we name it stick hill-climbing algorithm. For each k step, there are three possible circumstances in Algorithm 1 to determine whether to stop the iteration or not. The first case is $f(x)<f(x_k)$, for all $x\in O(x_k,\rho)$. The iteration is terminated, and the algorithm can declare that there is at least one maximizer in the neighbourhood of $\Omega(x_k,\rho)$ if the objective function f satisfies some conditions, such as continuity. The second situation is $f(x)\leq f(x_k)$, i.e., there exists $\bar{x}\in O(x_k,\rho)$ such as $f(\bar{x})=f(x_k)$. We let $x_{k+1}=\bar{x}$, and continue the iteration. The third case is $f(x)\equiv f(x_k)$ for all $x\in O(x_k,\rho)$. Then the method of current version can not infer the existence of maximizers or minimizers in $\Omega(x_k,\rho)$. However we can increase or decrease the search radius ρ as the below Algorithm 3 suggested, and further run Algorithm 1 to find the location of maximizers.

The search set of the developed algorithm is the surface $O(x_k,\rho)$ of the current iterate x_k with radius ρ, rather than the neighbourhoods of x_k as done in the traditional hill-climbing. The developed algorithm tries to search for the neighbourhood of maximizers rather than to find the maximizer directly. Moreover, the distance between the extreme points and the convergent iterate is not more than ρ. If ρ is driven to zero, the monotonic increasing sequence $\{f(x_k)\}$ produced by the stick hill-climbing algorithm implies $\{x_k\}$ approximates a maximizer. Therefore, the search radius ρ is a natural choice to represent the approximate error of the stick hill-climbing algorithm.

In practice, we have to sample the observed points set in each iteration for Algorithm 1. Obviously, there exist lots of ways to accomplish it, such as randomly, adaptively, or uniformly sampling. Here we choose a dynamic refinement principle to sample the search surface in each iteration rather than a dense mesh from the start, as Algorithm 2 shows. For k iteration in Algorithm 1, we firstly use a coarse mesh to sample $O(x_k,\rho_k)$. By comparing the function values of these samples with $f(x_k)$, Algorithm 2 determines

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

Algorithm 2 Dynamic Refinement Technique.

For k iteration in Algorithm 1

Step 1. Sample the surface $O(x_k,\rho_k)$ with a coarse mesh.

Step 2. Compare the function values of the samples with $f(x_k)$.
If there exists $\bar{x}$ such that $f(\bar{x})>f(x_k)$, goto **Step 4**.
Otherwise, goto **Step 3**.

Step 3. Subdivide $O(x_k,\rho_k)$, goto **Step 2**.

Step 4. Declare that the iteration is successful, and set $x_{k+1}=\bar{x}$.

whether the current grid shall be refined or not. The technique can reduce the number of the function evaluations. For example, in two dimensions, the search set is a circle. The schematic plot of the equally refined method is presented in Fig. 2 when the maximum sample points $N_{\max}=32$. For n-dimensional ($n\geq 3$) optimization problems, the search areas are restricted to an n-dimensional spherical surface. In these cases, we use the generalize spherical coordinate transformation to obtain the position of samples, i.e.,

$$\begin{aligned}
x_{k+1}^1 &= x_k^1+\rho_k\cos(\phi^1),\\
x_{k+1}^2 &= x_k^2+\rho_k\sin(\phi^1)\cos(\phi^2),\cdots,\\
x_{k+1}^{n-1} &= x_k^{n-1}+\rho_k\sin(\phi^1)\cdots\sin(\phi^{n-2})\cos(\phi^{n-1}),\\
x_{k+1}^n &= x_k^n+\rho_k\sin(\phi^1)\cdots\sin(\phi^{n-2})\sin(\phi^{n-1}),
\end{aligned}$$

where $\phi^1,\cdots,\phi^{n-2}\in[0,\pi]$, $\phi^{n-1}\in[0,2\pi)$. Then one can discrete the angular coordinates, $\phi^1,\cdots,\phi^{n-1}$, by the equally refined method in a dynamical way. In practice, we discretize $\phi^1,\phi^2,\cdots,\phi^{n-1}$ using $[m,m,\cdots,2m]$ points at start, and let $m\leftarrow m+k$ ($k\in\mathbb{Z}^+$) in the refinement process. The process is terminated when $2m^{n-1}$ is greater than a given $N_{\max}$.

