BAS: 天牛须搜索智能优化算法

基本情况

- **天牛须搜索**(Beetle Antennae Search-**BAS**)，也叫甲壳虫须搜索，是2017年提出的一种高效的智能优化算法。
- 类似于遗传算法、粒子群算法、模拟退火等智能优化算法，天牛须搜索**不需要知道函数的具体形式**，不需要梯度信息，就可以实现高效寻优。
- 相比于粒子群算法，天牛须搜索只需要**一个个体**，即一只天牛，运算量大大降低。

仿生原理

- **天牛须搜索是受到天牛觅食原理启发而开发的算法。**
- 天牛须搜索的生物原理：

  当天牛觅食时，天牛并不知道实物在哪里，而是根据食物气味的强弱来觅食。天牛有两只长触角，如果左边触角收到的气味强度比右边大，那下一步天牛就往左飞，否则就往右飞。依据这一简单原理天牛就可以有效找到食物。
- 天牛须搜索对我们的启发：

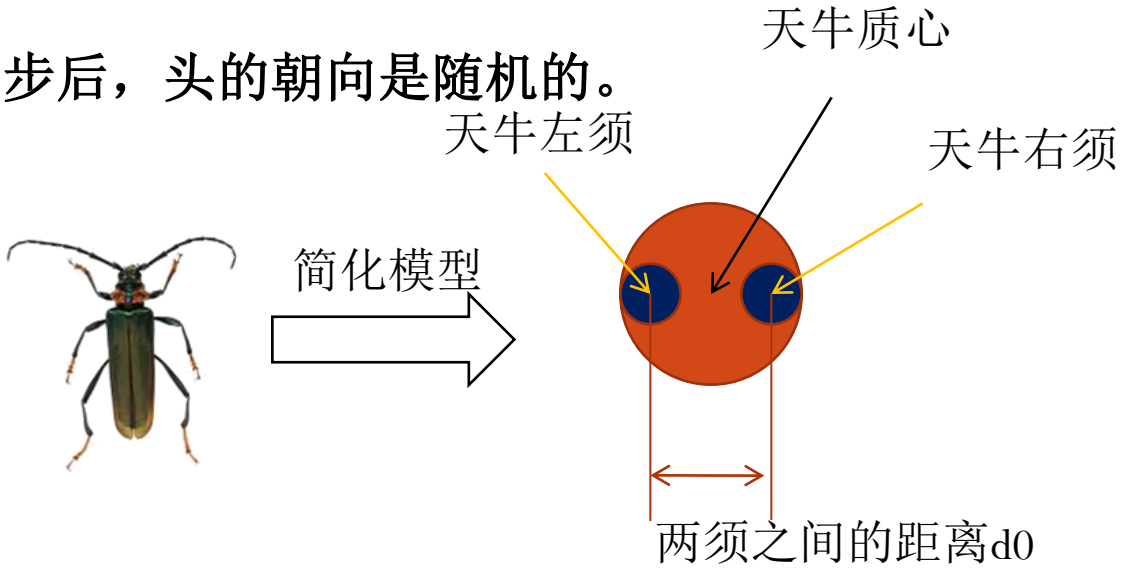
  食物的气味就相当于一个函数，这个函数在三维空间每个点值都不同，天牛两个须可以采集自身附近两点的气味值，天牛的目的是找到全局气味值最大的点。仿照天牛的行为，我们就可以高效的进行函数寻优。

