第 3 章 粒子群优化

3.1 引　　言

粒子群优化算法(PSO)起源于 1994 年提出的复杂适应系统理论(Complex Adaptive System,CAS)。PSO 从鸟群种群行为特性中获得启发,通过模拟鸟群的行为方式,来发掘同一鸟群同步飞行的模式,以及在最优形式重组时突然改变方向的模式,将其应用于求解优化问题。

设想这样的一个场景:一群鸟在随机地寻找食物,在整个区域内只存在一块食物,所有的鸟在初始时都不知道食物的位置,但它们知道自己的位置距离食物还有多远,那么最简单的觅食策略就是“搜索目前距离食物最近的那只鸟的位置的周围区域”。

在 PSO 中,每个优化问题的潜在解都可以理解成多维搜索空间上的一个点,即“粒子”(particle),所有粒子都有一个被目标函数决定的适应度值(fitness value),影响它们移动的方向和速度,决定它们移动的距离。然后粒子们就在这样的条件下开始追随当前的最优粒子在解空间中进行搜索。

3.2 基本粒子群优化

PSO 的基本思想是:通过群体中个体之间的协作和信息共享在给定的高维搜索空间中搜寻最优解。PSO 中的粒子都存在一种基本行为:效仿别的粒子的成功行为,并继续保持自身的成功行为。这种基本行为使得整个群体表现出能够搜寻最优解的群体行为。

一个 PSO 维护了一群粒子,每个粒子代表一个潜在的解。和进化计算范式进行类比,一个群类类似于一个种群,其中的每个粒子类似一个个体。每个粒子在高维搜索空间中飞行,通过自身的经验和其周围个体的经验来不断调整自己的位置。令 $x_i(t)$表示粒子 i 在时刻 t 时处于搜索空间中的位置,t 为离散的时间步。粒子的位置通过对当前位置增加速度 $v_i(t)$来改变,即

$$x_i(t+1)=x_i(t)+v_i(t+1) \tag{3-1}$$

其中,$x_i(0)\sim U(x_{\min},x_{\max})$。

速度向量 $\boldsymbol{v}$ 是驱动优化过程的关键因素,它反映了粒子自身的经验累积,以及它邻域内的各个粒子与它的交互信息。粒子自身的经验累积通常称为认知部分,与粒子到它找到过的最佳位置(称为个体最佳位置)的距离成正比。交互信息则称为速度方程式的社会部分。

根据邻域的大小不同,将基本粒子群优化分成了全局最佳粒子群优化(gbest PSO)和局

部最佳粒子群优化(lbest PSO)。

3.2.1 全局最佳粒子群优化

gbest PSO 是指每个粒子的邻域都是整个群体。gbest PSO 使用的社会网络是星型拓扑结构(详见 3.2.6 节),粒子速度方程式中的社会部分代表着粒子从整个群体中获取到的交互信息。因此,在 gbest PSO 中,社会部分就是整个群体所找到的最佳位置,记作 $\hat{y}(t)$。

对于 gbest PSO,粒子 i 的速度计算公式如下:

$$v_{ij}(t+1)=v_{ij}(t)+c_1 r_{1j}(t)[y_{ij}(t)-x_{ij}(t)]+c_2 r_{2j}(t)[\hat{y}_j(t)-x_{ij}(t)] \quad (3\text{-}2)$$

式中,$v_{ij}(t)$表示粒子 i 在时刻 t 时在维度 j 上的速度;

$x_{ij}(t)$表示粒子 i 在时刻 t 时在维度 j 上的位置;

c_1和c_2分别界定了认知部分和社会部分贡献的加速常数(都为正数);

r_{1j}和r_{2j}是处于区间[0,1]上的随机值,服从均匀分布,为算法引入了随机性。

y_i表示粒子 i 的个体最佳位置,是指从第一次迭代开始找到的最佳位置。在 $t+1$ 时刻,个体最佳位置可以按照如下公式进行计算:

$$y_i(t+1)=\begin{cases} y_i(t), & f(x_i(t+1))\geqslant f(y_i(t)) \\ x_i(t+1), & f(x_i(t+1))<f(y_i(t)) \end{cases} \quad (3\text{-}3)$$

式中,$f:\mathbf{R}^{n_x}\rightarrow\mathbf{R}$ 表示适应度函数,n_x 是搜索空间的维度。适应度函数反映了某个解距离最优解的远近,定量地描述了一个粒子(解)的好坏。

$\hat{y}(t)$ 表示 t 时刻的全局最佳位置,定义如下:

$$\hat{y}(t)\in\{y_0(t),\cdots,y_{n_s}(t)\}\mid f(\hat{y}(t))=\min\{f(y_0(t)),\cdots,f(y_{n_s}(t))\} \quad (3\text{-}4)$$

式中,n_s是群中的粒子总数。$\hat{y}(t)$ 表示所有粒子到目前为止找到的最佳位置。在实际的计算过程中,取个体最佳位置中的最优值。全局最佳位置也可以通过当前群中粒子位置的最优值获得,即

$$\hat{y}(t)=\min\{f(x_0(t)),\cdots,f(x_{n_s}(t))\} \quad (3\text{-}5)$$

gbest PSO 算法如下。

算法 3-1 gbest PSO

```
创建并初始化一个维度为n_x 的群;
repeat
    for 每个粒子 i=1,…,n_s do
        //设置个体最佳位置
        if f(x_i)<f(y_i) then
            y_i=x_i;
        end
        //设置全局最佳位置
        if f(y_i)< f(ŷ) then
            ŷ = y_i;
        end
    end
    for 每个粒子 i=1,…,n_s do
        更新每个粒子的速度v_ij(t+1);
```

更新每个粒子的位置 $x_i(t+1)$；
end
until 满足终止条件；

3.2.2　局部最佳粒子群优化

lbest PSO 中每个粒子的邻域都比较小，速度方程式中的社会部分反映了粒子邻域的信息交互，以及局部环境的知识。lbest 使用的社会网络是环型拓扑结构(详见 3.2.6 节)。社会部分对粒子速度的贡献跟粒子与该粒子邻域最佳位置之间的距离成正比。速度计算公式如下：

$$v_{ij}(t+1)=v_{ij}(t)+c_1 r_{1j}(t)[y_{ij}(t)-x_{ij}(t)]+c_2 r_{2j}(t)[\hat{y}_{ij}(t)-x_{ij}(t)] \quad (3\text{-}6)$$

式中，$\hat{y}_{ij}$ 表示粒子 i 处于维度 j 的邻域内找到的最佳位置。局部最优粒子的位置 $\hat{y}_i$ 是在邻域N_i中的最佳位置，定义如下：

$$\hat{y}_i(t+1) \in \{N_i \mid f(\hat{y}_i(t+1))=\min\{f(x)\}, \forall x \in N_i\} \quad (3\text{-}7)$$

邻域N_i的定义如下：

$$N_i=\{y_{i-nN_i}(t), y_{i-nN_i+1}(t), \cdots, y_{i-1}(t), y_i(t), y_{i+1}(t), \cdots, y_{i+nN_i}(t)\} \quad (3\text{-}8)$$

对于大小为 n N_i的邻域，局部最佳位置也称为邻域最佳位置。

在基本 PSO 中，同一个邻域中的粒子之间是没有关系的，邻域的选择是基于粒子索引进行的。随着 PSO 的发展，又有人提出了一些新的寻找邻域的策略，例如根据空间的相似性来确定邻域。

基于粒子索引来确定邻域有如下两个原因。

(1) 计算代价降低。基于粒子索引的方法不需要确定粒子的空间顺序，如果要确定粒子的空间距离来决定邻域，则需要计算所有粒子之间的欧氏距离，计算的时间复杂度为$O(n_s^2)$。

