

进化优化算法

基于仿生和种群的计算机智能方法



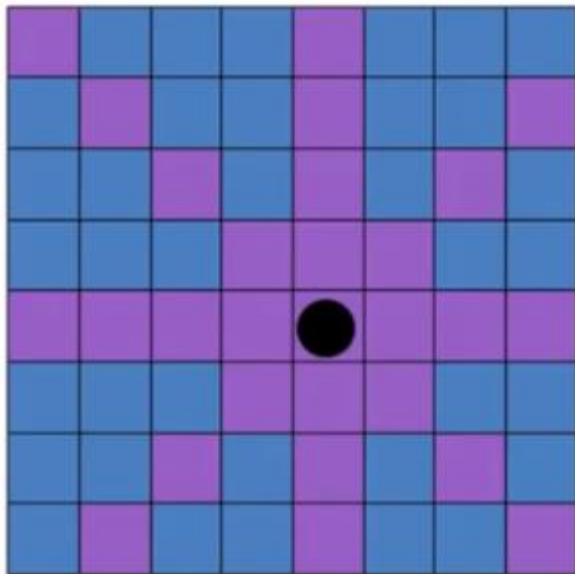
第五课：顺序编码遗传算法、差分进化算法

第二次作业

- 作业内容：八皇后问题
- 交作业时间：2024年11月18日之前
- 作业要求：
 1. 采用顺序编码遗传算法
 2. 编程语言不限
 3. 电子版作业报告一份：包含问题描述、编程思路及方法、源代码（添加适当的注释）
 4. 源代码一份，50次运行测试程序效率及准确性
 5. 程序自己编写，不要拷贝或抄袭现有的程序
 6. 每人提交一份作业

八皇后问题 (Eight queens)

问题：把8个皇后放在8×8的棋盘上，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上



八皇后问题如果用穷举法需要尝试 $8^8 = 16,777,216$ 种情况
对于 n 皇后问题，穷举法需要尝试 n^n 种情况

排序遗传算法中的交叉操作

- 单点排序交叉 (Single-point order crossover)
- 两点排序交叉 (Two-point order crossover)
- 部分映射交叉 (Partial mapped crossover)
- 基于位置的交叉 (Position based crossover)
- 边重组交叉 (Edge recombination crossover)

边重组交叉(Edge recombination crossover)

- 前面介绍的TSP交叉操作基本上考虑的是城市的位置和顺序，未考虑城市间的连接
- Grefenstette认为遗传算法应用于TSP，其遗传操作不仅要考虑城市的位置，而且有必要考虑城市间的关系，城市间的关系定义为边，让子个体继承父个体中边的信息，设计围绕边的遗传操作很有意义
- 1989年，Whitley等提出了一种被称为边重组交叉操作，使子个体能够从父个体继承边95%~99%的信息
- 边重组操作根据继承两个父个体定义的旅程中城市间的相邻状况生成子个体

边重组交叉(Edge recombination crossover)

- 例如一条路径

3	1	2	8	7	4	6	9	5
---	---	---	---	---	---	---	---	---

可以定义边有

(3, 1) (1, 2) (2, 8) (8, 7) (7, 4) (4, 6) (6, 9) (9, 5) (5, 3)

- TSP中, 最小化的目标函数是由合法路径中所有边的总和构成的, 从这个角度考虑, 路径中城市的位置不是特别重要
- 因此, 边重组操作中, 对于每一城市, 将任意一父个体中与之邻接的其他城市列出构成列表, 然后利用两个父个体对应的路径中的边表来设计交叉操作

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
										2	“1” 3 8
										3	2 4 5 9
										4	3 “5” 1
										5	“4” 6 3
										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

根据两个父代染色体生成
每一个城市的边表，其中两个父
个体中均有的边，数字打上标记
“”，边重组操作中产生子个体时优
先考虑打有标记的城市码

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
父代染色体为 P_1 和 P_2 ，子代染色体为 C_1 (初始为空)										2	“1” 3 8
步骤:										3	2 4 5 9
1) 从父代染色体 P_1 的初始城市 (X) 开始										4	3 “5” 1
2) 将 X 复制到子代染色体 C_1										5	“4” 6 3
3)从边表右侧列表中删除所有的 X										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

开始城市1

1								
---	--	--	--	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
步骤:										2	“1” 3 8
4) 从与城市 <i>X</i> 相邻的城市列表中选择下一个城市 <i>Y</i> , 优先选择具有 “ ” 标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个										3	2 4 5 9
5)重复第2步到第4步, 直至完成整个旅行										4	3 “5” 1
										5	“4” 6 3
										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

下一城市选择2

1	2							
---	---	--	--	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
步骤:										2	“1” 3 8
4) 从与城市 <i>x</i> 相邻的城市列表中选择下一个城市 <i>y</i> , 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个										3	2 4 5 9
5)重复第2步到第4步, 直至完成整个旅行										4	3 “5” 1
										5	“4” 6 3
										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

下一城市选择2

1	2							
---	---	--	--	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

城市	连接
1	9 “2” 4
2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市2
下一城市在3和8之间选择, 8的连接更少, 选择8

1	2	8						
---	---	---	--	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
步骤:										2	“1” 3 “8”
4) 从与城市 <i>x</i> 相邻的城市列表中选择下一个城市 <i>y</i> , 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个										3	2 4 5 9
5)重复第2步到第4步, 直至完成整个旅行										4	3 “5” 1
当前城市2										5	“4” 6 3
下一城市在3和8之间选择, 8的连接更少, 选择8										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

1	2	8						
---	---	---	--	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
步骤:										2	“1” 3 “8”
4) 从与城市 <i>x</i> 相邻的城市列表中选择下一个城市 <i>y</i> , 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个										3	2 4 5 9
5)重复第2步到第4步, 直至完成整个旅行										4	3 “5” 1
当前城市2										5	“4” 6 3
下一城市在3和8之间选择, 8的连接更少, 选择8										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

