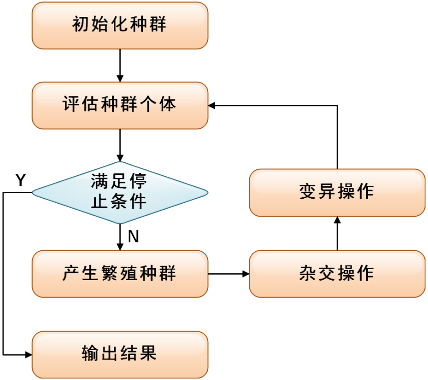


Figure: GA 流程图

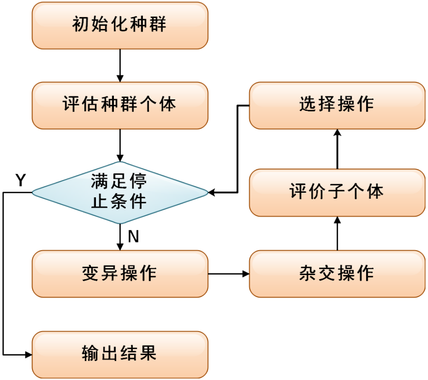
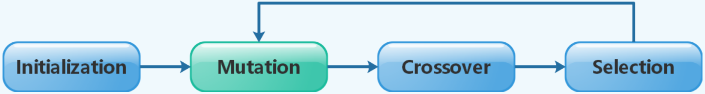


Figure: DE 流程图

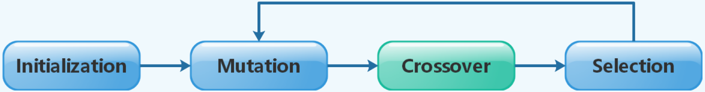
差分变异算子



差分演化算法的核心算子是差分变异(Differential mutation)算子, 其中, 经典算子“DE/rand/1”为:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right)$$

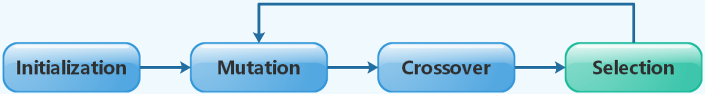
杂交算子



差分演化算法采用离散重组 (discrete recombination), 常用的二项式杂交算子为:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \mathrm{rndreal}(0,1) < Cr \mid\mid j == j_{\mathrm{rand}} \\ x_{i,j}, & \text{otherwise} \end{cases}$$

选择算子



差分演化算法采用**一对一锦标赛选择** (one-to-one tournament selection) 算子, 即目标向量 $\mathbf{x}_i$ 与实验向量 $\mathbf{u}_i$ 相比较, 较好个体保存到一下代:

$$\mathbf{x}_i' = \begin{cases} \mathbf{u}_i, & \text{if } f(\mathbf{u}_i) \le f(\mathbf{x}_i) \\ \mathbf{x}_i, & \text{otherwise} \end{cases}$$

变异, 重组 (杂交) , 选择算子重复执行, 直到终止条件达到.

智能优化技术

粒子群优化（Particle Swarm Optimization）

龚文引 (教授、博士生导师)

中国地质大学（武汉）计算机学院

Learning from Nature

“自然界的蚁群、鸟群、鱼群、羊群、牛群、蜂群等，其实时时刻刻都在给予我们以某种启示，只不过我们常常忽略了大自然对我们的最大恩赐！......”

——马良教授《蚁群优化算法》

1. 大纲

- 算法简介
- 算法流程
- 简单示例
- PSO 的构成要素
- 小结

2. 算法简介

- **算法简介**
- 算法流程
- 简单示例
- PSO 的构成要素
- 小结

2. 算法简介

2. 算法简介

粒子群优化（Particle swarm optimization）

- 简称 PSO;
- 由 Kennedy 和 Eberhart 于 1995 年提出;
- 群体迭代，粒子（particle）在解空间追随最优的粒子进行搜索;
- PSO 和差分演化算法已成为现代优化方法领域研究的热点.

Kennedy, J. and Eberhart, R. “Particle Swarm Optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.

2. 算法简介

算法优点

- 简单易行;
- 收敛速度快;
- 设置参数少.

2. 算法简介

算法优点

- 简单易行;
- 收敛速度快;
- 设置参数少.

