1 算法收敛的必要条件

传统意义上的“位置迭代类算法”中搜索位置的更新方程可以描述为如下式(1)所示：

$$X_{(i)}^{(n+1)} = X_{(i)}^{(n)} + Pace_{(i)}^{(n)} * Direction_{(i)}^{(n)} \quad (1)$$

这里$X_{(i)}^{(n)}$代指的是第 i 个搜索个体在第 n 次搜索的位置，$Pace_{(i)}^{(n)}$代表第 i 个个体在第 n 次位置更新时的前进步长，$Direction_{(i)}^{(n)}$代表第 i 个个体在第 n 次位置更新时的前进方向(单位向量)。(1)式将更新的量考虑成一种向量数乘的形式，即:长度*方向。

不难得出：“位置迭代类”算法的性能差异主要来源于于$Pace_{(i)}^{(n)}$和$Direction_{(i)}^{(n)}$的更新方式。

在正式进入具体的算法讨论之前，我们可以笼统地讲，任何一种“迭代类算法”均需要在更新$Pace_{(i)}^{(n)}$和$Direction_{(i)}^{(n)}$的过程中引入随机过程。

首先，考虑“超启发式算法”的应用背景：

1.无法提供目标函数的解析形式表达式，但可以根据位置的输入，获取对应的目标函数值。

2.可以提供目标函数的解析形式表达式，但形式过于复杂，无法获取相应位置的梯度值。

结合上面的两条，不难得出，“超启发式算法”本质上是一种“推测估计+假设搜索”的过程。具体来讲，就是说：在不获取全空间的目标函数值信息的前提下，对目标区域进行有限的、有选择性的搜索，进而通过历史搜索区域所反映出的信息，推测出“理想最优解”可能出现的区域，通过不断缩小和锁定搜索范围，最后在一定精度范围内确定“理想最优解”的位置。

这也就是说，仅仅通过有限次采样就认为已经了解了实际的全区间上的解的大致分布(这里的分布仅仅考虑目标函数的极值，并不是建立在全搜索区域上的目标函数的解空间)。这里可以将每一步的更新解释为：

1.Always believe the next will be better.

2.Always not sure the next is the best.

所以搜索的关键就在于：如何有效地扩大搜索区域；如何提高被搜索点的代

表性。

分析进行到这里，就可以很容易的得出结论：在搜索方程(1)中必须引入足够的随机项，才能保证搜索区域的全面性。

这里我们可以利用逻辑表达式的形式很轻松的解释这种随机项引入的必要性。

这里假设算法 A 在搜索空间 S 上遍历的轨迹点 D 的集合可以表示为 S 中所有符合算法 A 中的确定条件 C(这里的“确定”是指表达式不具有任何的随机内容，所有组成均为确定成分)的点，即如式（2）。

$$D = \{d \in S | \forall p \in C\} \tag{2}$$

此时我们可以看到在所有的被搜索点组成的集合 P 中，所有不符合条件 C 的点组成的集合就成为了“弃子”，考虑前面提到的“超启发式算法”的应用背景，在无法掌握被搜索空间内所有点对应的目标函数值的前提下，诸如上式(2)的搜索方案实际在搜索进行之前就已经将搜索区域中的点 d 被搜索到的概率分成了如下两类非 1 即 0 的情况，如下式(3)所示。

$$P(d\ will\ be\ searched) = \begin{cases} 1 & if\ d \in D \\ 0 & if\ d \notin D \end{cases} \tag{3}$$

而如果适当的在 C 中引入随机项之后，这时就相当于利用概率分布分割了搜索区域。即在每一次搜索位置确定之前，任何位置被搜索到的可能性均是非 0 的。

这里我们仅考虑在本算法中设计的机制，在位置更新方程(1)中，我们可以在 $Direction_{(i)}^{(n)}$ 中适当的加入随机项。

考虑到当前位置$X_{(i)}^{(n)}$的更新仅同$X_{(i)}^{(n-1)}$有关，因此可以将这里的位置更新过程视为一种马尔科夫过程。因此对于“位置迭代类算法”，有定理 1。

定理 1：

“位置迭代类算法”收敛的必要条件为：式(1)的位置更新过程为一个吸收态的马尔可夫过程。

但仅仅满足定理 1 的条件，最终输出解的全局最优性是无法保证的，也就是说，在这种条件下，只能保证收敛到某一个值。而这个值的具体价值并不能直接保证。而且吸收态这个大前提也侧面表明了“位置迭代类算法”的搜索范围一定会在有限步之内被锁定到一定的区域，也就是说，如果在被“某一解值”吸收之

前未完成足够大范围的搜索，算法的全局性即无法保证。因此，这里的吸收性和全局最优实际是算法设计过程中的一对互相矛盾的因素。

2 搜索方向的更新机制

结合上述分析可以看出，跳出机制的关键在于$Direction_{(i)}^{(n)}$，因为$Pace_{(i)}^{(n)}$决定了更新跨度，而$Direction_{(i)}^{(n)}$才是能够改变搜索位置的空间分布的关键。如果设计不当，就会出现屏蔽了大半个搜索区域的“更新屏蔽”问题。为了避免出现如同"PSO" 在高维搜索空间中的“更新屏蔽”问题。应该在$Direction_{(i)}^{(n)}$的更新过程中，根据待搜索区域的维数适当地增加信息的量。

这里先对“更新屏蔽”这种极具个性化的提法做一个说明，我们都知道:如果想要得到一个 Dim 维向量在线性向量空间内的 Dim 维线性分解，需要 Dim 个线性无关的 Dim 维向量作为基底。而"PSO"算法的更新式：

$$V_{(i)}^{(n+1)} = W * V_{(i)}^{(n)} + C1 * R1 * (Pbest_{(i)}^{(n)} - X_{(i)}^{(n)}) + C2 * R2 * (Gbest_{(i)}^{(n)} - X_{(i)}^{(n)}) \quad (4)$$

如果将上式中的$V_{(i)}^{(n+1)}$视为(1) 式中的$Pace_{(i)}^{(n)} * Direction_{(i)}^{(n)}$，即可以理解成仅仅选取了 2 个 Dim 维向量作为基底，通过随机加权求和的方式得到下一步的搜索位置更新量(这里仅将新引入的后两项作为更新量考虑，认为速度量是继承值，不是新生成的)。

这就意味着对于低于 2 维的问题，该更新方式造成了信息的冗余(因为这时的分量线性相关)。而对于高维问题，仅仅二维的方向更新相当于主观屏蔽了位于基向量张成的平面外的其他方向在下一步被搜索的可能性。

同时其随机函数的取值范围为[0,1)，这步操作本身就造成了极大的主观导向性。因为它相当于屏蔽了当前位置同历史最优位置以及全局最优位置的两向量所夹锐角之外的所有的值，如图 1 中灰色区域所示。

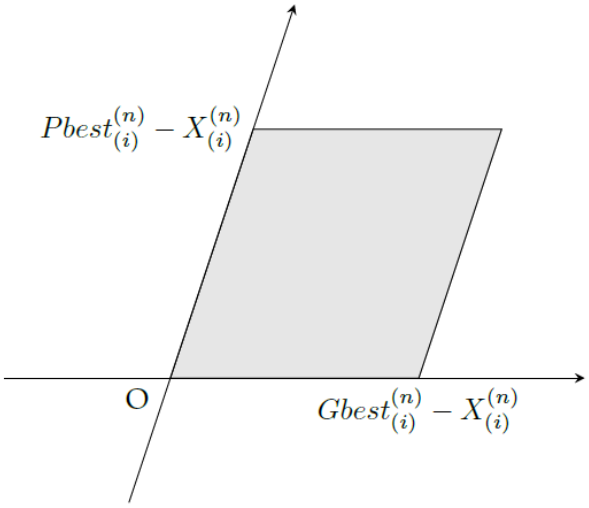

图 1 PSO 算法$V_{(i)}^{(n)}$更新方式示意图

在正式进入算法的讲述之前，先整理一下目前在算法进行到第 n 步时，我们可以收集到的信息：共计 n 次的历史搜索位置及其目标函数值。

基本信息就是这些。那么是不是所有的信息都是有用的，或者说可以引入算式(1)成为$Pace_{(i)}^{(n)} * Direction_{(i)}^{(n)}$一项或一部分？

很显然一股脑全扔到算式里是不现实的。首先这样会将某些极差的值引入其中，产生收敛速度的衰减。其次对于某些目标函数解空间单峰或少峰的情况，过多的引入“非主要因素”还十分容易对下一轮的搜索造成错误导向，出现信息干扰和冗余。

这里考虑目标函数中的极端情况，比如对一些目标函数极其复杂无法用初等函数的组合表示，或者表示起来也极其复杂，计算代价极大的情况(例如：本算法中中的优化问题)。这时，就需要尽量少的调用目标函数，尽可能地从历史搜索位置积累并发掘出更多的信息。如果从这种角度考虑的话，前述方式中既增加了单步位置更新式中的计算量，又因为收敛的迟缓而增加了目标函数的调用次数的方式是极不可取的。

同时在极高维的情况下，将每次搜索的解的信息悉数存储起来的方式将会占用内存中的极大空间。因此，为了增加被记录数据的典型性。许多算法都采用了记录个体历史最优解的方法，这种方式具有经济性和代表性。所以，此处对于历史信息的处理也采用这种记录个体历史最优解的方法。

这里将下一步的更新向量$Pace_{(i)}^{(n)} * Direction_{(i)}^{(n)}$分解为两个部分：步长$Pace_{(i)}^{(n)}$可以将其理解为更新半径)，$Direction_{(i)}^{(n)}$ (单位向量)更新方向，为了尽可能地实现全局性最需要避免的就是“将其可能的分布范围大面积地砍去”的这种直接找两个向量做个正向加权这种办法。