(2) 在所有粒子之间传播信息。基于粒子索引的方法会忽略粒子在搜索空间中的位置，从而将了解的信息传播到所有粒子。

需要注意的是邻域的重叠问题。一个粒子可能属于多个邻域，邻域的互相重叠有助于邻域之间的信息共享，并确保了群体能够收敛于一个点，也就是全局最优粒子。gbest PSO 其实就是$n_{N_i}=n_s$时的 lbest PSO。

lbest PSO 算法如下。

算法 3-2　lbest PSO

创建并初始化一个维度为n_x 的群；
repeat
　for 每个粒子 $i=1,\cdots,n_s$ **do**
　　//设置个体最佳位置
　　if $f(x_i)<f(y_i)$ **then**
　　$y_i=x_i$；
　　end
　　//设置邻域最佳位置
　　if $f(y_i)<f(\hat{y})$ **then**

```
            ŷ = y_i;
        end
    end
    for 每个粒子 i=1,…,n_s do
        更新每个粒子的速度v_ij(t+1);
        更新每个粒子的位置 x_i(t+1);
    end
until 满足终止条件;
```

3.2.3 gbest PSO 与 lbest PSO 的比较

以上介绍的两种版本的 PSO 在“速度更新的社会部分导致粒子向全局最优解移动”方面是相似的。

两种算法在收敛性方面的主要区别如下。

(1) gbest PSO 有更大的粒子互联度,所以它收敛得会比 lbest PSO 更快。然而,这种更为快速的收敛却导致了较差的多样性。

(2) lbest PSO 具有更好的多样性(能够覆盖搜索空间的绝大部分),因此不容易陷入局部极值。一般来说(取决于研究的问题),邻域的结构,例如 lbest PSO 中使用的环形拓扑结构,能够进一步提升性能。

3.2.4 速度成分

速度方程式由三部分组成。

(1) 之前的速度,$v_i(t)$,它代表之前飞行方向的记忆,即不久之前的移动情况。这种记忆可以看作一种动量或者趋势,能够防止粒子突然改变方向,继续偏向现在的方向。这个分量也被称为惯性分量。

(2) 认知部分,$c_1 r_1(y_i - x_i)$,可以评估粒子 i 相较于过去的表现。在某种程度上,认知部分模拟了粒子的最佳位置的个体记忆。这一项的作用是可以将粒子拉回到它们各自的最佳位置,表明了粒子回到过去最佳位置的一种趋势。Kennedy 和 Eberhart 也将认知部分称为粒子的“怀旧”。

(3) 社会部分,在 gbest PSO 中表示为 $c_2 r_2(\hat{y} - x_i)$,在 lbest PSO 中则表示为 $c_2 r_2(\hat{y}_i - x_i)$。它评估了粒子相对于群体或者其邻域的表现。社会部分可以认为是粒子试图达到的一种群体的标准,其作用是将粒子拉回到其邻居找到的最佳位置。

认知部分和社会部分的作用分别由随机变量 $c_1 r_1$ 和 $c_2 r_2$ 来控制。

3.2.5 几何描述

速度方程式的效果可以很容易地在一个二维向量空间中进行表示。为了更好地进行说明,先考虑二维搜索空间中的一个粒子。

图 3-1 展示了粒子的一种典型的移动方式。为了方便说明,图中粒子的下标被省略。图 3-1(a)描述了时刻 t 时的群体状态。注意新的位置 $x(t+1)$ 向全局最佳位置 $\hat{y}(t)$ 靠拢了。对于时间步 $t+1$,如图 3-1(b)所示,假定个体最佳位置不变,图中说明了惯性分量、认

知部分和社会部分是如何帮助粒子向全局最佳位置移动的。

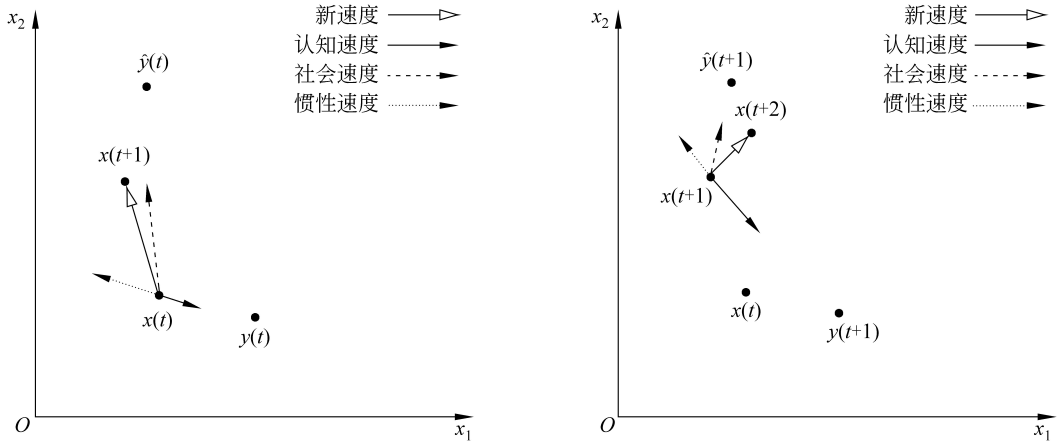


图 3-1　二维向量空间下的单一粒子的速度更新和位置更新示意图

当然，一个粒子的位置是有可能超越全局最佳位置的，主要是由于惯性成分的存在。这将会导致以下两种情况：

（1）新的位置，是在当前全局最佳位置的前提下产生的，因此有可能超越当前的全局最佳位置。在这种情况下，新的全局最佳位置就会吸引其他所有粒子向它移动。

（2）新的位置仍然不如当前的全局最佳位置，在之后的时间步骤之中，速度方程式中的认知部分和社会部分将使得粒子朝着全局最佳位置的方向移动。

粒子在所有位置上更新操作的累积效应就是每个粒子都会收敛到连接全局最佳位置和粒子个体最佳位置连线上的一点。

下面回到多个粒子的情况。图 3-2 中使用了 gbest PSO 优化，求解一个包含两个变量 x_1 和 x_2 的二维函数最小化问题。最优值在原点处，用符号“×”标记。图 3-2(a)表示了 8 个粒子的初始位置，同时全局最优值如图所示。在时刻 t 时，认知部分对每个粒子的速度贡献为零，只有社会部分会对粒子的移动产生影响。需要注意的是，全局最优值是不会改变的。

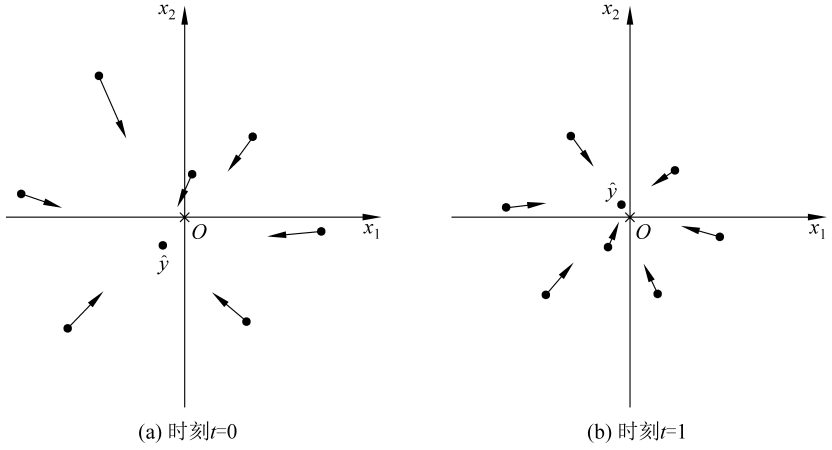
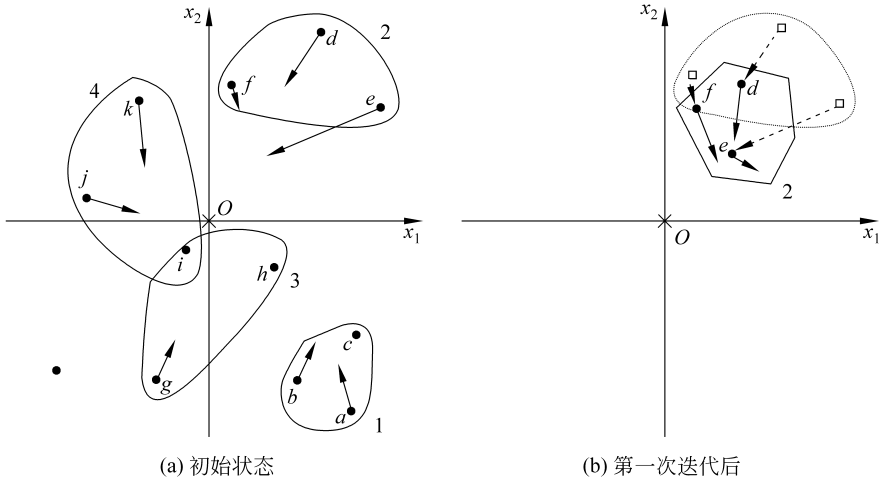