算法基本思想

- PSO 的思想源于对鸟群捕食行为的研究
- 模拟鸟集群飞行觅食的行为，鸟之间通过集体的协作使群体达到最优目的，是一种基于 Swarm Intelligence 的优化方法
- PSO 结合了社会行为（Social-only model）和个体认知（Cognition-only model）

2. 算法简介

算法介绍

- 每个寻优的问题解都被想像成一只鸟，称为“粒子”。所有粒子都在一个 n 维空间进行搜索
- 所有的粒子都由一个 fitness function 确定适应值以判断目前的位置（position）好坏
- 每一个粒子必须赋予记忆功能，能记住所搜寻到的最佳位置
- 每一个粒子还有一个速度（velocity）以决定飞行的距离和方向。这个速度根据它本身的飞行经验以及同伴的飞行经验进行动态调整

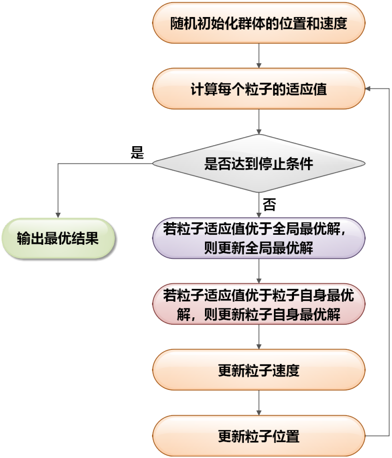
相关概念

- 群体（Swarm）：粒子的集合，相当于遗传算法的种群（population）；
- 粒子（Particle）：群体中的搜索个体
  - 位置（Position）：$\mathbf{x}_i = (x_{i,1}, \cdots, x_{i,n}) \in \mathbb{R}^n$
  - 速度（Velocity）：$\mathbf{v}_i = (v_{i,1}, \cdots, v_{i,n}) \in \mathbb{R}^n$
  - 当前粒子自身最优位置（Individual best position）: $\mathrm{PBest}_i = (p_{i,1}, \cdots, p_{i,n}) \in \mathbb{R}^n$
- 群体全局最优位置（Global best position）: $\mathrm{GBest} = (g_1, \cdots, g_n) \in \mathbb{R}^n$

3. 算法流程

- 算法简介
- **算法流程**
- 简单示例
- PSO 的构成要素
- 小结

3. 算法流程


随机初始化群体的位置和速度
计算每个粒子的适应值
是否达到停止条件
是
否
输出最优结果
若粒子适应值优于全局最优解，
则更新全局最优解
若粒子适应值优于粒子自身最优
解，则更新粒子自身最优解
更新粒子速度
更新粒子位置

3. 算法流程

初始化（Initialization）

- 初始化位置: $x_{i,d} = \mathrm{rndreal}(X_{\min,d}, X_{\max,d})$
- 初始化速度: $v_{i,d} = \mathrm{rndreal}(V_{\min,d}, V_{\max,d})$
- $i = 1, \cdots, NP$
- $d = 1, \cdots, n$

3. 算法流程

初始化（Initialization）

- 初始化位置: $x_{i,d} = \mathrm{rndreal}(X_{\min,d}, X_{\max,d})$
- 初始化速度: $v_{i,d} = \mathrm{rndreal}(V_{\min,d}, V_{\max,d})$
- $i = 1, \cdots, NP$
- $d = 1, \cdots, n$

粒子适应值计算（Fitness evaluation）

- $f_i = f(\mathbf{x}_i)$

3. 算法流程

粒子速度更新：$\mathbf{v}_i(t) \rightarrow \mathbf{v}_i(t+1)$

$$
\begin{aligned}
\mathbf{v}_i(t+1) &= \text{Inertial} + \text{Cognitive} + \text{Social} \\
\mathbf{v}_i(t+1) &= w \times \mathbf{v}_i(t) + \\
&\quad c_1 \times \mathrm{rndreal}_1() \times (\mathrm{PBest}_i - \mathbf{x}_i(t)) + \\
&\quad c_2 \times \mathrm{rndreal}_2() \times (\mathrm{GBest} - \mathbf{x}_i(t))
\end{aligned}
$$

- w 是惯性常数
- c_1 是个体认知常数
- c_2 是社会经验常数
- $\mathrm{rndreal}_1(), \mathrm{rndreal}_2()$ 是 $(0, 1)$ 之间的随机数

粒子位置更新：$\mathbf{x}_i(t) \to \mathbf{x}_i(t+1)$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

4. 简单示例

算法简介

算法流程

简单示例

PSO 的构成要素

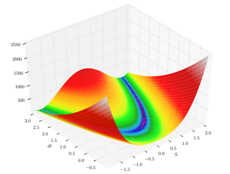
小结

问题

求解如下四维 Rosenbrock 函数的最小值：

$$f(\mathbf{x}) = \sum_{i=1}^{3} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