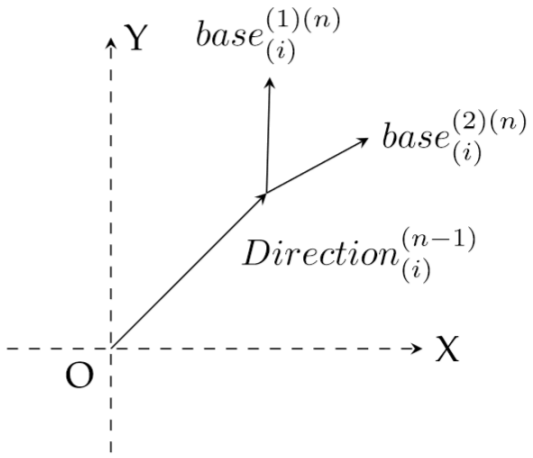

图 2 向量的更新示意图

正如前面提到的，如果想要得到一个 Dim 维向量在 Dim 维线性向量空间内的 Dim 维线性分解，需要 Dim 个线性无关的 Dim 维向量作为基底，这里的待搜索区域可以视为一个线性向量空间的子集合(不是子空间，因为运算不一定封闭)，那么此时需要的就是从个体历史最优解中总结出 Dim 个线性无关的单位向量，作为(在该待搜索区域最小包络线性向量空间下的)基底，进而生成新的更新方向$Direction_{(i)}^{(n)}$ (单位向量)。

这里用$Hbest_{(i)}^{(n)}$来表示第 i 个个体在前 n 步搜索中所经过的历史最优位置，$Hbestv_{(i)}^{(n)}$为其对应的目标函数值。利用$Hbest_{(i)}^{(n)}$中非$X_{(i)}^{(n)}$的量中，最优的 *Dim* 个值同$X_{(i)}^{(n)}$的差向量归一化之后的结果作更新时使用的基底，如下式(4)所示。

$$base_{(i)}^{(d)(n)} = normalize\left(Hbest_{(i)}^{(d)(n)} - X_{(i)}^{(n)}\right) \quad Hbest_{(i)}^{(d)(n)} \neq X_{(i)}^{(n)} \tag{4}$$

$Direction_{(i)}^{(n)}$的变化量$\delta_{(i)}^{(n)}$即可由图中的$base_{(i)}^{(1)(n)}$和$base_{(i)}^{(2)(n)}$的线性组合表示而成。这样的话$\delta_{(i)}^{(n)}$的端点就可以用在二维平面上单位圆上的点代替，通过改变单位圆上的点的概率分布，即可以调整$\delta_{(i)}^{(n)}$的概率分布，这里采用如下式(5)进

行表示：

$$\delta_{(i)}^{(n)} = \sum_{d=i}^{Dim} Hbelief_{(i)}^{(d)(n)} * base_{(i)}^{(d)(n)} \tag{5}$$

上式中的$Hbelief_{(i)}^{(d)(n)}$为该方向的置信系数，采用如下式(6)的生成方式。

$$Hbelief_{(i)}^{(d)(n)} = 2 * Random_{(i)}^{(d)(n)}(1,1) - Bias^{(n)} * Chaos_{(i)}^{(d)(n)} \tag{6}$$

式中 $Random_{(i)}^{(d)(n)}(1,1)$——[0,1)上的均匀分布的随机数;

$Bias^{(n)}$——第 n 次更新过程中的置信系数的整体偏置量;

$Chaos_{(i)}^{(d)(n)}$——第 n 次更新过程中的第 d 位上的偏置量;

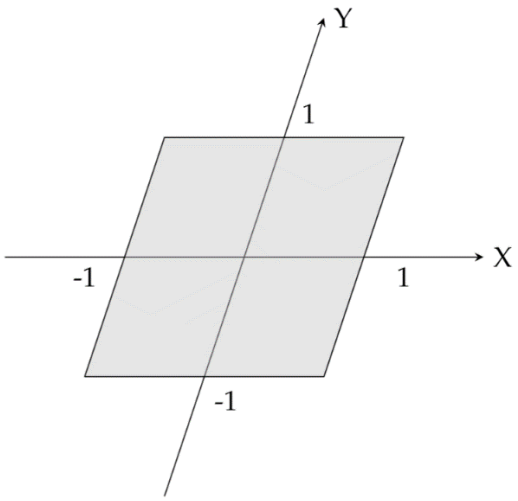

图 3 $Bias^{(n)} * Chaos_{(i)}^{(d)(n)} = 1$时，向量$\delta_{(i)}^{(n)}$端点分布示意图

此处先以二维向量为例，对上式(6)中各项的组成及相应的作用进行说明。这里直接建立以$base_{(i)}^{(1)(n)}$和$base_{(i)}^{(2)(n)}$两个单位向量为基底的斜角坐标系如上图 3,我们先暂且不管(6)后两个参数的具体形式，仅将其作为整体，对其所表现的数学含义进行讨论。那么当$Bias^{(n)} * Chaos_{(i)}^{(d)(n)} = 1$时，$Hbelief_{(i)}^{(d)(n)}$为分布在[-1,1)上的服从均匀分布的随机数。相应的，$Direction_{(i)}^{(n)}$的变化量向量$\delta_{(i)}^{(n)}$的端点可能区域即可以表示为由图 3 中阴影部分所示的区域表示。

由于后续过程中会对$\delta_{(i)}^{(n)}$进行归一化操作，因此这里更需要着重考虑的是生成指向某一固定方向的$\delta_{(i)}^{(n)}$向量簇的概率。由于(5)式中各个轴的置信系数在生成过程中相互独立，因此，可以在类似图 3 的 Dim 维斜角坐标系下，得出δ的如下式(7)的概率密度函数。

$$Pdf\left(\delta^{(1)},\delta^{(2)},\ldots,\delta^{(Dim)}\right)=\begin{cases}0.5^{Dim} & if\ \delta\in D\\ 0 & else\end{cases} \tag{7}$$

由(7)式可以得出，分布在图 3 中阴影区域 D 上的每一点的概率密度相同，那么此时，就可以用对应象限(卦限或区域)的面积(或体积)来代表$\delta^{(n)}$出现在对应区域的概率。同时对于$\delta^{(n)}$出现在某一方向上的概率，可以用其相应的射线(端点同$\delta^{(n)}$重合，延展方向同$\delta^{(n)}$一致)截阴影区域的长度来表示。

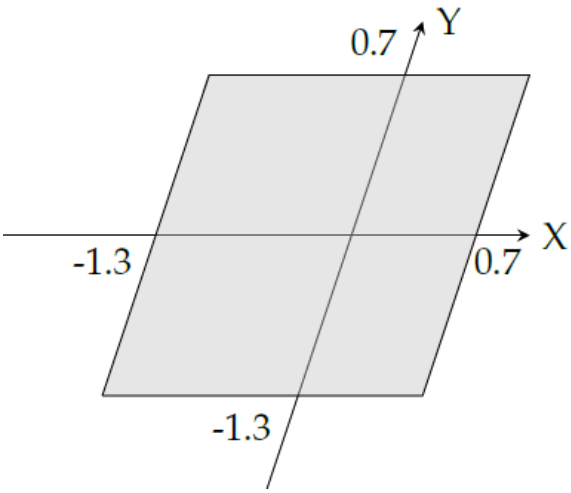

图 4 $Bias^{(n)}*Chaos_{(i)}^{(d)(n)}=1.3$时，向量$\delta_{(i)}^{(n)}$端点分布示意图

基于上述分析，不难得出，只要调整(6)式中$Bias^{(n)}*Chaos_{(i)}^{(d)(n)}$的值，就可以改变对应的$Hbelief_{(i)}^{(d)(n)}$这的分布情况，相应的就可以改变$\delta^{(n)}$的概率密度分布图。如上图 4，即为当$Bias^{(n)}*Chaos_{(i)}^{(d)(n)}=1.3$时，向量$\delta_{(i)}^{(n)}$端点的分布情况。可以清楚的看到，这时指向历史最优解群的第一象限内的阴影部分区域的面积明显减小，这时的算法趋向于向该方向相反的方向运动，有利于搜索区域的扩大。

这样的机制有利于鞍点位置上的搜索，如图 5 所示：

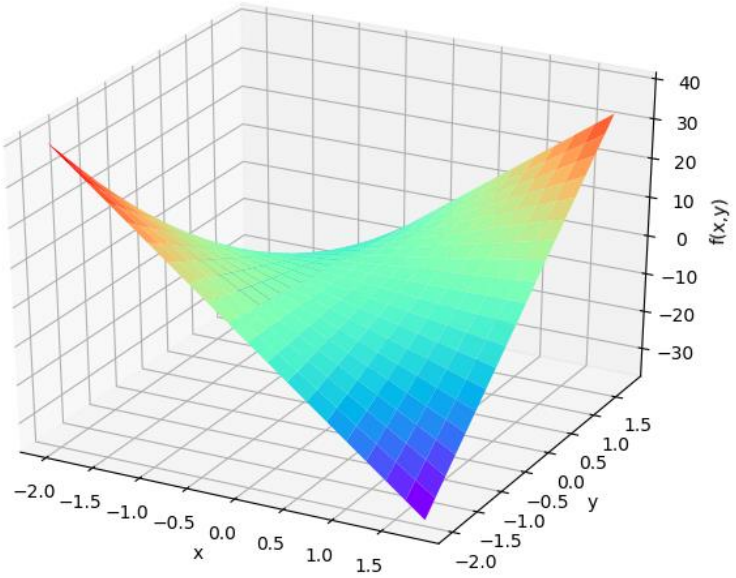

图 5 马鞍面示意图，曲面方程为Z = XY

如果某一搜索点的当前位置恰好位于鞍点附近，而大多数历史搜索点均位于鞍面一侧，此时由式(5)，(6)组成的搜索方向更新机制依然可以保证鞍面的对策的区域可以被搜索到。

下面着重讨论$Bias^{(n)}$和$Chaos_{(i)}^{(d)(n)}$两个参数的形式和作用。

考虑到搜索初期，由于信息采集较少，指向当前的所谓“最优位置”的方向并不一定就是最终收敛解的相对方向。因此，这时在对于$Base_{(i)}^{(d)(n)}$作线性合成求$\delta_{(i)}^{(n)}$时，$Hbelief_{(i)}^{(d)(n)}$中的偏置量仍需要引入一定范围的随机数。

这里将该偏置量分为两部分$Bias^{(n)}$用于控制第 n 次信息更新时，偏置量的大小，$Chaos_{(i)}^{(d)(n)}$用于控制第 n 次更新时第 d 位上的偏置的范围。

结合实际搜索过程中“最优位置的导向性和可参考性逐渐增强”的特点，需要将$Bias^{(n)}$设计成一个可以随时间变化最终逐渐收敛的函数，由于其与后面的搜索步长更新机制有着紧密的关系，因此其具体的内涵将在下一节进行详述。

在不清楚具体的目标函数组成的前提下，每一个$Base_{(i)}^{(d)(n)}$向量的指向性和可信度实际上都是不确定的，因此这里将统一通过$Chaos_{(i)}^{(d)(n)}$项进行合理推测。

