粒子群算法简介

1 粒子群算法的历史

粒子群算法源于复杂适应系统（Complex Adaptive System,CAS）。CAS理论于1994年正式提出，CAS中的成员称为主体。比如研究鸟群系统，每个鸟在这个系统中就称为主体。主体有适应性，它能够与环境及其他的主体进行交流，并且根据交流的过程“学习”或“积累经验”改变自身结构与行为。整个系统的演变或进化包括：新层次的产生（小鸟的出生）；分化和多样性的出现（鸟群中的鸟分成许多小的群）；新的主题的出现（鸟寻找食物过程中，不断发现新的食物）。

所以CAS系统中的主体具有4个基本特点（这些特点是粒子群算法发展变化的依据）：

- 首先，主体是主动的、活动的。
- 主体与环境及其他主体是相互影响、相互作用的，这种影响是系统发展变化的主要动力。
- 环境的影响是宏观的，主体之间的影响是微观的，宏观与微观要有机结合。
- 最后，整个系统可能还要受一些随机因素的影响。

粒子群算法就是对一个CAS系统 - - - 鸟群社会系统的研究得出的。

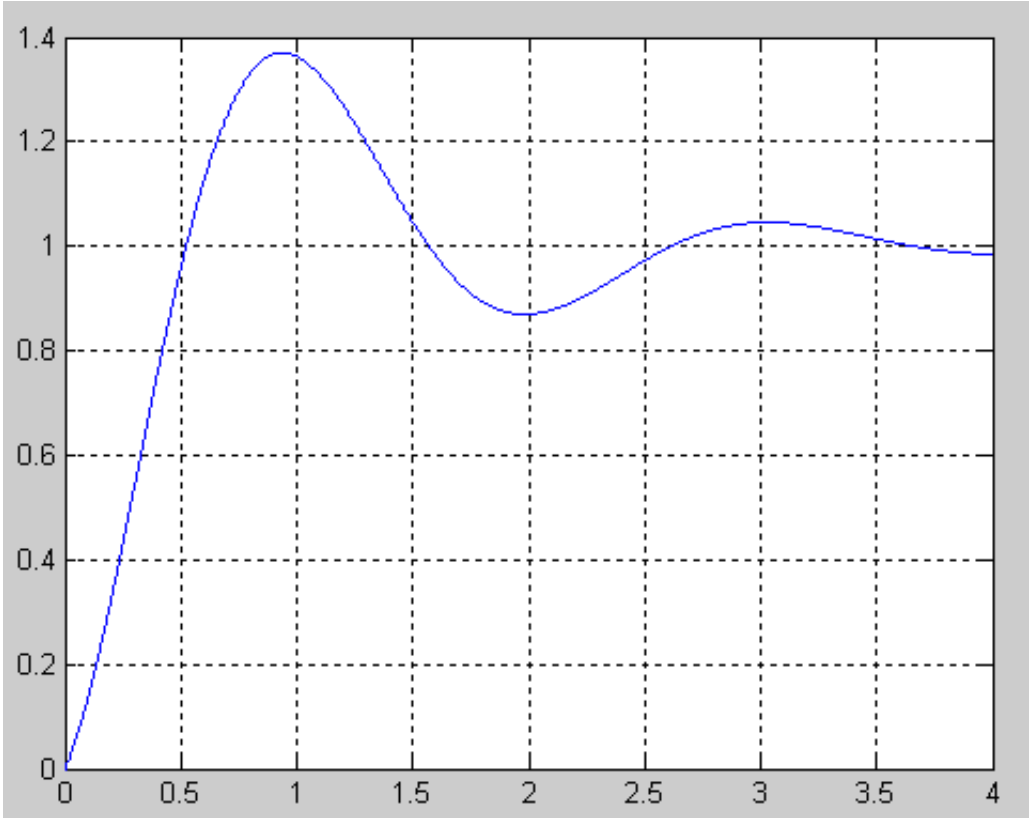
粒子群算法（ Particle Swarm Optimization, PSO）最早是由Eberhart和Kennedy于1995年提出，它的基本概念源于对鸟群觅食行为的研究。设想这样一个场景:一群鸟在随机搜寻食物，在这个区域里只有一块食物，所有的鸟都不知道食物在哪里，但是它们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢?最简单有效的就是搜寻目前离食物最近的鸟的周围区域。

PSO算法就从这种生物种群行为特性中得到启发并用于求解优化问题。在PSO中，每个优化问题的潜在解都可以想象成d维搜索空间上的一个点，我们称之为“粒子”（Particle），所有的粒子都有一个被目标函数决定的适应值(Fitness Value)，每个粒子还有一个速度决定他们飞翔的方向和距离，然后粒子们就追随当前的最优粒子在解空间中搜索。Reynolds对鸟群飞行的研究发现。鸟仅仅是追踪它有限数量的邻居但最终的整体结果是整个鸟群好像在一个中心的控制之下.即复杂的全局行为是由简单规则的相互作用引起的。