其中 $x_i \in [-30, 30], i = 1, 2, 3, 4.$



算法参数设置

- 群体大小：$NP = 5$
- 粒子速度范围：$[-60, 60]$
- $w = 1$
- $c_1 = c_2 = 2$

4. 简单示例

初始化位置

- $\mathbf{x}_1^{(0)} = \{21.721, -9.13677, 6.62244, 3.84079\}$
- $\mathbf{x}_2^{(0)} = \{-13.5001, -23.6131, 17.4462, -29.0515\}$
- $\mathbf{x}_3^{(0)} = \{-29.6563, -0.871811, -27.8912, 17.7425\}$
- $\mathbf{x}_4^{(0)} = \{23.6218, -16.4885, -22.7019, 25.4033\}$
- $\mathbf{x}_5^{(0)} = \{-28.0992, 22.6482, 0.675616, -8.43752\}$

初始化速度

- $\mathbf{v}_1^{(0)} = \{-19.9048, 29.562, -22.104, -5.45346\}$
- $\mathbf{v}_2^{(0)} = \{-20.5922, -28.6944, -26.3216, 19.0615\}$
- $\mathbf{v}_3^{(0)} = \{-7.83576, -55.7173, -40.9177, 28.255\}$
- $\mathbf{v}_4^{(0)} = \{-11.6373, -41.0138, 17.7311, -14.87\}$
- $\mathbf{v}_5^{(0)} = \{17.561, -13.5365, 51.2722, -56.098\}$

计算粒子适应值

- $f(\mathbf{x}_1^{(0)}) = 2.38817 \times 10^7$
- $f(\mathbf{x}_2^{(0)}) = 4.45306 \times 10^7$
- $f(\mathbf{x}_3^{(0)}) = 1.35376 \times 10^8$
- $f(\mathbf{x}_4^{(0)}) = 6.56888 \times 10^7$
- $f(\mathbf{x}_5^{(0)}) = 8.50674 \times 10^7$

保存最优个体

- 群体历史最优解: $\mathrm{GBest} = \mathbf{x}_1^{(0)}$
- 个体历史最优解: $\mathrm{PBest}_i = \mathbf{x}_i^{(0)}, i = 1, 2, 3, 4, 5$

粒子速度更新：$\mathbf{v}_i(t) \rightarrow \mathbf{v}_i(t+1)$

$$
\begin{aligned}
\mathbf{v}_i(t+1) &= \text{Inertial} + \text{Cognitive} + \text{Social} \\
\mathbf{v}_i(t+1) &= w \times \mathbf{v}_i(t) + \\
&\quad c_1 \times \mathrm{rndreal}_1() \times (\mathrm{PBest}_i - \mathbf{x}_i(t)) + \\
&\quad c_2 \times \mathrm{rndreal}_2() \times (\mathrm{GBest} - \mathbf{x}_i(t))
\end{aligned}
$$

更新速度

- $\mathbf{v}_1^{(1)} = \{-19.9048, 29.562, -22.104, -5.45346\}$
- $\mathbf{v}_2^{(1)} = \{40.0498, -3.76972, -44.9573, 75.6939\}$
- $\mathbf{v}_3^{(1)} = \{14.8665, -59.3694, -25.667, 22.1122\}$
- $\mathbf{v}_4^{(1)} = \{-13.843, -32.4824, 51.7604, -39.892\}$
- $\mathbf{v}_5^{(1)} = \{95.9018, -63.5174, 60.6234, -36.7907\}$

4. 简单示例

粒子位置更新：$\mathbf{x}_i(t) \to \mathbf{x}_i(t+1)$

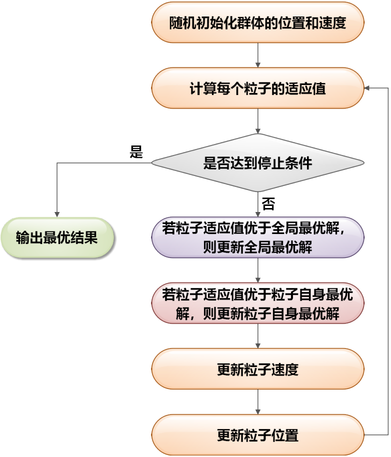
$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

4. 简单示例

更新位置

- $\mathbf{x}_1^{(1)} = \{1.81621, 20.4252, -15.4816, -1.61267\}$
- $\mathbf{x}_2^{(1)} = \{26.5497, -27.3829, -27.5112, 30.9485\}$
- $\mathbf{x}_3^{(1)} = \{-14.7898, -60.2412, -53.5582, 39.8547\}$
- $\mathbf{x}_4^{(1)} = \{9.77877, -48.971, 29.0584, -14.4887\}$
- $\mathbf{x}_5^{(1)} = \{31.9008, -37.3518, 60.6756, -45.2282\}$