由于分布的位置和目前的目标函数值的排序并不能直接决定该搜索方向的优先采用性，因此在实际过程中，应该将所谓的置信偏置值设置为一定范围内的随机数。

具体考虑到实际的搜索过程中，如果采用服从正态分布的随机数的话，导向性和主观性会过强，因此这里$Chaos_{(i)}^{(d)(n)}$依旧采用服从均匀分布的随机数，随机数的分布区间一般采取相对于 1 对称，分布的邻域半径在[0.2,0.3]之间，这里推荐采用分布于[0.8,1.2]上的服从均匀分布的随机数作为$Chaos_{(i)}^{(d)(n)}$的构成。

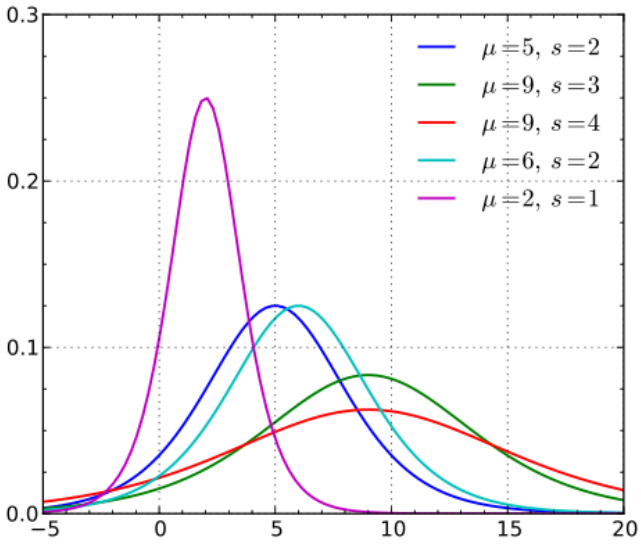

图 6 logistic 分布概率密度曲线示意图

但考虑到搜索初期，由于采集的信息较少，这时更新产生的$\delta^{(n)}$的置信度不高因此在这里引入另一个随机置信参数$Gbelief_{(i)}^{(n)}$。

此处的$Gbelief_{(i)}^{(n)}$采用服从 logistic 分布的随机数，这里考虑到 logistic 分布的峰度(Kurtosis)较大，因此更新时既能保证中心集中，又能使区域不至于过分居中。Logistic 分布的详细说明见下式(8)

$$Pdf(x,\mu,s)=\frac{exp\left(-\frac{x-\mu}{s}\right)}{s*\left(1+exp\left(-\frac{x-\mu}{s}\right)\right)^2}$$

$$Variance=\frac{s^2\pi^2}{3}$$

$$Mean=\mu \tag{8}$$

$$Skewness=0$$

$$Kurtosis=1.2$$

这里通过设置 Trust 函数(即均值μ)，进而改变随着算法的进行，$Gbelief_{(i)}^{(n)}$的分布情况。考虑到位置更新量$\delta_{(i)}^{(n)}$的置信度会逐渐提升，结合实际情况将 Trust 函数也设置为 Type1(凸函数)，Type2(线性函数)，Type3(凹函数)三种递增函数，

具体的形式如下式(9)所示：

$$
\begin{aligned}
\textbf{Type1:}\begin{cases} A & = (T_{[end]} - T_{[init]})/Cycle^{0.5} \\ B & = T_{[end]} \\ Trust^{(n)} & = A * n^{0.5} + B \end{cases} \\
\textbf{Type2:}\begin{cases} A & = (T_{[end]} - T_{[init]})/Cycle \\ B & = T_{[end]} \\ Trust^{(n)} & = A * n + B \end{cases} \\
\textbf{Type3:}\begin{cases} A & = (T_{[end]} - T_{[init]})/Cycle^{2} \\ B & = T_{[end]} \\ Trust^{(n)} & = A * n^{2} + B \end{cases}
\end{aligned} \tag{9}
$$

3 搜索步长的更新机制

如果说方向确定的是搜索区域，那么步长确定的就是搜索范围。可以很直接的说：在不清楚目标函数在目标区域的对应映射下的拓扑空间的前提下，任何一种算法都无法保证能够得到全局最优的解。

这里能做的只是尽可能地提高搜索范围，并且假设搜索的区域可以被碎片化成为一个个相对独立的单峰结构。优化算法的目标即为：首先锁定一个单峰的碎片区域，然后在区域内进行精细搜索，直至达到所需精度。

在搜索步长上，应该考虑结合当前搜索位置的分布信息，进行调整。由于每一步搜索过程中所得到的位置信息及相应的目标函数值的排序并不代表绝对的可信度，因此，仅仅通过同指定的几个所谓“最优点”相对距离归纳出的步长更新机制显然欠妥。

在本节主要考虑设计一种基于对当前历史记录中“最优位置”分布情况的评估指标以及上一步的更新步长等信息的推测更新机制，这样就可以更广泛的适应在多变的搜索环境，同时也避免了在“群体性算法”中由于搜索点分布过于分散而导致的后期收敛困难的情况。

首先考虑对当前采集的“历史最优位置”的分布评估指标的建立。

该分布评估指标应满足：1.可以一定程度上反应每一个分量上的分布离散程度 2.尽可能多的反映绝大多数个体的分布情况，忽略边缘个体。

在本算法中采用先对$Hbest^{(n)}$的每一个维度的分量组成的数组求标准差，再对各个维度上的标准差组成的 Dim 维向量取 2-范数。即如下式(10)所示：

$$std^{(d)}\left(Hbest^{(d)(n)}\right)=\sqrt{\frac{1}{M-1}\sum_{m=1}^{M}Hbest^{(d)(n)(m)}}$$

$$Radius^{(n)}=\sqrt{\sum_{d=1}^{Dim}[std^{(d)}(Hbest^{(d)(n)})]^2} \tag{10}$$

式中 $Hbest^{(d)(n)(m)}$——第 n 次更新后 *Hbest* 上的第 d 维分量中的第 m 个值

这里我们选取在二维平面上各个分量互相独立地服从均匀分布，正态分布，logistic 分布，采用如上式(10)的方式，分别求取 Radius，结果如下图 7，8，9 所示。

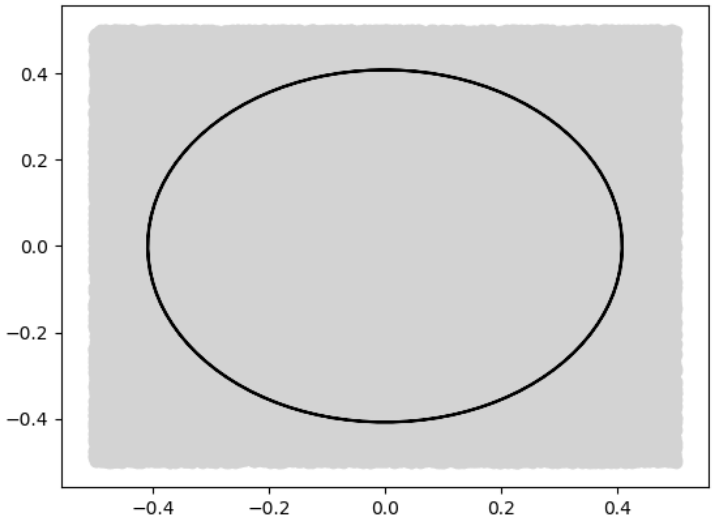

图 7 [-1,1]上的均匀分布，Radius= 0.8143

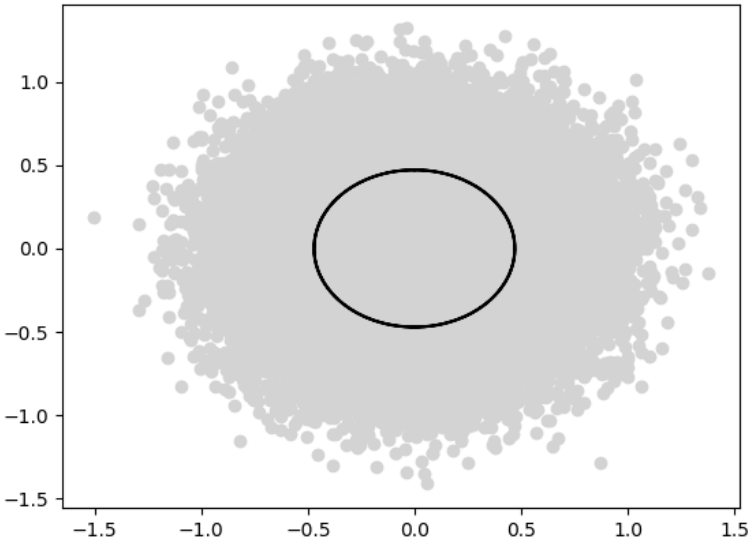

图 8 μ=0、σ=1/3 时的正态分布，Radius= 0.4736

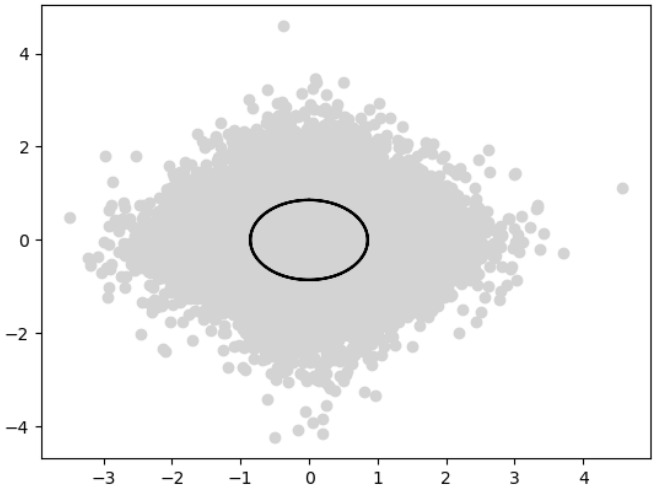

图 9 μ=0、s=1/3 时的 logistic 分布，Radius= 0.8585

可以从上面的三个例子中看到所求得的 Radius 的值同每一个分量上的方差的组成的向量的 2-范数一致。

方差的数学意义是某一维度的分量上的值相对于该分量上值得中心的离散程度的度量，而范数既是一种对空间的度量，也同时可以视为一种对向量各个分量的一种平均化。

考虑到这里要尽可能地反应平均状态，因此选择了 2-范数，具体原因详见第 3 章，这里不再重复。

下面就步长$Pace^{(n)}$的更新机制进行讨论，在此说明本小节在以下公式推导

和证明过程中均采用$p^{(n)}$代替$Pace^{(n)}$，$r^{(n)}$代替$Radius^{(n)}$，这里的$r^{(n)}$是指第 n 次位置更新之后所得到的$Hbest^{(n)}$生成的 *Radius*，$p^{(n)}$是指第 *n* 次更新时采用的步长 *Pace*。

