计算智能——粒子群算法

作者 柯良军

西安交通大学 电信学院

October 27, 2015

目录

概述

1995年，*James Kennedy*和的*Russell Eberhart*提出了粒子群优化算法。

概述

1995年，*James Kennedy*和的*Russell Eberhart*提出了粒子群优化算法。

图：自然界中的鸟群和鱼群.

概述

1995年，*James Kennedy*和的*Russell Eberhart*提出了粒子群优化算法。

图：自然界中的鸟群和鱼群.

它是在动物社会心理学的研究成果基础上，模拟生物群体的智能行为和应用计算机对这些认知行为进行仿真后的产物。

思想来源

社会心理学（特别是动态社会影响理论）是粒子群算法发展的重要来源。

思想来源

社会心理学（特别是动态社会影响理论）是粒子群算法发展的重要来源。

个体可以根据环境的变化及时的调整他们的信念和态度从而和群体的行为保持一致。

思想来源

社会心理学（特别是动态社会影响理论）是粒子群算法发展的重要来源。

个体可以根据环境的变化及时的调整他们的信念和态度从而和群体的行为保持一致。

粒子群优化算法源于对鸟类觅食行为的研究：

鸟类觅食时，搜寻当前距离食物最近的鸟周围区域。

生物群体中个体与个体之间、个体与群体之间相互影响、相互作用，在群体中的信息共享。

基本思想

粒子群算法是从生物种群的智能行为中得到的启发来求解优化问题。

每个粒子（*particle*）具有三个属性：

位置 $\longrightarrow$ 解

速度决定飞行的方向和距离，速度根据自身和其他粒子的移动经验进行动态的调整，以此来实现个体在可行解空间中寻优。

适应度值，适应度值通过适应度函数计算，其大小反映了粒子的优劣程度。

基本思想

粒子群算法是从生物种群的智能行为中得到的启发来求解优化问题。

每个粒子（*particle*）具有三个属性：

位置 $\longrightarrow$ 解

速度决定飞行的方向和距离，速度根据自身和其他粒子的移动经验进行动态的调整，以此来实现个体在可行解空间中寻优。

适应度值，适应度值通过适应度函数计算，其大小反映了粒子的优劣程度。

粒子在解空间中运动时，其位置更新时通过跟踪个体最优解$Pbest$和群体最优解$Gbest$来完成的。

相关术语

一个由n个粒子组成的种群$x = (x_1, x_2, \cdots, x_n)$在$D$维搜索空间中以一定的速度飞行，每个粒子在搜索时，融合当前信息和历史信息，进行位置（或状态、解）的变化。

相关变量定义如下：

第i个粒子的位置：$x_i = (x_{i1}, x_{i2}, \cdots, x_{iD})^T$；

第i个粒子的速度：$v_i = (v_{i1}, v_{i2}, \cdots, v_{iD})^T$；

第i个粒子的个体最优解：$p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})^T$；

种群内所有粒子的群体最优解：$p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})^T$。

其中$1 \le i \le m, 1 \le d \le D$，$i$表示粒子编号，$D$表示求解问题的维数。

一般来说，粒子的位置和速度都是在连续的实数空间内取值。

速度和位置更新

在每次迭代过程中，粒子通过个体最优解和群体最优解更新自身的速度和位置信息，其数学表达式如下：

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_1 (p_{id}^{k} - x_{id}^{k}) + c_2 r_2 (p_{gd}^{k} - x_{id}^{k}) \tag{1}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{2}$$