仿生原理

天牛寻找食物图解

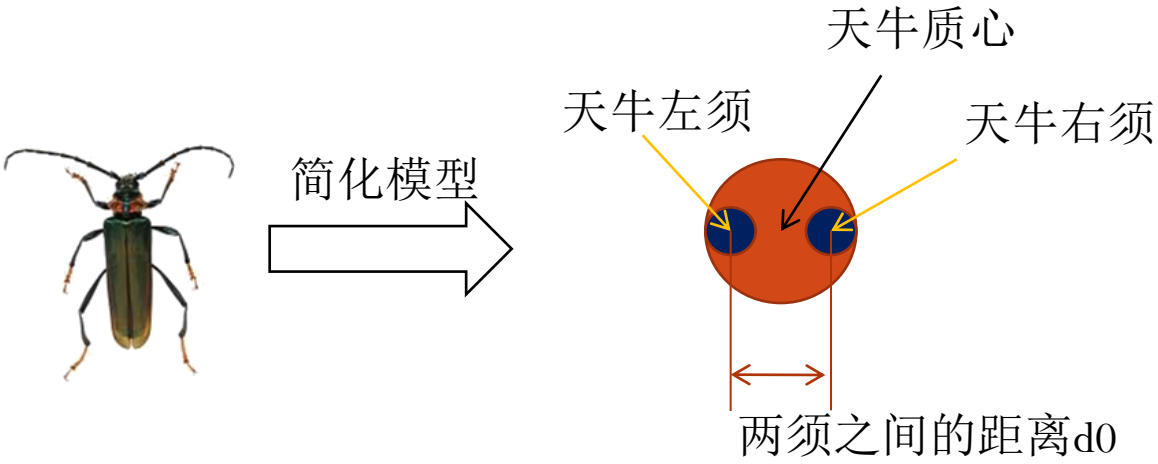
算法

- 天牛在三维空间运动，而天牛须搜索需要对任意维函数都有效才可以。因而， 天牛须搜索是对天牛生物行为在任意维空间的推广。
- 我们采用如下的简化模型假设描述天牛：
  - 1. 天牛左右两须位于质心两边。
  - 2. 天牛步长step与两须之间距离d0的比是个固定常数即step=c*d0其中c是常数。即，大天牛（两须距离长）走大步，小天牛走小步。
  - 3. 天牛飞到下一步后，头的朝向是随机的。



建模（n维空间函数f最小化）

- 第一步：对于一个n维空间的优化问题，我们用xl表示左须坐标，xr表示右须坐标，x表示质心坐标，用d0表示两须之间距离。根据假设3， 天牛头朝向任意，因而从天牛右须指向左须的向量的朝向也是任意的，所以可以产生一个随机向量dir=rands(n,1)来表示它。对此归一化：dir=dir/norm(dir);我们这样可以得到xl-xr=d0*dir； 显然， xl，xr还可以表示成质心的表达式：xl=x+d0*dir/2;xr=x-d0*dir/2.



建模

- **第二步：对于待优化函数f，求取左右两须的值：**

fleft=f(xl); fright=f(xr); 判断两个值大小，

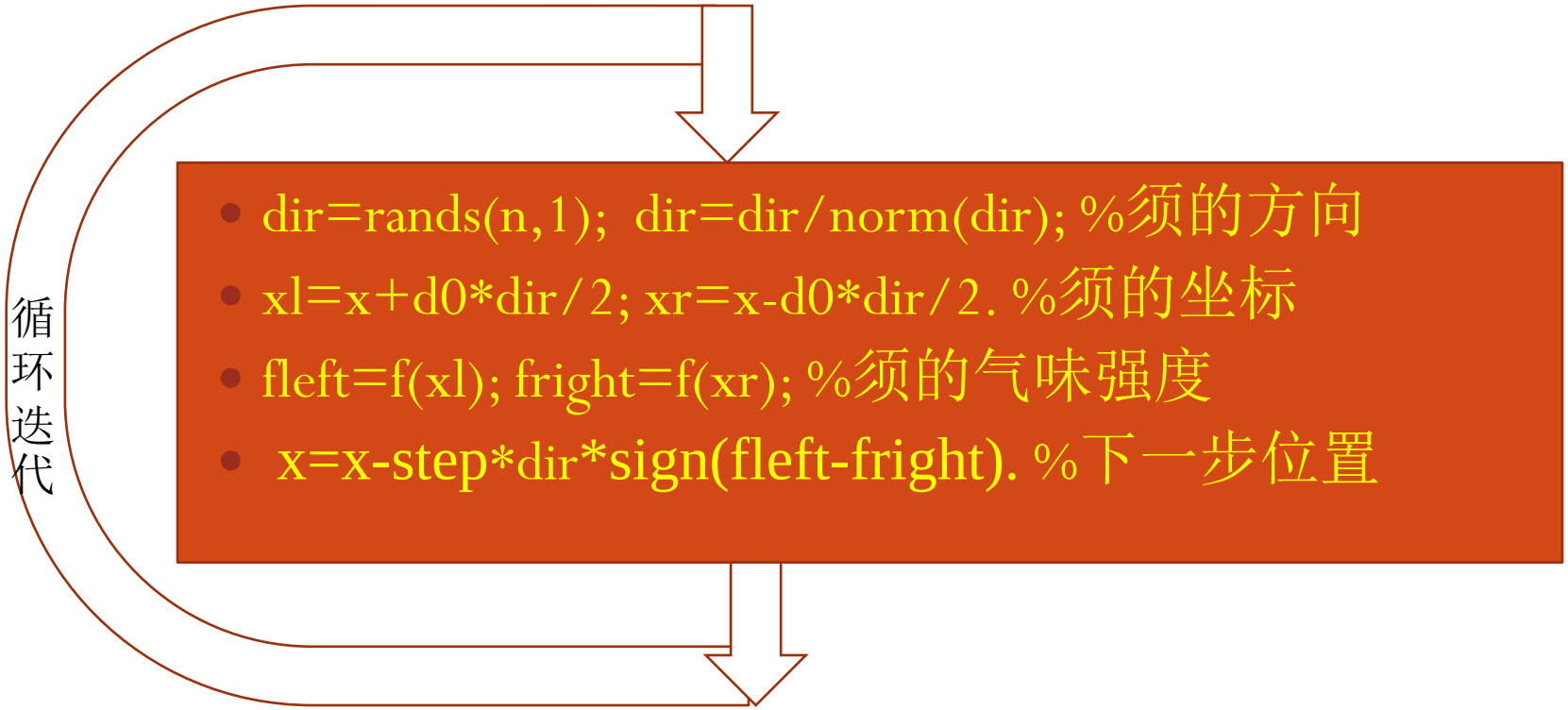
- 如果fleft<fright,为了探寻f的最小值，则天牛向着左须方向行进距离step，即x=x+step*normal(xl-xr)；
- 如果fleft>fright,为了探寻f的最小值，则天牛向着右须方向行进距离step，即x=x-step*normal (xl-xr)；
- **如上两种情况可以采用符号函数sign统一写成：**