这里采用的步长更新机制如下式(11)所示:

$$p^{(n)} = \frac{A * r^{(n-1)}}{1 + exp\left(\frac{p^{(n-1)}}{r^{(n-1)} + \epsilon} - 1\right)} \tag{11}$$

式中 A——常数

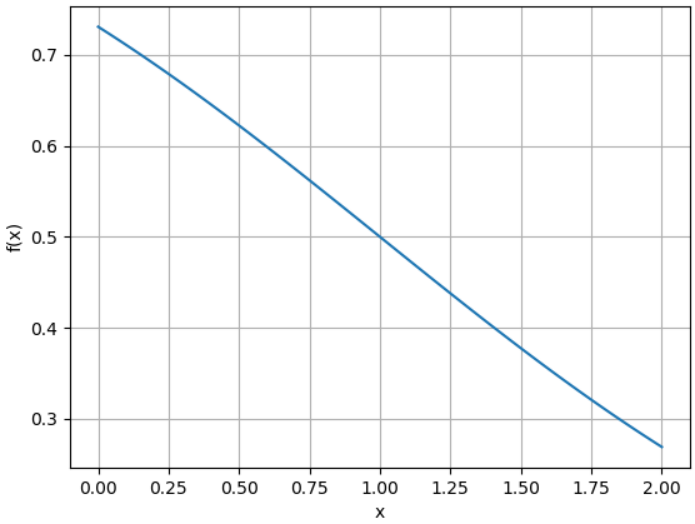

图 10 在[0,2]上的函数$1/(1 + exp(x - 1))$的图像

可以看到函数$1/(1 + exp(x - 1))$的图像如上图 10 所示。这里通过系数 A 来控制$Pace^{(n)}$的增长幅度。

在正式进入式(11)的参数讨论之前，先对该更新方式的稳定性的必要条件进行讨论。

定理 2：

(11)稳定的必要条件是，当$t > C, \exists C \in (0, +\infty)$时，满足$p^{(n)} = \alpha * r^{(n-1)}$

不难得出，在搜索的最后，要时刻控制$p^{(n)}$同$r^{(n-1)}$成一定比例。这样才可以维持 *Radius* 的值在一个定值附近。

由定理 2，很容易推出式(12)

$$\Delta p^{(n)} = p^{(n)} - p^{(n)}$$

$$\Delta r^{(n)} = r^{(n)} - r^{(n-1)} \tag{12}$$

将(12)反代入(11)中可以得到式(13)

$$\alpha = \frac{A}{1 + exp\left(\frac{p^{(n)} - \Delta\, p^{(n)}}{r^{(n-1)}} - 1\right)} \tag{13}$$

考虑到$\lim_{n\to\infty} \Delta p^{(n)} = 0$，因此式(13)可以得出如下式(14)

$$A = \alpha * (1 + exp(\alpha - 1)) \tag{14}$$

上式(14)可以标明在式(11)中的常系数 A 同α 直接相关。这里对α 的取值作详细讨论。

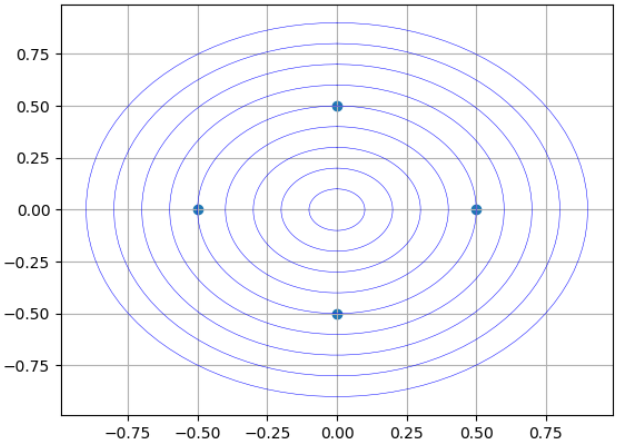

图 11 ABCD四点(图中实点)初始分布示意图

这里先以$x^2 + y^2 = z$这个曲面为例，讨论其在(0,0)附近的等高线图如上图 11 所示。这里我们考虑当上述的四个点位于同一等高线附近时，如上图 11 中的四个点。

如果希望$Hbest$在下一步能够得到更新解，那么下一步的更新步长就不能大于$2 * Radius$，因为如果大于$2 * Radius$，那么此时下一次的搜索位置将永远无法位于该等高线内侧，因为此时下一次的搜索的点所在的圆均内含该等高线，如图 12 所示($\alpha = 2.5$)，中间的粗红线为元 ABCD 四点分布的等高线，其余四个细线圆为 ABCD 在下一步中可能更新的位置。此时，很明显，算法将无法在下一步的搜索中获得更新值。

但考虑到实际情况中各个点的分布位置不可能如此均匀，因此这里我们可以得出能够使算法得到良好更新条件是：$0 < \alpha < 2 + C$(C为某一大于 0 的常数)

这里我们以二维平面为例，假设存在 ABCD 四个点，此四点均布于圆上，如

下图 13 所示。

这里考虑$0 < \alpha < 2$的情况，此时，更新后的四个点可能为于图 14 中的对应的四个圆上。

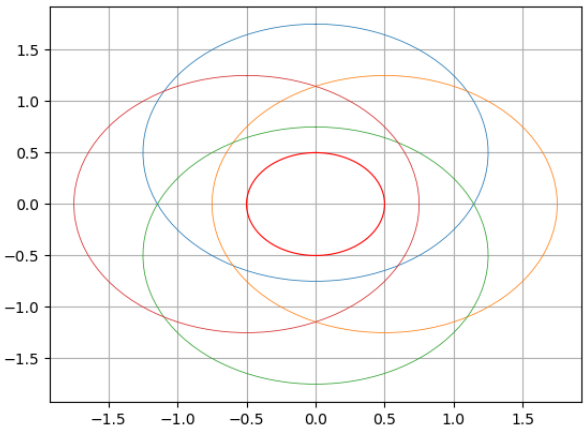

图 12 ABCD四点更新位置示意图

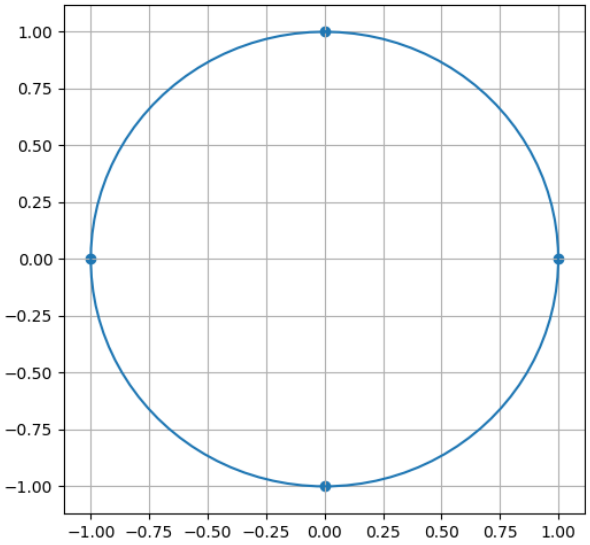

图 13 ABCD四点初始位置示意图

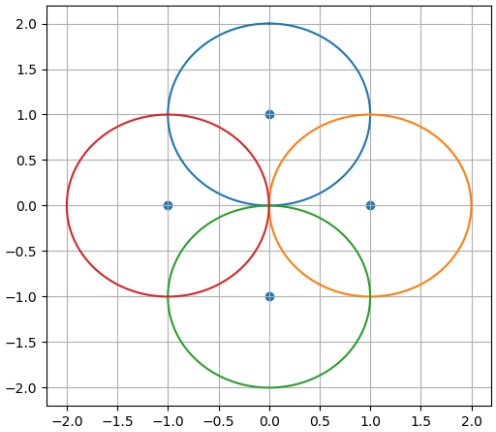

图 14 ABCD四点更新位置示意图

我们可以得出此时$r^{(n)} \le (1+\alpha) * r^{(n-1)}$，结合式(11)(14)，有(15)：

$$p^{(n+1)} = \frac{\alpha * (1 + exp(\alpha - 1)) * r^{(n)}}{1 + exp\left(\frac{p^{(n)}}{r^{(n)} + \epsilon} - 1\right)} \tag{15}$$

进一步的，由于$\alpha = p^{(n+1)}/r^{(n)}$，于是有(16)：

$$\alpha = \frac{\alpha * (1 + exp(\alpha - 1))}{1 + exp\left(\frac{\alpha}{1+\alpha} - 1\right)} \tag{16}$$

考虑$\alpha \le 2$，结合(16)可以得出如下式(17)的函数式

$$f(\alpha) = 2 * \left(1 + exp\left(\frac{\alpha}{1+\alpha} - 1\right)\right) - \alpha * (1 + exp(\alpha - 1)) \tag{17}$$

问题即转化为寻找$f(\alpha) > 0$的α 值的问题，$f(\alpha)$在[1.1,1.7]之间的函数图像如下图 15 所示。

但考虑实际的情况下，既是α 略大于 1.8 也可以保证收敛特性，但后期的 Pace 值震荡较大，因此结合实际情况推荐如下的参数值，这里的 A 为在如下区间内的 α 带入(14)所产生的对应值，这里$A - \alpha$ 图像，如下图 16 所示。

$$\alpha \in [1.2,1.6]$$

$$A \in [2.7,5]$$

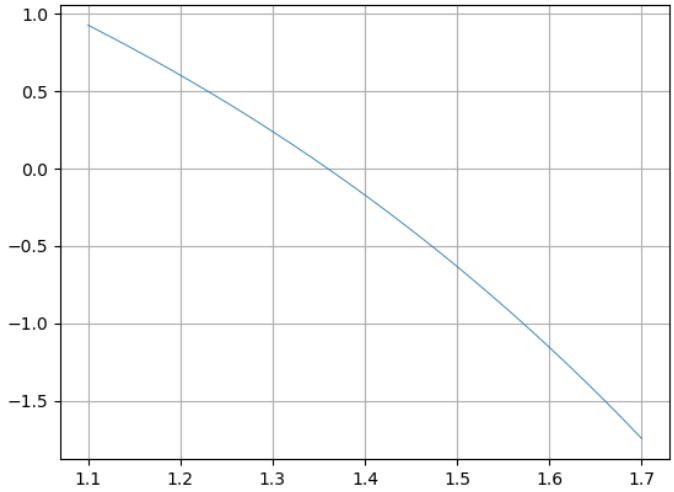

图 15 函数$2 * \left(1 + exp\left(\frac{\alpha}{1+\alpha} - 1\right)\right) - \alpha * (1 + exp(\alpha - 1))$在α ∈[1.1,1.7]上的图像

