第七章　粒子群优化算法

一. 前言

1. 粒子群优化算法

- **Particle Swarm Optimization (PSO)**
- **美国学者Kennedy 和Eberhart于1995年提出**
- 灵感来自于对鸟群和鱼群社会行为的研究
- 个体与个体、个体与群体之间的相互作用和相互影响

Kennedy, J. and Eberhart, R.: Particle Swarm Optimization. Proceedings of the Fourth IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center 1942-1948, 1995.

一. 前言

2. 发展历程

- 鸟群行为分析
  - **Reynolds(1987)和Grenander(1990)发现，由数目庞大的个体组成的鸟群飞行中可以改变方向，散开，或者队形的重组等等，那么一定有某种潜在的能力或者规则保证了这些同步的行为。这些科学家都认为上述行为是基于不可预知的鸟类社会行为中的群体动态学。**

一. 前言

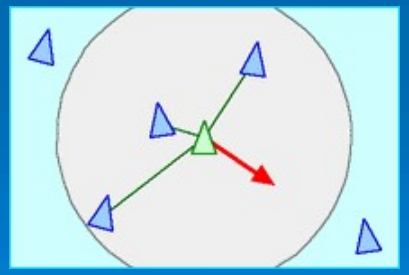
2. 发展历程

- 鱼群行为分析
  - **Wilson(1975)**在论文中阐述了对鱼群的研究并提出：“至少在理论上，鱼群的个体成员能够受益于群体中其他个体在寻找食物的过程中发现的和以前的经验，这种受益是明显的，它超过了个体之间的竞争所带来的利益消耗，不管任何时候食物资源不可预知的分散于四处。”这说明，同种生物之间信息的社会共享能够带来好处。

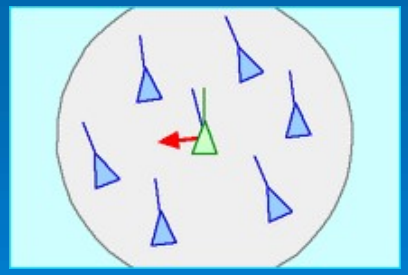
一. 前言

2. 发展历程

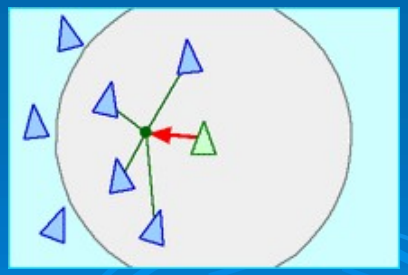
- **Reynolds提出了Boids模型来模拟鸟类群体的这种社会交互行为**
  - 分离性：远离周围个体的趋势，防止发生碰撞
  - 对齐性：倾向与伙伴的移动速度和方向保持一致
  - 一致性：倾向伙伴中心位置移动的趋势