1	2	8						
---	---	---	--	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

当前城市8
下一城市优先选择7

1	2	8	7					
---	---	---	---	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市8
下一城市优先选择7

城市	连接
1	9 “2” 4
2	“1” 3 “8”
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	7 9 2
9	8 1 6 3

1	2	8	7					
---	---	---	---	--	--	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

城市	连接
1	9 “2” 4
2	“1” 3 “8”
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	7 9 2
9	8 1 6 3

当前城市7
下一城市选择6

1	2	8	7	6			
---	---	---	---	---	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

城市	连接
1	9 “2” 4
2	“1” 3 “8”
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

当前城市7
下一城市选择6

1	2	8	7	6			
---	---	---	---	---	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市6
下一城市在5和9之间选择, 9的连接更少, 选择9

城市	连接
1	9 “2” 4
2	“1” 3 “8”
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

1	2	8	7	6	9			
---	---	---	---	---	---	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
										2	“1” 3 8
										3	2 4 5 9
										4	3 “5” 1
										5	“4” 6 3
										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市6

下一城市在5和9之间选择, 9的连接更少, 选择9

1	2	8	7	6	9			
---	---	---	---	---	---	--	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*，优先选择具有“”标记的城市码，如果没有标记，优先列表中具有最少连接的城市，如果连接数相同，任意选择一个

5)重复第2步到第4步，直至完成整个旅行

城市	连接
1	9 “2” 4
2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

当前城市9
下一城市选择3

1	2	8	7	6	9	3		
---	---	---	---	---	---	---	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

城市	连接
1	9 “2” 4
2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

当前城市9
下一城市选择3

1	2	8	7	6	9	3		
---	---	---	---	---	---	---	--	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市3
下一城市在4和5之间选择, 4和5
的连接数相同, 随机选择一个,
假设选择5

2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

1	2	8	7	6	9	3	5	
---	---	---	---	---	---	---	---	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市3

下一城市在4和5之间选择, 4和5的连接数相同, 随机选择一个, 假设选择5

城市	连接
1	9 “2” 4
2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

1	2	8	7	6	9	3	5	
---	---	---	---	---	---	---	---	--

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9
父代2:	4	1	2	8	7	6	9	3	5

步骤:

4) 从与城市*x*相邻的城市列表中选择下一个城市*y*, 优先选择具有“”标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个

5)重复第2步到第4步, 直至完成整个旅行

当前城市5
下一城市选择4

城市	连接
1	9 “2” 4
2	“1” 3 8
3	2 4 5 9
4	3 “5” 1
5	“4” 6 3
6	5 “7” 9
7	“6” “8”
8	“7” 9 2
9	8 1 6 3

1	2	8	7	6	9	3	5	4
---	---	---	---	---	---	---	---	---

边重组交叉(Edge recombination crossover)

父代1:	1	2	3	4	5	6	7	8	9	城市	连接
父代2:	4	1	2	8	7	6	9	3	5	1	9 “2” 4
步骤:										2	“1” 3 8
4) 从与城市 <i>x</i> 相邻的城市列表中选择下一个城市 <i>y</i> , 优先选择具有 “ ” 标记的城市码, 如果没有标记, 优先列表中具有最少连接的城市, 如果连接数相同, 任意选择一个										3	2 4 5 9
5)重复第2步到第4步, 直至完成整个旅行										4	3 “5” 1
										5	“4” 6 3
										6	5 “7” 9
										7	“6” “8”
										8	“7” 9 2
										9	8 1 6 3

当前城市4, 得到C1
同样可以得到C2

C1:	1	2	8	7	6	9	3	5	4
-----	---	---	---	---	---	---	---	---	---

顺序遗传算法中的变异操作

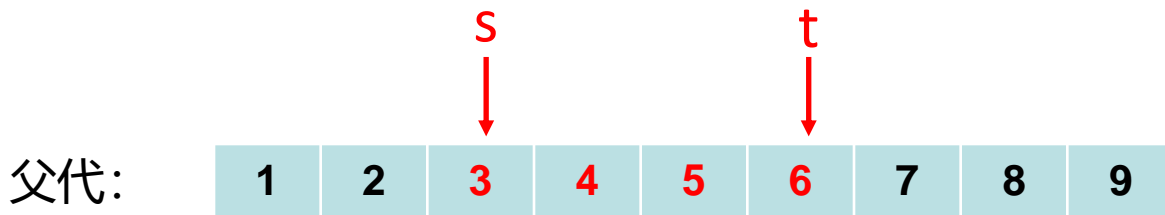
- 相反变异 (Inversion Mutation)
- 插入变异 (Insertion Mutation)
- 交换变异 (Swap Mutation)
- 启发式变异 (Heuristic Operators)

相反变异 (Inversion Mutation)

假设 P 是一个父代染色体, 染色体的长度为 L

步骤:

- 1) 在 $[1: L - 1]$ 之间随机产生一个位置 s , 满足 $1 \leq s \leq L - 1$
- 2) 在 $[s + 1 : L]$ 之间随机产生一个位置 t , 满足 $s + 1 \leq t \leq L$
- 3) s 和 t 之间的基因构成了选择的子字符串
- 4) 将选择的子字符串取反来产生子代



相反变异 (Inversion Mutation)伪代码

procedure: Inversion Mutation

input: 染色体 v 染色体长度 l

output: 子代染色体 v'

begin

$s \leftarrow \text{random}[1:l-1]$;

$t \leftarrow \text{random}[s+1:l]$;

$S \leftarrow \text{invert}(v[s:t])$;

$v' \leftarrow v[1:s-1] // S // v[t+1:l]$;

output 子代染色体 v' ;

end

其中, v 是父代染色体, l 是染色体的长度, v' 是子代染色体, s 为子字符串的起始位置, t 为子字符串的终止位置, S 为子字符串的反向排序字符串, $\text{invert}(\text{string})$ 表示取 string 相反顺序。