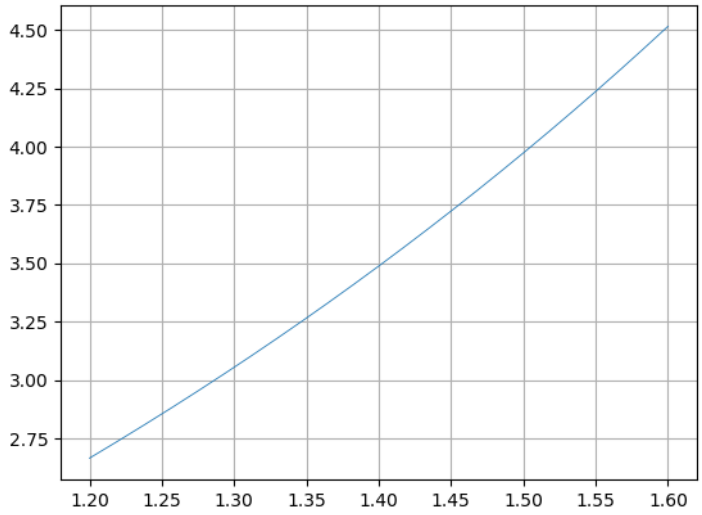

图 16 函数$\alpha * (1 + exp(\alpha - 1))$在α ∈[1.2,1.6]上的图像

下面对(11)式的更新机制，进行仿真验证，进行如下图 17，18 所示的两次验证。

这里可以很明显的看出在经过短暂的震荡之后，$Pace$的曲线为$Radius$曲线拉伸之后的曲线，对应点的幅值比最终等于常数，近似为 1.5，这同前面分析的过程基本一致

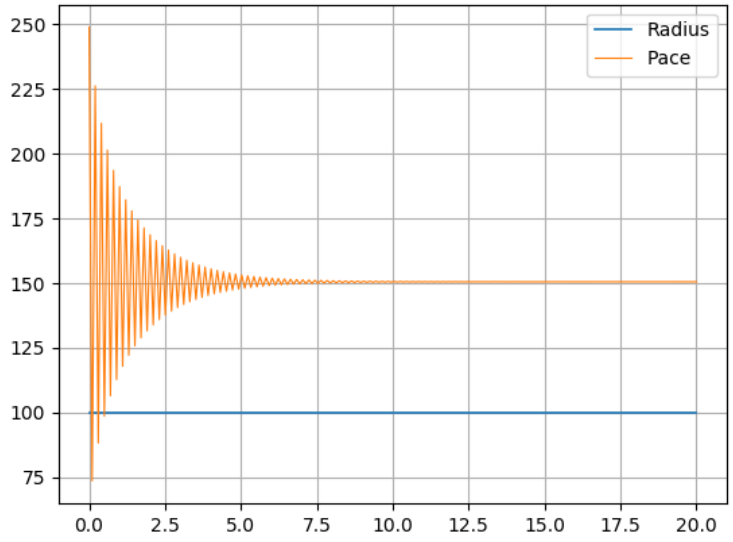

图 17 $p^{(0)} = 50, r^{(n)} = 100$, A=4 时，两变量的变化曲线

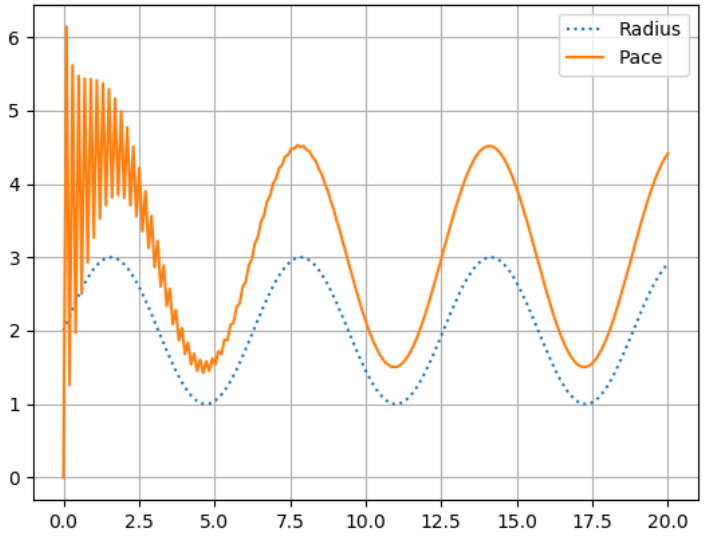

图 18 $p^{(0)} = 50, r^{(n)} = sin(n) + 2 + 1e - 15, A_{[Pace]} = 4$时，两变量的变化曲线

最后，对$Bias^{(n)}$的更新机制进行讨论。正如上一小节所述，$Bias^{(n)}$的主要功能是提供一个基于当前位置的对于目前相对最优位置导向性的置信概率偏置，考虑到当$Pace^{(n)}$,$Radius^{(n)}$的变化剧烈时，此时的相对最优位置的导向性的可信度较低，因此这时的$Bias^{(n)}$的值应较大，而反之在算法即将收敛与某一值时，$Pace^{(n)}$,$Radius^{(n)}$的变化将相应地减缓，这时的$Bias^{(n)}$的值也应相应的稳定下来。而由上面的分析可以得出，$Judge^{(n)}$可以反映$Pace^{(n)}$,$Radius^{(n)}$的相对变化程度，因此采用$Judge^{(n)}$乘以常系数的方式更新$Bias^{(n)}$。$Judge^{(n)}$的具体数学形式如下

式(18)所示：

$$Judge^{(n)} = \frac{1}{1 + \exp\left(\frac{Pace^{(n-1)}}{Radius^{(n-1)}} - 1\right)} \tag{18}$$

4 算法伪代码

Input：*Cycle*(循环次数)，*Dim*(待搜索空间的维数)，*HNum*(参与搜索的成员数)，

$X_{(i)}^{(0)}$ (第 i 个搜索成员的初始位置，*Dim* 维向量)，

A_D，B_D(两个初始系数用于生成初始的搜索范围)，

$T_{[init]}$，$T_{[end]}$(用于生成 *Trust* 函数)

$A_{[Pace]}$ (*Pace* 函数幅值，推荐取[3.5,5])

$A_{[Bias]}$ (*Bias* 函数幅值，推荐取[1,1.5])

$A_{[Chaos]}$，$B_{[Chaos]}$(用于生成 *Chaos*，推荐$A_{[Chaos]}$=0.4，$B_{[Chaos]} = 0.8$)

Initialization：

$$Direction_{(i)}^{(0)} = A_D * Random(Dim, 1) + B_D$$

$$X_{(i)}^{1} = X_{(i)}^{0} + Direction_{(i)}^{(0)}$$

$Direction_{(i)}^{(0)}$归一化

Cycle：

$$Trust^{(n)} = T_{[end]} - n * (T_{[end]} - T_{[init]})/Cycle$$

$$Gbelief_{(i)}^{(n)} = logistic(Trust^{(n)}, \frac{1 - Trust^{(n)}}{3}, Dim)$$

$$Radius^{(n-1)} = \overline{std}(Hbest^{(n-1)})$$

(**Attention**：$std\left(Hbest^{(n-1)}\right)$ 为$Hbest^{(n-1)}$每一位上对应值的组合的标准差)

(**Attention**：$Radius^{(n-1)}$为$std\left(Hbest^{(n-1)}\right)$的 2 范数)

$$Judge^{(n)} = \frac{1}{1 + \exp\left(\frac{Pace^{(n-1)}}{Radius^{(n-1)}} - 1\right)}$$

$$Pace^{(n)} \coloneqq A_{[Pace]} * Radius^{(n-1)} * Judge^{(n)}$$

$$Bias^{(n)} \coloneqq A_{[Bias]} * Judge^{(n)}$$

$$Chaos_{(i)}^{(d)(n)} = A_{[Chaos]} * Random_{(i)}^{(n)}(Dim, 1) + B_{[Chaos]}$$

$$Hbelief_{(i)}^{(d)(n)} = 2 * Random_{(i)}^{(d)(n)}(1,1) - Bias^{(n)} * Chaos_{(i)}^{(d)(n)}$$

$Hbest_{(i)}^{(n)}$排序

$\left(\textbf{Attention}: RefBest^{(d)(n)} \neq X_{(i)}^{(n)}，RefBest^{(d)(n)}\text{为相对较好的 HNum 个位置}\right)$

$$base_{(i)}^{(d)(n)} = RefBest_{(i)}^{(d)(n)} - X_{(i)}^{(n)}$$

$base_{(i)}^{(d)(n)}$归一化

$$\delta^{(n)} = \sum_{d=1}^{Dim} Hbelief_{(i)}^{(d)(n)} * base_{(i)}^{(d)(n)}$$

$\delta^{(n)}$归一化

$$Direction_{(i)}^{(n)} = \left(1 - Gbelief_{(i)}^{(n)}\right) * Direction_{(i)}^{(n-1)} + Gbelief_{(i)}^{(n)} * \delta^{(n)}$$