If the stick hill-climbing algorithm converges, it is evident that the search space is shrunk to a ball with the radius ρ. The convergent point x_k provides a good initial value for other optimization methods, including derivative-free approaches, and derivative-based algorithms if the objective function is differentiable. We will demonstrate this by several numerical experiments in Section 3. Meanwhile, we can further exploit the potential of stick hill-climbing algorithm to improve the precision by tuning the search radius

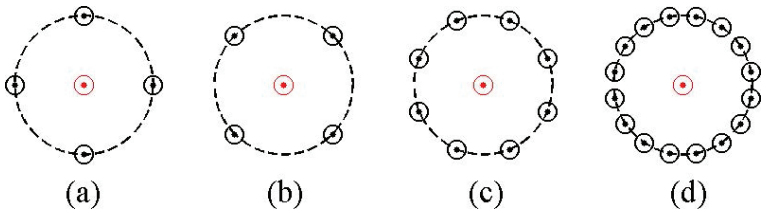


Figure 2: Schematic diagram of equally refined approach for sampling the search surface $O(x_k,\rho)$ in two dimensions, i.e., a circle. Subfigure (a) demonstrates the initial samples, and images (b)-(d) show new sample points in the equally refined process sequentially.

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

ρ. Algorithm 3 presents a strategy to resize the search radius ρ. Significant difference of the version from Algorithm 1 is adjusting the search radius ρ when Algorithm 1 fails to find $f(\bar{x})>f(x_k)$, $\bar{x}\in O(x_k,\rho)$ with fixed ρ. The natural stop criterion of Algorithm 3 is to terminate the run when ρ is smaller than a prescribed numerical accuracy. Certainly, the search surface can be expanded by setting control factor $\eta>1$ if required. In fact, Algorithm 3 provides a restart technique by fixed the k-iterate with different search radius ρ.

Algorithm 3 Changeable Stick Hill-Climbing Algorithm.

Initialization: Choose x_0, ρ, and control factor η.
For $k=0,1,2,\cdots$,
Use Algorithm 1 to find $\bar{x}\in O(x_k,\rho)$ such that $f(\bar{x})>f(x_k)$.
If such a point is found, then set $x_{k+1}=\bar{x}$.
Otherwise, change the search radius $\rho=\eta\cdot\rho$.

3 Numerical results

In this section, we choose three kinds of test functions, including single extreme point function, multi-extreme points function, and continuous but indifferentiable function, to demonstrate the performance of the stick hill-climbing algorithm. The maximum number of sampling points $N_{\max}$ for search set in Algorithm 2 defaults to 16 in two dimensional cases if not specified (see Figs. 2(a)-(c)).

3.1 Single extreme point function

For the first example we consider two-dimensional Gaussian function

$$f(x,y)=h\exp\Big(-\frac{x^2}{s_x}-\frac{y^2}{s_y}\Big), \tag{3.1}$$

where $h=10$, $s_x=1.0$, $s_y=0.5$. The graph of a Gaussian function likes a "bell" sharp, and has a unique maximum, $(x^*,y^*)=(0,0)$, and $f(0,0)=10$. The objective function is differentiable in $\mathbb{R}^2$, however, it quickly falls off towards zero out of the "bell".

We firstly investigate the influence of initial values for the stick hill-climbing algorithm. For this set of tests of 20 runs, the search radius ρ is fixed as 1.0, start points are randomly generated in the space $(-10,10)^2$. For each experiment, the stick hill-climbing method can capture a neighbourhood of the peak $(0,0)$ in finite iterations. Fig. 3 gives the required iterations for convergence in 20 numerical experiments. In these 20 runs, the convergent iterations are about 12 on average, while the maximum is 20, and the minimum is 5. The number of iterations is inversely proportional to the distance of the initial point and the optimal point. When the initial values are far away from the optimal point,

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

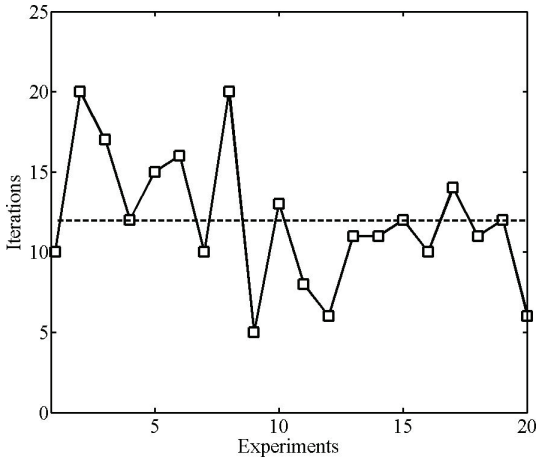


Figure 3: The iterations of convergence of the stick hill-climbing algorithm to the single-peak Gaussian function (3.1) in 20 runs. Start points are randomly generated in the space $(-10,10)^2$, and $\rho=1.0$. The flat dashed line shows the average.

the algorithm needs more iterations. In contrast, when the initial points are close to the maximizer, the method requires less iterations. We also note that when the start point is far away from the peak, the derivative-based methods, such as steepest descent method, conjugate gradient method and Newton method, will fail since the gradient value is almost zero. However, the stick hill-climbing algorithm can always approximate the peak point. The initial values may yield a few more iterations but NOT affect the convergence.

