# 基于鹰栖息的无模型优化

今天介绍一种新颖的优化算法—鹰栖息优化（eagle perching optimizer，EPO）[1]，该算法模仿了老鹰栖息的天性，由Ameer Tamoor Khan和Shuai Li于2018年提出的，这篇文章也是李老师推荐给我学习的。总体感觉该算法原理简单，实现容易，参数较少，性能较优。

## 灵感

鹰是许多大型食肉鸟类的统称，它们属于鹰科，也是食肉动物。它们的捕猎方式很独特，它们会飞到可能很高的地方，然后开始定位猎物。猎物一旦被跟踪，它们就俯冲下去捕获猎物。老鹰居住在高处，通常是高树、峭壁和大山上。它们有着天生的算法可以帮助追踪最高的地方，首先老鹰先高高地飞向空中，观察地面并采样一些点，在这些采样点中寻找最高位置，然后它们下降到那个点，随着越来越靠近，它们会再次进行采样，进一步明确最高点位置。就是通过这样迭代执行该过程并进行微调以寻找驻留的最高位置。图1展示了它们内置的栖息算法的本质。

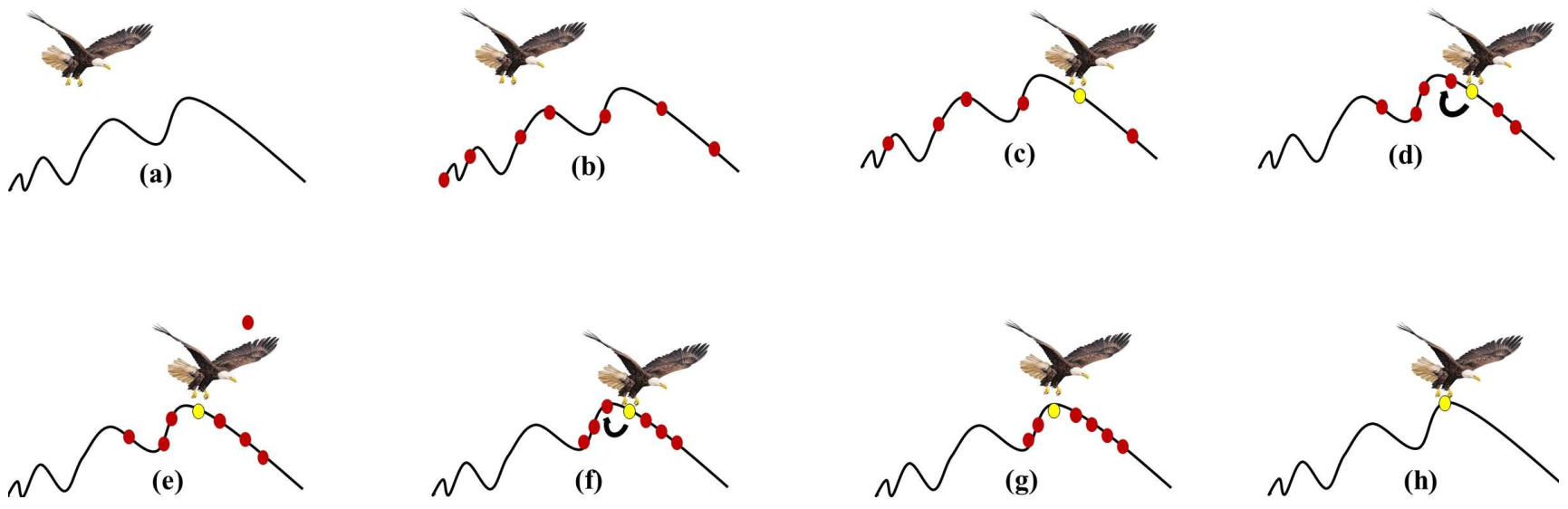


图1 老鹰栖息的天性。（a）老鹰在搜索空间中盘旋。（b）老鹰进行采样并在其中寻找最高点。（c）老鹰到达采样点。（d）老鹰进一步对搜索空间采样，不过此时空间很小，同样寻找最高点。（e~h）老鹰遵循相同的模式，直至到达最高点。

EPO就是利用了这种特性，然后用于获得最优解。在该算法中会有一群老鹰各自寻找最佳的驻留高度，然后再从所有的老鹰中寻找最优解。

## EPO数学表达

老鹰通过一种简单但独特的方式探索地形，在高空飞行时，它通过取样几个点来观察四周，然后向最高点移动，到达最高点后，它再次扫视四周，重复同样的过程，这种重复的过程使鹰能够到达最高点。当老鹰飞向高空俯瞰地面时，由于视野较为开阔，可以看到整体地貌，这实际上为探索的过程，当老鹰飞向采样点时，又会在其周围寻找是否还有更高的点，这就是利用的过程。在函数优化时，可以利用这种思想，先全局搜索再局部利用，从而达到较好的优化，而从探索到利用的转换是随机优化算法的关键所在，EPO算法通过以下实现这种转换：