图 3-2　多粒子的 gbest PSO 优化示意图

图 3-2(b)表示经过一次迭代后的所有粒子的新位置,并且已经找到了一个新的全局最佳位置。图中所有粒子的速度方程式都受到影响,都朝着新的全局最佳位置移动。

最后,关于 lbest PSO,图 3-3 展示了粒子是如何受到其邻居的影响的。为了保证图示的可读性,只展示了部分粒子的移动情况,并只标出了向全局最优值聚合的速度方向。在邻域 1 中,粒子 a 和粒子 b 都向粒子 c 移动,c 是此邻域内的最优值。对于邻域 2 来说,粒子 d 和粒子 e 都向粒子 f 移动。在下一次迭代中,粒子 e 将会是邻域 2 内的最优值。图 3-3(b)显示粒子 d 和粒子 f 都向粒子 e 移动。图中的方块代表粒子之前的位置。值得注意的是,e 依然是邻域 2 内的最优值,整体的运动都是朝着最小值的方向移动的。



(a) 初始状态　　(b) 第一次迭代后

图 3-3　多粒子 lbest PSO 优化示意图

3.2.6　社会网络结构

社交互动是驱动 PSO 工作的本质属性。群体中的粒子之间相互学习,并根据获取的知识,使自身变得更接近“比自己更好”的邻居粒子。PSO 的社会结构由重叠的邻域信息决定,邻域内的粒子之间互相影响。这个特点类似动物观察的行为,区域中的生物个体最有可能受到其周围邻居个体的影响,并且“更为成功”的个体对其周围个体的影响力也会更大。

在 PSO 中,同一个邻域内的粒子通过和每个粒子交换成功信息来进行交流。所有粒子都向着被认为是更好的位置移动。PSO 的表现很大程度上依赖社会网络的结构。通过社会网络的信息流取决于以下三方面。

(1) 网络节点之间的连接程度。

(2) 聚类的数量(当一个节点的邻居也是其他节点的邻居时会发生聚类)。

(3) 节点之间的平均最短距离。

在高度互联的社会网络中,大多数的个体是可以互通的,使得最佳位置的信息能够迅速地在社会网络中进行传递。从优化的角度来说,这意味着搜索最优解的收敛速度将会更快。然而,对于高度互联的网络,更快的收敛速度将会导致更容易陷入局部极值,主要是因为搜

索空间的覆盖范围小于互联程度较低的网络。对于在邻域中存在大量聚类的稀疏连接网络,还可能发生搜索空间未被充分覆盖,导致不能获得最优解的情况,因为每个聚类只能包含搜索空间的一小部分。在这样的网络结构中存在若干聚类,而不同的聚类之间通常联系很少,因此,搜索空间某一有限部分的信息在聚类之间的流动速度将会非常缓慢。

目前已经存在多种 PSO 的社会网络结构,这里介绍一些主要的社会网络结构。其他部分可以参考文献中的内容。

- 星型(star)社会网络结构。如图 3-4(a)所示,在这种网络结构中,所有粒子互相连接,因此每个粒子都可以和其他所有粒子进行通信。在这种情况下,每个粒子都向着全局最佳位置移动。一般 gbest PSO 都是用这种网络结构实现的。
- 环型(ring)社会网络结构。如图 3-4(b)所示,每个粒子和其直接邻居进行通信,试图从邻域空间中寻找最佳位置,并向着最佳位置移动。信息在社交网络中的流动速度较慢,因此收敛速度也会较慢。但是,与星型社会网络结构相比,搜索空间的大部分都会被覆盖。这种网络结构能够在处理多模问题时提供更好的性能。一般 lbest PSO 都是用这种网络结构实现的。
- 轮型(wheel)社会网络结构。如图 3-4(c)所示,邻域中的个体是相互独立的。一个粒子作为中心焦点,所有信息都通过焦点粒子进行传递。焦点粒子比较邻域中所有粒子的性能,并根据最优的邻居调整自己的位置。如果焦点粒子的新位置代表了更好的解,那么它就会将这个信息传递给所有的粒子。这种网络结构同样降低了信息的传播速度。
- 金字塔型(pyramid)社会网络结构。如图 3-4(d)所示,这种网络结构形成了三维有线连接框架。
- 四聚类型(four clusters)社会网络结构。如图 3-4(e)所示,四个集群由集群之间的两个连接构成,集群内的粒子分别和五个邻居粒子相连。
- 冯·诺依曼(von Neumann)社会网络结构。如图 3-4(f)所示,粒子以网络结构连接,这种结构在许多实证研究中显示出了比其他网络结构更好的性能。

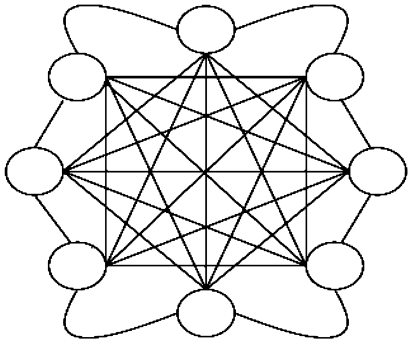
3.2.7 算法的其他部分

1. 算法初始化

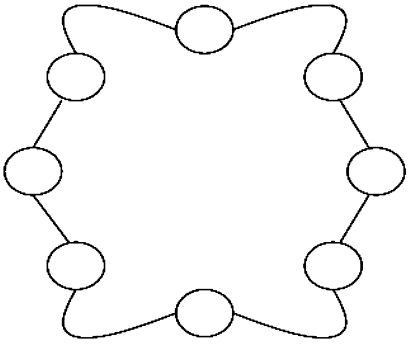
PSO 的第一步是群的初始化,并且设置控制参数。在基本 PSO 优化中,控制参数c_1和c_2、初始速度、粒子位置和个体最佳位置都需要赋初始值。在 lbest PSO 中,还需要制定邻域范围内的粒子个数。

一般情况下,粒子的初始位置是在搜索空间中均匀分布的,但是,PSO 的搜索效果会很大程度上依赖初始群的分布,即覆盖了多少搜索空间以及粒子的分布情况。如果初始群没有覆盖到极值所在的区域,那么 PSO 优化将很难搜索到这个值。或者粒子的速度方程式的惯性因素将粒子引向了未覆盖区域,那么也有可能搜索到这个极值。

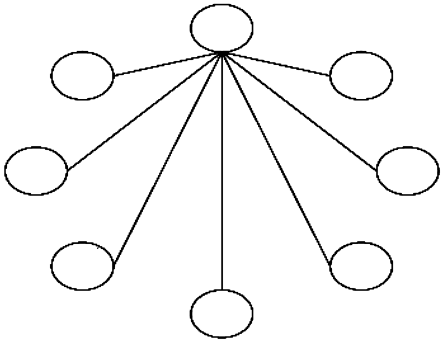
假设极值在$x_{\min}$和$x_{\max}$所定义的区域内,它们分别表示在每一维上的最小值和最大值,粒子位置的初始化可以表示为