Then we take an example to further observe the numerical behavior of the stick hill-climbing algorithm. Table 1 shows the iterative procedure of the stick hill-climbing approach in detail when the start point is $(x_0,y_0)=(3.3,2.9)$ with fixed search radius $\rho=1.0$. The results show that the iterates can be updated efficiently to capture the neighbourhood of the maximizer $(0,0)$ within 6 steps. The convergent result reduces the search space and provides a good start position $(0.3,-0.1)$ to further approximate the maximizer to high precision with other optimization algorithms.

Table 1: The iterative procedure of stick hill-climbing algorithm with $\rho=1.0$ when the initial value is $(x_0,y_0)=(3.3,2.9)$.

k	(x_k,y_k)	$f(x_k,y_k)$	$\lvert f(x_k,y_k)-f(x^*,y^*)\rvert$
0	(3.3, 2.9)	9.24059×10^{-12}	9.999999999990759
1	(3.3, 1.9)	1.36435×10^{-7}	9.999999863564643
2	(2.3, 1.9)	3.68957×10^{-5}	9.999963104276466
3	(2.3, 0.9)	9.97758×10^{-3}	9.990022422035157
4	(1.3, 0.9)	0.36516	9.634838262462596
5	(0.3, 0.9)	1.80866	8.191342073828778
6	(0.3, -0.1)	8.95834	1.041658647034717

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

Table 2: The iterative procedure of the changeable ρ compass hill-climbing Algorithm 3 with control factor $\eta=0.5$ when the start point is $(x_0,y_0)=(0.3,-0.1)$, and initial search radius $\rho=0.8$. k_ρ represents the number of iterations for a given ρ.

ρ	k_ρ	(x_k,y_k)	$f(x_k,y_k)$	$\lvert f(x_k,y_k)-f(x^*,y^*)\rvert$
0.8	1	(0.3, -0.1)	8.9583413	1.04166
0.4	2	(-0.1, -0.1)	9.7044553	2.95545×10^{-2}
0.2	2	$(4.14214\times10^{-2},4.14214\times10^{-2})$	9.9486604	5.13396×10^{-2}
0.1	2	$(-2.92893\times10^{-2},-2.92893\times10^{-2})$	9.9742972	2.57028×10^{-2}
5×10^{-2}	4	$(-1.46447\times10^{-2},-1.46447\times10^{-2})$	9.9935681	6.43191×10^{-3}
2.5×10^{-2}	4	$(-7.32233\times10^{-3},-7.32233\times10^{-3})$	9.9983916	1.60837×10^{-3}
1.25×10^{-2}	4	$(-3.66117\times10^{-3},-3.66117\times10^{-3})$	9.9995979	4.02116×10^{-4}
6.25×10^{-3}	4	$(7.58252\times10^{-4},7.58252\times10^{-4})$	9.9999272	7.27699×10^{-5}
3.125×10^{-3}	4	$(-9.15291\times10^{-4},-9.15291\times10^{-4})$	9.9999749	2.51327×10^{-5}
1.5625×10^{-3}	2	$(1.895630\times10^{-4},1.895630\times10^{-4})$	9.9999989	1.07802×10^{-6}

Due to the good analytical nature of objective function, the derivative-based methods are good choices for finding the maximizer with the above convergent results. For our purpose, we still adopt the stick hill-climbing method, but with changeable ρ to approximate the peak (see Algorithm 3), and observe its numerical behavior. The start node is $(0.3,-0.1)$, the initial search radius is $\rho_0=0.8$. The control factor η in Algorithm 3 is set to be 0.5 under which circumstances the search radius can be driven to decrease during iteration. Table 2 shows that, in 29 ($\sum k_\rho$) steps, Algorithm 3 can efficiently obtain a neighbourhood of the maximizer $(0,0)$, while the error of $\lvert f(x_{29},y_{29})-f(0,0)\rvert$ achieves about $\mathcal{O}(10^{-6})$.

3.2 Multi-extreme points function

The second tested function is the Ackley function [18], a benchmark function, widely used for testing optimization algorithms. The expression of the Ackley function can be written as

$$f(x)=-a\cdot\exp\left(-b\cdot\sqrt{\frac{1}{n}\sum_{i=1}^{2}x_i^2}\right)-\exp\left(\frac{1}{n}\cos(cx_i)\right)+a+\exp(1), \tag{3.2}$$

where $a=20$, $b=0.2$, $c=2\pi$, n is the dimension. Ackley function has many local minima and a unique global minimum of $(0,0)$ with $f(0,0)=0$, which poses a risk for optimization algorithms to be trapped into one of local minima, such as the traditional hill-climbing method [21]. In this subsection, we will apply the stick hill-climbing algorithm to 2-, 4-, and 8-dimensional Ackley function.