2 粒子群算法的具体表述

上面罗嗦了半天，那些都是科研工作者写论文的语气，不过，PSO的历史就像上面说的那样。下面通俗的解释PSO算法。

PSO算法就是模拟一群鸟寻找食物的过程，每个鸟就是PSO中的粒子，也就是我们需要求解问题的可能解，这些鸟在寻找食物的过程中，不停改变自己在空中飞行的位置与速度。大家也可以观察一下，鸟群在寻找食物的过程中，开始鸟群比较分散，逐渐这些鸟就会聚成一群，这个群忽高忽低、忽左忽右，直到最后找到食物。这个过程我们转化为一个数学问题。寻找函数 `y=1-cos(3*x)*exp(-x)` 的在[0,4]最大值。该函数的图形如下：

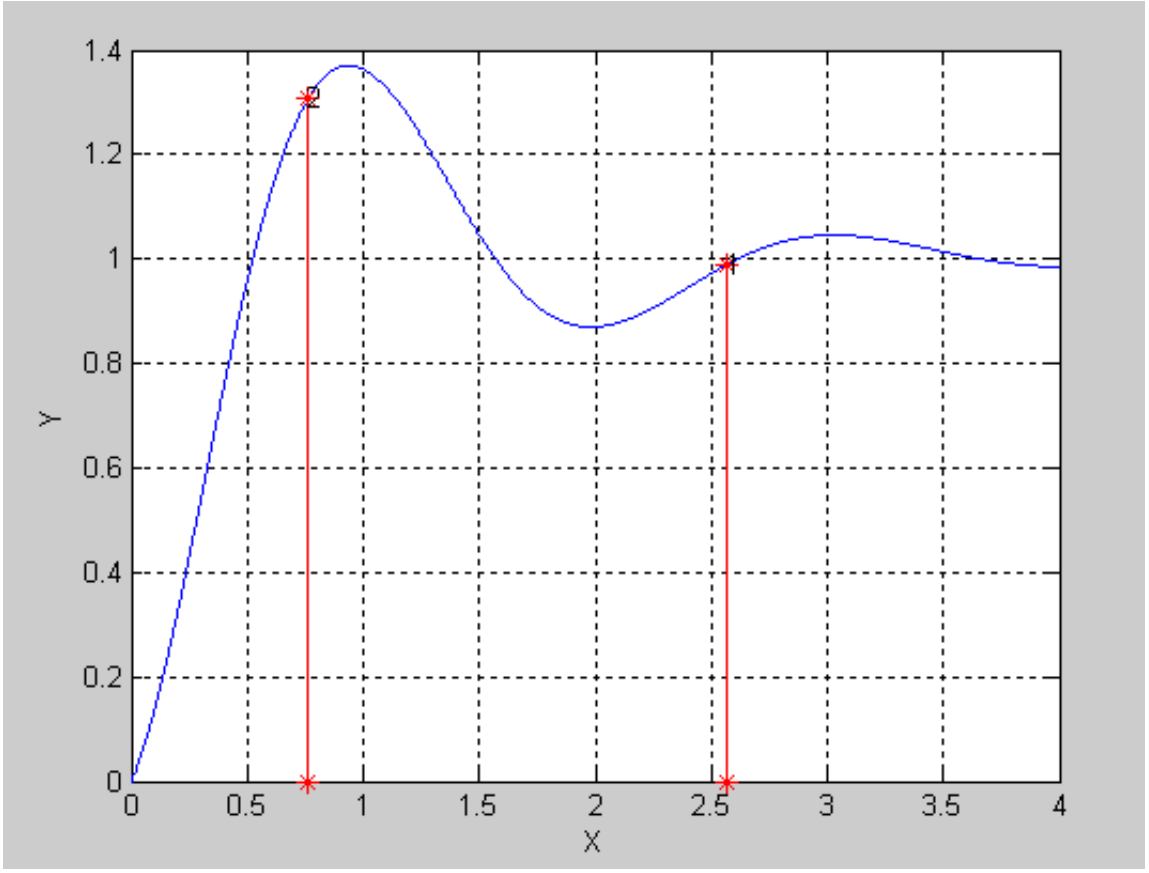


当x=0.9350-0.9450，达到最大值y=1.3706。为了得到该函数的最大值，我们在[0，4]之间随机的洒一些点，为了演示，我们放置两个点，并且计算这两个点的函数值，同时给这两个点设置在[0，4]之间的一个速度。下面这些点就会按照一定的公式更改自己的位置，到达新位置后，再计算这两个点的值，然后再按照一定的公式更新自己的位置。直到最后在y=1.3706这个点停止自己的更新。这个过程与粒子群算法作为对照如下：