排序遗传算法中的变异操作

- 相反变异 (Inversion Mutation)
- 插入变异 (Insertion Mutation)
- 交换变异 (Swap Mutation)
- 启发式变异 (Heuristic Operators)

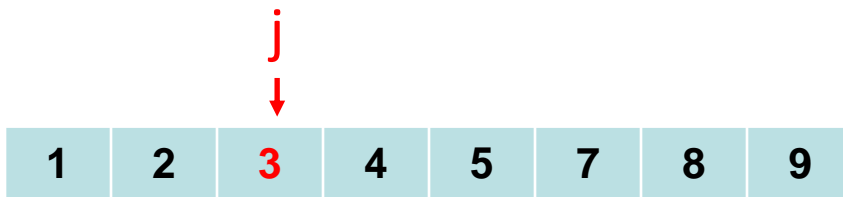
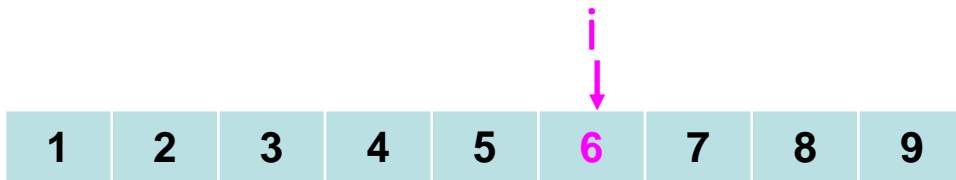
插入变异(Insertion Mutation)

假设 P 是一个父代染色体, 染色体的长度为 L

步骤:

- 1) 在 $[1: L]$ 之间随机产生一个位置 i , 满足 $1 \leq i \leq L$
- 2) 在 $[1: L - 1]$ 之间随机产生一个位置 j , 满足 $1 \leq j \leq L - 1$
- 3) 将父代染色体中第 i 个基因值插入到父代个体中第 j 个基因的前方

父代:



子代:



插入变异(Insertion Mutation)伪代码

procedure: 插入变异

input: 染色体 v , 染色体长度 l

output: 子代染色体 v'

begin

$i \leftarrow \text{random}[1:l]$;

$j \leftarrow \text{random}[1:l-1]$;

$W \leftarrow v[1:i-1] \text{ // } v[i+1:l]$;

$v' \leftarrow W[1:j-1] \text{ // } v[i] \text{ // } W[j:l-1]$;

output 子代染色体 v' ;

end

其中, v 是父代染色体, l 是染色体的长度, v' 是子代染色体, i 是在父代个体中选择的位置 1, j 是在父代个体中选择的位置 2, W 为工作数据集。

排序遗传算法中的变异操作

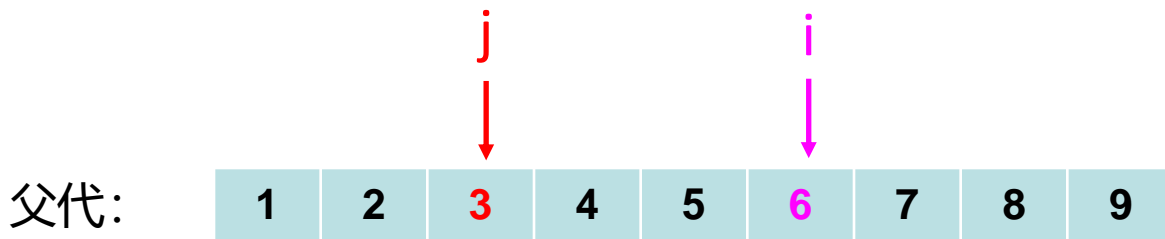
- 相反变异 (Inversion Mutation)
- 插入变异 (Insertion Mutation)
- 交换变异 (Swap Mutation)
- 启发式变异 (Heuristic Operators)

交换变异 (Swap Mutation)

假设 P 是一个父代染色体, 染色体的长度为 L

步骤:

- 1) 在 $[1: L]$ 之间随机产生两个不同位置 i, j , 满足 $1 \leq i \neq j \leq L$
- 2) 将父代染色体中第 i 个基因值与父代个体中第 j 个基因进行交换产生子代染色体



交换变异 (Swap Mutation)伪代码

procedure: Swap Mutation

input: 染色体 v 染色体长度 l

output: 子代染色体 v'

begin

$i \leftarrow \text{random}[1:l-1]$;

$j \leftarrow \text{random}[i+1:l]$;

$v' \leftarrow v[1:i-1] // v[j] // v[i+1:j-1] // v[i] // v[j+1:l]$;

output 子代染色体 v' ;

end

其中, v 是父代染色体, l 是染色体的长度, v' 是子代染色体, i 、 j 是随机选择的两个位置。

排序遗传算法中的变异操作

- 相反变异 (Inversion Mutation)
- 插入变异 (Insertion Mutation)
- 交换变异 (Swap Mutation)
- 启发式变异 (Heuristic Operators)

启发式变异(Heuristic Operators)

假设 P 是一个父代染色体, 染色体的长度为 L , m 为选择位置的总数, r 为选择的位置, N 是邻域染色体的总数, $F(p[i])$ 为 $p[i]$ 的适应度值

步骤:

- 1) 选择位置并产生邻域。给定如下的父代染色体, 从而产生5个邻域染色体
- 2) 通过计算邻域来产生子代, 假设适应度数值越小, 表示当前个体越好, 因此选出的染色体为邻域中第4条染色体

										$F(p[i])$
父代:	1	2	3	4	5	6	7	8	9	32
子代1:	1	2	3	4	5	8	7	6	9	27
子代2:	1	2	8	4	5	3	7	6	9	54
子代3:	1	2	8	4	5	6	7	3	9	45
子代4:	1	2	6	4	5	8	7	3	9	23
子代5:	1	2	6	4	5	3	7	8	9	56