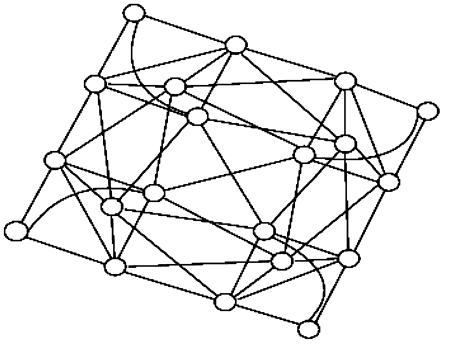
(a) 星型社会网络结构



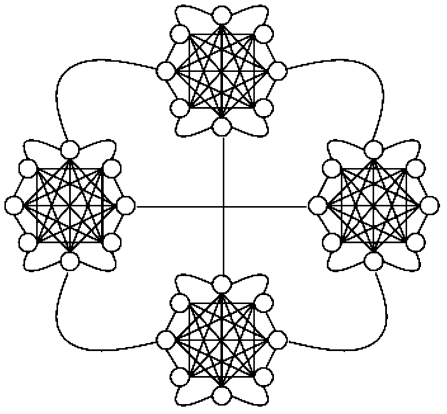
(b) 环型社会网络结构



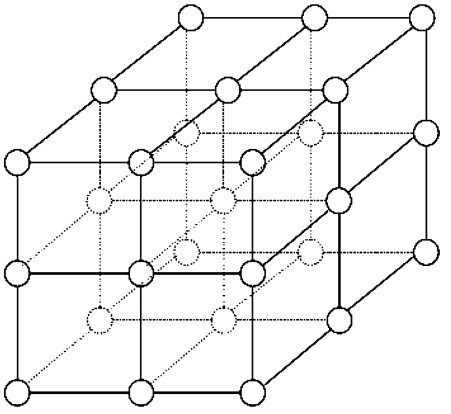
(c) 轮型社会网络结构



(d) 金字塔型社会网络结构



(e) 四聚类型社会网络结构



(f) 冯·诺依曼社会网络结构

图 3-4　社会网络结构示例

$$x(0)=x_{\min,j}+r_j(x_{\max,j}-x_{\min,j}),j=1,\cdots,n_x \tag{3-9}$$

式中，$r_j\sim U(0,1)$。

初始速度可以初始化为 0，即

$$v_i(0)=0,\ i=1,\cdots,n_s \tag{3-10}$$

位置的随机初始化可以保证搜索的随机性，速度也可以随机初始化，但不能太大，过大的初始速度会包含较大的惯性成分，导致每次位置更新时的幅度较大，这样很容易引起粒子偏离搜索区域，使得群难以收敛。

粒子的个体最佳位置可以初始化为粒子在时刻 $t=0$ 时的位置，即

$$y_i(0)=x_i(0),i=1,\cdots,n_s \tag{3-11}$$

2. 算法的终止条件

选择终止条件时，需要考虑两个因素：

(1) 终止条件不能在种群尚未进化完全时停止优化，否则会得到次优解而不是最优解。

(2) 终止条件不能导致频繁地计算适应度函数，否则计算的复杂度会急剧提升。

目前经常选用的终止条件如下：

(1) 设置最大迭代次数。当达到最大迭代次数时，就终止算法。

(2) 设置可接受的解的范围。当解达到可接受的程度时，即终止算法。

(3) 在一定迭代次数内，解没有得到优化，则终止算法。

3.3　粒 子 轨 迹

3.3.1　简化 PSO 模型的粒子轨迹

对于一个简化的 PSO 模型，只包含一个粒子($n_s=1$)和一个维度($n_x=1$)，同时假设个体最佳和全局最佳位置相同且保持不变，即 $\hat{y}(t)=y(t)=p$，进而假设一个确定的 PSO 模型(没有随机因素的存在)，没有惯性权重、速度限制以及收缩系数。在以上假设下，速度和位置更新方程式表示如下：

$$v(t)=v(t-1)+(\phi_1+\phi_2)(p-x(t-1)) \tag{3-12}$$

$$x(t)=x(t-1)+v(t) \tag{3-13}$$

式中，$\phi_1=c_1r_1$，$\phi_2=c_2r_2$。粒子和维数的下标都被省略掉了，因为只考虑了单个粒子和一个维度的情况。令 $\phi=\phi_1+\phi_2$，则速度方程式和位置方程式变为

$$v(t)=v(t-1)+\phi x(t-1)+\phi p \tag{3-14}$$

$$x(t)=x(t-1)+v(t-1)+\phi x(t-1)+\phi p \tag{3-15}$$

进一步可得

$$x(t-1)=x(t-2)+v(t-1) \tag{3-16}$$

$$v(t-1)=x(t-1)+x(t-2) \tag{3-17}$$

综上各式，可以得到如下非齐次递归关系：

$$x(t)=(2-\phi)x(t-1)+x(t-2)+\phi p \tag{3-18}$$

初始条件为 $x(0)=x_0$，$x(1)=(1-\phi)+v_0+\phi p$。

改写成矩阵-向量乘法形式：

$$\begin{bmatrix} x(t) \\ x(t-1) \\ 1 \end{bmatrix}=\begin{bmatrix} 2-\phi & -1 & \phi p \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x(t-1) \\ x(t-2) \\ 1 \end{bmatrix} \tag{3-19}$$

根据上述矩阵方程式，可得特征方程式如下：

$$(1-\lambda)(1-\lambda(2-\phi)+\lambda^2)=0 \tag{3-20}$$

可得如下解：

$$\begin{cases}\alpha=\dfrac{2-\phi+\gamma}{2}\\ \beta=\dfrac{2-\phi-\gamma}{2}\\ \gamma=\sqrt{\phi^2-4\phi}\end{cases}$$

因此，非齐次递归式(3-18)可以写成

$$x(t)=k_1+k_2\,\alpha^t+k_3\,\beta^t \tag{3-21}$$

式中的三个系数是由系统的初始条件所确定的常数，由于有三个未知系数，因此需要列方程式如下：

$$\begin{bmatrix}x(0)\\ x(1)\\ x(2)\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\ 1 & \alpha & \beta\\ 1 & \alpha^2 & \beta^2\end{bmatrix}\begin{bmatrix}k_1\\ k_2\\ k_3\end{bmatrix} \tag{3-22}$$

通过求解上述方程式，可得

$$k_1=p$$
$$k_2=x_0-p-k_3$$
$$k_3=\frac{(x_0-p)(\gamma+\phi)}{2\gamma}-\frac{v_0}{\gamma}$$

令 $x_0=p$，可得粒子的轨迹方程式：

$$x(t)=\frac{v_0}{\gamma}(\alpha^t-\beta^t)+p \tag{3-23}$$

式中，$\gamma=\sqrt{\phi^2-4\phi}$。当$\phi^2-4\phi>0$时，$\gamma\in\mathbf{R}$，否则 $\gamma\in\mathbf{C}$。下面分情况进行讨论。

1. $\gamma\in\mathbf{R}$

(1) $\phi=0$，可得递归轨迹方程式如下：

$$x(t)=2x(t-1)-x(t-2) \tag{3-24}$$

粒子的轨迹如下：

$$x(t)=(x_0-p)v_0t \tag{3-25}$$

如果 $x_0=p$，则粒子会沿着初始方向一直移动下去。

(2) $\phi=4$，可得递归轨迹方程式如下：

$$x(t)=-2x(t-1)-x(t-2)+4p \tag{3-26}$$

粒子的轨迹如下：

$$x(t)=((x_0-p)+(2(x_0-p)-v_0)t)(-1)^t+p \tag{3-27}$$

如果 $x_0=p$，则轨迹方程式变成

$$x(t)=-v_0t\,(-1)^t+p \tag{3-28}$$

由于$(-1)^t$的存在，粒子将在连续的时间步上做相反方向的运动。

(3) $\phi>4$，粒子轨迹如式(3-23)所示，粒子将做振动运动(以指数函数为界)。初始速度v_0越大，粒子的步长也会越大。

综上可知，当 $\gamma\in\mathbf{R}$ 时，粒子的步长的增大需要加以限制，以保证粒子不会偏离搜索空间。

2. $\gamma \in \mathbf{C}$

(1) $0<\phi<4$，轨迹方程式如下：

$$x(t)=k_2\,\alpha^t+k_3\,\beta^t \tag{3-29}$$

式中，

$$\alpha=\frac{2-\phi+\gamma'}{2}$$

$$\beta=\frac{2-\phi-\gamma'}{2}$$

$$k_3=\frac{(x_0-p)\,(\gamma'+\phi)}{2\,\gamma'}-\frac{v_0}{\gamma'}$$

$$k_2=x_0-p-k_3$$

$$\gamma'=\mathrm{i}\sqrt{|\,\phi^2-4\phi\,|}$$

对于任意一个复数 $z\in\mathbf{C}$，它的模由 l_2 范数度量，即

$$\|z\|=\sqrt{(R(z))^2+(G(z))^2} \tag{3-30}$$

需要注意的是，复数可以写成如下形式：

$$z^t=(\|z\|\mathrm{e}^{\mathrm{i}\theta})^t=\|z\|^t\,\mathrm{e}^{\mathrm{i}\theta t}=\|z\|^t(\cos\theta t+\mathrm{i}\sin\theta t) \tag{3-31}$$