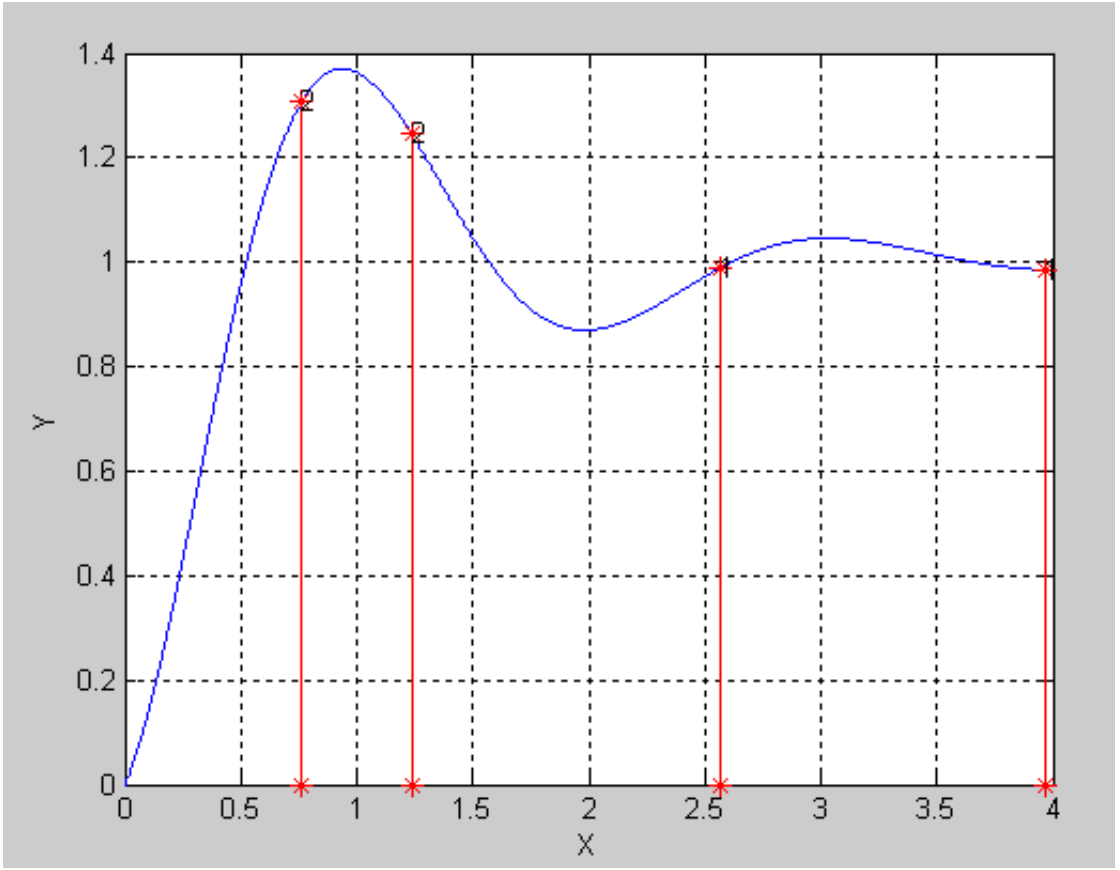
- 这两个点就是粒子群算法中的粒子。
- 该函数的最大值就是鸟群中的食物
- 计算两个点函数值就是粒子群算法中的适应值，计算用的函数就是粒子群算法中的适应度函数。
- 更新自己位置的一定公式就是粒子群算法中的位置速度更新公式。