启发式变异(Heuristic Operators)伪代码

procedure: Heuristic Mutation

input: 染色体 v , 染色体长度 l

output: 子代染色体 v'

begin

$P \leftarrow \phi$;

for $i=1$ **to** m {

$r \leftarrow \text{random}[1:l]$;

$P \leftarrow P \cup nb(v[r])$;}

$w \leftarrow F(p[1])$;

for $i=2$ **to** $|P|$

if $w > F(p[i])$ **then**

$w \leftarrow F(p[i])$, $n \leftarrow i$;

$v' \leftarrow p[n]$;

output 子代染色体 v' ;

end

其中, v 是父代染色体, l 是染色体的长度, v' 是子代染色体, m 为选择位置的总数, r 为选择的位置, N 是邻域染色体的总数, w 为工作数据, n 是 P 中具有最好适应度值的染色体的位置, $P\{p[i]\}$, $i=1, 2, \dots, N$ 是邻域染色体的集合, $nb(v[r])$ 用于搜索第 r 个基因的邻域, $F(p[i])$ 为 $p[i]$ 的适应度值。

-
- 顺序编码遗传算法
 - 举例说明顺序编码遗传算法的工作机理
 - 遗传算法参数经验设置
 - 遗传算法的优点和局限

旅行商 (TSP) 问题

旅行商 (TSP) 问题是一类受到广泛研究的组合优化问题。TSP问题描述很简单, 即旅行商要访问 n 个城市, 那他要以怎样的访问顺序来访问这 n 个城市才能使得行走路径最短。

TSP问题中常用的符号

(1) 索引

i, j : 城市索引, 其中 $i, j = 1, 2, \dots, n$

(2) 参数

n : 城市总数

d_{ij} : 城市 i 到城市 j 的距离, 由 d_{ij} 构成的矩阵是对称矩阵

(3) 决策变量

x_{ij} : 0, 1决策变量, 若选择路线 (i, j) , 则 $x_{ij} = 1$, 否则取值为0

TSP问题的数学模型

*TSP*问题一般可以表述为如下形式：

$$\min \quad Z = \sum_{i=1}^n \sum_{j=1}^n (d_{ij} x_{ij} + d_{n1} x_{n1})$$

$$s.t. \quad \sum_{i=1}^n x_{ij} = 1, \quad j = 1, 2, 3, \dots, n$$

$$\sum_{j=1}^n x_{ij} = 1, \quad i = 1, 2, 3, \dots, n$$

其中，

$$x_{ij} = \begin{cases} 1, & \text{若选中 } route(i, j) \\ 0, & \text{否则} \end{cases}$$

TSP问题的表示方法：直接表示法

直接表示法是TSP问题最自然的表示方式，在这种表示方式下城市按照它们被访问的顺序列出。我们给出如下的染色体

1 2 3 4 5 6 7 8 9

染色体3 2 5 4 7 1 6 9 8

它表示访问城市的列表为3 - 2 - 5 - 4 - 7 - 1 - 6 - 9 - 8

下面给出序列编码和解码的一般过程：

procedure: 序列编码	procedure: 序列解码
input: 城市集合, 城市总数 N	input: 染色体 v , 城市总数 N
output: 染色体 v	output: 访问列表 L
<div>begin for $j=1$ to N $v[j] \leftarrow j$; for $i=1$ to $\lceil N/2 \rceil$ 重复如下操作 $j \leftarrow \text{random}[1, N]$; $l \leftarrow \text{random}[1, N]$; 直到 $l \neq j$ 交换 $v[j]$和 $v[l]$; output 染色体 v; end</div>	<div>begin $L \leftarrow \phi$; for $i=1$ to N $L \leftarrow L \cup v[i]$; output 访问列表 L; end</div>

TSP问题的表示方法：随机数表示法

随机数表示法从 $(0, 1)$ 间选取随机数来编码一个解，之后将产生的若干随机数从小到大排序，其排序顺序就是相应城市的访问顺序
具体地，我们给出如下染色体

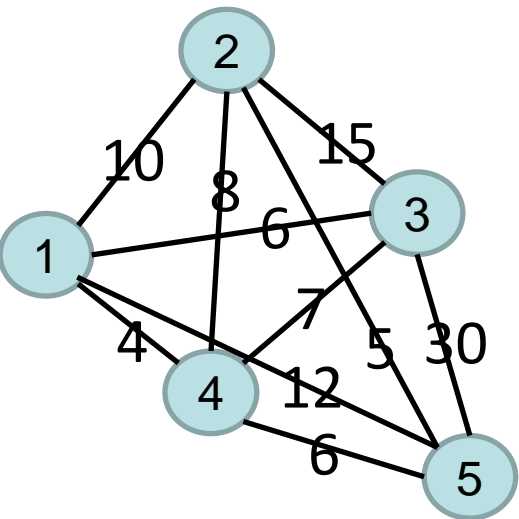
	1	2	3	4	5	6	7	8	9
染色体	0.23	0.82	0.45	0.74	0.87	0.11	0.56	0.69	0.78

其中，染色体中基因的位置 i 就代表了城市 i ，将上述染色体中的随机数按照从小到大排序后，可以得到访问城市的列表为 $6 - 1 - 3 - 7 - 8 - 4 - 9 - 2 - 5$

下面给出随机数编码和解码的一般过程：

procedure: 随机数编码	procedure: 随机数解码
input: 城市集合, 城市总数 N output: 染色体 v begin for $i=1$ to N $v[i] \leftarrow \text{random}[0, 1]$; output 染色体 v ; end	input: 染色体 v , 城市总数 N output: 访问列表 L begin $L \leftarrow \phi$ for $i=1$ to N $L \leftarrow L \cup i$; 根据 $v[i]$ 对 L 排序; output 访问列表 L ; end

示例：5个城市的TSP问题



d_{ij}	1	2	3	4	5
1	0	10	6	4	12
2	10	0	15	8	5
3	6	15	0	7	30
4	4	8	7	0	6
5	12	5	30	6	0