速度和位置更新

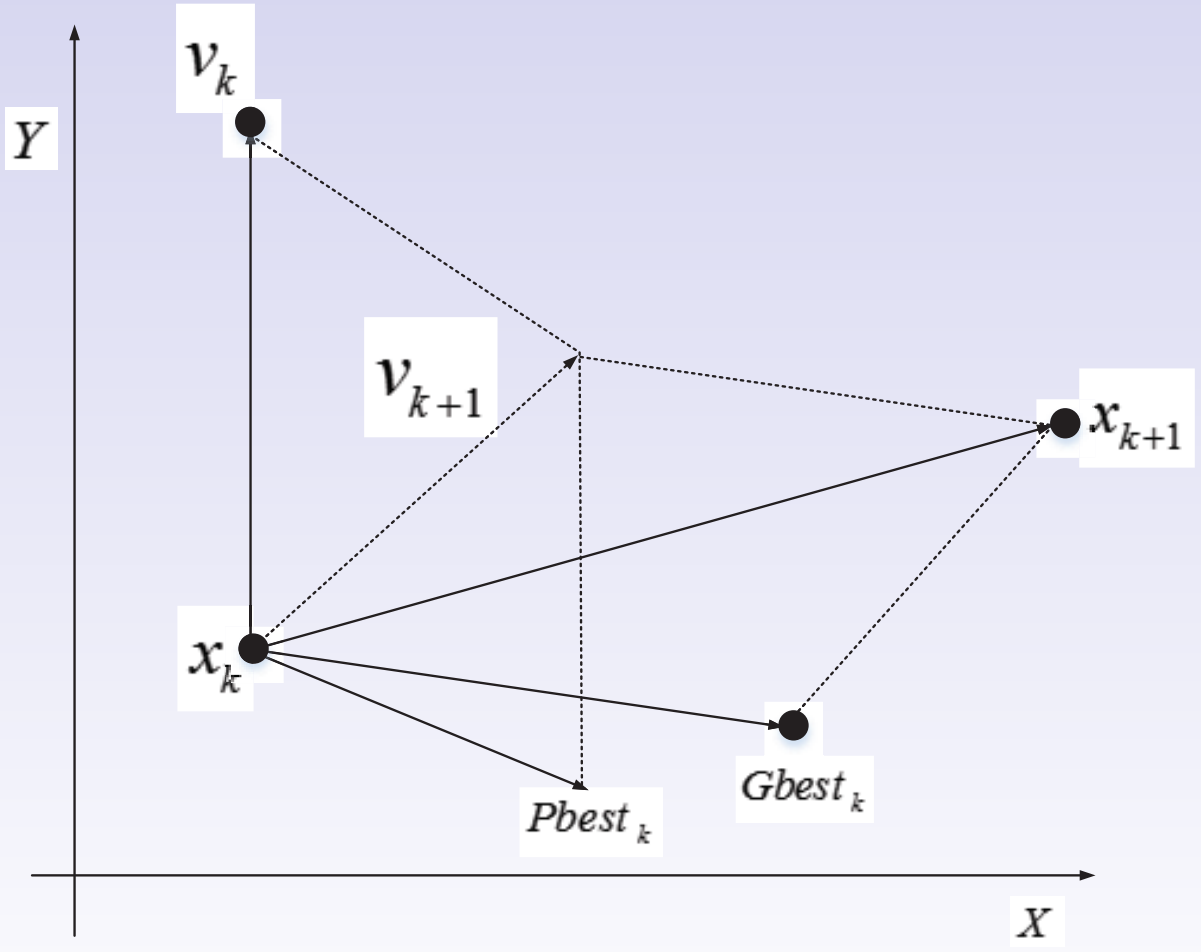
在每次迭代过程中，粒子通过个体最优解和群体最优解更新自身的速度和位置信息，其数学表达式如下：

$$v_{id}^{k+1} = v_{id}^{k} + c_1 r_1 (p_{id}^{k} - x_{id}^{k}) + c_2 r_2 (p_{gd}^{k} - x_{id}^{k}) \tag{1}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{2}$$

其中$d = 1, 2, \cdots, D$；$i = 1, 2, \cdots, n$；k为当前的迭代次数；$c_1, c_2 \in R_+$称为学习因子或加速系数，通常情况下，c_1和c_2取为2；r_1和r_2是介于$[0, 1]$之间均匀分布的随机数。一般将其位置和速度分别限制在区间$[-X_{max}, X_{max}]$和$[-V_{max}, V_{max}]$ 的范围内。

速度和位置的关系



速度和位置更新

*Shi*和*Eberhart*引入了惯性权重（*inertia weight*），将速度更新方程修改为：

$$v_{id}^{k+1} = \omega v_{id}^{k} + c_1 r_1 (p_{id}^{k} - x_{id}^{k}) + c_2 r_2 (p_{gd}^{k} - x_{id}^{k}) \tag{3}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{4}$$