下面演示一下这个算法运行一次的大概过程：

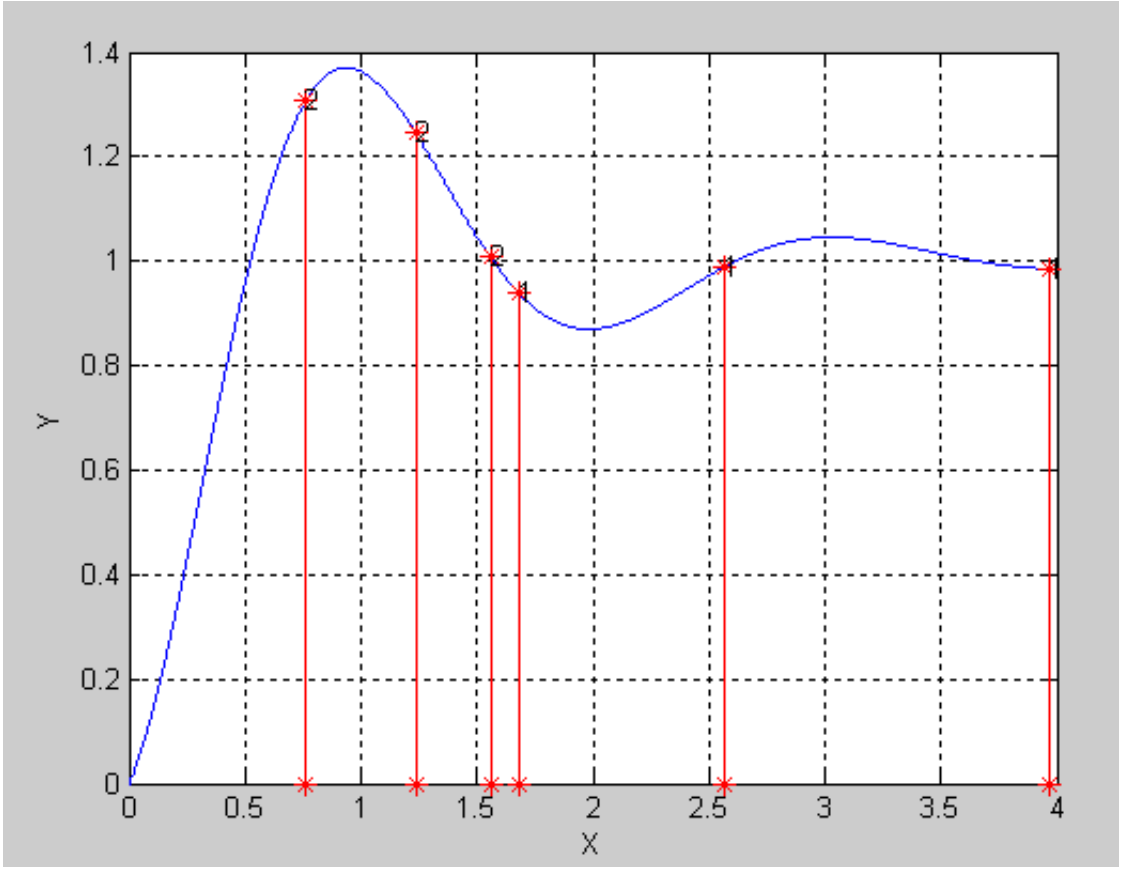
第一次初始化:



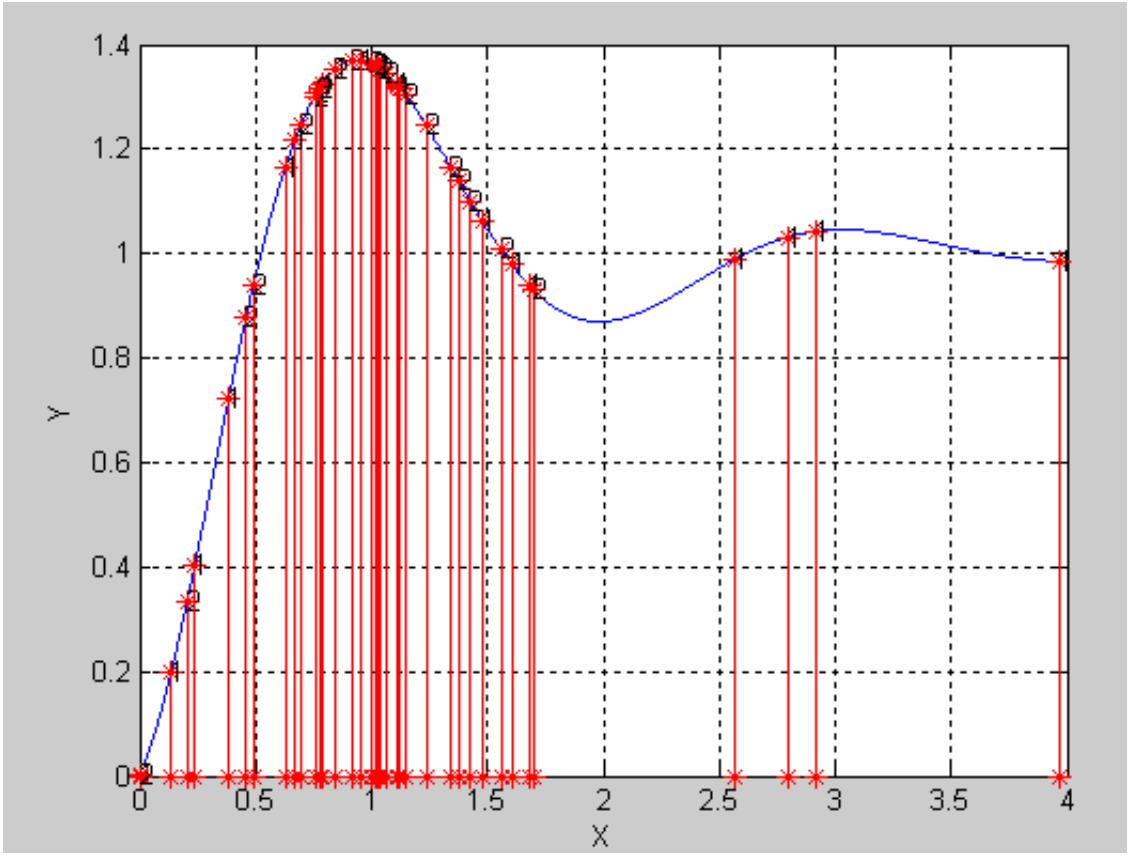
第一次更新位置:



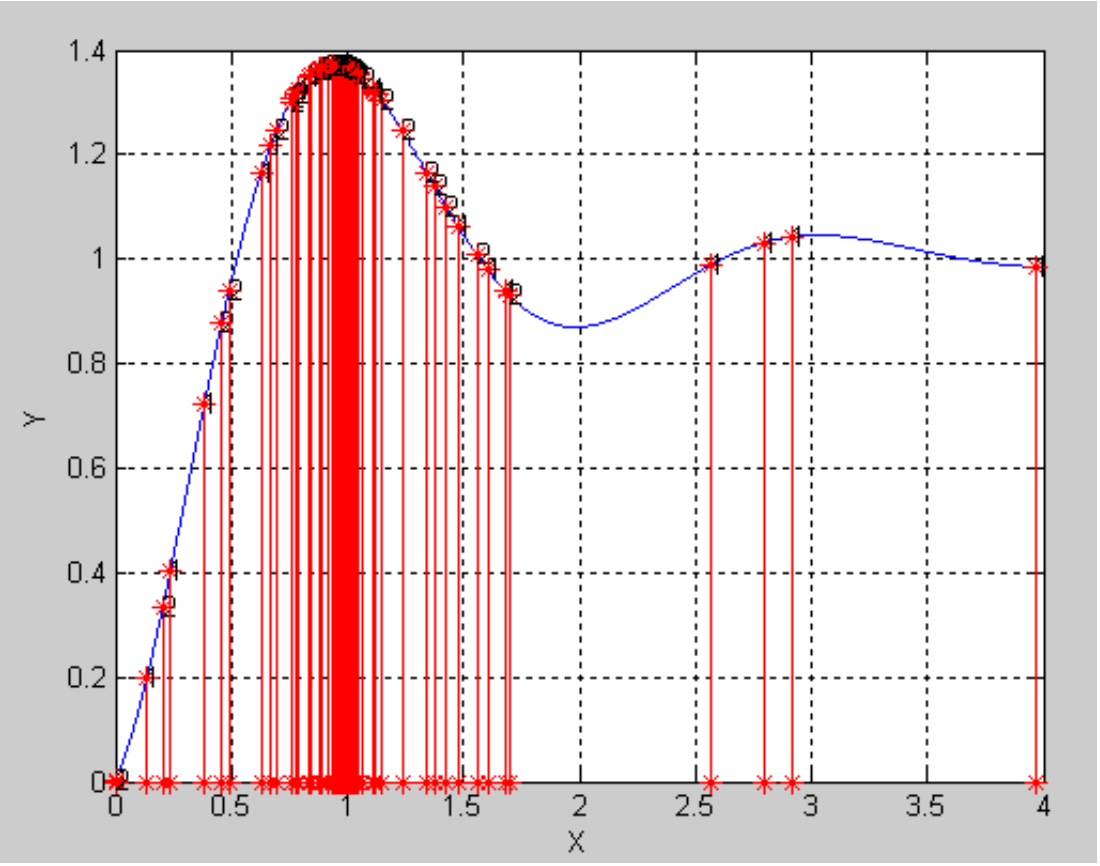
第二次更新位置:



第21次更新:



最后的结果（30次迭代）:



最后所有的点都集中在最大值的地方。

在粒子群算法的大概思想就说到这里。下节介绍标准的粒子群算法.