x=x-step*normal (xl-xr) *sign(fleft-fright)=x-step***dir** *sign(fleft-fright).

（注：其中normal是归一化函数）

建模

基本步骤就这两步。总结下：



◆几点说明：

1. **核心代码如上，只有4行。**
2. **实用中可以设置可变步长，由于假设2中我们认为step=c*d0其中c是常数，变步长意味着d0=step/c为变化的。**

关于步长

◆关于变步长，推荐如下两种：

1. 每步迭代中采用**step=eta*step**，其中**eta**在**0,1**之间靠近**1**，通常可取**eta=0.95**；
2. 引入新变量**temp**和最终分辨率**step0**，**temp=eta*temp**，**step=temp+step0**。

◆ 关于初始步长：初始步长可以尽可能大，最好与自变量最大长度相当。

Matlab 程序

- function bas()
- clear all
- close all
- %初始化部分
- eta=0.95;
- c=5;%ratio between step and d0
- step=1;%initial step set as the largest input range
- n=100;%iterations
- k=20;%space dimension
- x=rands(k,1);%intial value
- xbest=x;
- fbest=f(xbest);
- fbest_store=fbest;
- x_store=[0;x;fbest];
- display(['0:','xbest=[',num2str(xbest'),'],fbest=',num2str(fbest)])