分离性



对齐性



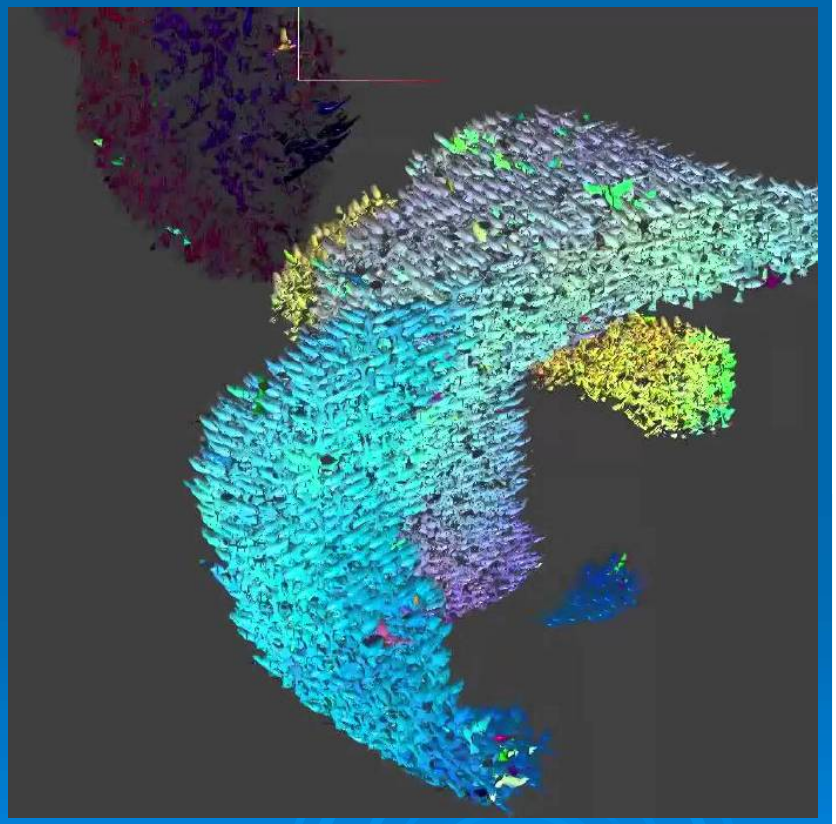
一致性

Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. Computer Graphics, 21(4):25-34, 1987

一. 前言

2. 发展历程

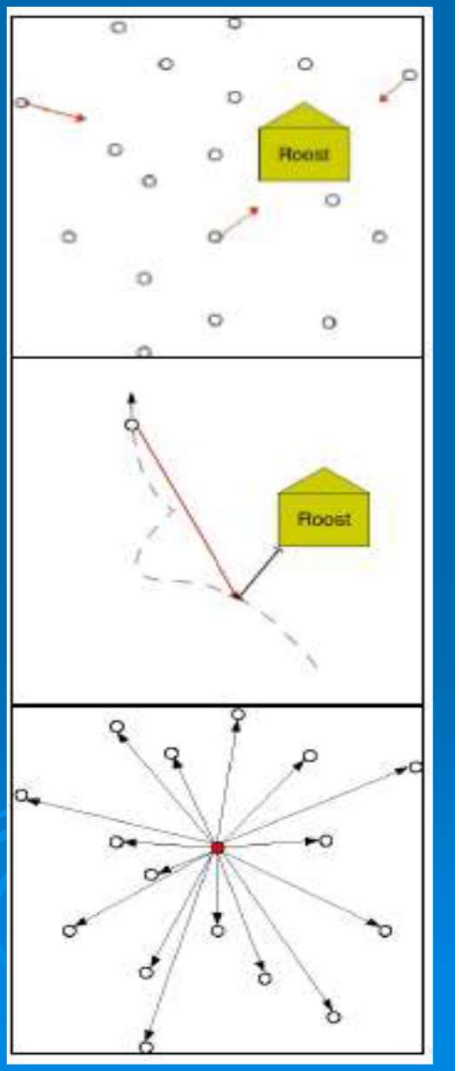
- **Boids模型：每个 boid** 个体都可以得知整体的几何参数，但群体要求其只对其周围某个小邻近范围作出反应。



一. 前言

2. 发展历程

- **Kennedy和Eberhart构建了一个类似于Boids的仿真系统**
  - **agent会被roost的位置所吸引**
  - **agent记得它曾经到过的离roost更近的位置信息**
  - **agent与neighbors共享关于离roost更近的位置信息**
- **最后所有的agent都收敛于roost**



一. 前言

2. 发展历程

- 鸟的捕食策略
  - 所有的鸟都在搜索空间里随机的飞行
  - 搜索空间里每个点都对应着一定数量的食物
  - 每只鸟都向飞到拥有最多食物的地点
  - 任何一只鸟都不知道这个最佳地点在哪
  - 对每只鸟来说，采用什么样的策略才能保证自己最后停在能够发现拥有尽可能多食物的地点呢？