其中ω称为惯性权重，基本粒子群算法是惯性权重为1的特例。

粒子的社会行为分析

从粒子的速度更新方程（3）可以看出，标准粒子群算法中，粒子的速度主要由三部分构成。

第一部分：**粒子当前的速度**，是粒子飞行中的惯性作用和其能够飞行的基本保证，表明粒子当前的状态，防止粒子大幅度改变方向，平衡了粒子的全局和局部搜索能力。

第二部分：**个人认知部分**，表示粒子在飞行中凭借自身的经验，向自己曾经找到过的最优位置靠近，使粒子有足够强的全局搜索能力，避免局部最优。

第三部分：**社会经验部分**，表示粒子在飞行中考虑到社会经验，向邻域中其他粒子学习，通过借鉴和信息共享，使粒子在飞行时向邻域内所有粒子曾经找到过的最好点靠近。

粒子群算法的要素

粒子群算法的要素包括算法的相关参数和算法设计中的相关问题两部分。

粒子群算法的要素

粒子群算法的要素包括算法的相关参数和算法设计中的相关问题两部分。

相关参数有种群大小、学习因子、最大速度、惯性权重；

算法设计中的相关问题有：邻域拓扑结构、粒子空间的初始化和停止条件。

粒子群算法的要素

1. 种群规模n

种群规模n过小，算法容易陷入局部最优，而种群规模n过大，将导致计算时间大幅增加，并且当种群数目增大到一定水平时，再增大将不再有显著的作用。

当$n = 1$时，粒子群算法变为基于个体搜索的算法，一旦陷于局部最优，很难跳出；

当n趋于无穷大时，粒子群算法的全局搜索能力会很好，可是收敛速度缓慢。

粒子群算法的要素

2. 学习因子c_1和c_2

学习因子使粒子具有自我总结和向种群中优秀个体学习的能力，从而向种群内或邻域内最优位置靠近。

粒子群算法的要素

2. 学习因子c_1和c_2

学习因子使粒子具有自我总结和向种群中优秀个体学习的能力，从而向种群内或邻域内最优位置靠近。

3. 最大速度V_{max}

最大速度V_{max}决定了粒子在一次迭代中最大的移动距离。

V_{max} 较大，搜索能力增强，但是粒子容易飞过最好解；

V_{max} 较小时，开发能力增强，但是容易陷于局部最优。

通过分析和实验发现，V_{max}的设定可以通过惯性权重的调整来实现。目前的粒子群算法基本上使用V_{max}进行初始化，将V_{max}设定为每一维变量的变化范围，而不需要进行细致的选择和调节。

粒子群算法的要素

4. 惯性权重ω

惯性权重ω决定了粒子对当前速度继承的多少。

合理的取值会增强粒子均衡的探索能力和开发能力。

较大的惯性权重使粒子能在自己原来的方向上具有更大的速度，从而在原方向上飞行更远，具有更好的探索能力；

较小的惯性权重使粒子继承了较少的原方向的速度而飞行较近，具有更好的开发能力。由此可见，通过调节惯性权重ω能够调节粒子群的探索能力。

粒子群算法的要素

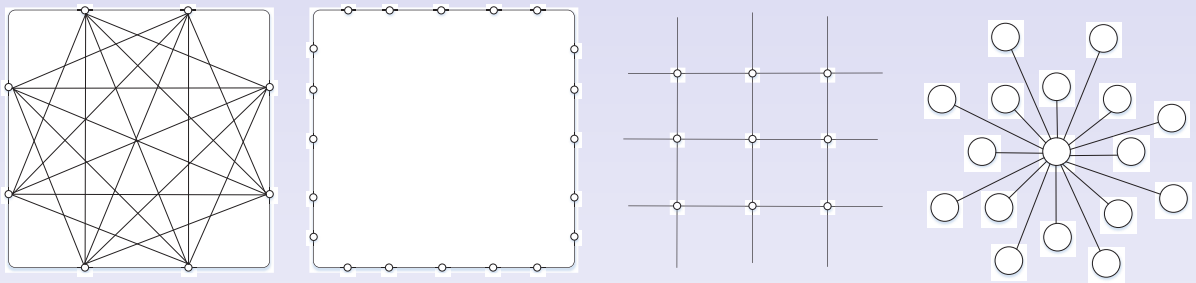
5. 邻域的拓扑（*topology*）结构

在粒子群算法中，由n个粒子组成的集合$x = (x_1, x_2, \cdots, x_n)$，每个粒子对应可行解空间中的一个潜在解，一个粒子从它的邻域N_i接收信息（其中N_i是解空间的子区域）。

标准的粒子群算法中，**粒子和粒子之间的关系**可以用一个无向图$G = \{V, E\}$ 来表示，其中V是粒子的集合，E是边的集合，表示粒子与邻域粒子之间的配对关系。这个无向图通常被称作粒子群的拓扑结构。

粒子群算法中常见的**拓扑结构**有星型结构、环形结构、冯· 诺依曼结构和齿形结构等。

粒子群算法的要素



（a）星型结构　（b）环型结构　（c）冯• 诺依曼结构　（d）齿型结构

图：粒子的拓扑结构示意图.