示例：5个城市的TSP问题

*TSP*问题一般可以表述为如下形式：

$$\min \quad Z = \sum_{i=1}^n \sum_{j=1}^n (d_{ij} x_{ij} + d_{n1} x_{n1})$$

第一步：确定种群规模, 迭代次数, 交叉概率, 变异概率

$$N_p = 4 \quad T = 10 \quad p_c = 0.8 \quad p_m = 0.3$$

示例：5个城市的TSP问题

第二步：随机生成初始种群, 并且计算适应度函数 $f(x) = 1 / z$

$$P = \begin{bmatrix} 2 & 4 & 1 & 3 & 5 \\ 3 & 2 & 4 & 5 & 1 \\ 3 & 5 & 2 & 1 & 4 \\ 5 & 3 & 1 & 2 & 4 \end{bmatrix} \quad z = \begin{bmatrix} 53 \\ 47 \\ 56 \\ 60 \end{bmatrix} \quad f = \begin{bmatrix} 0.0189 \\ 0.0213 \\ 0.0179 \\ 0.0167 \end{bmatrix}$$

选择：轮盘赌选择

第三步：利用轮盘赌方式选择一对父代解，轮盘赌运行2次

$$P = \begin{bmatrix} 2 & 4 & 1 & 3 & 5 \\ 3 & 2 & 4 & 5 & 1 \\ 3 & 5 & 2 & 1 & 4 \\ 5 & 3 & 1 & 2 & 4 \end{bmatrix} \quad Z = \begin{bmatrix} 53 \\ 47 \\ 56 \\ 60 \end{bmatrix}$$
$$f = \begin{bmatrix} 0.0189 \\ 0.0213 \\ 0.0179 \\ 0.0167 \end{bmatrix} \quad \text{选择的概率} p = \begin{bmatrix} 0.253 \\ 0.285 \\ 0.239 \\ 0.223 \end{bmatrix}$$

假设运行两次轮盘赌得到的第一对父代解为

$$Parent_1 = [2 \quad 4 \quad 1 \quad 3 \quad 5]$$

$$Parent_2 = [3 \quad 2 \quad 4 \quad 5 \quad 1]$$

交叉：单点排序交叉

第四步：随机生成一个随机数决定是否进行交叉

假设 $r = 0.4$

$r < p_c = 0.8 \rightarrow$ 进行交叉操作

第五步：随机选择一个交叉点(采用单点排序交叉)

假设 $r = 2$

$$\begin{array}{ll} \text{Parent}_1 = \begin{bmatrix} 2 & 4 & 1 & 3 & 5 \end{bmatrix} & \text{offspring}_1 = \begin{bmatrix} 2 & 4 & & & \end{bmatrix} \\ \text{Parent}_2 = \begin{bmatrix} 3 & 2 & 4 & 5 & 1 \end{bmatrix} & \text{offspring}_2 = \begin{bmatrix} 3 & 2 & & & \end{bmatrix} \end{array}$$

交叉：单点排序交叉

第四步：随机生成一个随机数决定是否进行交叉

假设 $r = 0.4$

$r < p_c = 0.8 \rightarrow$ 进行交叉操作

第五步：随机选择一个交叉点(采用单点排序交叉)

假设 $r = 2$

$Parent_1 = [2 \quad 4 \quad 1 \quad 3 \quad 5]$

$Parent_2 = [3 \quad 2 \quad 4 \quad 5 \quad 1]$

$offspring_1 = [2 \quad 4 \quad 3 \quad 5 \quad 1]$

$offspring_2 = [3 \quad 2 \quad 4 \quad 1 \quad 5]$

交叉：单点排序交叉

第六步：利用轮盘赌方式选择第二对父代解，轮盘赌运行2次

$$P = \begin{bmatrix} 2 & 4 & 1 & 3 & 5 \\ 3 & 2 & 4 & 5 & 1 \\ 3 & 5 & 2 & 1 & 4 \\ 5 & 3 & 1 & 2 & 4 \end{bmatrix} \quad z = \begin{bmatrix} 53 \\ 47 \\ 56 \\ 60 \end{bmatrix}$$
$$f = \begin{bmatrix} 0.0189 \\ 0.0213 \\ 0.0179 \\ 0.0167 \end{bmatrix} \quad \text{选择的概率} p = \begin{bmatrix} 0.253 \\ 0.285 \\ 0.239 \\ 0.223 \end{bmatrix}$$

假设运行两次轮盘赌得到的第二对父代解为

$$Parent_1 = [3 \quad 5 \quad 2 \quad 1 \quad 4]$$
$$Parent_2 = [5 \quad 3 \quad 1 \quad 2 \quad 4]$$

交叉：单点排序交叉

第七步：随机生成一个随机数决定是否进行交叉

假设 $r = 0.3$

$r < p_c = 0.8 \rightarrow$ 进行交叉操作

第五步：随机选择一个交叉点(采用单点排序交叉)

假设 $r = 1$

$$\begin{array}{ll} Parent_3 = \begin{bmatrix} 3 & 5 & 2 & 1 & 4 \end{bmatrix} & offspring_3 = \begin{bmatrix} 3 & & & & \end{bmatrix} \\ Parent_4 = \begin{bmatrix} 5 & 3 & 1 & 2 & 4 \end{bmatrix} & offspring_4 = \begin{bmatrix} 5 & & & & \end{bmatrix} \end{array}$$

交叉：单点排序交叉

第七步：随机生成一个随机数决定是否进行交叉

假设 $r = 0.3$

$r < p_c = 0.8 \rightarrow$ 进行交叉操作

第五步：随机选择一个交叉点(采用单点排序交叉)