一. 前言

2. 发展历程

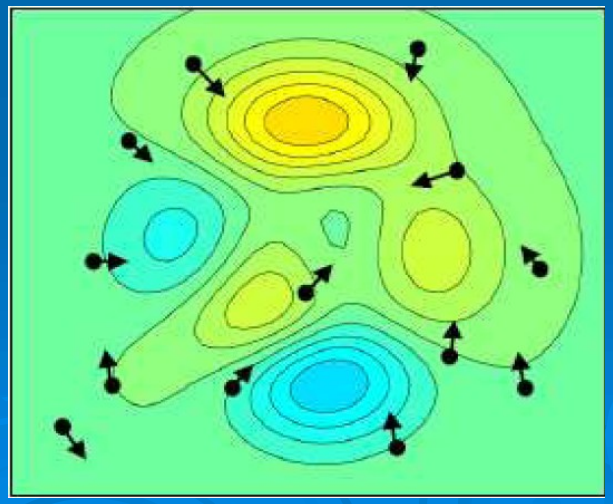
- 鸟的捕食策略
  - 所有的鸟都在搜索空间里随机的飞行
  - 搜索空间里每个点都对应着一定数量的食物
  - 每只鸟都想飞到拥有最多食物的地点
  - 任何一只鸟都不知道这个最佳地点在哪
  - 对每只鸟来说，采用什么样的策略才能保证自己最后停在能够发现拥有尽可能多食物的地点呢？

Follow the bird with most food!

一. 前言

3. 基本思想

- 模拟鸟群的捕食行为，通过群体中个体之间的协作和信息共享来寻找最优解，即个体的搜索行为将受到其自身发现的最优位置信息以及其邻居发现的最优位置信息所影响



一. 前言

4. 名称的由来：Swarm和Particle

- **Swarm**：在美国传统字典中有三个意思
  - 一大群尤指正在行进中的一大群昆虫或其它细小生物
  - 蜂群由蜂王带领迁移到别处建立一新据点的一群蜜蜂
  - 一大群尤指处于骚乱中或成群出动的一大批喧闹的人或动物
  - 作者引用此词是借用了**Millonas**在**1994**年的论文中的人工生命的一个应用模型中的提法

一. 前言

3. 名称的由来：Swarm和Particle

- **Particle:**
  - 算法中有速度和加速度的字眼，这比较适合于粒子。**Reeves在1983年的论文中讨论了粒子系统包括基本粒子团和云、火、烟雾等弥漫性物体**
  - 作者的想法是让粒子尽量具有一种普遍性的意义
  - 用粒子在超空间（**Hyperspace**）的飞行来模拟个体的社会交互行为

二. 基本算法

1. 算法描述

- 种群中**m**个个体分布在一个**D**维搜索空间中
- 每个个体均具有当前位置、速度以及历史最优位置三个属性
- 种群具有一定的拓扑结构，个体可以基于种群拓扑结构与其邻域内的其他个体进行相互作用
- 算法迭代时，每个个体会根据自身信息（认知行为）和邻域内其他个体的信息（社会行为）进行状态更新（位置和速度）

二. 基本算法

2. 算法模型

- 编号：
  - $1 \le i \le m$
  - $1 \le d \le D$
- 粒子i：
  - $\vec{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iD}\}$ 当前位置
  - $\vec{v}_i = \{v_{i1}, v_{i2}, \dots, v_{iD}\}$ 当前速度
  - $\vec{p}_i = \{p_{i1}, p_{i2}, \dots, p_{iD}\}$ 历史最好位置**pbest**
- 粒子群
  - $\vec{p}_g = \{p_{g1}, p_{g2}, \dots, p_{gD}\}$

注意：这里仅考虑整个粒子群共享唯一的最好位置**gbest**

二. 基本算法

2. 算法模型

- 更新公式：
  - $v_{id} = v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ **(1)**
  - $x_{id} = x_{id} + v_{id}$ **(2)**
  - 这里**c1**和**c2**为正常数，ξ和η为**random(0,1)**