粒子群算法的要素

6. 粒子空间的初始化

随机设定每个粒子的初始值，包括位置x_i和速度v_i，将个体的历史最优$Pbest$设为当前位置，种群中最优的个体作为当前的$Gbest$。较好地选择粒子的初始化空间，将大大缩短收敛时间。

7. 停止条件

一般地，停止条件设定为一个足够好的适应值或达到一个预设的最大迭代次数。

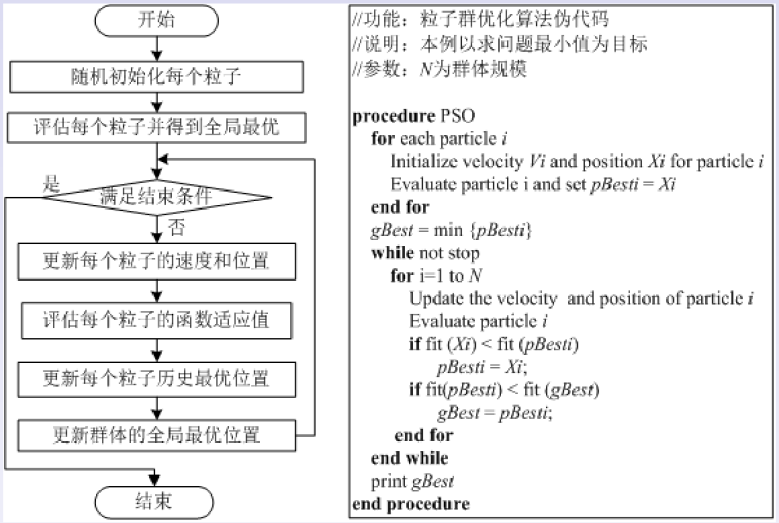
步骤

通过前面的介绍，可以将标准粒子群算法的步骤归纳如下：

$Step\ 1$：初始化粒子群，随机设定位置x_i和速度v_i；

$Step\ 2$：在每一次进化（迭代）中计算每个粒子的适应值；

流程图



图：粒子群算法流程图.

步骤

$Step\ 3$：对于每个粒子x_i，如果其适应度值比所经历过的历史最好位置$Pbest$的适应度值好，则用当前位置p_i更新个体历史最好的位置$Pbest$；

$Step\ 4$：对于每个粒子x_i，如果其历史最优适应度值比群体内或邻域内所经历的最好位置$Gbest$的适应度值好，则用当前的全局最好位置p_g更新种群的历史最好的位置$Gbest$；

步骤

Step 5：根据公式（3）和（4）对粒子的位置x_i和速度v_i进行更新；

Step 6：若未达到停止条件，则转到*Step* 2。

如果把种群中所有粒子都作为邻域成员时，得到粒子群算法的**全局版本**；如果把种群中部分粒子作为邻域时，得到粒子群算法的**局部版本**。

在局部版本中，一般有两种方式组成邻域，一种是索引号相邻的粒子组成邻域，比如环形结构、轮型结构和星型结构等。另一种是位置相邻的粒子组成邻域，它是基于距离的拓扑结构。粒子群算法的邻域定义策略又可以称为粒子群的邻域拓扑结构。

算例

本节通过一个例子来说明标准粒子群算法的运行过程。

例1 使用标准粒子群算法求解$\min f(x) = 4x^3 + 3x^2 - 6x + 1, x \in [-1, 1]$，请写出算法执行的关键步骤，精度到小数点后2位。

步骤1

解：根据标准粒子群算法的步骤，求解$\min f(x)$的步骤如下：

$Step\ 1$：初始化粒子群。设种群大小为$n=3$,随机设定位置x_i和速度v_i，计算每个粒子的适应值，得到粒子的历史最优位置$Pbest_i$和全局最好位置$Gbest$。

p_1: $v_1=3, x_1=-1, f_1=4\times(-1)^3+3\times(-1)^2-6\times(-1)+1=7, Pbest_1=x_1=-1$

p_2: $v_2=-3, x_2=1, f_2=4\times 1^3+3\times 1^2-6\times 1+1=2, Pbest_2=x_2=1$

p_3: $v_3=2, x_3=-0.5, f_3=4\times 0^3+3\times 0^2-6\times 0+1=1, Pbest_3=x_3=-0.5$

$Gbest=Pbest_3=-0.5$

步骤

Step 2 ：更新粒子的速度和位置。根据粒子的历史最优位置$Pbest_i$和全局最好位置$Gbest$，更新粒子的位置x_i和速度v_i。

p_1: $v_1^2 = \omega v_1^1 + c_1 r_1 (p_1^1 - x_1^1) + c_2 r_2 (p_g^1 - x_1^1) = 0.5 \times 3 + 0 + 2 \times 0.1 \times (-0.5 - (-1)) = 0.1$