$Direction_{(i)}^{(n)}$归一化

$$X_{(i)}^{(n)} = X_{(i)}^{(n-1)} + Pace^{(n)} * Direction_{(i)}^{(n)}$$

$$Evaluation_{(i)}^{(n)} = f\left(X_{(i)}^{(n)}\right)$$

$\boldsymbol{if}\ Hbest_{(i)}^{(n)} > Evaluation_{(i)}^{(n)}$:

$$Hbest_{(i)}^{(n)} = X_{(i)}^{(n)}$$

$$Hbestv_{(i)}^{(n)} = Evaluation_{(i)}^{(n)}$$

$\boldsymbol{End\ If}$

$Return：min(Hbestv) \rightarrow Hbest$

5 仿真分析

由于本算法主要针对可行域为高维的情况因此主要采用 Rastrigin，Sphere，Grienwank 函数作为主要的验证函数。具体的函数形式如下式(18)所示。

$$Rastrigin(X) = \sum_{d=1}^{Dim} 10 + x_d^2 - 10 * cos(2\pi x_d)$$

$$Sphere(X) = \sum_{d=1}^{Dim} x_d^2 \tag{18}$$

$$Grienwank(X) = 1 + \frac{1}{4000}\sum_{d=1}^{Dim} x_d^2 - \prod_{d=1}^{Dim} \cos(\frac{x_d}{\sqrt{d}})$$

上述三个函数在二维向量空间下的部分图像如下图 19，20，21 所示

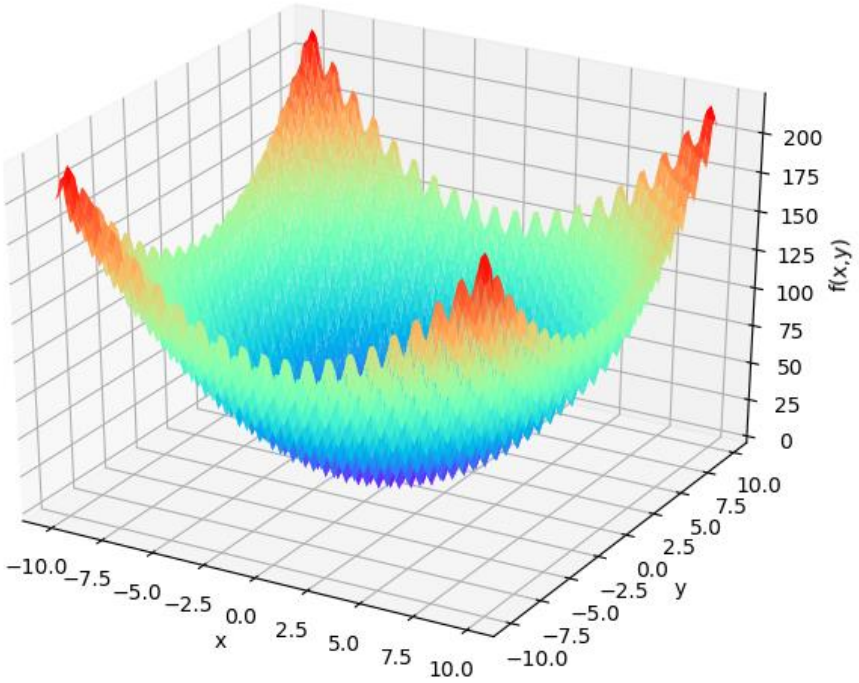

图 19 Rastrigin 函数在二维定义域下的图像

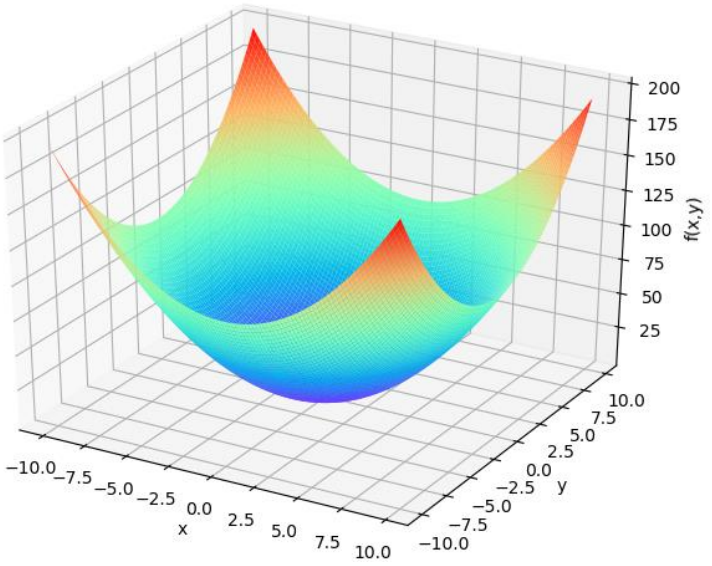

图 20 Sphere 函数在二维定义域下的图像

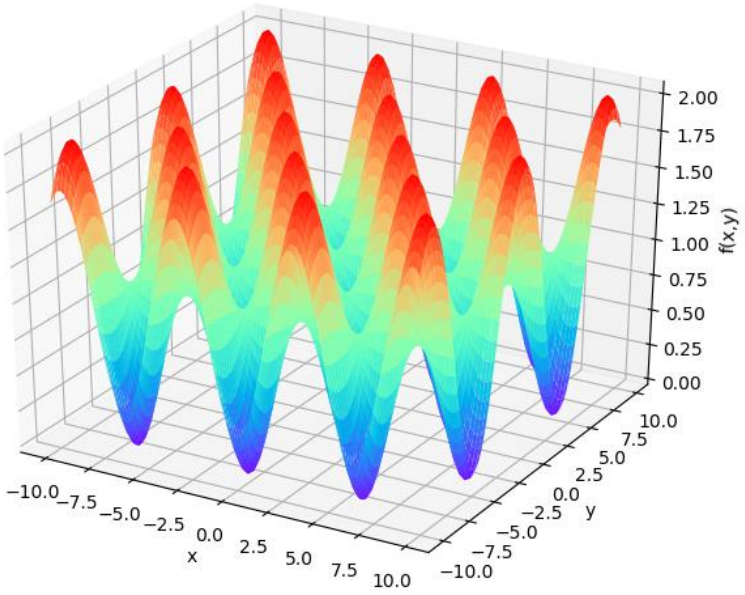

图 21 Grienwank 函数在二维定义域下的图像

这里为了考量算法在极端条件下的性能，分别在 15 维情况下同 PSO 算法进行了性能对比，同时将初值均设定为(1000,1000,…,1000)，$A_D = 1$，$B_D = -0.5$。具体的性能及算法中的参数变化图如下图 21 至 35 所示。