二. 基本算法

2. 算法模型

- 更新公式：
  - $v_{id} = v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ (1)
  - $x_{id} = x_{id} + v_{id}$ (2)
  - 这里c1和c2为正常数，ξ和η为random(0,1)

公式(1)主要通过三个部分来更新粒子i的速度：一是粒子i前一时刻的速度，说明粒子的当前状态；二是粒子i当前位置与自己最好位置之间的距离，表示粒子对自身的认知；三是粒子i当前位置与群体最好位置之间的距离，体现群体的社会共享行为。

二. 基本算法

2. 算法模型

- 更新公式：
  - $v_{id} = v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ (1)
  - $x_{id} = x_{id} + v_{id}$ (2)
  - 这里c1和c2为正常数，ξ和η为random(0,1)

公式(1)主要通过三个部分来更新粒子i的速度：一是粒子i前一时刻的速度，说明粒子的当前状态；二是粒子i当前位置与自己最好位置之间的距离，表示粒子对自身的认知；三是粒子i当前位置与群体最好位置之间的距离，体现群体的社会共享行为。

二. 基本算法

2. 算法模型

- 更新公式：
  - $v_{id} = v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ (1)
  - $x_{id} = x_{id} + v_{id}$ (2)
  - 这里c1和c2为正常数，ξ和η为random(0,1)

公式(1)主要通过三个部分来更新粒子i的速度：一是粒子i前一时刻的速度，说明粒子的当前状态；二是粒子i当前位置与自己最好位置之间的距离，表示粒子对自身的认知；三是粒子i当前位置与群体最好位置之间的距离，体现群体的社会共享行为。

二. 基本算法

2. 算法模型

- 更新公式：
  - $v_{id} = v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ (1)
  - $x_{id} = x_{id} + v_{id}$ (2)
  - 这里c1和c2为正常数，ξ和η为random(0,1)

公式(1)主要通过三个部分来更新粒子i的速度：一是粒子i前一时刻的速度，说明粒子的当前状态；二是粒子i当前位置与自己最好位置之间的距离，表示粒子对自身的认知；三是粒子i当前位置与群体最好位置之间的距离，体现群体的社会共享行为。

二. 基本算法

2. 算法模型

- 更新公式：
  - $v_{id} = v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ (1)
  - $x_{id} = x_{id} + v_{id}$ (2)
  - 这里c1和c2为正常数，ξ和η为random(0,1)

公式(2)利用粒子i的当前位置和下一时刻的速度更新下一时刻的位置。

二. 基本算法

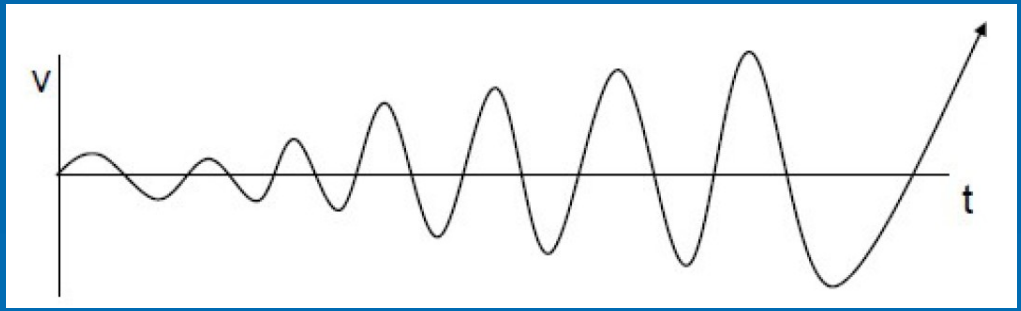
2. 算法模型

- 学习因子或加速系数c1和c2：
  - 参数c1使个体具有自我总结的能力
  - 参数c2使个体具有向种群中优秀个体学习的能力
  - 若c1=c2=0，完全随机搜索
  - 若c1>0, c2=0，个体均是独立的hill-climber
  - 若c1=0, c2>0，种群是一个随机的hill-climber
  - 若c1=c2>0，个体对pbest和gbest的学习程度相同
  - 若c1>c2，更适用于multimodel问题
  - 若c2>c1，更适用于unimodel问题
  - 若c1和c2均很小，尽可能保持原始速度
  - 若若c1和c2均很大，尽可能加速