 （1）

$$

l\_{\text {scale}}=l\_{\text {scale}} \* \text {eta}

$$

其中*l*scale是缩放变量（scaling variable），该值随着迭代的进行会不断降低，使得算法由探索转向利用。eta是一收缩常量，满足0<eta<1，其可以根据最终值分辨率计算得到：

 （2）

$$

e t a=\left(\frac{r e s}{l\_{s c a l e}}\right)^{1 / t\_{s}}

$$

其中*t*s是最大迭代次数，*res*为分辨率范围参数，满足0<*res*<*l*scale以将*eta*限制在0到1之间。注意如果*eta*>1，那么就无法实现从探索到利用的目标，因为随着算法的运行，探索的空间会越来越大。

为了实现更快的优化，文中采用了一群老鹰协同进行空间搜索。

 （3）

$$

X = \left[ {\begin{array}{\*{20}{c}}

{{X\_{1,1}}}&{{X\_{1,2}}}&{{X\_{1,3}}}& \ldots &{{X\_{1,m}}}&{}\\

{{X\_{2,1}}}&{{X\_{2,2}}}&{{X\_{2,3}}}& \ldots &{{X\_{2,m}}}&{}\\

{{X\_{3,1}}}&{{X\_{3,2}}}&{{X\_{3,3}}}& \ldots &{{X\_{3,m}}}&{}\\

{{X\_{3,1}}}&{{X\_{3,2}}}&{{X\_{3,3}}}& \ldots &{{X\_{3,m}}}&{}\\

{{X\_{4,1}}}&{{X\_{4,2}}}&{{X\_{4,3}}}& \ldots &{{X\_{4,m}}}&{}\\

\vdots & \vdots & \vdots &{}& \vdots &{}\\

{{X\_{n,1}}}&{{X\_{n,2}}}&{{X\_{n,3}}}& \ldots &{{X\_{n,m}}}&{}

\end{array}} \right]

$$

其中n表示粒子（老鹰）数量，m为决策空间维度。

为了理解粒子(老鹰)在搜索空间中的移动，考虑一个位于x位置的粒子，它可以自由地、随机地向所有可能的方向移动。在每次迭代时，在当前位置增加一项ΔX（随机值），即X+ΔX，则有：

 （4）

$$

X=X+\Delta X

$$

其中

 （5）

\Delta X = \left[ {\begin{array}{\*{20}{c}}

{{R\_{1,1}}}&{{R\_{1,2}}}&{{R\_{1,3}}}& \ldots &{{R\_{1,m}}}&{}\\

{{R\_{2,1}}}&{{R\_{2,2}}}&{{R\_{2,3}}}& \ldots &{{R\_{2,m}}}&{}\\

{{R\_{3,1}}}&{{R\_{3,2}}}&{{R\_{3,3}}}& \ldots &{{R\_{3,m}}}&{}\\

{{R\_{4,1}}}&{{R\_{4,2}}}&{{R\_{4,3}}}& \ldots &{{R\_{4,m}}}&{}\\

\vdots & \vdots & \vdots & \vdots & \vdots &{}\\

{{R\_{n,1}}}&{{R\_{n,2}}}&{{R\_{n,3}}}& \ldots &{{R\_{n,m}}}&{}

\end{array}} \right]

R∈(0,1)表示随机值，对于X的每一个元素有

 （6）

$$

X\_{i, j}=X\_{i, j}+\Delta X\_{i, j}

$$

其中i代表第i个粒子，j代表当前位置的第j维。

群体中的每一只老鹰通过下式评估采样点的高度：

 （7）

$$

Y\_{i, j}=f\left(X\_{i, j}\right)

$$

对于最小化函数，为了寻找最小的Ymin，定义两个变量YBest和XBest，其进化过程如下：



$$

\begin{aligned} if: f\left(X\_{i, j}\right) &<Y\_{\text {Best}} \\ Y\_{\text {Best}} &=f\left(X\_{i, j}\right) \\ X\_{\text {Best}} &=X\_{i, j} \end{aligned}