式中，$\theta=\arg(z)$。

若 $x_0=p$，则轨迹方程式变成

$$x(t)=\frac{2\,v_0}{\|\gamma'\|}\sin\left(\arctan\left(\frac{\|\gamma'\|t}{2-\phi}\right)\right) \tag{3-32}$$

式(3-22)表明，一个具有复数值 γ 的粒子的轨迹是一条正弦波，其中初始条件和参数的选择将决定波的振幅和频率，其实决定的是粒子的方向和步长。振幅和频率的表达形式如下：

$$A=\frac{2\,v_0}{\|\gamma'\|} \tag{3-33}$$

$$V=\frac{\arctan\left(\dfrac{\|\gamma'\|t}{2-\phi}\right)}{2\pi} \tag{3-34}$$

周期 $T=\dfrac{1}{V}$，轨迹方程式的周期性可以使粒子重复地搜索已经访问过的空间，除非其邻域中的另一个粒子找到了更好的解。

(2) $\phi=2$，由式(3-18)可得

$$x(t)=-x(t-2)+\phi p=v_0\sin\left(\frac{t\pi}{2}\right)+2p \tag{3-35}$$

(3) $0<\phi<2$ 或 $2<\phi<4$，轨迹方程式如下：

$$x(t)=T_s\sin\theta t+r_c\cos\theta t \tag{3-36}$$

式中，

$$T_s=\frac{2\,v_0-\phi x_0-\phi p}{\gamma}$$

$$T_c=x_0-p$$

$$\theta=\arctan\left(\frac{\gamma}{|\,2-\phi\,|}\right)$$

$$\gamma=\sqrt{|\phi^2-4\phi|}$$

当 $x_0=p$，$0<\phi\leqslant 2-\sqrt{3}$ 和 $2+\sqrt{3}<\phi<4$ 时，正弦波的振幅随着 ϕ 的减小而增大，因为 $\|\gamma'\|<1$，频率也随着周期的增大而减小。

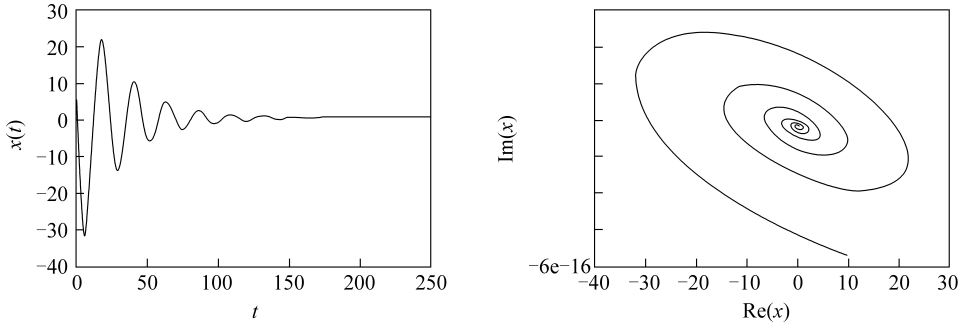
当 $2-\sqrt{3}<\phi\leqslant 2$ 和 $2<\phi\leqslant 2+\sqrt{3}$ 时，$\|\gamma'\|>1$，由于 $\|\gamma'\|$ 的上界为 $\sqrt{5}$，正弦波的振幅接近 v_0。

3.3.2 轨迹示例

下面是一个简化 PSO 系统的部分粒子轨迹展示。这些图例展示出了一个简化粒子的不同类型的行为，比如收敛、周期性和发散等行为。其中的参数 ϕ_1 和 ϕ_2 是随机选取的，由此产生的轨迹图像展示出了随机粒子的探索能力。

为了简化模型，令 $y=1.0$，$\hat{y}=0$，初始条件为 $x(0)=10$，$x(1)=10-9\phi_1-10\phi_2$。由此可以构建一个简单的模型。下面是模型的部分行为的图例展示。

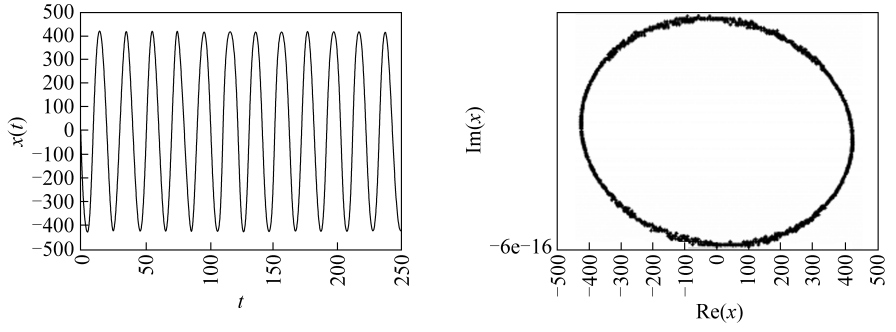
如图 3-5 所示，简化的系统收敛到了一个平衡点。图 3-5(a)显示的是粒子的探索行为，在探索的初期，粒子需要覆盖尽可能多的搜索空间，随着时间步数的增大，振荡幅度逐渐减小，直至归零。其中的参数 $\omega=0.5$，$\phi_1=\phi_2<1.4$。



(a) 时间域　　(b) 相空间

图 3-5　简化 PSO 系统的收敛行为

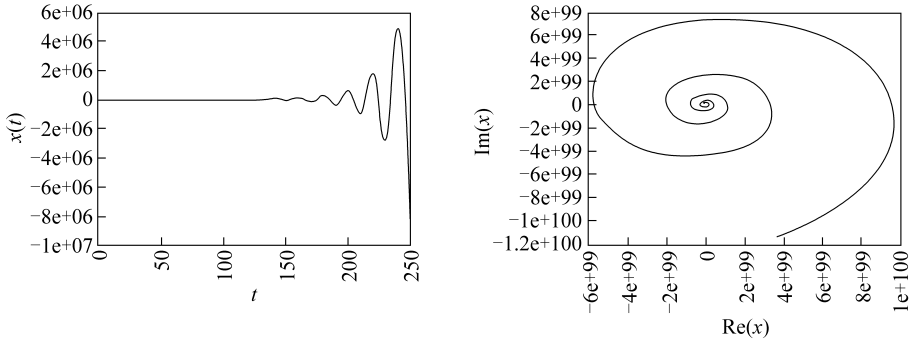
图 3-6 展示的是系统的周期性行为，粒子并没有收敛到平衡点。其中的参数 $\omega=1.0$，$\phi_1=\phi_2=1.999$。



(a) 时间域　　(b) 相空间

图 3-6　简化 PSO 系统的周期性行为

图 3-7 展示的是系统的发散行为，在探索初期，粒子的振荡幅度很小，随着时间步数的增大，振荡幅度逐渐变大，最后导致粒子偏离了搜索空间。



(a) 时间域　　(b) 相空间

图 3-7　简化 PSO 系统的发散行为

3.4　收敛性证明

3.4.1　局部收敛性

关于基本 PSO 优化的收敛性证明，证明基本 PSO 不是一个局部最小化算法。首先考虑单模最优化问题。

定义：

$$x_0 = \underset{x_i}{\operatorname{argmax}}\{f(x_i)\}, \quad i = 1,2,\cdots,n_s \tag{3-37}$$

在最小化问题中，x_0 使得 f 的值最大，因此 x_0 表示种群中最差的粒子，作为初始位置。

定义紧集：

$$L_0 = \{x \in S : f(x) \leqslant f(x_0)\} \tag{3-38}$$

紧集表示的是所有满足 f 值小于或等于 $f(x_0)$ 的粒子。

假设所有粒子都在函数 f 的同一个“盆地”中，因此 $x_i, y_i \in L_0$。由速度方程式和位置方程式可得，PSO 的方程式 A 定义如下：

$$A(\hat{y}_t, x_{i,t}) = \begin{cases} \hat{y}_t, & f(g(x_{i,t})) \geqslant f(\hat{y}_t) \\ g(x_{i,t}), & f(g(x_{i,t})) < f(\hat{y}_t) \end{cases} \tag{3-39}$$