$x_1^2 = x_1^1 + v_1^2 = -1 + 0.1 = -0.9$

p_2: $v_2^2 = \omega v_2^1 + c_1 r_1 (p_2^1 - x_2^1) + c_2 r_2 (p_g^1 - x_2^1) = 0.5 \times (-3) + 0 + 2 \times 0.2 \times (-0.5 - 1) = -0.18$

$x_2^2 = x_2^1 + v_2^2 = 1 + (-0.18) = 0.82$

p_3: $v_3^2 = \omega v_3^1 + c_1 r_1 (p_3^1 - x_3^1) + c_2 r_2 (p_g^1 - x_3^1) = 0.5 \times 2 + 0 + 0 = 1$

$x_3^2 = x_3^1 + v_3^2 = -0,5 + 1 = 0.5$

其中，w为惯性权值，一般取$w \in [0, 1.4]$，这里去0.5；c_1和c_2为加速因子，取固定值2.0；r_1和r_2取介于$[0, 1]$之间均匀分布的伪随机数。

步骤

$Step\ 3$：评估粒子的适应度函数值，更新$Pbest$和$Gbest$。

p_1:$\hat{f}_1 = 4 \times (-0.9)^3 + 3 \times (-0.9)^2 - 6 \times (-0.9) + 1 = 5.91 < f_1 = 7$

$f_1 = \hat{f}_1 = 5.91$

p_2:$\hat{f}_2 = 4 \times 0.82^3 + 3 \times 0.82^2 - 6 \times 0.82 + 1 = 0.30 < f_2 = 2$

$f_2 = \hat{f}_1 = 0.30$

p_3:$\hat{f}_3 = 4 \times 0.5^3 + 3 \times 0.5^2 - 6 \times 0.5 + 1 = -0.75 < f_3 = 1$

$f_3 = \hat{f}_1 = -0.75$

$Gbest = Pbest_3 = -0.75$

步骤

$Step\ 4$：若未达到停止条件，则转到$Step\ 2$。

种群规模改进

从标准的粒子群算法可以看出，种群规模的大小是影响算法性能的主要因素之一，**根据经验，粒子数取值范围一般为** $n \in (20, 40)$。

种群规模的设置要根据具体求解的问题来定，一般来说，种群规模的大小在求解过程中是保持不变的，目前还不能给出一个最具合理的值。

种群规模改进

一种动态调整种群规模策略如下：

1. 如果粒子得到充分改进，且粒子是邻域中最差的，则删除种群中的粒子；
2. 如果粒子没得到充分改进，且粒子是邻域中最佳的，则复制粒子。

惯性权值设置

在速度更新公式中，第一部分是惯性作用和其能够飞行的基本保证，其作用是**平衡粒子的全局和局部搜索能力**。

如果在更新公式中，令$c_1 = 0, c_2 = 0$，假设初始速度非零，那么当$w > 1$时，粒子的速度会随着飞行时间的增加而增大到最大速度V_{max}；当$w < 1$时,粒子的速度会随着飞行时间的增加而减小到0，停止粒子的运动。

实验研究表明w的合理取值范围为$\in [0, 1.4]$，当$w \in [0.8, 1.2]$时，算法会加速收敛；当$w > 1.2$时，容易导致算法收敛失败。

惯性权值设置

1. 固定权值

固定权值的优点是粒子在中飞行中能始终保持相同的搜索能力。问题不同，权值也是不同的。

研究发现：种群规模越小，对惯性权重的值要求越大，只有这样才可以弥补粒子不足而搜索能力保持不变，防止粒子过早收敛；相反，种群规模越大，对惯性权值的要求越小，因为粒子可以更多的关注邻近的区域。

惯性权值设置

2. 线性时变权值

为了保持粒子的全局搜索能力，在算法运行初期设置较大的惯性权值，而后期为了保持粒子的局部搜索能力，使用较小的惯性权值。

惯性权值的值可以通过如下公式来计算：

$$w = (w_1 - w_2) \times \frac{MaxIter - CurIter}{MaxIter} + w_2 \tag{5}$$

其中w_1和w_2表示惯性权值的初始值和最终值，$MaxIter$表示最大迭代数，$CurIter$表示当前迭代数。

这是一种线性递减的变化过程，也可以采用非线性递减的方式，线性时变权值是实际应用中最为广泛的一种。

惯性权值改进

3. 模糊权值

利用模糊系统来动态的调整权值。模糊系统由以下部分组成：

- 两个输入，一个表示全局最优位置的适应度，另一个表示惯性权值的当前值；
- 一个输出，表示惯性权值的改变；
- 三个模糊集，分别为LOW、$MEDIUM$和$HIGH$；
- 若干模糊规则，惯性权值根据这些规则来计算改变值。