$$

但在原文中式（8）应该是写错了。

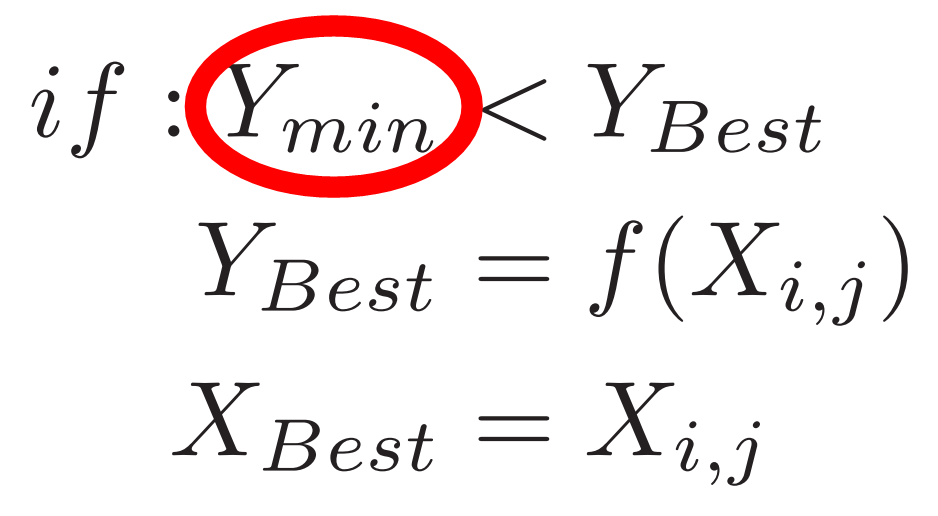


图2 原文中的错误

## EPO算法

通过以上数学描述，可以给出如下的EPO的伪代码。

procedure

初始化所有变量

for <最大迭代次数 do

根据式（5）计算ΔX

根据式（4）计算X

for <总粒子个数> do

根据式（7）评估Y

end for

根据式（7）评估Ymin

通过式（8）比较Ymin

if 式（8）满足 then

执行式（9）和（10）

根据式（1）重新评估lscale

end if

end for

end procedure

## 改进的EPO

为了加速EPO的收敛，文中还对eta的计算进行了一点改进，每次迭代时，eta按下式进行计算：

 （11）

$$

e t a=e t a\_{\max }-t \* \frac{e t a\_{\max }-e t a\_{\min }}{t\_{s}}

$$

其中etamax和etamin分别代表eta的最大值（初始值）和最小值（最终值），该eta可以使得从探索向利用的转换更贱快速和有效。文中还给出了具有可变eta的改进EPO算法。

procedure

初始化所有变量

for <最大迭代次数 do

根据式（5）计算ΔX

根据式（4）计算X

for <总粒子个数> do

根据式（7）评估Y

end for

根据式（7）评估Ymin

通过式（8）比较Ymin

if 式（8）满足 then

执行式（9）和（10）

根据式（1）重新评估lscale

根据式（11）计算eta

end if

end for

end procedure

另一个改进在于，根据式（7）评估完所有粒子的值后，又根据式（12）对解按照从优到劣的顺序进行排序，从中选择n个最好的解，并将这些解对应的坐标存于式（14）中的数组中，再对数组中的元素进行平均（式（15）），得到的结果用于根据式（7）计算函数值。这个改进的意义在于，通过平均值Xavg而不是单个最优Xbest，进一步扩大算法范围。

 （12）

$$

Y\_{\text {sort}}=\left[Y\_{\text {best}\_{1}} \quad Y\_{\text {best}\_{2}} \quad Y\_{\text {best}\_{3}} \quad Y\_{\text {best}\_{4}} \quad \ldots Y\_{\text {worst}}\right]

$$

 （13）

$$

Y\_{\text {sort}}=\left[Y\_{\text {best}\_{1}} \quad Y\_{\text {best}\_{2}} \quad Y\_{\text {best}\_{3}} \quad Y\_{\text {best}\_{4}} \ldots Y\_{\text {best}\_{n}}\right]

$$

 （14）

$$

X\_{\text {sort}}=\left[X\_{\text {best}\_{1}} \quad X\_{\text {best}\_{2}} \quad X\_{\text {best}\_{3}} \quad X\_{\text {best}\_{4}} \ldots X\_{\text {best}\_{n}}\right]

$$

 （15）

$$

X\_{a v g}=\frac{X\_{b e s t\_{1}}+X\_{b e s t\_{2}}+X\_{b e s t\_{3}} \ldots+X\_{b e s t\_{n}}}{n}

$$

文中在单峰函数和多峰函数上比较了两种算法的性能，结论是改进的EPO在精度和标准差上更优。

## 算法对比与应用

文中比较了EPO与其他元启发式算法（蚁狮优化-ALO、蜻蜓优化-DA、粒子群-PSO、遗传算法-GA、花朵授粉算法-FPA、物质状态搜索算法-SMA、布谷鸟搜索-CS、蝙蝠算法-BA和萤火虫算法-FA）的性能，结果表明在单峰和多峰函数上都能得到很高的精度。

在约束优化中，主要用EPO解决了悬臂梁设计问题、三杆桁架设计问题和齿轮系设计问题。

## 结论

其实论文中还给出了EPO算法的收敛性证明，以及一些关键参数对算法性能的影响，这里我没有详细地讲述，感兴趣的读者可以点击“阅读全文”查看原文。我说说一下自己的感受吧，论文中主要采用了一种从探索到利用的控制策略，是一个由粗调到细调的过程，但是有一点论文中没有指出，参数lscale到底是如何控制变量的取值空间的，是影响ΔX吗还是什么，相关公式和算法中都没有提及，最后的验证部分本质上还都是数值优化问题，这与算法对比部分有点重复，个人感觉如果可以解决一些组合优化问题就更好了。

1. Tamoor Khan, A., et al. *Model-Free Optimization Using Eagle Perching Optimizer*. arXiv e-prints, 2018.