Firstly we consider the 2-dimensional Ackley function whose morphology is shown in Fig. 4. We want to examine the ability of Algorithm 1 to approximate the global minimum. The first numerical examples are also set to observe the influence of initial values

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

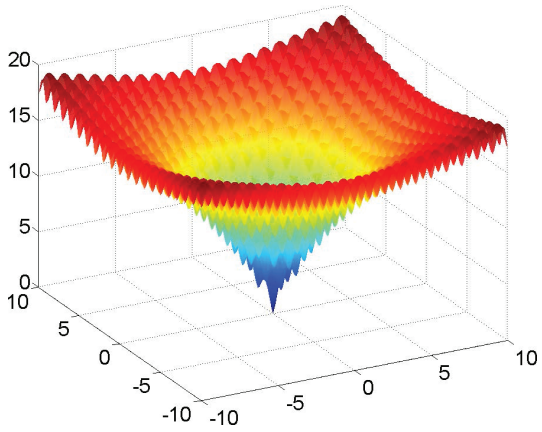


Figure 4: Morphology of 2-dimensional Ackley function (3.2).

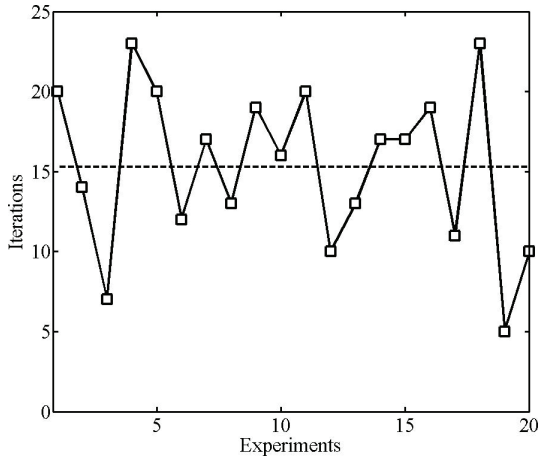


Figure 5: The required iterations of convergence by the stick hill-climbing algorithm 1 for Ackley function (3.2) in 20 numerical experiments. Start points are randomly generated in the space $(-12,12)^2$ with fixed $\rho=1.0$. The flat dashed line shows the average.

with fixed ρ. When $\rho=1.0$, 20 numerical tests are performed with random initial points generated in the space of $(-12,12)^2$. The convergent domain in each test contains the global minimizer $(0,0)$. Fig. 5 shows the required iterations for convergence. In the 20 numerical tests, the average convergent steps are about 15, while the maximum number of iterations is 23 and the minimum is 5. The initial positions also only affect the speed of convergence, but not the ability of catching the neighbourhood of the global minimum. To make this point clear, Table 3 gives the iterative procedure in detail when the start position is $(x_0,y_0)=(4.1,3.4)$, and $\rho=1.0$. One can see that the stick hill-climbing algorithm has ability to find a neighbourhood of the global minimum in 9 iterations. The convergent point (x_9,y_9) is close to the global minimizer $(0,0)$. As the above subsection

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

Table 3: The iterative procedure of the stick hill-climbing algorithm with fixed $\rho=1.0$ when the initial value is $(x_0,y_0)=(4.1,3.4)$. For Ackley function, the global minimum is $(x^*,y^*)=(0,0)$, and $f(x^*,y^*)=0$.

k	(x_k,y_k)	$\lvert f(x_k,y_k)-f(x^*,y^*)\rvert$
0	(4.1, 3.4)	12.301695
1	(3.1, 3.4)	11.284588
2	(3.1, 2.4)	10.230367
3	(2.1, 2.4)	8.978451
4	(2.1, 1.4)	7.721871
5	(1.1, 1.4)	6.170178
6	(1.1, 0.4)	4.769384
7	(0.1, 0.4)	2.851124
8	(1.02388, 0.01732)	2.719632
9	(0.02388, 0.01732)	0.106464

Table 4: The convergent results and required iterations of the stick hill-climbing algorithm with different ρ when the initial value is $(x_0,y_0)=(4.1,3.4)$. The locations of different local minima are marked in Fig. 6.

ρ	0.1	0.3	0.5	0.6	0.7	0.73	0.74	0.75	1.0
Iterations	6	3	2	5	3	3	3	11	10
Minimizer	Loc_1	Loc_1	Loc_1	Loc_4	Loc_2	Loc_2	Loc_3	Global Min.	Global Min.

done, we can also resize the search radius ρ in the stick hill-climbing algorithm to further approximate the global minimum. If the initial search radius is $\rho_0=1.0$, and control factor $\eta=0.5$, the changeable stick hill-climbing scheme costs 47 iterations to reach an error of $\rho_{\text{tol}}=10^{-6}$. The convergent point is $(3.170987\times10^{-7},3.475996\times10^{-7})$ which is much closer to the global minimum $(0,0)$.

The above tests have demonstrated that the stick hill-climbing algorithm can capture the neighbourhood of the global minimum of the Ackley function. Subsequently we plan to manifest the ability of the stick hill-climbing algorithm to obtain these local minimizers. As discussed above, the unique regulatable parameter in the stick hill-climbing scheme is the search radius ρ. Therefore we will test the numerical behaviors with different size of ρ. In the set of tests, the initial value is always $(x_0,y_0)=(4.1,3.4)$. Table 4 gives the convergent results with different ρ. Fig. 6 marks corresponding local minima.