惯性权值设置

4. **随机权值**

为了构建非线性的搜索算法，可按如下方式选择权值：

$$w = 0.5 + \frac{rand()}{2} \tag{6}$$

其中$rand()$为$[0,1]$之间的随机数。

通过分析可以看出，惯性权值的平均值为0.75，w在$[0.5,1]$之间变化。

实验表明：该方法在算法执行初期能够加速粒子收敛，并且对大多数优化函数都能找到比较好的解。

学习因子设置

粒子的自身经验和群体经验对粒子的飞行轨迹会产生很大的影响，所以调整粒子的自身经验和群体经验有利于粒子飞向最好的位置，这两部分可以通过调整粒子的学习因子（加速因子）c_1 和c_2来实现。因此对学习因子的改进显得很有必要。

学习因子设置

c_1和c_2主要反映了粒子之间的信息交流和共享。

如果$c_1 > c_2$，每个粒子会更多地飞向自己的个体最佳位置，导致过多的漫游；

如果$c_1 < c_2$，粒子过多的飞向全局最优的位置，导致粒子过早地冲向最优解点。

c_1和c_2设置较低，得到比较平滑的粒子轨迹，使得粒子在飞回最优区域之前漫游到较差的区域；

c_1和c_2设置较高，导致粒子具有更大的加速度，使得粒子突然的移向或者错过最好的区域。

学习因子设置

同步法：

将学习因子c_1和c_2的取值范围设置为$[c_{min}, c_{max}]$，最大迭代次数为$Iter_{max}$，则第i次迭代的学习因子可表示为：

$$c_1 = c_2 = c_{max} - \frac{c_{max} - c_{min}}{Iter_{max}} \times i \tag{7}$$

公式7 中c_1和c_2同步线性减小。

学习因子设置

异步法：

其目标的是在算法执行的初期加强全局搜索，在算法执行的后期促使粒子收敛到全局最优解。

学习因子设置

异步法：

其目标的是在算法执行的初期加强全局搜索，在算法执行的后期促使粒子收敛到全局最优解。

具体做法：c_1不断减小和c_2不断增大。

学习因子设置

异步法：
其目标的是在算法执行的初期加强全局搜索，在算法执行的后期促使粒子收敛到全局最优解。
具体做法：c_1不断减小和c_2不断增大。
学习因子调整如下：

$$c_1 = (c_{1e} - c_{1s})\frac{CurIter}{MaxIter} + c_{1s} \tag{8}$$

$$c_2 = (c_{2e} - c_{2s})\frac{CurIter}{MaxIter} + c_{2s} \tag{9}$$

其中$MaxIter$表示最大迭代次数，$CurIter$表示当前迭代次数，c_{1s}和c_{1e}分别表示c_1的初始值和最终值，c_{2s}和c_{2e}分别表示c_2的初始值和最终值。

学习因子设置

研究中发现，学习因子设置为：$c_{1e} = 0.5, c_{1s} = 2.5, c_{2e} = 2.5, c_{2s} = 0.5$，
优化效果较好。

邻域拓扑结构设置

粒子的拓扑结构也称作社会结构，每个粒子在各自的邻域内互相共享信息，

粒子的拓扑结构一般分为**基于索引号的拓扑结构和基于距离的拓扑结构**。

邻域拓扑结构改进

1. **基于索引号的拓扑结构**

基于索引号的拓扑结构的特点是不考虑粒子间的相对位置

邻域拓扑结构改进

1. **基于索引号的拓扑结构**

基于索引号的拓扑结构的特点是不考虑粒子间的相对位置

优点： 计算开销小。

邻域拓扑结构改进

1. 基于索引号的拓扑结构

基于索引号的拓扑结构的特点是不考虑粒子间的相对位置

优点： 计算开销小。

（1）环形结构

环形结构是一种基本的邻域拓扑结构，每个粒子只与其索引号相近的N个粒子相连构成该粒子的邻域。

邻域拓扑结构改进

1. 基于索引号的拓扑结构

基于索引号的拓扑结构的特点是不考虑粒子间的相对位置

优点： 计算开销小。

（1）环形结构

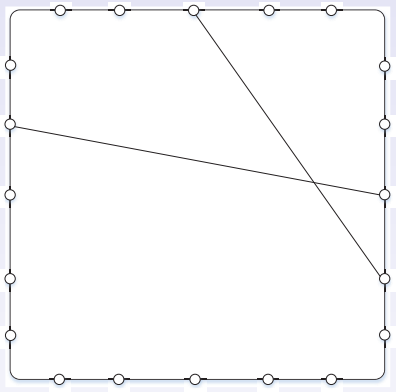
环形结构是一种基本的邻域拓扑结构，每个粒子只与其索引号相近的N个粒子相连构成该粒子的邻域。