假设 $r = 1$

$$\begin{array}{lcl} \text{Parent}_3 = \begin{array}{|c|c|c|c|} \hline 3 & 5 & 2 & 1 & 4 \\ \hline \end{array} & & \text{offspring}_3 = \begin{array}{|c|c|c|c|} \hline 3 & 5 & 1 & 2 & 4 \\ \hline \end{array} \\ \text{Parent}_4 = \begin{array}{|c|c|c|c|} \hline 5 & 3 & 1 & 2 & 4 \\ \hline \end{array} & & \text{offspring}_4 = \begin{array}{|c|c|c|c|} \hline 5 & 3 & 2 & 1 & 4 \\ \hline \end{array} \end{array}$$

变异：交换变异

第九步：对所有子代进行变异操作

$$p_m = 0.3$$

	Offspring	Random number for mutation	New offspring
O_1	[2 4 3 5 1]	0.1 执行交换变异, 假设位置为1和4	[5 4 3 2 1]
O_2	[3 2 4 1 5]	0.2 执行交换变异, 假设位置为2和3	[3 4 2 1 5]
O_3	[3 5 1 2 4]	0.1 执行交换变异, 假设位置为1和4	[2 5 1 3 4]
O_4	[5 3 2 1 4]	0.5 不执行交换变异	[5 3 2 1 4]

$$O = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 3 & 4 & 2 & 1 & 5 \\ 2 & 5 & 1 & 3 & 4 \\ 5 & 3 & 2 & 1 & 4 \end{bmatrix}$$

$$Z = \begin{bmatrix} 50 \\ 67 \\ 38 \\ 65 \end{bmatrix}$$

$$f = \begin{bmatrix} 0.0200 \\ 0.0149 \\ 0.0263 \\ 0.0154 \end{bmatrix}$$

幸存策略

第十步：合并所有的解，选择最佳的 N_p ($N_p = 4$) 个解

$$P = \begin{bmatrix} 2 & 4 & 1 & 3 & 5 \\ 3 & 2 & 4 & 5 & 1 \\ 3 & 5 & 2 & 1 & 4 \\ 5 & 3 & 1 & 2 & 4 \end{bmatrix}$$

$$z = \begin{bmatrix} 53 \\ 47 \\ 56 \\ 60 \end{bmatrix}$$

$$f = \begin{bmatrix} 0.0189 \\ 0.0213 \\ 0.0179 \\ 0.0167 \end{bmatrix}$$

$$O = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 3 & 4 & 2 & 1 & 5 \\ 2 & 5 & 1 & 3 & 4 \\ 5 & 3 & 2 & 1 & 4 \end{bmatrix}$$

$$z = \begin{bmatrix} 50 \\ 67 \\ 38 \\ 65 \end{bmatrix}$$

$$f = \begin{bmatrix} 0.0200 \\ 0.0149 \\ 0.0263 \\ 0.0154 \end{bmatrix}$$

下一代种群

$$P = \begin{bmatrix} 2 & 4 & 1 & 3 & 5 \\ 3 & 2 & 4 & 5 & 1 \\ 5 & 4 & 3 & 2 & 1 \\ 2 & 5 & 1 & 3 & 4 \end{bmatrix}$$

$$z = \begin{bmatrix} 53 \\ 47 \\ 50 \\ 38 \end{bmatrix}$$

$$f = \begin{bmatrix} 0.0189 \\ 0.0213 \\ 0.0200 \\ 0.0263 \end{bmatrix}$$

TSP问题伪代码

procedure: GA 用于 TSP 问题

input: TSP 问题数据集, GA 参数

output: 最优旅行路线

begin

$t \leftarrow 0;$

通过序列编码或随机数编码产生 $P(t)$;

通过序列解码或随机数解码计算适应度 $\text{eval}(P)$;

while (终止条件未满足) **do**

应用部分映射交叉算子对 $P(t)$ 实施交叉操作以产生 $C(t)$;

应用交换变异算子对 $P(t)$ 实施变异操作以产生 $C(t)$;

应用序列解码或随机数解码计算适应度 $\text{eval}(C)$;

从 $P(t)$ 与 $C(t)$ 中选出 $P(t+1)$;

$t \leftarrow t+1;$

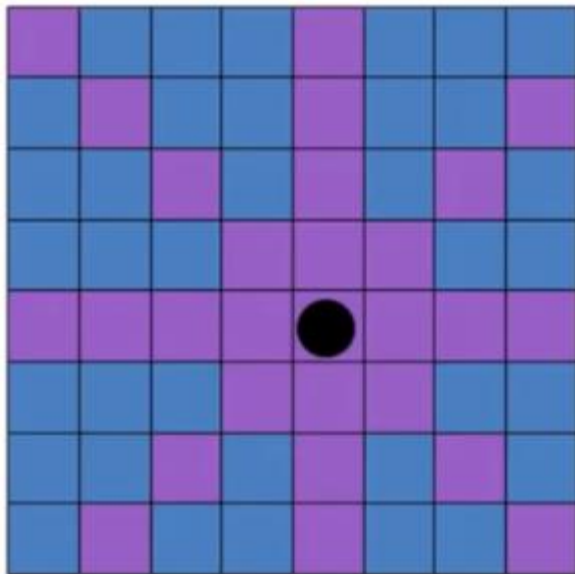
end

output 最优旅行路线;

end

八皇后问题 (Eight queens)