标准的粒子群算法

在上一节的叙述中，唯一没有给大家介绍的就是函数的这些随机的点（粒子）是如何运动的，只是说按照一定的公式更新。这个公式就是粒子群算法中的位置速度更新公式。下面就介绍这个公式是什么。在上一节中我们求取函数y=1-cos(3*x*)exp(-x)的在[0,4]最大值。并在[0,4]之间放置了两个随机的点，这些点的坐标假设为$x_1 = 1.5$ ；$x_2 = 2.5$ ；这里的点是一个标量，但是我们经常遇到的问题可能是更一般的情况 - - x为一个矢量的情况，比如二维的情况$z = 2 \star x_1 + 3 \star x$的情况。这个时候我们的每个粒子为二维，记粒子$P_1 = (x_{11}, x_{12}), P2 = (x_{21}, x_{22}), P3 = (x_{31}, x_{32})$，$.....Pn = (x_{n1}, x_{n2})$ 。这里n为粒子群群体的规模，也就是这个群中粒子的个数，每个粒子的维数为2。更一般的是粒子的维数为q，这样在这个种群中有n个粒子，每个粒子为q 维。

由n个粒子组成的群体对Q维（就是每个粒子的维数）空间进行搜索。每个粒子表示为：$xi = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iQ})$ ，每个粒子对应的速度可以表示为$vi = (v_{i1}, v_{i2}, v_{i3}, \ldots., v_{iQ})$ ，每个粒子在搜索时要考虑两个因素：

1. 自己搜索到的历史最优值 p_i ，$p_i = (p_{i1}, p_{i2}, \ldots., p_{iQ})$ ，i=1,2,3,....,n。
2. 全部粒子搜索到的最优值p_g ，$p_g = (p_{g1}, p_{g2}, \ldots., p_{gQ})$ ，注意这里的p_g只有一个。

下面给出粒子群算法的位置速度更新公式：

$$v_{id}^{k+1} = wv_{id}^{k} + c_1\xi(p_{id}^{k} - x_{id}^{k}) + c_2\eta(p_{gd}^{k} - x_{id}^{k})$$

$$x_{id}^{k+1} = x_{id}^{k} + rv_{id}^{k+1}$$

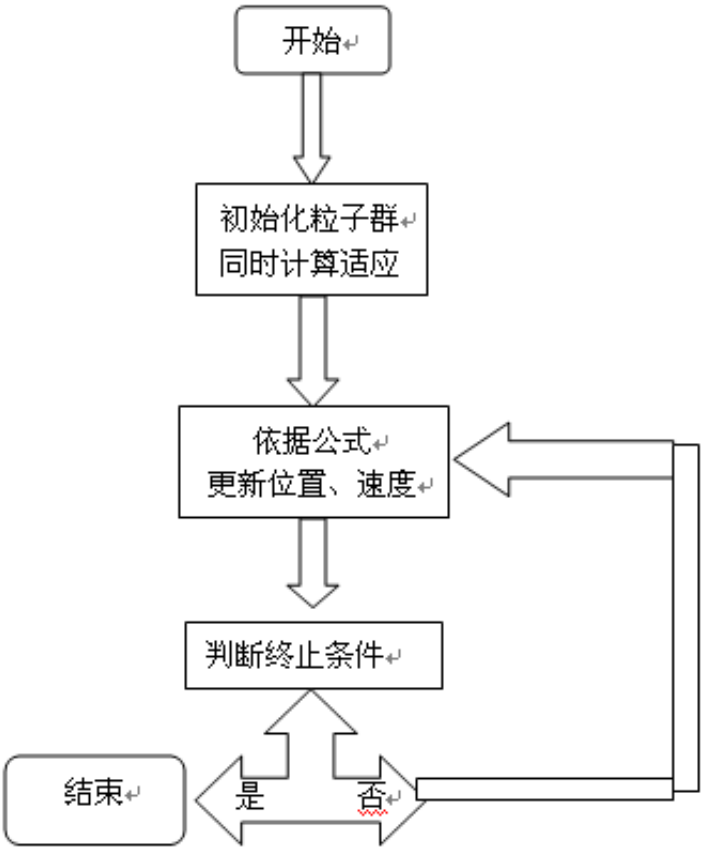
这里有几个重要的参数需要大家记忆，因为在以后的讲解中将会经常用到,它们是：

w是保持原来速度的系数，所以叫做**惯性权重**。
c_1是粒子跟踪自己历史最优值的权重系数，它表示粒子自身的认识，所以叫“**认知**”。通常设置为2。

$\xi\eta$是[0,1]区间内均匀分布的随机数。
r是对位置更新的时候，在速度前面加的一个系数，这个系数我们叫做**约束因子**。通常设置为1。

这样一个标准的粒子群算法就结束了。

下面对整个基本的粒子群的过程给一个简单的图形表示：



判断终止条件可是设置适应值到达一定的数值或者循环一定的次数。

注意：这里的粒子是同时跟踪自己的历史最优值与全局（群体）最优值来改变自己的位置预速度的，所以又叫做**全局版本的标准粒子群优化算法**。

标准的粒子群算法(局部版本)