二. 基本算法

2. 算法模型

- 学习因子或加速系数c1和c2：
  - 通常c1和c2需要设置足够大
  - 过高的取值会造成个体的速度越来越大



  - 粒子的速度被限制在[-Vmax, Vmax]的范围内，以防止溢出和保证算法稳定

二. 基本算法

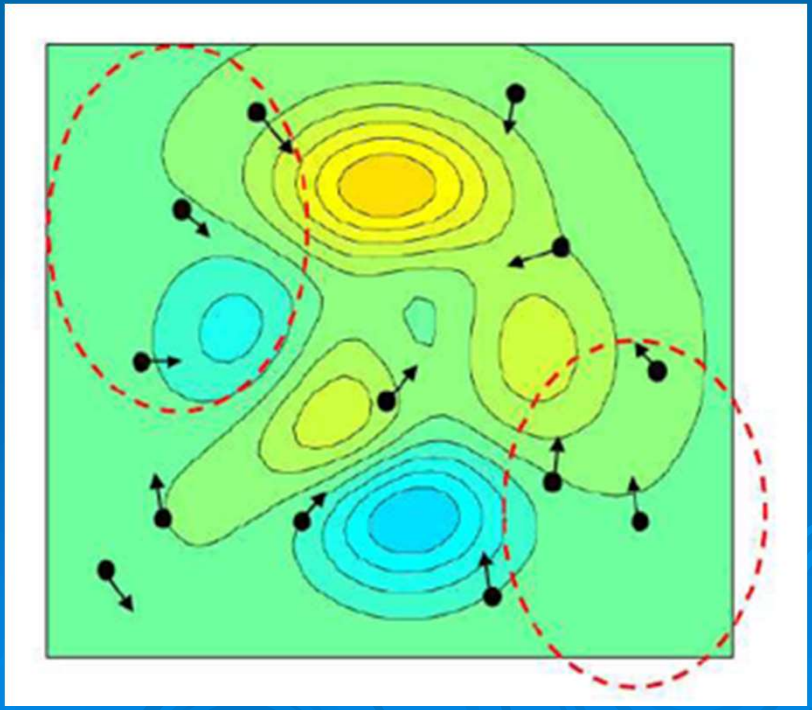
2. 算法模型

- 粒子群的拓扑结构
  - 全局版本算法将整个群体看作是一个全连通图，群体内所有个体共享一个邻域最优 **gbest**
  - 局部版本算法中每个个体的邻域将是整个群体的一个子集，此时影响个体的**gbest**取决于具体的拓扑结构。