Matlab 程序

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%迭代部分
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:n
    d0=step/c;
    dir=rands(k,1);
    dir=dir/(eps+norm(dir));
    xleft=x+dir*d0;
    fleft=f(xleft);
    xright=x-dir*d0;
    fright=f(xright);
    x=x-step*dir*sign(fleft-fright);
    f=f(x);
    %%%%%%%%%%%
    if f<fbest
        xbest=x;
        fbest=f;
    end
    %%%%%%%%%%%
```

Matlab 程序

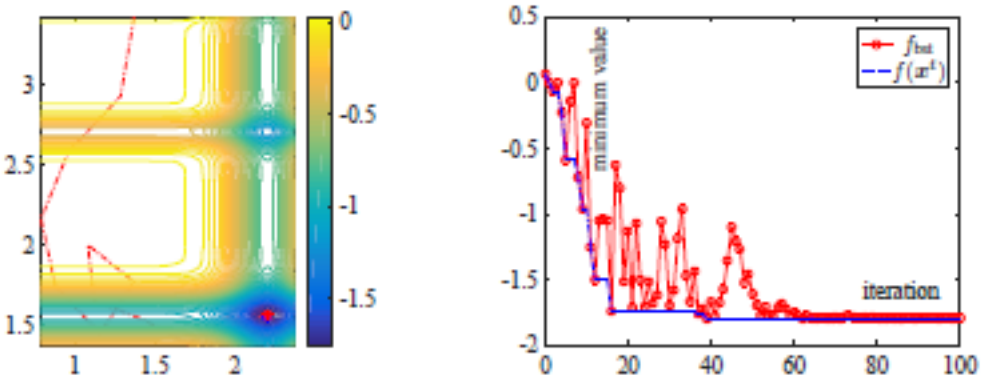
```
 x_store=cat(2,x_store,[i;x;f]);
   fbest_store=[fbest_store;fbest];
   display([num2str(i),':xbest=[',num2str(xbest'),'],fbest=',num2str(fbest)])
   %%%%%%%%%%
   step=step*eta;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%数据显示部分
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1),clf(1),
plot(x_store(1,:),x_store(end,:),'r-o')
hold on,
plot(x_store(1,:),fbest_store,'b-.')
xlabel('iteration')
ylabel('minimum value')
end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%被优化的函数，这部分需要换用你自己的被优化函数
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=f(x)
y=norm(x);
end
```

实测效果

测试1. Michalewicz函数的最小化

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \sin(x_i)[\sin(\frac{ix_i^2}{\pi})]^{2m}, \qquad (6)$$

where $m = 10$ and $i = 1, 2, \cdots$, the minimized value



(a) (b)

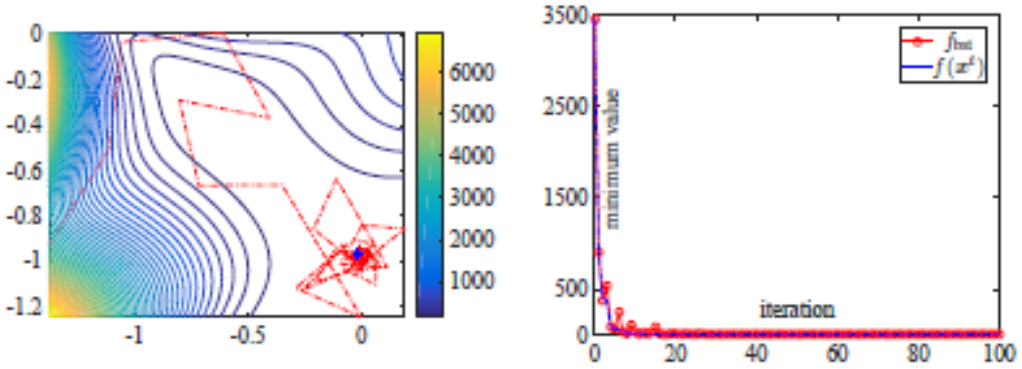
Fig. 2. Performances of the proposed BAS algorithm to search the global optimum of Michalewicz function (6) through 100 iteration steps. (a) The searching trajectory of BAS algorithm when x lays in a 2-D space. (b) Convergence of the minimum value along iteration step t.

实测效果

测试2. Goldstein-Price函数的约束最小化

$$
\begin{aligned}
f(x) \quad = \quad & [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2 \\
& +6x_1x_2+3x_2^2)][30+(2x_1-3X_2)^2(18-32x_1 \\
& +12x_1^2+48x_2-36x_1x_2+27x_2^2)] \qquad (7)
\end{aligned}
$$

where the input domain is usually on the square $\{x|x_i \in [-2,2], i=1,2\}$.



(a) (b)

Fig. 3. Performances of simulation results for the proposed BAS algorithm to solve the Goldstein and Price problem (7) through 100 iteration steps. (a) The searching trajectory of BAS algorithm when x lays in a 2-D space. (b) Convergence of the minimum value along iteration step t.

实测效果总结

1. 运算量非常小，收敛非常快
2. 具有全局寻优能力
3. 代码非常短，容易实现

- 这也告诉我们一个道理：

天牛，或者一般来说甲壳虫，是一种很笨很傻的动物，如果上天公平的话，在残酷的生存竞争中，经过亿万年的进化，这种低等动物早该被淘汰了。但是，事实是他没有被淘汰，因为：

甲壳虫看起来低等，但是从动物行为学上来看，他们一点也不低等，反而很优秀。而正是因此，他们直到今天还活的很好。

参考文献

Xiangyuan Jiang and Shuai Li, BAS: Beetle Antennae Search Algorithm for Optimization Problems, arXiv:1710.10724v1

欢迎使用天牛须搜索算法！相关问题，可以联系我们： jxy@upc.edu.cn 和 lishuai8@gmail.com