式中，$g(x_{i,t})$ 表示 PSO 优化中的更新操作，$\hat{y}_t$ 表示所有粒子在时间步 t 之前所找到的最佳位置。

证明的过程考虑 $x_{i,t}$ 值的计算是对 g_1、g_2 和 g_3 三个方程式的成功应用。每个方程式都在原有结果的基础上增加了一项，代表着位置更新中的一部分。即

$$x_{i,t+1} = g(x_{i,t}) = g_1(x_{i,t}) + g_2(x_{i,t}) + g_3(x_{i,t}) \tag{3-40}$$

式中，

$$g_1(x_{i,t}) = x_{i,t} + \omega v_{i,t} \tag{3-41}$$

$$g_2(x_{i,t})_j = c_1 r_{1j}(t)(y_{ij,t} - x_{ij,t}), \quad j = 1,\cdots,n_x \tag{3-42}$$

$$g_3(x_{i,t})_j = c_2 r_{2,j}(t)(y_{ij,t} - x_{ij,t}), \quad j = 1,\cdots,n_x \tag{3-43}$$

$g_k(x_{i,t})_j$ 表示向量方程式 g_k 的第 j 维。

根据紧集的定义，假设所有粒子在初始状态时都属于 L_0，即

$$x_{i,0}, y_{i,0} \in L_0 \tag{3-44}$$

因此 $\hat{y}_0 \in L_0$，令 $g^N(x_{i,t})$ 表示更新方程式 g 在 $x_{i,t}$ 上的 N 次应用，则有引理 3.1。

引理 3.1 存在一个值 $N(1 \leqslant N \leqslant +\infty)$，使得 $\|g^{t'}(x_{i,t}) - g^{t'+1}(x_{i,t})\| < \epsilon$，$\forall t' \geqslant N$，$\epsilon > 0$，在 $\max\{\|\alpha\|, \|\beta\|\} < 1$ 时，受参数 ω、ϕ_1、ϕ_2 的值选择的影响。

证明：

$$\lim_{t \to +\infty} x(t) = \lim_{t \to +\infty} (k_1 + k_2 \alpha^t + k_3 \beta^t) = \frac{\phi_1 y + \phi_2 \hat{y}}{\phi_1 + \phi_2} \tag{3-45}$$

由递推关系可得

$$x(t+1) - x(t) = v(t+1) \tag{3-46}$$

因此，当 $\max\{\|\alpha\|, \|\beta\|\} < 1$ 时，有

$$\lim_{t \to +\infty} v(t+1) = \lim_{t \to +\infty} (x(t+1) - x(t)) = \lim_{t \to +\infty} (k_2 \alpha^t (\alpha - 1) + k_3 \beta^t (\beta - 1)) = 0 \tag{3-47}$$

进一步，有

$$x(t+1) = x(t) + v(t+1) = x(t) + \omega v(t) - x(t)(\phi_1 + \phi_2) + y\phi_1 + \hat{y}\phi_2 \tag{3-48}$$

根据式(3-45)可得，在 t 的极限情况下，$x(t+1) = x(t)$。根据式(3-46)可得，在 t 的极限情况下，$v(t) = 0$。因此，在极限条件下，有

$$x(t) = x(t) - x(t)(\phi_1 + \phi_2) + y\phi_1 + \hat{y}\phi_2 \to 0 \tag{3-49}$$

当 $x(t) = y = \hat{y}$ 时，式(3-49)成立，系统将会收敛。需要注意的是，这个点不一定是一个局部最优值。

根据上述证明可以发现，一旦算法达到如下状态，即 $x(t) = y = \hat{y}$，表示算法不会取得进步，则算法就会终止，但此状态中的 $\hat{y}$ 不一定是全局最优值，可能只是一个局部最优值。

综上所述，基本 PSO 并非是一个局部搜索算法，因为它不能保证从任意一个初始状态都能收敛到一个局部最优值。

3.4.2 全局收敛性

3.4.1 节已经证明了，基本 PSO 不能保证收敛到一个局部极小值，因此也不能保证可以收敛到全局极小值。在本节中继续讨论基本 PSO 的全局收敛性。

由收敛性条件可知，任意一个集合 $A \subset S$，必须有一个非零的概率被算法能够无限地进行检查。因此，条件必须在所有时间步上都要满足，或者在有限个数的时间步上例外。由于 $R_\epsilon \subset S$，则存在一个非零的概率使得样本可以在 S 的任何子集中产生，并且一个样本 $x' \in R_\epsilon$ 最终会产生。同时，一个算法如果一致性地忽略搜索空间的任何区域，则不能保证可以产生一个样本 x' 使得 $x' \in R_\epsilon$，除非算法有关于全局最小值的先验信息。因此，为了保证全局最小值能够以渐近概率 1 被发现，一个算法必须可以产生遍布于 S 的无限多的样本。

引理 3.2 基本 PSO 不满足全局搜索的收敛条件。

证明： 令 $\mathcal{M}_{i,t}$ 表示粒子 i 在第 t 步的采样空间的支撑。所有粒子的采样空间的并集必须覆盖 S，即

$$S \subseteq \bigcup_{i=1}^{n_s} \mathcal{M}_{i,t} \tag{3-50}$$

根据非齐次递推关系可得，$\mathcal{M}_{i,t}$的计算公式如下：

$$\begin{aligned}\mathcal{M}_{i,t} &= (1+\omega-\phi_1-\phi_2)x_{ij,t} - \omega x_{ij,t-2} + \phi_1 y_{ij} - \phi_2 \hat{y}_j \\ &= x_{ij,t-1} + \omega(x_{ij,t-1} - x_{ij,t-2}) + \phi_1(y_{ij} - x_{ij,t-1}) + \phi_2(\hat{y}_j - x_{ij,t-1})\end{aligned} \tag{3-51}$$

式中，$x_{ij,t}$表示第 i 个粒子在时刻 t 时处于维度 j 时的值。同时注意 $0 \leqslant \phi_1 \leqslant c_1$，$0 \leqslant \phi_2 \leqslant c_2$。

因此，$\mathcal{M}_{i,t}$是一个超矩形，其中的参数由 ϕ_1 和 ϕ_2 决定，其中一个角定义为 $(\phi_1, \phi_2) = (0,0)$，另一个角定义为 $(\phi_1, \phi_2) = (c_1, c_2)$，很明显，当满足如下条件时，

$$\max\{c_1 | y_{ij} - x_{ij,t-1} |, c_2 | \hat{y}_j - x_{ij,t-1} |\} < 0.5 \times \text{diameter}_j(S) \tag{3-52}$$

有 $\mathcal{L}(\mathcal{M}_{i,t} \cap S) < m(S)$，而不管超矩形的角的位置如何；其中 $\text{diameter}_j(S)$ 是 S 位于维度 j 上的长度。

根据 3.4.1 节的内容可知，每一个 x_i 会收敛到一个稳定的点，$(c_1 y_i + c_2 \hat{y})/(c_1 + c_2)$，因此，当 $t \to +\infty$ 时，$\mathcal{M}_{i,t}$的长度趋于 0。由于每个 $\mathcal{M}_{i,t}$ 的体积会随着时间步 t 的增大而减小，因此 $\mathcal{L}\left(\bigcup_{i=1}^{n_s} \mathcal{M}_{i,t}\right)$ 也会随之减小。这表明，除了 $t < t'$（其中 t' 是有限的），有

$$\mathcal{L}\left(\bigcup_{i=1}^{n_s} \mathcal{M}_{i,t} \cap S\right) < L(S) \tag{3-53}$$