From these numerical experiments, one can find that the search radius ρ plays a filter role in catching different local minimum by setting different values. When ρ is greater than 0.75, the stick hill-climbing approach can approximate the global minimum, otherwise, the method converges to the local minima. If starting from other initial position, the stick hill-climbing can find other minima.

Next we consider the high-dimensional Ackley function, especially, 4-, and 8-dimensional problems. In these numerical experiments, we also choose the equally refine approach to sample the observed spherical surface in a dynamic way in each iteration as mentioned in Section 2. The initial point $m=3$, and in each refinement procedure, we let $m=m+1$. For 4-, and 8-dimensional problems, the maximum samples $N_{\max}$ is set as

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

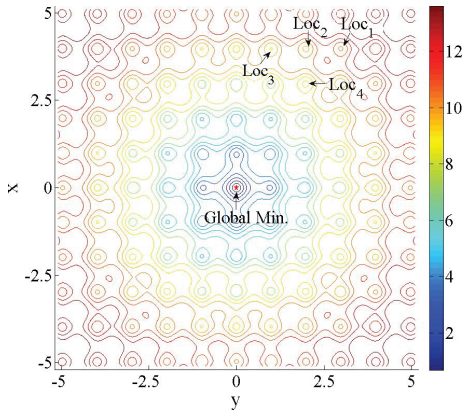


Figure 6: The locations of some local minima and the global minimum of the Ackley function.

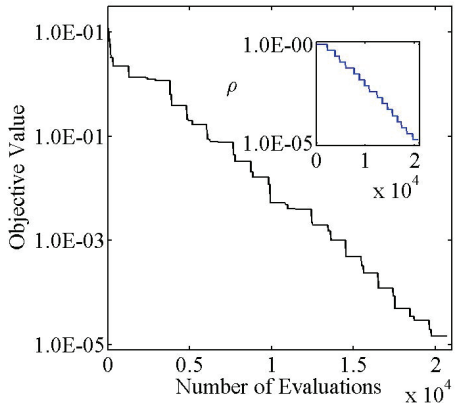


Figure 7: Numerical behavior of Algorithm 3 when solving 4-dimensional Ackley function. The initial state is $x_0=(4.1,3.4,3.45,3.12)$, and the maximum sampling points $N_{\max}$ is 800. The inset is the convergent behavior of search radius ρ.

800 and 8000, respectively. The search radius ρ is 1.0 for 4-dimensional case, and 2.0 for 8-dimensional problem. The convergent criterion is $\rho<1.0^{-5}$ in both cases. The changeable stick hill-climbing method, i.e., Algorithm 3, is applied to these high-dimensional optimization problems.

The numerical behavior of Algorithm 3 when solving 4-dimensional Ackley function is given in Fig. 7 with the initial value $x_0=(4.1,3.4,3.45,3.12)$. After 20681 function evaluates, Algorithm 3 reaches the given error of 1.0^{-5}. The convergent solution is $(3.06\times10^{-7},4.09\times10^{-6},5.90\times10^{-6},-7.60\times10^{-7})$ and function value is 1.446×10^{-5}. Starting from the initial state $x_0=(2.12,5.426,-3.45,3.12,-3.0,0.99,1.57,-4.11)$, the numerical behavior of Algorithm 3 for solving 8-dimensional Ackley function is given in Fig. 8. Algorithm 3 costs 840826 function evaluations to achieve the given error. The convergent solution is $(-6.32\times10^{-7},-1.78\times10^{-6},-3.78\times10^{-6},-6.66\times10^{-7},7.02\times$

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

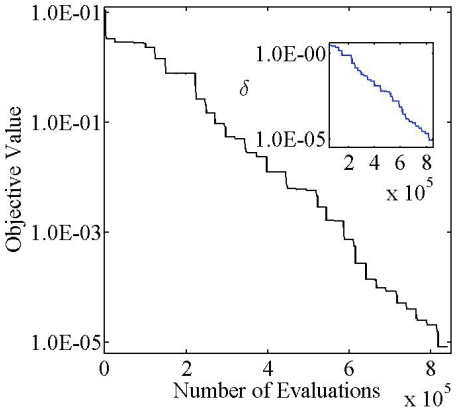


Figure 8: Numerical behavior of Algorithm 3 when solving 8-dimensional Ackley function. The initial state is $x_0=(2.12,5.426,-3.45,3.12,-3.0,0.99,1.57,-4.11)$, and the maximum sampling points $N_{\max}$ is 8000. The inset is the convergent behavior of search radius ρ.

$10^{-7},1.19\times10^{-6},-1.82\times10^{-7},-3.73\times10^{-6})$ with function value 8.262×10^{-6}. For both high-dimensional Ackley problems, Algorithm 3 can efficiently obtain a neighbourhood of the global minimizer with a controllable number of function evaluations.