4. 简单示例



随机初始化群体的位置和速度
计算每个粒子的适应值
是否达到停止条件
是
否
输出最优结果
若粒子适应值优于全局最优解，
则更新全局最优解
若粒子适应值优于粒子自身最优
解，则更新粒子自身最优解
更新粒子速度
更新粒子位置

5. PSO 的构成要素

5. PSO 的构成要素

群体大小（NP）

- NP 是一个整数
- NP 很小时，陷入局部最优的可能性很大
- NP 很大时，PSO 的优化能力很好，但算法收敛慢
- 当群体数目增长至一定水平时，再增长将不再有显著作用

5. PSO 的构成要素

惯性权重因子（w）

- w 是一个非负数
- $w = 1$ 基本 PSO 算法
- $w = 0$ 失去粒子本身的速度记忆

5. PSO 的构成要素

惯性权重因子（w）

- w 是一个非负数
- $w = 1$ 基本 PSO 算法
- $w = 0$ 失去粒子本身的速度记忆

个体认知常数（c_1）

- $c_1 = 0$ 无私型 PSO, “只有社会，没有自我"
- 迅速丧失群体多样性，易陷入局部最优而无法跳出

5. PSO 的构成要素

惯性权重因子（w）

- w 是一个非负数
- $w = 1$ 基本 PSO 算法
- $w = 0$ 失去粒子本身的速度记忆

个体认知常数（c_1）

- $c_1 = 0$ 无私型 PSO, “只有社会，没有自我"
- 迅速丧失群体多样性，易陷入局部最优而无法跳出

社会经验常数（c_2）

- $c_2 = 0$ 自私型 PSO, “只有自我，没有社会"
- 完全没有信息的社会共享，导致算法收敛速度缓慢

5. PSO 的构成要素

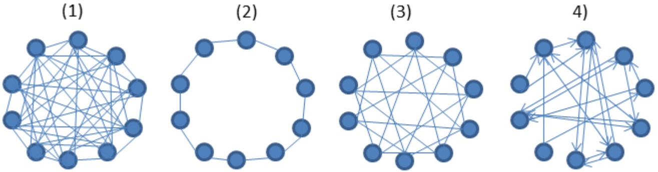
最大速度（$V_{\max}$）

- 用于维护算法的勘探能力与开采能力的平衡
- $V_{\max}$ 较大时，勘探能力强，但粒子容易飞过最优解
- $V_{\max}$ 较小时，开采能力强，但容易陷入局部最优解
- $V_{\max}$ 一般设为每维变量变化范围的$10\% - 20\%$

5. PSO 的构成要素

邻域（Neighborhood）拓扑结构

- 全局历史最优解：GBest
- 局部历史最优解：LBest_i



Graphical representation of (1) fully connected, (2) ring, (3) von Neumann and (4) random topology

5. PSO 的构成要素

思考

- PSO 的产生与测试在何处？
- PSO 的搜索方向和搜索步长如何确定？
- PSO 如何实现信息共享？
- PSO 如何实现资源竞争？
- PSO 的勘探与开采如何实现？

5. PSO 的构成要素

课外作业

在网上下载 PSO 源程序，读懂，并尝试独立实现。要求：

1. 对每一行代码添加中文注释

6. 小结

6. 小结

本章小结

1. PSO 简介
2. 算法基本流程
3. 简单示例
4. PSO 的构成要素

6. 小结

思考

在网上下载 PSO 源程序，读懂，并尝试独立实现。思考以下问题：

1. 不同邻域拓扑结构对 PSO 性能的影响如何？
2. PSO 和 DE 两种优化方法的区别与联系？
3. 尝试通过实验对比 PSO 和 DE 两种优化方法在不同测试函数上的优化性能。

6. 小结

进一步阅读资料

- M. Clerc and J. Kennedy, “The particle swarm - explosion, stability, and convergence in a multidimensional complex space,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.
- Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez and R. G. Harley, “Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171-195, 2008.
- M. R. Bonyadi and Z. Michalewicz, “Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review,” *Evolutionary Computation*, vol. 25, no. 1, pp. 1-54, 2017.

Thank you!

AUTHOR: GONG, Wenyin
ADDRESS: School of Computer Science,
China University of Geosciences,
Wuhan, 430074, China
E-MAIL: wygong@cug.edu.cn
HOMEPAGE: http://grzy.cug.edu.cn/gongwenyin