式中，$\mathcal{M}_{i,t}, i=1$ 不能覆盖整个 S，因此，存在一个有限的 t''（$t''=0$ 也是有效状态），使得对于所有 $t' \geqslant t''$，存在一个可能的不相交的集合 $A \subset S$，并且满足

$$\sum_{i=0}^{n_s} \mu_{i,t}(A) = 0 \tag{3-54}$$

因此，基本 PSO 不满足全局搜索的收敛条件，不能作为一种全局优化算法。

3.5 单解与多解粒子群优化

基于基本 PSO 优化，提出了大量的变体，主要目的是提高解的精度、群体的多样性和收敛性。大部分变体用于求解单解问题，部分变体用于求解多解、多目标优化、动态优化、约束优化以及在离散空间进行优化的问题。

3.5.1 单解粒子群优化

根据操作的不同，单解 PSO 可以分为以下 6 种类型。

(1) 基于社会结构的 PSO：通过引入一个新的社会拓扑结构，或者改变计算个体最佳与邻域最佳位置的方法的 PSO 变体。

(2) 混合算法：结合了进化计算的思想（如选择、交叉、变异等）、蚁群优化算法（ACO）的概念和操作的 PSO 变体。

(3) 基于子群的 PSO：实现子群策略的 PSO，这些子群之间可以是互相关联的，也可以是各自独立的。

(4) 拟基因算法：基于基本 PSO，并加入了局部搜索策略的拟基因算法。

(5) 多次重启 PSO：在满足一定条件的情况下，重启整个种群或者部分种群的 PSO

变体。

(6) 排斥算法：为了提高粒子的多样性，从而加入了避免粒子之间互相排斥或碰撞的机制的 PSO。

以下介绍基于社会结构的 PSO、混合算法、基于子群的 PSO 和多次重启 PSO。

1. 基于社会结构的 PSO

1) 空间社会网络

这种情况下的邻域一般是在粒子索引的基础上形成的。例如，在一个简单的环型社会网络中，索引为 i 的粒子的直接邻居的索引是$(i-1 \bmod n_s)$和$(i+1 \bmod n_s)$，其中的n_s是群中粒子的总数。一般情况下，邻域是基于粒子之间的欧氏距离确定的。大小为 m 的邻域，便是由最靠近中心粒子的 m 个粒子组成的。

空间邻域的计算，需要在每一代中计算所有粒子之间的欧氏距离，因此会明显提升算法的计算复杂度。如果算法执行了n_t代，则空间邻域的计算复杂度就会提升 $O(n_t n_s^2)$。但这种根据距离来确定邻域的方法可以在每一代中动态地改变邻域。

2) 基于适应度值的空间网络

基于适应度值的空间网络是空间邻域的一个变种，其中的粒子会朝着已经找到了较好解的邻域粒子移动。假设一个最小化问题，粒子 i 的邻域定义为使以下表达式的值最小的 m 个粒子：

$$\varepsilon(x_i, x_{i'}) \times f(x_{i'}) \tag{3-55}$$

式中，$\varepsilon(x_i, x_{i'})$表示粒子 i 和粒子 i'之间的欧氏距离，$f(x_{i'})$表示粒子 i'的适应度值。在这种计算机制下，邻域是有可能重叠的。

基于以上决定邻域的机制，使用标准 lbest PSO 速度方程式，以及式(3-55)来确定邻域的最佳位置 $\hat{y}_i$。

算法 3-3 空间邻域的计算

N_i为粒子 i 的邻域；
计算粒子之间的欧氏距离 $\varepsilon(x_{i_1}, x_{i_2})$，$\forall i_1, i_2 = 1, \cdots, n_s$；
$S = \{i; i = 1, \cdots, n_s\}$；
for $i = 1, \cdots, n_s$ **do**
    $S' = S$；
    for $i' = 1, \cdots, m$ **do**
        $N_i = N_i \cup \{x_{i''}: \varepsilon(x_i, x_{i''}) = \min\{\varepsilon(x_i, x_{i''}), \forall x_{i''}\}\}$；
        $S' = S' \setminus \{x_{i''}\}$；
    end
end

3) 增长的邻域

交互行为较低的社交网络的收敛速度会比较慢，这样可以对搜索空间进行更大范围的探索。连接紧密的拓扑结构更容易收敛，但也会因此而忽略掉部分搜索空间。为了结合邻域结构的探索能力和高度连接网络的快速收敛性，提出了一种综合方法。搜索过程使用 $n_N = 2$(即最小邻域)的 lbest PSO 来初始化，随着迭代次数的增大，不断增大邻域的大小，直到每个邻域都包含整个群(即$n_N = n_s$)。

如果粒子位置 $x_{i_2}(t)$ 和粒子位置 $x_{i_1}(t)$ 满足如式(3-56)所示的关系，则将粒子位置 $x_{i_2}(t)$ 添加到粒子位置 $x_{i_1}(t)$ 的邻域之中。

$$\frac{\|x_{i_1}(t)-x_{i_2}(t)\|_2}{d_{\max}}<\varepsilon \tag{3-56}$$

式中，$d_{\max}$ 表示粒子之间的最大距离，ε 满足如下关系：

$$\varepsilon=\frac{3t+0.6\,n_t}{n_t} \tag{3-57}$$

式中，n_t 表示最大迭代次数。

这样的操作，可以在开始的几次搜索迭代中探索更多的搜索空间，在之后的迭代中则会更快地收敛。

4) 超立方体结构

如果问题中的值是二进制的，则可以使用一种超立方体结构。如果粒子索引位置的比特表示两个粒子之间的汉明距离为 1，则两个粒子位于一个邻域。为了利用超立方体的拓扑结构，粒子的总数必须是 2 的幂，粒子的索引从 0 到$2^{n_N}-1$，因此，超立方体结构具有如下属性：

(1) 每个邻域内正好有n_N个粒子。

(2) 粒子之间的最大距离恰好为n_N。

(3) 如果粒子i_1和粒子i_2是邻居关系，则它们不会再有其他的共同邻居。

针对二进制问题，超立方体结构的表现会优于 gbest PSO。

5) 全信息 PSO(全知 PSO)

根据标准速度更新方程式，每个粒子的位置更新受个体最佳位置和邻域最佳位置的影响，一个粒子不仅受到另一个粒子的影响，而且受到其邻域的统计状态的影响。基于这个思想，粒子的速度更新方程式将受到邻域内所有粒子的影响。因此，此种 PSO 变体也称为全信息 PSO 或者全知 PSO(fully informed PSO)。

但这种 PSO 变体没有考虑到不同粒子的影响有可能互相抵消的问题。

2. 混合算法

1) 基于选择的 PSO

基于进化计算中的选择思想，在速度更新之前进行择优选择操作，算法如下。

算法 3-4 基于选择的 PSO

```
计算所有粒子的适应度值;
for 粒子 i=1,…,n_s do
    随机选择n_ts个粒子;
    评估粒子 i 对这些随机选取的粒子的表现;
end
根据评估的表现得分对群进行排序;
用较好的一半粒子替换较差的一半,不改变个体最佳位置
```

表现较差的粒子虽然被替换，但它们找到的最优解却并未丢失，搜索过程依然会在已有的基础上继续进行。这种方法会增强基于选择的 PSO 的搜索能力，但却降低了种群的多样性，因为一半的粒子被替换掉了。

2）基于繁殖的 PSO

基于当前的粒子，可以产生后代。例如一个粒子在产生新的粒子之后，自身会消亡，新的粒子会根据环境修改自身的速度方程式中的系数。或者一个粒子的邻域如果没有得到改进，则在邻域内产生一个新的粒子；如果粒子的邻域得到了改进，则剔除表现最差的粒子。

在不同的情况下，可以使用不同的繁殖策略。

3）基于变异的 PSO

基于变异的 PSO，是在原有的位置更新 $x_i(t+1)$ 的基础上，加入了新的随机变异值。例如基于高斯变异的 PSO，位置更新的修改如下：

$$x_i(t+1)=x'_i(t+1)+\eta' N(0,1) \tag{3-58}$$

3. 基于子群的 PSO

将主群分成若干子群，这些子群可以是各自独立的，也可以有协作的机制。此处以协同分裂 PSO 和捕食 PSO 为例进行介绍。

1）协同分裂 PSO

协同分裂 PSO 由 van den Bergh 和 Engel Brecht 提出，是基于协同进化遗传算法的 PSO 变体。在协同分裂 PSO 中，每个粒子分裂成更小维度的 K 个独立的部分，每个部分都用一个单独的子群来进行优化。如果 $K=n_x$，每一维都由一个独立子群优化，可以使用任意 PSO。分裂的数目 K 称为分裂因子。