3.3 Continuous but indifferentiable function

The last model is a variant of Dennis-Woods function [16,22],

$$f(x)=\frac{1}{2}\max\{\|x-c_1\|^2,\|x-c_2\|^2\},\quad x=(x,y), \tag{3.3}$$

where $c_1=(1,-1)^T$, $c_2=-c_1$, $\|\cdot\|$ denotes l^2 norm. This objective function has a unique minimizer of $(0,0)$, and $f(0,0)=1$, indicated by a red star in the contour plot of Fig. 9. The function is continuous and strictly convex everywhere, but its gradient is discontinuous along the line $x=y$. It has been shown that the Nelder-Mead simplex algorithm fails to converge to the minimizer of Dennis-Woods function (3.3) in [22]. In this subsection, we will investigate the performance of our proposed method.

Firstly, we still examine the efficiency of the stick hill-climbing algorithm for seeking for the neighbourhood of the minimizer $(0,0)$ of the Dennis-Woods function. Twenty numerical experiments are performed with random initial values generated in the space of $(-5,5)^2$. The search radius is always fixed as $\rho=0.5$ for the set of tests. The stick hill-climbing algorithm can find a neighbourhood of the minimizer $(0,0)$ in each test no matter which initial value is used. The required iterations for convergence is shown in Fig. 10, while the average iterations are about 13.5. In these cases, the stick hill-climbing method demonstrates a pronounced convergent behavior, even within 2 steps.

Subsequently, we will take an example to show the numerical behavior of the stick hill-climbing algorithm in detail. The initial value is $(x_0,y_0)=(1.0,1.5)$, and the search

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

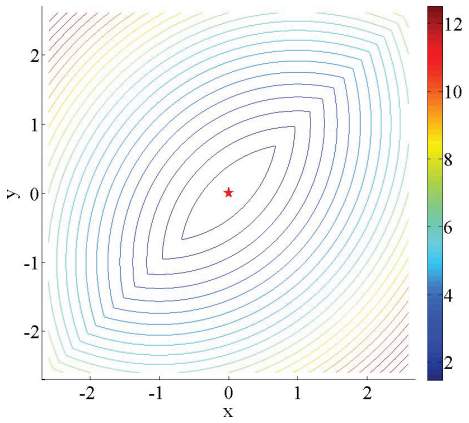


Figure 9: Contours of the variant of the Dennis-Woods function (3.3).

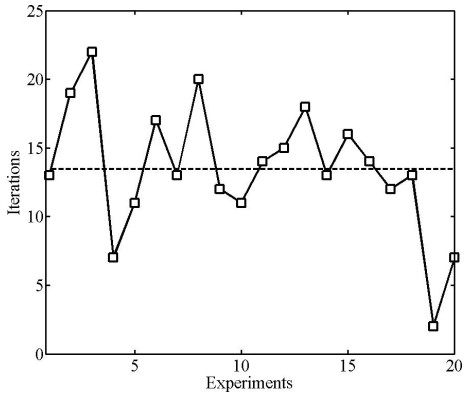


Figure 10: The iterations of convergence of the stick hill-climbing method to the Dennis-Woods function (3.3) in 20 test examples. Start points are randomly generated in the space $(-5,5)^2$, and $\rho=0.5$. The flat dashed line shows the average.

radius is $\rho=0.5$. Table 5 gives the iterative procedure. The stick hill-climbing method can detect the decrease quickly, and find a neighbourhood of the minimizer in 4 iterations. The convergent point is $(-0.06066,-0.06066)$. Using the convergent point as initial value, one can also adjust ρ in the stick hill-climbing scheme to further approach the minimizer.

Finally we compare the performance of the stick hill-climbing algorithm with one of the standard directional direct-search methods, the coordinate-search (CS) method, for the Dennis-Woods function. Before we go further, a short introduction of the CS method is necessary. More details about the CS method can be found, for instance, in a recent monograph [4] or [16]. The CS method makes use of the positive bases $\mathcal{D}_{\oplus}$ which spans the $\mathbb{R}^2$ with positive coefficients. Let x_k be the current iterate and λ the current search step length. The CS method evaluates the function f at the points in the set

$$\mathcal{P}_k=\{x_k+\lambda d: d\in\mathcal{D}_{\oplus}\},$$

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

Table 5: The convergent results of the stick hill-climbing algorithm with $\rho=0.5$ when the initial value is $(x_0,y_0)=(1.0,1.5)$. $(x^*,y^*)=(0,0)$, $f(x^*,y^*)=0$.

k	(x_k,y_k)	$\lvert f(x_k,y_k)-f(x^*,y^*)\rvert$
0	(1.0, 1.5)	3.125
1	(1.0, 1.0)	2.0
2	(0.64645, 0.64645)	1.417893
3	(0.29289, 0.29289)	1.085786
4	(-0.06066, -0.06066)	1.003680

following some specified order, trying to find a point in $\mathcal{P}_k$ that decreases the objective function value. When that happens, the method defines a new iterate $x_{k+1}=x_k+\lambda d\in\mathcal{P}_k$ such that $f(x_{k+1})<f(x_k)$. In such a case, one either leaves the parameter λ unchanged, or increases it, or decreases it. Here we will change λ during the iterative procedure. If none of the points in $\mathcal{P}_k$ leads to a decrease in f, then the parameter λ is reduced (here by a factor of $1/2$) and the next iteration at the same point ($x_{k+1}=x_k$). The algorithm is terminated when the λ is smaller than a given tolerance.