问题：把8个皇后放在8×8的棋盘上，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上



八皇后问题如果用穷举法需要尝试 $8^8 = 16,777,216$ 种情况
对于 n 皇后问题，穷举法需要尝试 n^n 种情况

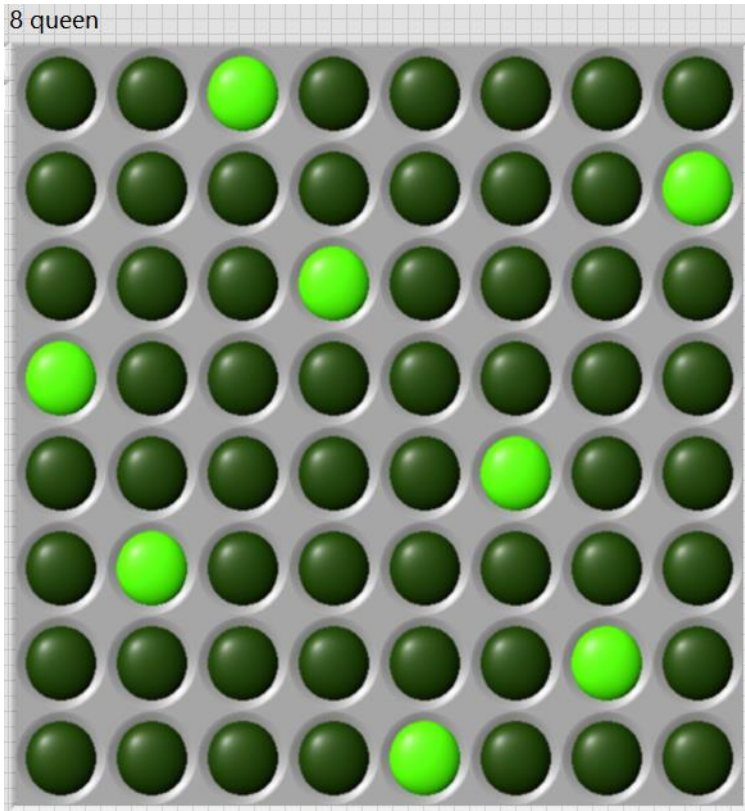
利用遗传算法求解八皇后问题

- 第一步：个体解的染色体表示
- 第二步：生成初始种群
- 第三步：应用适应度值函数
- 第四步：根据适应度值选择父代解构成配对池
- 第五步：父代解通过交叉操作产生子代解
- 第六步：子代解进行变异生成新的子代解
- 第七步：幸存者策略产生新的种群
- 第八步：重复第三步至第七步，直至找到最优解

第一步：个体解的染色体表示

4	6	1	3	8	5	7	2
---	---	---	---	---	---	---	---

- 每个皇后放在不同的列，染色体的长度为8，分别代表第1~8列，每一位的基因值代表该皇后位于哪一行
- 染色体由1~8组成，每个数字只出现一次

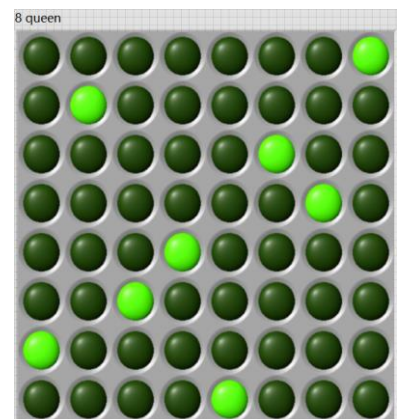
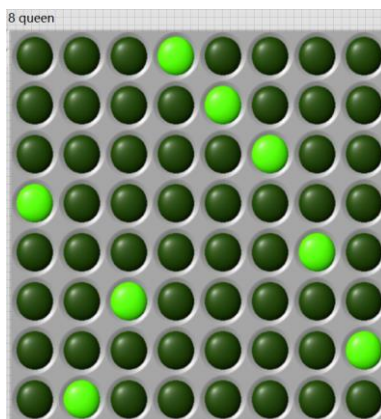
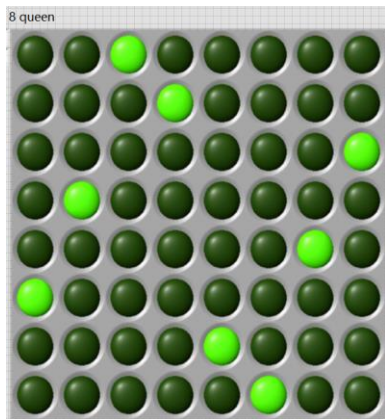
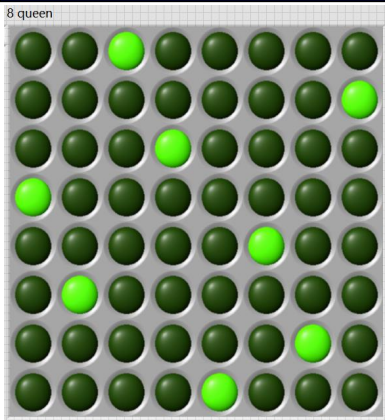


第二步：生成初始种群

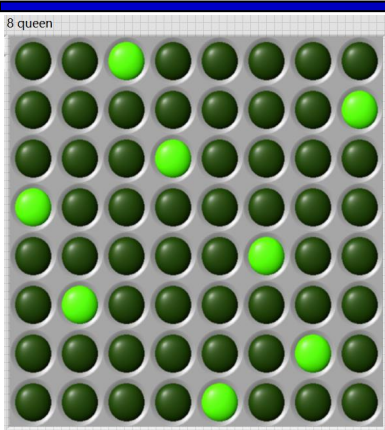
- 假设种群的规模为4，随机生成初始种群

Initial Population

4	6	1	3	8	5	7	2
6	4	1	2	7	8	5	3
4	8	6	1	2	3	5	7
7	2	6	5	8	3	4	1



第三步：应用适应度值函数



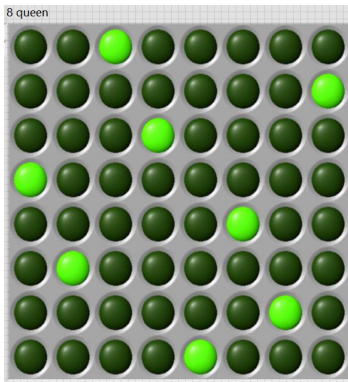
攻击皇后的数量

- Queen 1: 1
- Queen 2: 0
- Queen 3: 0
- Queen 4: 1
- Queen 5: 1
- Queen 6: 1
- Queen 7: 0
- Queen 8: 0
- Total (n): 4
- Max 适应度函数值 $f(n)=1/(1+n)$