协同分裂 PSO 的难点在于粒子适应度值的计算，子群中的每个粒子的适应度值需要结合其他子群来进行计算，因此每个子群中的粒子仅代表着完整的 n_x 维中的解的一部分。为了解决这一问题，完整的解可以通过维护一个上下文信息来进行表示。最简单的方法是将全局最佳位置直接连成一串。除了与子群相关的元素之外，上下文向量中的其他元素都是常数。之后子群中的粒子替换向量中的对应位置，再使用原始的适应度值计算函数来评估向量的适应度值。然后将获得的适应度值指定为子群的相应粒子的适应度。

这个过程的优点在于适应度函数的值 f 在上下文向量的每个子部分更新完之后才会被评估，这样可以进行更细粒度的搜索。对完整的 n_x 维问题进行优化存在一个问题，即使适应度值获得改善，但向量中的一部分分量也有可能偏离最佳值。改善的适应度可以通过向其他分量的最佳值移动获得，而协同分裂 PSO 则通过独立调整解的各个子部分来解决这一问题。

虽然协同分裂 PSO 已经被证明可以取得比基本 PSO 更好的结果，但需要注意的是，当相关元素被分裂到不同的子群时，性能会降低。如果可以确定哪些参数是相关的，则可以将这些参数分组到相同的子群中，由此问题便可解决。然而实际情况下，这样的先验知识往往是无法得到的。当且仅当一个粒子改善了其上下文向量的适应度值时，它可以成为其所在子群的全局最优值或个体最优值，那么这个问题在实际操作中便可以得到一定程度的解决。

2）捕食 PSO

捕食者与被捕食者的关系在自然界中是天然存在的，是自然界竞争关系的一种反映。这种行为也被应用到 PSO 之中，用来平衡探索和挖掘的过程。通过引入捕食者粒子来构建第二个群，捕食者粒子的存在会导致被捕食者粒子被驱赶，因此被捕食者粒子会变得更为分散。这样的操作能够探索更大范围的搜索空间。

4. 多次重启 PSO

基本 PSO 的一个主要问题是：当粒子开始收敛于同一点时，多样性就会开始大幅地下降。为了防止 PSO 的迭代过早结束，开始引入多种方法来为群体增大随机性和混乱性。此类方法被统称为多次重启 PSO。

多次重启 PSO 的主要目的是提高多样性，从而探索更大的搜索空间。向群中引入混乱因素，从而造成负熵。需要注意的是，随机位置的连续注入会导致群永远都不能达到平衡状态。虽然并不是所有的方法都能考虑这一问题，但都可以通过减少时间来降低混乱度，以此来在一定程度上解决这一问题。

Kennedy 和 Eberhart 最早提出了随机重新初始化粒子的优势，他们将这个过程称为“疯狂”。当考虑为群增大随机性时，需要考虑一些因素，例如该初始化什么，什么时候应该初始化，如何进行初始化，以及初始化的过程会影响哪些部分等。

3.5.2 多解粒子群优化

虽然基本 PSO 是用于寻找优化问题的单一解的，但 PSO 能够实现对特殊函数的选择，因此标准 PSO 也存在着寻找多个解的潜在能力。Agrafiotis 和 Cedeno 在观察某个具体问题时，发现粒子能够进入几个局部极小值。

PSO 的主要驱动部分分为认知部分和社会部分。社会部分是阻止物种形成的主要因素。例如 gbest PSO，社会部分使得所有粒子被吸引至群体所获得的最佳位置，而如果全局自身位置和粒子自身的个体最佳位置不同，则认知部分对粒子自身的最佳位置施加反作用力，由此产生的效果使得粒子汇聚到全局最佳位置和个体最佳位置之中的一个点。如果 PSO 一直被执行直到群体达到平衡，则每个粒子都会收敛到全局最佳位置和各自的个体最佳位置之一，并且最终速度都会归零。

Brits 等通过实验证明了 lbest PSO 成功找到了一些函数的多个极值。然而，这些研究在算法超过函数评估的最大数目之后便会终止，而并不是达到平衡状态才停止的。因此粒子还会保留一些惯性因素，能够进一步探索搜索空间。在粒子数目增大时，更多的解被定位的可能性也会随之增大。如果运行的过程持续进行，则受到社会部分的影响，会有更多粒子收敛到全局最佳位置以达到平衡。

Niche PSO 被设计用于解决一般多模态环境的多解问题。Niche PSO 的基本原理是将粒子自组织成互相独立的子群。每个子群都被定位于并被保持在一个小的生存环境中。信息交互的范围仅被保留在子群内部，子群与子群之间没有信息交互的过程。子群之间的这种独立性能够使得子群保持在适当的位置。需要注意的是：每一个子群都是一个稳定、独立的群体，独立地进行演化，与其他子群中的粒子保持相互独立。

Niche PSO 起始于一个群(被称为主群，包含了所有的粒子)。一旦一个粒子收敛于一个潜在的解，就会很快地把靠近潜在解的粒子拉近组成一个子群，然后将这些粒子从主群中移除，并在子群内部继续进行细化并维持解。随着时间的推移，主群的规模会因子群的不断产生而减小，当子群不再改进它们所代表的解时，则认为 Niche PSO 已经收敛，最后将每个子群的全局最佳位置作为最优解，这样就有可能产生多个最优解。

Niche PSO 算法如下。

算法 3-5 Niche PSO

创建并初始化一个n_x维的主群，S；
repeat
　使用认知模型训练一代主群，S；
　更新主群中每个粒子的适应度，$S.x_i$；
　for 每个子群S_k **do**
　　使用全模型 PSO 来训练每个子群中的粒子，$S_k.x_i$；
　　更新每个粒子的适应度值；
　　更新种群的半径$S_k.R$；
　end
　如果可能，合并子群；
　允许子群从主群中吸收任何要进入子群的粒子；
　如果可能，创建新的子群；
until 终止条件满足；
返回每个子群的最优值 $S_k.\hat{y}$ 作为最后的解

1. 主群训练

主群使用认知模型来促进对搜索空间的探索。由于社会部分会吸引所有的粒子，因此需要将其从速度更新方程式中移除，这样就能使得粒子在搜索空间中的不同部分汇聚。

需要额外注意的是，主群中的各粒子的速度必须初始化为零。

2. 子群训练

子群是各自独立的群，使用全模型基本 PSO 进行训练。在使用完整的模型来训练子群时，允许粒子根据自身的经验（认知部分）和邻域群体的信息（社会部分）来调整自身的位置。虽然任何全模型 PSO 都可以用来训练子群，但 Niche PSO 是使用保证收敛的 PSO（GCPSO）进行训练的，因为它能够保证收敛到局部极值，此外在群的规模较小时，GCPSO 也具有比较好的性能。第二个特性尤为必要，因为子群最初只包含两个粒子，而 gbest PSO 往往会在这种小规模群体上停滞不前。

3. 创建子群

当一个粒子收敛到一个解时，就形成了一个子群。如果一个粒子的适应度值在多次迭代中都没有发生变化，则用这个粒子及其最近的拓扑邻居来创建一个子群。从形式上来看，观察每个粒子的适应度函数 $f(x_i)$ 的标准差σ_i，如果$\sigma_i<\varepsilon$，则创建一个子群。为了避免问题依赖，σ_i需要根据域进行归一化处理。使用欧氏距离来计算与粒子 i 的位置 x_i 的最近邻 l，具体计算公式如下：

$$l=\underset{a}{\operatorname{argmin}}\{\|x_i-x_a\|\} \tag{3-59}$$

式中，$1\leqslant i,a\leqslant S.n_s,i\neq a$，$S.n_s$是主群的大小。

创建子群的算法如下。

算法 3-6 Niche PSO 创建子群算法

if $\sigma_i<\varepsilon$ **then**
　$k=k+1$；
　创建子群$S_k=\{x_i,x_l\}$；
　$Q\cup S_k\rightarrow Q,S\backslash S_k\rightarrow S$；
end