优点是：

环形结构中，种群中的一部分粒子可能**聚集于局部最优**，另一部分粒子**可能聚集于全局最优**，或继续在解空间中寻优，这样就避免了粒子过早陷于局部最优的情况。

邻域拓扑结构设置

有学者在环形拓扑结构中加入两条捷径（*shortcut*）得到带有捷径的环形拓扑结构，
为了保持环路的连通性，粒子**随机地选取**种群中的另一个粒子作为自己的邻域成员，从而**加强了不同粒子之间的信息交流和共享**，缩短了邻域间的距离，使得种群能够更快收敛。



图：带捷径环形拓扑结构示意图.

邻域拓扑结构设置

（2）齿形结构

齿形结构是有一个**焦点粒子**，其他粒子除了与该焦点粒子相连外它们之间并不相连，所有的粒子只与焦点粒子进行信息交流和共享，从而实现了粒子之间的分离。

邻域拓扑结构设置

（2）齿形结构

齿形结构是有一个**焦点粒子**，其他粒子除了与该焦点粒子相连外它们之间并不相连，所有的粒子只与焦点粒子进行信息交流和共享，从而实现了粒子之间的分离。

焦点粒子通过比较其与邻域其他粒子的表现来调节其飞行轨迹并向最好位置靠近，这样就实现了位置的更新。然后焦点粒子再将信息扩散到邻域的粒子。

邻域拓扑结构设置

（2）齿形结构

齿形结构是有一个**焦点粒子**，其他粒子除了与该焦点粒子相连外它们之间并不相连，所有的粒子只与焦点粒子进行信息交流和共享，从而实现了粒子之间的分离。

焦点粒子通过比较其与邻域其他粒子的表现来调节其飞行轨迹并向最好位置靠近，这样就实现了位置的更新。然后焦点粒子再将信息扩散到邻域的粒子。

焦点粒子起缓冲器的作用，有效的减缓了较好解在种群中的扩散速度。

邻域拓扑结构设置

也可以在齿形结构中加入两条捷径得到带有捷径的齿形结构，如图5所示。

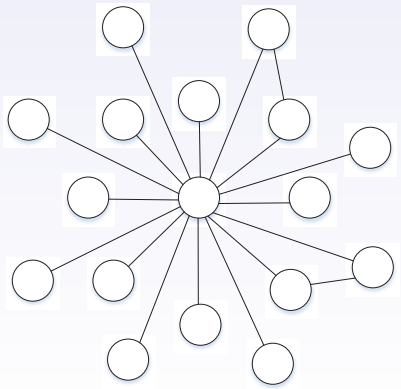
这种拓扑结构的特点是：

（1）能够产生迷你邻域（$Mini-Neighborhood$）

与焦点粒子相连的粒子直接影响着迷你邻域中的外围粒子，这将导致与迷你邻域交流信息的子种群可以更快地收敛，而焦点粒子的缓冲器又有效的阻止了整个种群过早收敛于局部最优。

（2）也可能产生孤岛现象

影响子种群间的联系，导致子种群内部进行信息共享，独立地进行问题的优化，从而导致信息交流的减少，使那些孤立的个体不能得到整个种群的最好位置；也使种群其他粒子不能分享被孤立的个体搜索中获得的成功信息。



邻域拓扑结构设置

（3）星型结构

星型结构中的每个粒子都可以看做一个焦点，将整个种群当作自己的邻域

邻域拓扑结构设置

（4）随机结构

随机结构是一种比较简单的拓扑结构，在种群规模为N的粒子间随机地建立N个对称的两两联结。

邻域拓扑结构设置

2. 基于距离的拓扑结构

计算一个粒子与种群中其他粒子之间的距离，然后根据距离来确定该粒子的邻域结构。

邻域拓扑结构设置

2. 基于距离的拓扑结构

计算一个粒子与种群中其他粒子之间的距离，然后根据距离来确定该粒子的邻域结构。

一种典型的做法是：在搜索开始时，粒子的邻域只有粒子本身，将其个体最优解作为邻域最优解，随着迭代次数的不断增加，邻域逐渐扩大，直到最后将群体中所有粒子都作为自己的邻域。这样做的优点是初始迭代时，粒子有较好的探索性能，而在后期迭代时有较好的开发性能。