二. 基本算法

2. 算法模型

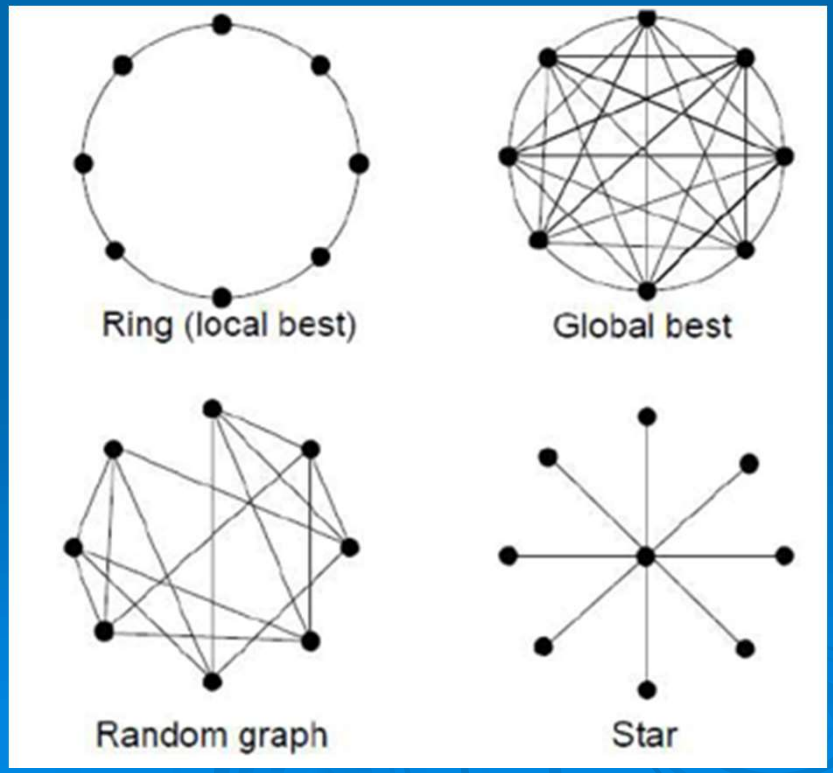
- 粒子群的拓扑结构
  - 基于空间的拓扑结构



二. 基本算法

2. 算法模型

- 粒子群的拓扑结构
  - 基于关联网络拓扑结构



二. 基本算法

3. 算法流程图

```
begin
   initialize a swarm of particles with random location and velocity;
   repeat
      for each particle i do
         adapt the velocity of particle i using Equation (1);
         update the position of particle i using Equation (2);
         evaluate objective function at the location of particle i;
         update pbest of particle i;
      endfor
      for each particle i do
         update gbest of particle i;
      endfor
   until a stop condition is met
end
```

三. 标准算法

1. 带有惯性权重的粒子群算法

- **Shi和Eberhart(1998)在基本算法中引入了惯性权重的概念，将公式(1)修改为：**
  - $v_{id} = wv_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta(p_{gd} - x_{id})$ **(3)**
  - w称为惯性权重，其大小决定了对粒子当前速度继承的多少，合适的选择可以使粒子具有均衡的探索和开发能力
    - ✓ $w \geq 1$：粒子加速，种群发散
    - ✓ $0 < w < 1$：粒子减速，种群收敛依赖于c_1和c_2
  - 标准算法仅使用Vmax进行初始化，将Vmax设定为每维变量的变化范围，而不必进行细致选择与调节

Shi, Y., Eberhart, R., A modified particle swarm optimizer, Proceedings of the 1998 IEEE World Congress on Computational Intelligence, pp. 69-73, 1998.

三. 标准算法

2. 带有收缩因子的粒子群算法

- **Clerc和Kennedy(2002)在基本算法中引入收缩因子χ，将公式(1)修改为：**
  - $v_{id} = \chi\left(v_{id} + c_1\xi(p_{id} - x_{id}) + c_2\eta\left(p_{gd} - x_{id}\right)\right)$ **(4)**
  - 这里$\varphi = c_1 + c_2 > 4$，$\chi = \frac{2}{\varphi+\sqrt{\varphi^2-4\varphi}}$
  - 经验设置: $c_1 = c_2 = 2.05$，$\chi = 0.7298$
  - 若采用这样的参数设置：$w = 0.7298$，$c_1 = c_2 = 1.49618$，则两种标准算法等价

Clerc, M., Kennedy, J., The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, 6(1):58-73, 2002.