Like the CS method, the stick hill-climbing algorithm has similar algorithmic steps to update iterates and change the search radius. Unlike the CS method, however, the stick hill-climbing algorithm does not predetermine the search directions, such as $\mathcal{D}_\oplus$ in the CS scheme. In k iteration, the stick hill-climbing evaluates functions on the search surface $O(x_k,\rho)$, and compares them with $f(x_k)$. In practice, the search surface $O(x_k,\rho)$ is dynamically sampled as discussed in Algorithm 2. From another perspective, the stick hill-climbing algorithm provides an adaptive mechanism to adjust the search directions according to objective function.

In the following numerical tests, the initial value is $(x_0,y_0)=(1.1,0.9)$, the initial search radius is $\rho=1.0$. In the stick hill-climbing algorithm, the maximum sample points $N_{\max}$ of $O(x_k,\rho)$ is 8. In the CS method, the positive bases is $\mathcal{D}_\oplus=\{(1,0),(0,1),(-1,0),(0,-1)\}$ of standard choice. The control factor of adjusting search radius $\eta=0.5$ in both methods.

Fig. 11 shows the error of $|f(x_k,y_k)-f(x^*,y^*)|$ against the number of function evaluations for both the stick hill-climbing and the CS method. As observed in Fig. 11 the CS method stalls at the error of $\mathcal{O}(10^{-2})$, however, the stick hill-climbing method can continue to reduce error. At an error of 2×10^{-9}, the stick hill-climbing scheme spends 349 function evaluations, while the iterate point is $(1.50249\times10^{-9},4.48273\times10^{-9})$, and search radius becomes 7.45058×10^{-9}. The reason is that when the iterate approaches to the minimizer, the CS method can not detect a sufficient decrease, even no decrease, along the predetermined search directions in $\mathcal{D}_\oplus$. This results in the stagnation phenomenon. However, the mechanism of detecting smaller values in the stick hill-climbing algorithm is dependent on the feature of $f(x_k)$ rather than along predetermined directions. It yields the efficient and flexible performance of the stick hill-climbing method when approximating the minimizer.

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

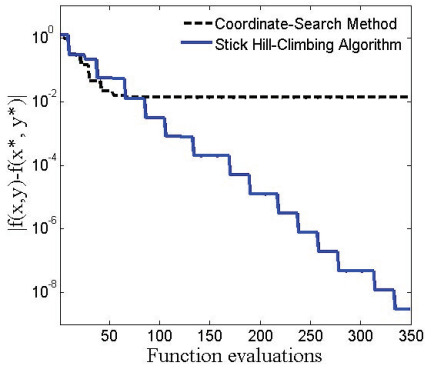


Figure 11: Application of the stick hill-climbing ($N_{\max}=8$) and the CS (with predetermined search directions $\mathcal{D}_{\oplus}$) methods to the Dennis-Woods function starting from $(x_0,y_0)=(1.1,0.9)$. In the stick hill-climbing algorithm, the initial search radius $\rho=1.0$, and in the CS approach the initial search step length $\lambda=1.0$. The control factor $\eta=0.5$ in both methods. $f(x^*,y^*)$ is the global minimizer.

4 Conclusions and discussion

In this paper, we have developed a new and simple direct search method, the stick hill-climbing algorithm, to solve unconstrained optimization problems. The proposed method is inspired from the behavior of the blind for hill-climbing with a stick to detect a higher position using circular motions. The main idea of the new algorithm, at each iteration, is comparing function values on a surface of the current point with the nodal value, rather than a neighbourhood of the current node. The proposed approach has a unique parameter, the search radius determined by the motion of the stick, which makes the algorithm convenient in numerical implementation. At the same time, a concrete and feasible numerical scheme of the stick hill-climbing method, and an algorithm of tuning search radius are also presented.