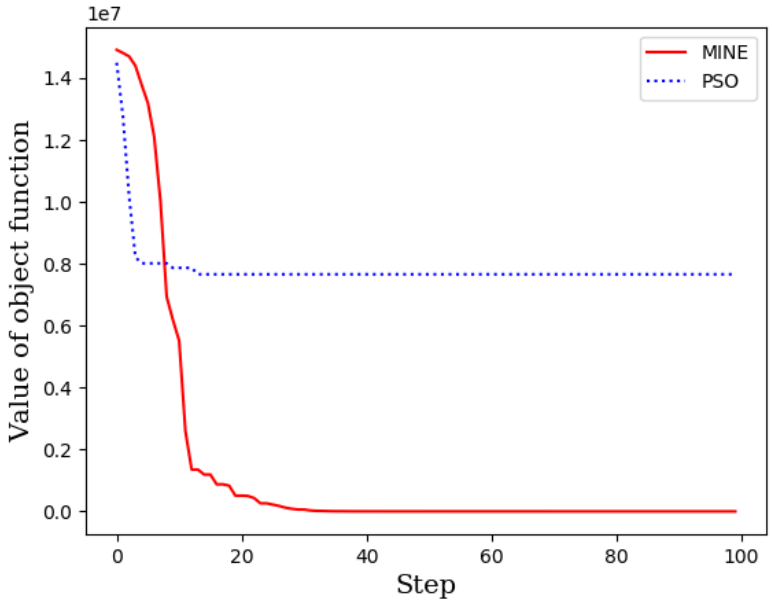

图 21 以 Rastrigin 函数为对象的同 PSO 算法仿真性能对比

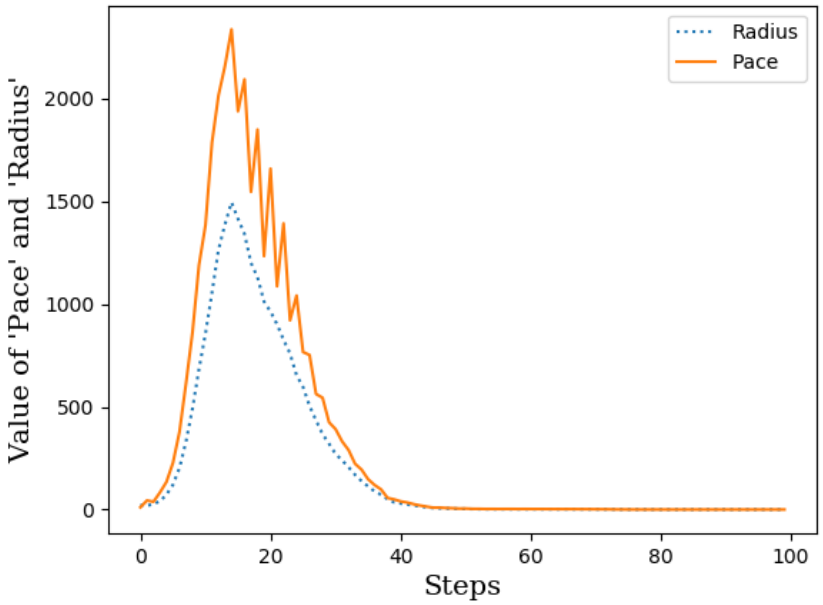

图 22 以 Rastrigin 函数为对象的算法中 Pace 和 Radius 变化图

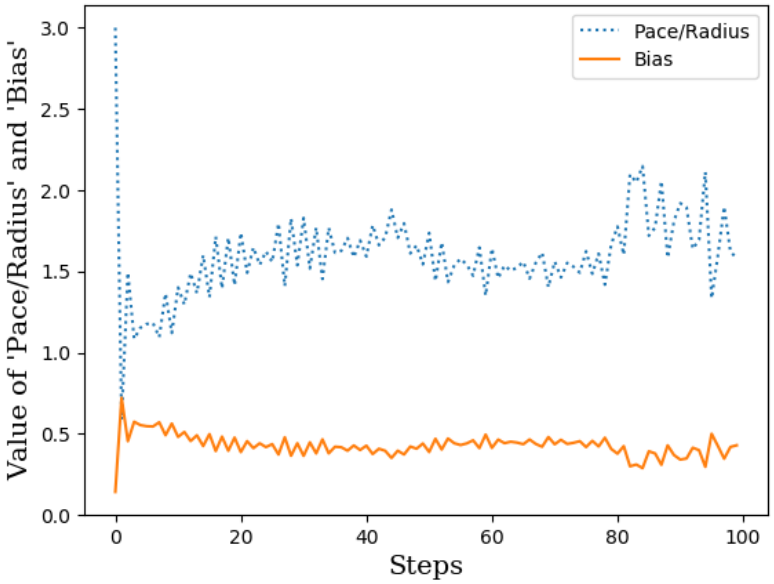

图 23 以 Rastrigin 函数为对象的算法中 Pace*Radius 和 Bias 变化图

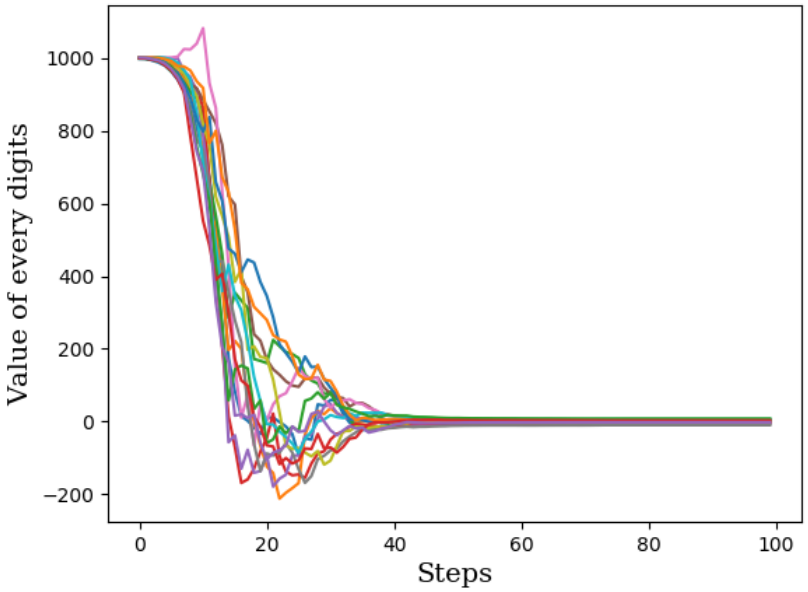

图 24 以 Rastrigin 函数为对象的算法最优点各分量值变化图

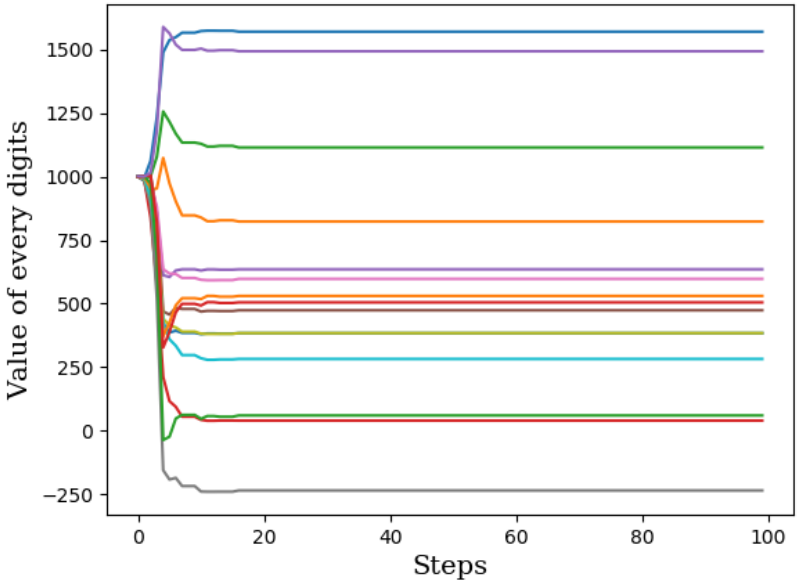

图 25 以 Rastrigin 函数为对象的算法最优点各分量值变化图(PSO)

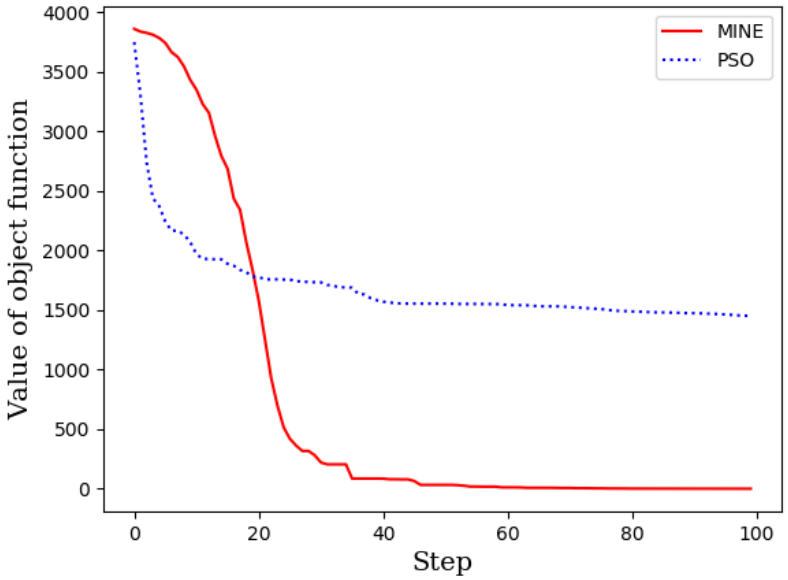

图 26 以 Sphere 函数为对象的同 PSO 算法仿真性能对比

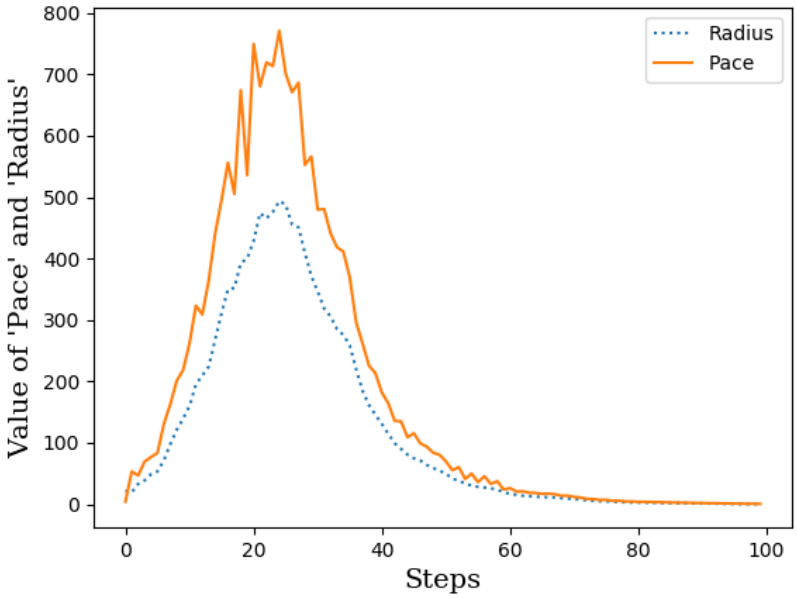

图 27 以 Sphere 函数为对象的算法中 Pace 和 Radius 变化图

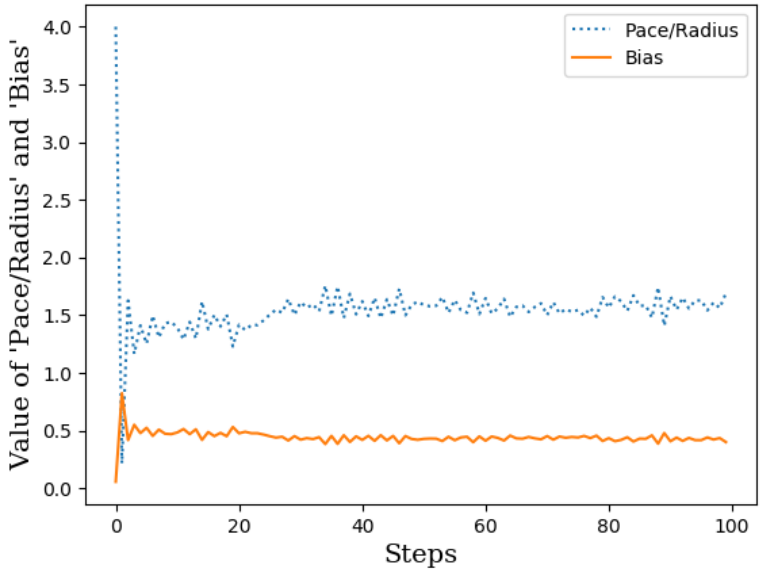

图 28 以 Sphere 函数为对象的算法中 Pace*Radius 和 Bias 变化图

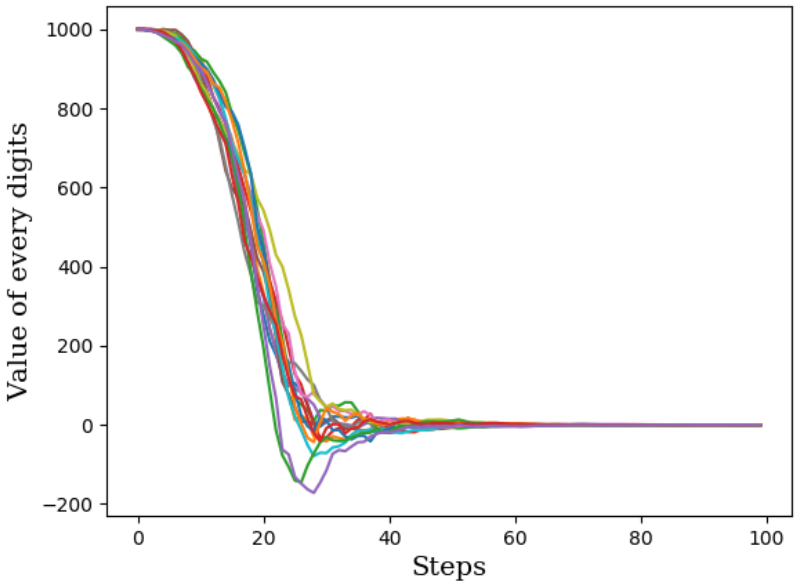

图 29 以 Sphere 函数为对象的算法最优点各分量值变化图

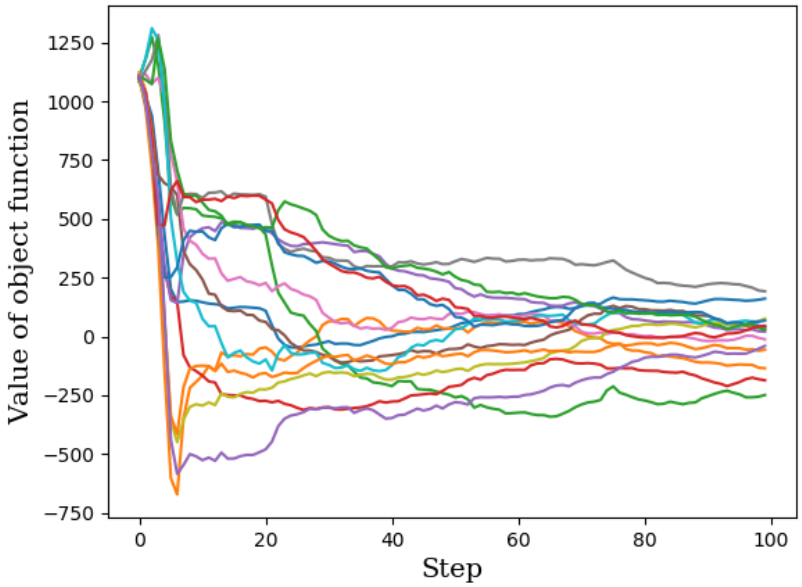

图 30 以 Sphere 函数为对象的算法最优点各分量值变化图(PSO)