在全局版的标准粒子群算法中，每个粒子的速度的更新是根据两个因素来变化的，这两个因素是：

1. 粒子自己历史最优值p_i。
2. 粒子群体的全局最优值p_g。

如果改变粒子速度更新公式，让每个粒子的速度的更新根据以下两个因素更新，

1. 粒子自己历史最优值p_i。
2. 粒子**邻域内**粒子的最优值pn_k。

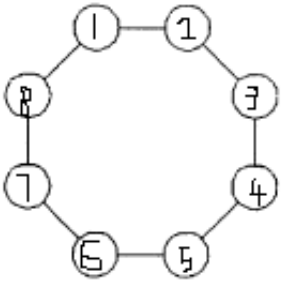
其余保持跟全局版的标准粒子群算法一样，这个算法就变为局部版的粒子群算法。

一般一个粒子i 的邻域随着迭代次数的增加而逐渐增加，开始第一次迭代，它的邻域为0，随着迭代次数邻域线性变大，最后邻域扩展到整个粒子群，这时就变成全局版本的粒子群算法了。经过实践证明：全局版本的粒子群算法收敛速度快，但是容易陷入局部最优。局部版本的粒子群算法收敛速度慢，但是很难陷入局部最优。现在的粒子群算法大都在**收敛速度与摆脱局部最优**这两个方面下功夫。其实这两个方面是矛盾的。看如何更好的折中了。

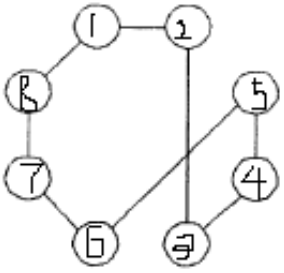
根据取邻域的方式的不同，局部版本的粒子群算法有很多不同的实现方法.

第一种方法：按照粒子的编号取粒子的邻域，取法有四种：

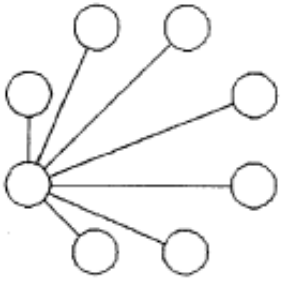
1. 环形取法



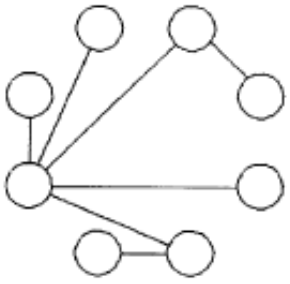
2. 随机环形取法



3. 轮形取法



4. 随机轮形取法。



因为后面有以环形取法实现的算法，对环形取法在这里做一点点说明：以粒子1为例，当邻域是0的时候，邻域是它本身，当邻域是1时，邻域为2，8；当邻域是2时，邻域是2，3，7，8；……，以此类推，一直到邻域为4，这个时候，邻域扩展到整个例子群体。据文献介绍（**国外的文献**），采用轮形拓扑结构，PSO的效果很好。

第二种方法：按照粒子的欧式距离取粒子的邻域

在第一种方法中，按照粒子的编号来得到粒子的邻域，但是这些粒子其实可能在实际位置上并不相邻，于是Suganthan提出基于空间距离的划分方案，在迭代中计算每一个粒子与群中其他粒子的距离。记录任何2个粒子间的的最大距离为dm。对每一粒子按照$||x_a - x_b||/dm$ 计算一个比值。其中||xa-xb||是当前粒子a到b的距离。而选择阈值frac根据迭代次数而变化。当另一粒子b满足$||x_a - x_b||/dm < frac$ 时，认为b成为当前粒子的邻域。

这种办法经过实验，取得较好的应用效果，但是由于要计算所有粒子之间的距离，计算量大，且需要很大的存

储空间，所以，**该方法一般不经常使用。**

粒子群算法分类

粒子群算法主要分为4个大的分支：

1 标准粒子群算法的变形

在这个分支中，主要是对标准粒子群算法的惯性因子、收敛因子（约束因子）、“认知”部分的c_1，“社会”部分的c_2进行变化与调节，希望获得好的效果。

惯性因子的原始版本是保持不变的，后来有人提出随着算法迭代的进行，惯性因子需要逐渐减小的思想。算法开始阶段，大的惯性因子可以使算法不容易陷入局部最优，到算法的后期，小的惯性因子可以使收敛速度加快，使收敛更加平稳，不至于出现振荡现象。**经过本人测试**，动态的减小惯性因子w，的确可以使算法更加稳定，效果比较好。但是递减惯性因子采用什么样的方法呢？人们首先想到的是线型递减，这种策略的确很好，但是是不是最优的呢？于是有人对递减的策略作了研究，研究结果指出：线型函数的递减优于凸函数的递减策略，但是凹函数的递减策略又优于线型的递减，经过本人测试，实验结果基本符合这个结论，但是效果不是很明显。