Three different types of objective functions, including single extreme point function, multi-extreme points function, and continuous but undifferentiable function, were designed to test the performance of the stick hill-climbing algorithms. So far, there is no theoretical result about the convergence and the speed of convergence for the stick hill-climbing algorithm. However, numerical results demonstrate that the stick hill-climbing algorithm is a robust scheme, and can be applied to find the neighbourhood of extreme points when objective functions are continuous. Furthermore, the algorithm has the following advantages. For an appropriate search radius, the stick hill-climbing approach can converge in finite iterations which depends only on the search radius rather than the initial values. It also has potential to overcome oscillations, and find the global minimizer, even for high-dimensional problems, by using an appropriate search radius ρ, as Section 3.2 presented. For an objective function with multi-extreme points, the proposed algorithm possesses multi-resolution feature to distinguish the local and global optimum points with different values of ρ. The convergent result of the stick hill-climbing method

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

shrinks the search space and provides a good initial value for further approximating the extreme point. The distance between the stationary point and convergent point is less than the search radius ρ. Therefore the stick hill-climbing scheme provides a good initial guess when integrating with other optimization methods. In the stick hill-climbing algorithm, the search radius plays the key role instead of the search direction. It can approximate a maximum or a minimum to high accuracy by itself through tuning ρ during iterative procedure. Based on these good features, it is believed that the stick hill-climbing algorithm has potential applications in computational science. Meanwhile, the stick hill-climbing algorithm can be easily applied to higher-dimensional problems and constrained optimization.

This work can be extended in several directions. The rigorous mathematical theory should be built up, such as the convergence. The algorithm can be applied to more higher-dimensional problems using some well-designed techniques. Finally, applying the stick hill-climbing algorithm to scientific problems is also important.

Acknowledgements

This work was supported by National Science Foundation of China (Nos. 91430213, 11401504). K. Jiang was partially supported by the research grand from Hunan Science Foundation of China (No. 2015JJ3127).

References

[1] W. Y. Sun and Y. Yuan, Optimization Theory and Methods: Nonlinear Programming, New York: Springer, 2006.
[2] A. R. Conn, N. I. M. Gould and P. L. Toint, Trust Region Methods, Philadelphia: SIAM, 2000.
[3] J. Nocedal and S. J. Wright, Numerical Optimization, Berlin: Springer-Verlag, 2nd ed., 2006.
[4] A. R. Conn, K. Scheinberg and L. N. Vicente, Introduction to Derivative-Free Optimization, Philadelphia: SIAM, 2009.
[5] L. M. Rios and N. V. Sahinidis, *Derivative-free optimization: a review of algorithms and comparison of software implementations*, J. Global Optim., 56 (2013), pp. 1247–1293.
[6] M. J. D. Powell, *UOBYQA: unconstrained optimization by quadratic approximation,* Technical Report DAMTP NA2000/14, CMS, University of Cambridge, 2000.
[7] M. J. D. Powell, *On trust region methods for unconstrained minimization without derivatives,* Technical Report DAMTP NA2002/NA02, CMS, University of Cambridge, February 2002.
[8] T. Wu, Y. Yang, L. Sun, and H. Shao, *A heuristic iterated-subspace minimization method with pattern search for unconstrained optimization,* Comput. Math. Appl., 58 (2009), pp. 2051–2059.
[9] Z. Zhang, *Sobolev seminorm of quadratic functions with applications to derivative-free optimization,* Math. Program., 146 (2014), pp. 77–96.
[10] Z. Michalewicz and D. B. Fogel, How to Solve It: Modern Heuristics, Springer, 2004.

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press

[11] Y. LeCun, Bengio, Y. Hinton and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 521–436.
[12] R. Hooke and T. A. Jeeves, *"Direct search" solution of numerical and statistical problems,* J. ACM, 8 (1961), pp. 212–229.
[13] R. M. Lewis, V. Torczon and M. W. Trosset, *Direct search methods: then and now,* J. Comput. Appl. Math., 124 (2000), pp. 191–207.
[14] J. A. Nelder and R. Mead, *A simplex method for function minimization,* Comput. J., 7 (1965), pp. 308–313.
[15] V. Torczon, *On the convergence of pattern search algorithms,* SIAM J. Optim., 7 (1997), pp. 1–25.
[16] T. G. Kolda, R. W. Lewis and V. Torczon, *Optimization by direct search: new perspectives on some classical and modern methods,* SIAM Rev., 45 (2003), pp. 385–482.
[17] J. E. Jr Dennis and V. Torczon, *Direct search methods on parallel machines,* SIAM J. Optim., 1 (1991), pp. 448–474.
[18] J. M. Dieterich and B. Hartke, *Empirical review of standard benchmark functions using evolutionary global optimization,* Appl. Math. 3 (2012), pp. 1552–1564.
[19] S. Gratton, C. W. Royer, L. N. Vicente and Z. Zhang, *Direct search based on probabilistic descent,* SIAM J. Optim., 25 (2015), pp. 1515–1541.
[20] S. J. Russell and P. Norvig, Artificial Intelligence: a Modern Approach, 3rd ed., Prentice Hall, 2010.
[21] T. Bäck, Evolutionary Algorithms in Theory And Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.
[22] J. E. Jr Dennis and D. J. Woods, *Optimization on microcomputers: The Nelder-Mead simplex algorithm*, In: New Computing Environments: Microcomputers in Large-Scale Computing, A. Wouk ed., Philadelphia: SIAM, 1987.

https://doi.org/10.4208/aamm.2016.m1481 Published online by Cambridge University Press