邻域拓扑结构设置

2. 基于距离的拓扑结构

对于邻域中的第i个粒子，计算其与种群中其他所有粒子之间的距离。第j个粒子与第i个粒子（$j \neq i$）之间的距离记作$dist[j]$，粒子之间的最大距离记作$dist_{max}$。

定义一个表示当前迭代次数的函数$frac$:

$$frac = \frac{3.0 \times Iter + 0.6 \times MaxIter}{MaxIter} \tag{10}$$

如果$frac < 0.9$，$\frac{dist[l]}{dist_{max}} < frac$，那么该粒子组成当前粒子$i$的邻域；

如果$frac \geqslant 0.9$，那么种群中所有粒子作为当前粒子i的邻域。

混合算法

粒子群算法与其他智能优化算法结合寻优也引起了一些学者的关注，形成了各种各样的基于粒子群算法的混合智能算法。这些混合算法大概可以分为：

（1）与传统进化算法中的算子融合，比如融合遗传算法中的选择算子、交叉算子和变异算子等；

（2）直接和其他智能算法结合，比如结合模拟退火、查分进化、免疫算法和禁忌搜索等；

（3）融合数学、物理、生物等相关学科中的一些思想和技术来设置粒子群算法。

离散版本

粒子群算法在解决连续优化问题时表现出了很好的优越性，成功的解决了许多实际问题。之前许多学者的研究也主要集中在求解连续函数上，粒子的速度和加速度等参数都是连续的。但是现实生活中有许多工程问题都是离散的，例如比较经典的离散组合优化问题有$0-1$背包问题、TSP问题、调度问题、和路由规划问题等。

离散版本

二进制PSO算法:

速度更新公式:

$$v_{id} \leftarrow v_{id} + \varphi(p_{id} - x_{id}) + \varphi(p_{gd} - x_{id})$$

位置更新公式:

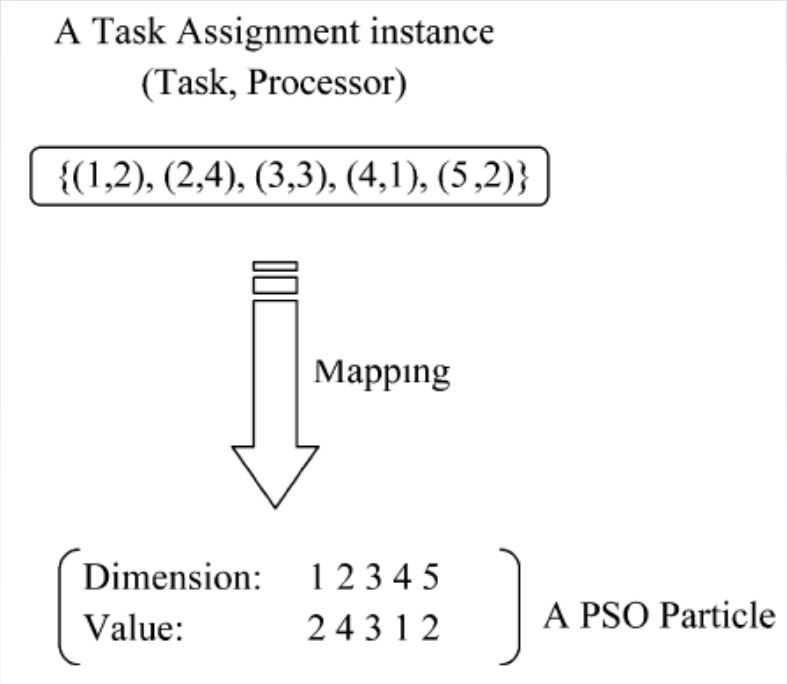
$$x_{id} = \begin{cases} 1 & \text{if} \quad rand() < sig(v_{id}) \\ 0 & otherwise \end{cases}$$

$$sig(v_{id}) = 1/(1 + exp(-v_{id}))$$

离散版本

四舍五入法：

用任务图和连接图来分别表示待分配的任务和处理器，通过一定的规则将任务图和处理器连接图映射到连续粒子的运动空间。



离散版本

加减法乘法:
引入交换子和交换序列的概念。在一个交换子$S = Swap(i, j)$指交换粒子第i个和第j个元素的位置，一个交换序列是指一组以特定顺序的交换子。粒子的速度是指粒子为达到最优解而需要对当前位置执行的基本交换序。

$$V_{t+1} = c_1 V_t \oplus c_2(p_{it} - X_t) \oplus c_3(p_{gt} - X_t) \tag{11}$$
$$X_{t+1} = X_t + V_{t+1}$$