对于收敛因子，经过证明如果收敛因子取0.729,可以确保算法的收敛，但是不能保证算法收敛到全局最优，经过本人测试，取收敛因子为0.729效果较好。对于社会与认知的系数c_1,c_2也有人提出：c_1先大后小，而c_2先小后大的思想，因为在算法运行初期，每个鸟要有大的自己的认知部分而又比较小的社会部分，这个与我们自己一群人找东西的情形比较接近，因为在我们找东西的初期，我们基本依靠自己的知识取寻找，而后来，我们积累的经验越来越丰富，于是大家开始逐渐达成共识（社会知识），这样我们就开始依靠社会知识来寻找东西了。

2007年希腊的两位学者提出将收敛速度比较快的全局版本的粒子群算法与不容易陷入局部最优的局部版本的粒子群算法相结合的办法，利用的公式是:

速度更新公式，v代表速度

$$v = n * v(\text{全局版本}) + (1 - n) * v(\text{局部版本})$$

位置更新公式

$$w(k+1) = w(k) + v$$

该算法在文献中讨论了系数n取各种不同情况的情况，并且运行来了20000次来分析各种系数的结果。

2 粒子群算法的混合

这个分支主要是将粒子群算法与各种算法相混合，有人将它与模拟退火算法相混合，有些人将它与单纯形方法相混合。但是最多的是将它与遗传算法的混合。根据遗传算法的三种不同算子可以生成3中不同的混合算法。

粒子群算法与选择算子的结合

这里相混合的思想是：在原来的粒子群算法中，我们选择粒子群群体的最优值作为p_g，但是相结合的版本是根据所有粒子的适应度的大小给每个粒子赋予一个被选中的概率，然后依据概率对这些粒子进行选择，被选中的粒子作为p_g，其它的情况都不变。这样的算法可以在算法运行过程中保持粒子群的多样性，但是致命的缺点是收敛速度缓慢。

粒子群算法与杂交算子的结合

结合的思想与遗传算法的基本一样，在算法运行过程中根据适应度的大小，粒子之间可以两两杂交，比如用一个很简单的公式:

$$w(新) = n \times w_1 + (1 - n) \times w_2$$

w1与w2就是这个新粒子的父辈粒子。

这种算法可以在算法的运行过程中引入新的粒子，但是算法一旦陷入局部最优，那么粒子群算法将很难摆脱局部最优。

粒子群算法与变异算子的结合

结合的思想：测试所有粒子与当前最优的距离，当距离小于一定的数值的时候，可以拿出所有粒子的一个百分比（如10%）的粒子进行随机初始化，让这些粒子重新寻找最优值。

3 二进制粒子群算法

最初的PSO是从解决连续优化问题发展起来的.Eberhart等又提出了PSO的离散二进制版.用来解决工程实际中的组合优化问题。他们在提出的模型中将粒子的每一维及粒子本身的历史最优、全局最优限制为1或0，而速度不作这种限制。用速度更新位置时，设定一个阈值，当速度高于该阈值时，粒子的位置取1，否则取0。二进制PSO与遗传算法在形式上很相似，但实验结果显示，在大多数测试函数中，二进制PSO比遗传算法速度快，尤其在问题的维数增加时.

4 协同粒子群算法

协同PSO，该方法将粒子的D维分到D个粒子群中，每个粒子群优化一维向量，评价适应度时将这些分量合并为一个完整的向量。例如第i个粒子群，除第i个分量外，其他D-1个分量都设为最优值，不断用第i个粒子群中的粒子替换第i个分量，直到得到第i维的最优值，其他维相同。为将有联系的分量划分在一个群，可将D维向量分配到m个粒子群优化，则前D mod m个粒子群的维数是D/m的向上取整。后m - (D mod m)个粒子群的维数是D/m的向下取整。协同PSO在某些问题上有更快的收敛速度，但该算法容易被欺骗。

基本的粒子群算法的分支就着4个，大部分的粒子群算法都围绕着这4个分支在变化，其中粒子群算法的变形居多，从根本上来说，几乎没有什么新的思想的提出。