三. 标准算法

3. 计算举例

- 求解无约束优化问题：2维Sphere函数（D=2）

$$minf(x) = \sum_{i=1}^{D} x_i^2$$

三. 标准算法

3. 计算举例

- 算法参数设置
  - 初始化范围：设定为[-10，10]
  - 种群大小：m=5
  - 惯性权重：$w = 0.7298$
  - 学习因子：$c_1 = c_2 = 1.49618$

初始化

x		v		f	pbest		gbest	
4.27	-74.36	83.37	79.24	5548.03	4.27	-74.36	-0.97	-23.08
-0.97	-23.08	-82.39	-61.80	533.65	-0.97	-23.08		
-14.43	-80.03	-74.09	-70.03	6613.72	-14.43	-80.03		
-37.28	37.97	-10.92	92.32	2831.87	-37.28	37.97		
6.55	64.04	-81.78	-13.89	4143.88	6.55	64.04		

第一代

x		v		f	pbest		gbest		ζ	η
62.45	-16.54	58.18	57.83	4173.52	62.45	-16.54	-0.97	-23.08	0.58	0.34
-61.10	-68.18	-60.13	-45.10	8381.20	-0.97	-23.08			0.13	0.61
-66.77	-131.15	-52.34	-51.11	21657.54	-14.43	-80.03			0.07	0.09
-34.25	105.35	3.03	67.37	12270.59	-37.28	37.97			0.17	0.20
-59.28	53.90	-65.82	-10.14	6418.72	6.55	64.04			0.28	0.55

第二代

x		v		f	pbest		gbest		ζ	η
									0.87	0.64
									0.54	0.73
									0.31	0.70
									0.06	0.19
									0.82	0.67

四. 算法设计

1. 惯性权重

- 固定权重
  - 即赋予惯性权重以一个常数值，一般来说，该值在0和1之间。固定的惯性权重使粒子在飞行中始终具有相同的探索和开发能力。显然，对于不同的问题，获得最好优化效果的这个常数是不同的，要找到这个值需要大量的实验。
  - 通过实验我们发现：种群规模越小，需要的惯性权重越大，因为此时种群需要更好的探索能力来弥补粒子数量的不足，否则粒子极易收敛；种群规模越大，需要的惯性权重越小，因为每个粒子可以更专注于搜索自己附近的区域。

四. 算法设计

1. 惯性权重

- 时变权重
  - 一般来说，希望粒子群在飞行开始的时候具有较好的探索能力，而随着迭代次数的增加，特别是在飞行的后期，希望具有较好的开发能力。所以希望动态调节惯性权重。
  - 可以通过时变权重的设置来实现，惯性权重的取值范围为：$[w_{min}, w_{max}]$，最大迭代次数为Iter_max，则第i次迭代时的惯性权重可以取为：

$$w_i = w_{max} - \frac{w_{max} - w_{min}}{Iter\_max} \times i$$

四. 算法设计

1. 惯性权重

- 随机权重
  - 随机权重是在一定范围内随机取值。例如可以取值如下：$w = 0.5 + \frac{Random}{2}$
  - 其中，**Random**为**0**到**1**之间的随机数。这样，惯性权重将在**0.5**到**1**之间随机变化，均值为**0.75**。

四. 算法设计

2. 学习因子

- **c1和c2同步时变**
  - 参照时变惯性权重的设置方法，将学习因子设置如下：设学习因子c1和c2的取值范围为$[c_{min}, c_{max}]$，最大迭代次数为Iter_max，则第i次迭代时学习因子取值为：

$$c_1 = c_2 = c_{max} - \frac{c_{max} - c_{min}}{Iter\_max} \times i$$

  - 一组较好的设置方案为：

$$c_{max} = 3，\ c_{min} = 0.25$$

四. 算法设计

2. 学习因子

- **c1和c2异步时变**
  - 使两个学习因子在优化过程中随时间进行不同的变化，所以我们这里称之为异步时变。这种设置的目的是在优化初期加强全局搜索，而在搜索后期促使粒子收敛于全局最优解。这种想法可以通过随着时间不断减小自我学习因子**c1**和不断增大社会学习因子**c2**来实现。
  - 在优化的初始阶段，粒子具有较大的自我学习能力和较小的社会学习能力，这样粒子可以倾向于在整个搜索空间飞行，而不是很快就飞向群体最优解；在优化的后期，粒子具有较大的社会学习能力和较小的自我学习能力，使粒子倾向于飞向全局最优解。