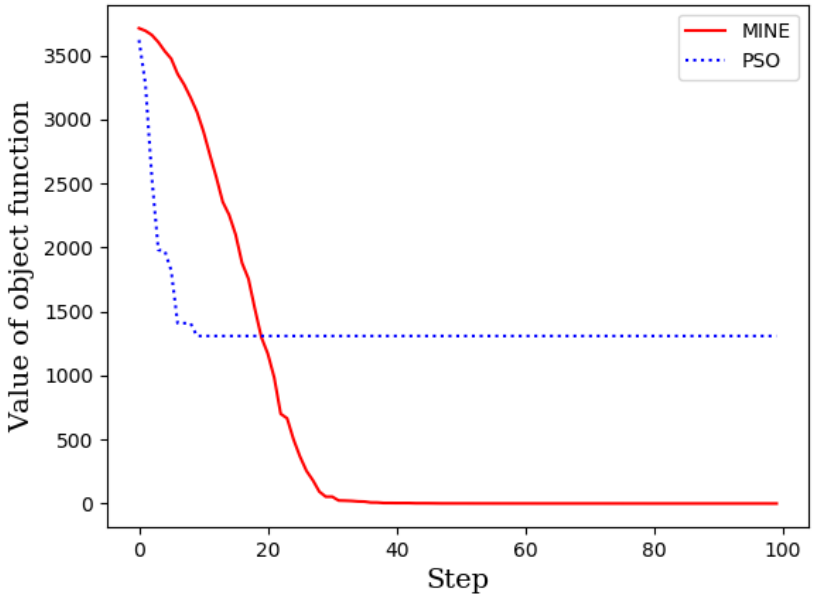

图 31 以 Grienwank 函数为对象的同 PSO 算法仿真性能对比

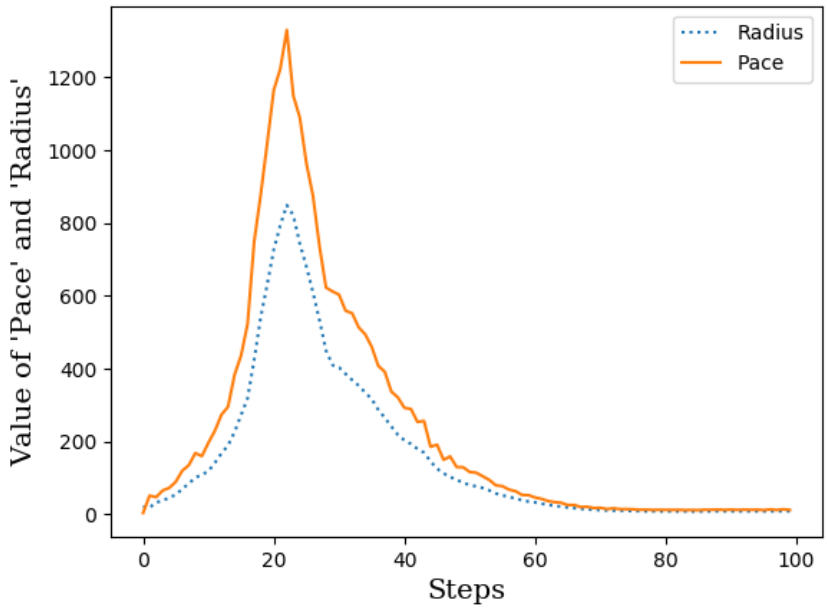

图 32 以 Grienwank 函数为对象的算法中 Pace 和 Radius 变化图

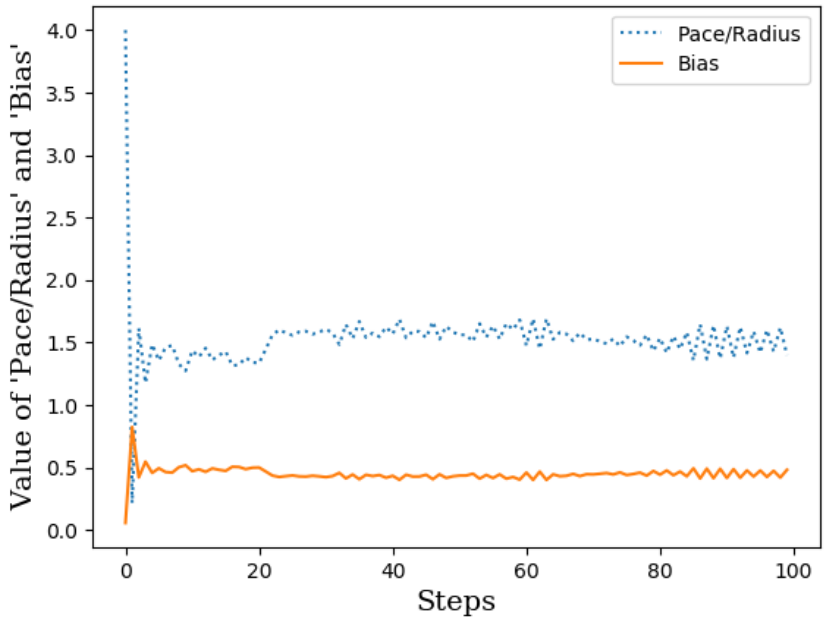

图 33 以 Grienwank 函数为对象的算法中 Pace*Radius 和 Bias 变化图

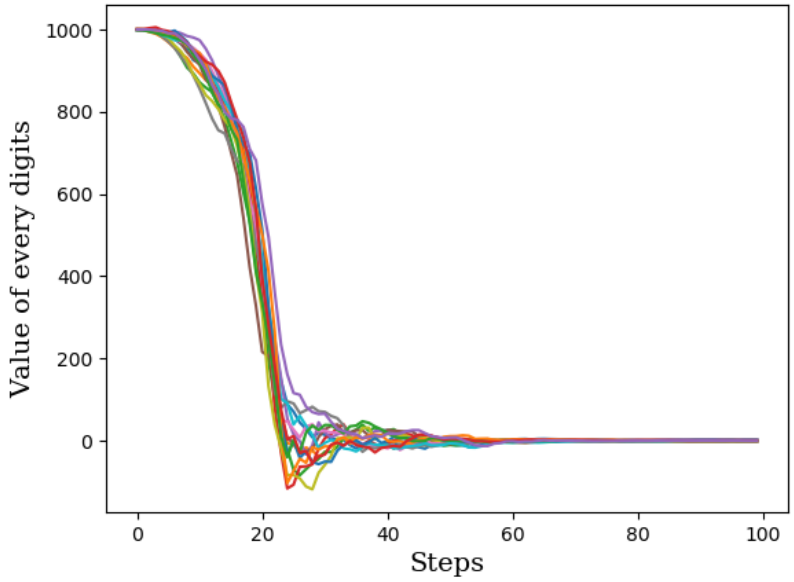

图 34 以 Sphere 函数为对象的算法最优点各分量值变化图

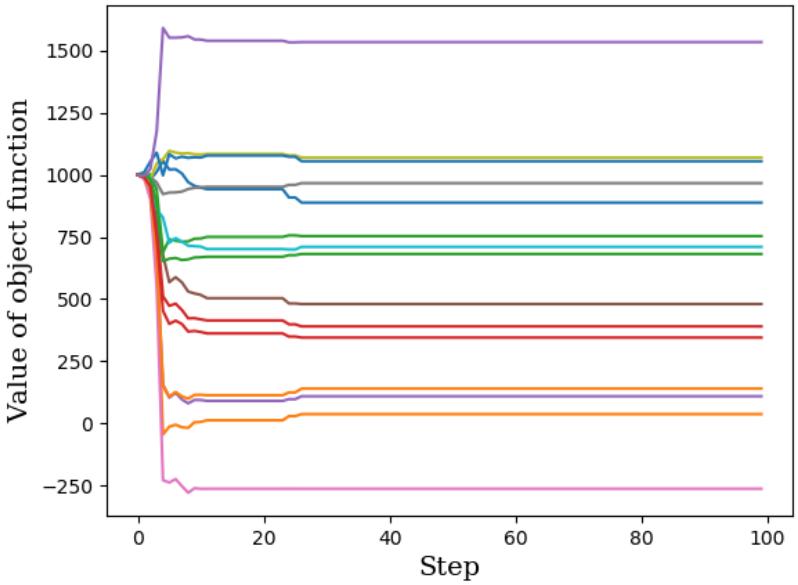

图 35 以 Sphere 函数为对象的算法最优点各分量值变化图(PSO)

这里由上图分析可以得出，本算法在搜索的过程中是先均匀地扩大搜索地范围。当遇到某一收敛值时，将以收敛解的附近的位置为中心不断缩小搜索区域，直到满足算法的终止条件为止。并且通过对比可以看到，同 PSO 仅能实现某一位上的全局最优不同，本算法可以实现在各个维度上的同步收敛，也就是说，在不增加搜索群体的基础上，可以实现可行域全维度上的更深层更均匀的搜索。并且由于步长的更新是自动根据当前所记录的历史最优位置的分布情况进行更新

的，因此在参数设置实现了简化(并不需要进行多余的手动调整)。