四. 算法设计

2. 学习因子

- **c1和c2异步时变**
  - 具体实现方式如下：

$$c_1 = (c_{1f} - c_{1i})\frac{iter}{Iter\_max} + c_{1i}$$

$$c_2 = (c_{2f} - c_{2i})\frac{iter}{Iter\_max} + c_{2i}$$

  - 这里c_{1i}，c_{1f}，c_{2i}，c_{2f}为常数，分别表示c_1和c_2的初始值和最终值，Iter_max为最大迭代次数，Iter为当前迭代数。
  - 研究中发现：$c_{1i} = 2.5$，$c_{1f} = 0.5$，$c_{2i} = 0.5$，$c_{2f} = 2.5$，这样的参数效果较好。

四. 算法设计

3. 同步更新与异步更新

- 同步更新

```
begin
   initialize a swarm of particles with random location and velocity;
   repeat
      for each particle i do
         adapt the velocity of particle i;
         update the position of particle i;
         evaluate objective function at the location of particle i;
         update pbest of particle i;
         update gbest of particle i;
      endfor
   until a stop condition is met
end
```

四. 算法设计

3. 同步更新与异步更新

- 同步更新
  - 粒子**pbest**和**gbest**的位置和速度更新同步
  - 群体搜索信息反馈较快
  - 适用于局部版本的粒子群算法

四. 算法设计

3. 同步更新与异步更新

- 异步更新

```
begin
   initialize a swarm of particles with random location and velocity;
   repeat
      for each particle i do
         adapt the velocity of particle i using Equation (1);
         update the position of particle i using Equation (2);
         evaluate objective function at the location of particle i;
         update pbest of particle i;
      endfor
      for each particle i do
         update gbest of particle i;
      endfor
   until a stop condition is met
end
```

四. 算法设计

3. 同步更新与异步更新

- 异步更新
  - 粒子**pbest**和**gbest**的位置和速度更新不同步
  - 群体搜索信息反馈较慢
  - 适用于全局版本的粒子群算法

四. 算法设计

4. Fully Informed Particle Swarm (FIPS)

- 个体的更新被领域内所有**neighbors**影响
  - 速度更新公式(1)修改为：

$$v_{id} = \chi\left(v_{id} + \frac{1}{K_i}\sum_{j=1}^{K_i}\xi_j(p_{jd} - x_{id})\right) \quad (5)$$

  - **FIPS**在很多问题上优于常规粒子群算法
  - **FIPS**的性能更加依赖于邻域拓扑结构

五. 算法应用实例

PSO求解函数优化问题

$$min\ f(x,y) = 3cos(xy) + x + y^2$$

$$\text{s.t. } x, y \in [-4, 4]$$

算法仿真过程：

（1）种群大小为100，粒子维度为2，迭代次数为200，学习因子均为1.5，惯性权重最大值和最小值分别设为0.8和0.4，速度最大值和最小值分别设为1和-1。

（2）初始化种群粒子位置和速度，粒子个体最优位置和最优值，粒子群全局最优位置和最优值。

五. 算法应用实例

PSO求解函数优化问题

$$min\ f(x,y) = 3cos(xy) + x + y^2$$

$$\text{s.t. } x, y \in [-4, 4]$$

算法仿真过程：

（3）计算动态惯性权重值，更新粒子位置和速度值，判断是否替换粒子最优位置和最优值，以及粒子群全局最优位置和最优值。

（4）判断是否满足终止条件：若满足，则结束搜索过程，输出最优解；若不满足，则继续进行迭代寻优。

五. 算法应用实例

PSO求解函数优化问题

$$min\ f(x,y) = 3cos(xy) + x + y^2$$

$$\text{s.t. } x, y \in [-4, 4]$$

作业要求：

- 参考代码，调试程序；
- 分析不同算法参数（种群大小、迭代次数、学习率、惯性权重）对算法性能的影响；
- 尝试**FIPS**模型，设计并检验算法求解效果（选作）。