第三步：应用适应度值函数

➤ Max 适应度函数值 $f(n)=1/(1+n)$

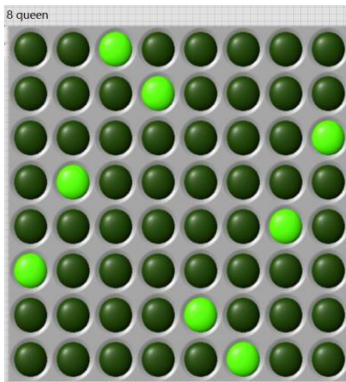
A



4	6	1	3	8	5	7	2
---	---	---	---	---	---	---	---

$$f(n)=1/5=0.2$$

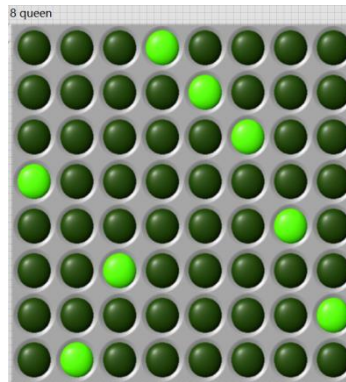
B



6	4	1	2	7	8	5	3
---	---	---	---	---	---	---	---

$$f(n)=1/17=0.059$$

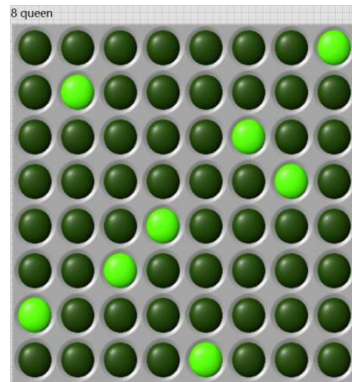
C



4	8	6	1	2	3	5	7
---	---	---	---	---	---	---	---

$$f(n)=1/13=0.077$$

D

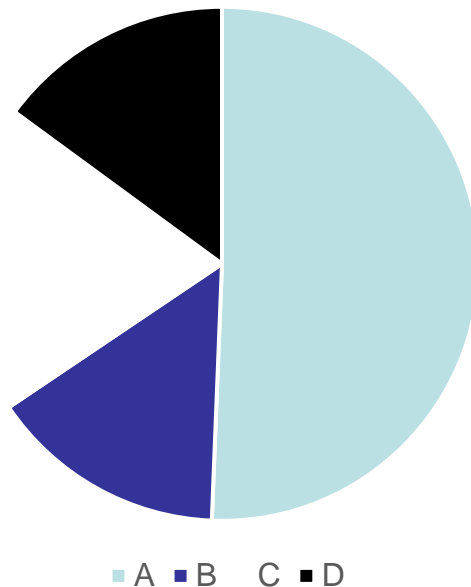


7	2	6	5	8	3	4	1
---	---	---	---	---	---	---	---

$$f(n)=1/17=0.059$$

第四步：根据适应度值选择父代解构成配对池

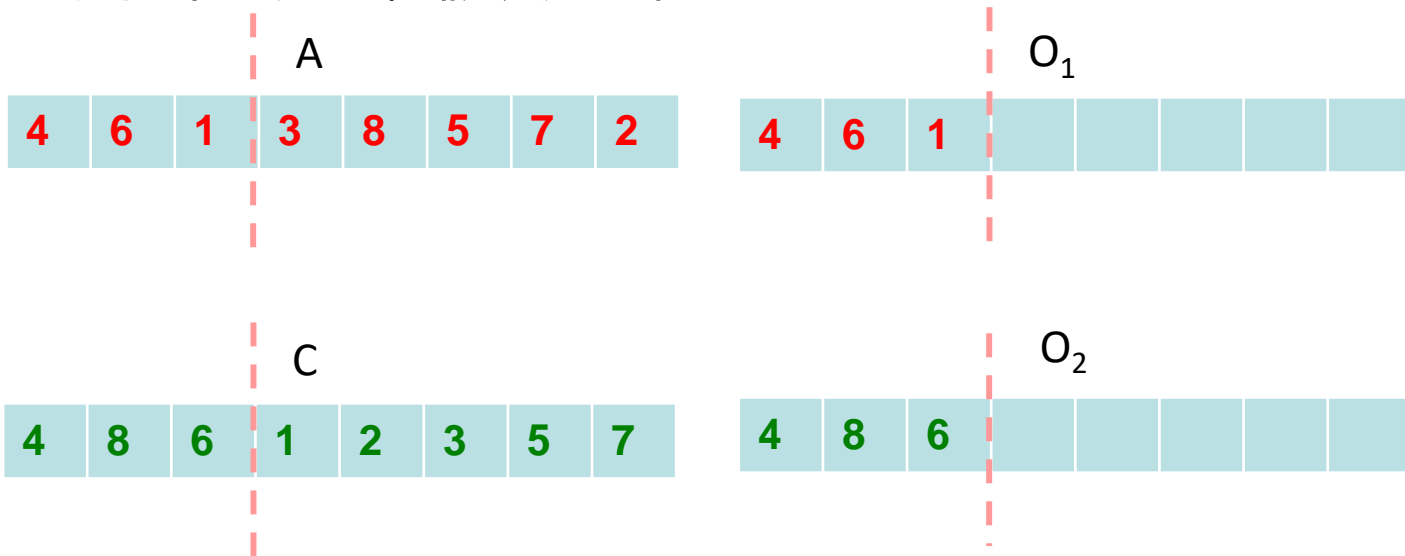
个体 (i)	适应度 函数值	轮盘 (面积)
A	0.2	50.6%
B	0.059	14.9%
C	0.077	19.5%
D	0.059	14.9%



- 采用轮盘赌方式选择父代解
- 假设选择的第一对父代解为A、C
- 第二对父代解为A、D

第五步：父代解通过交叉操作产生子代解

- 假设交叉概率 $p_c=0.8$ ，生成随机数 $r=0.3 < p_c$ ，执行交叉操作
- 执行单点交叉，假设交叉位置 $k=3$



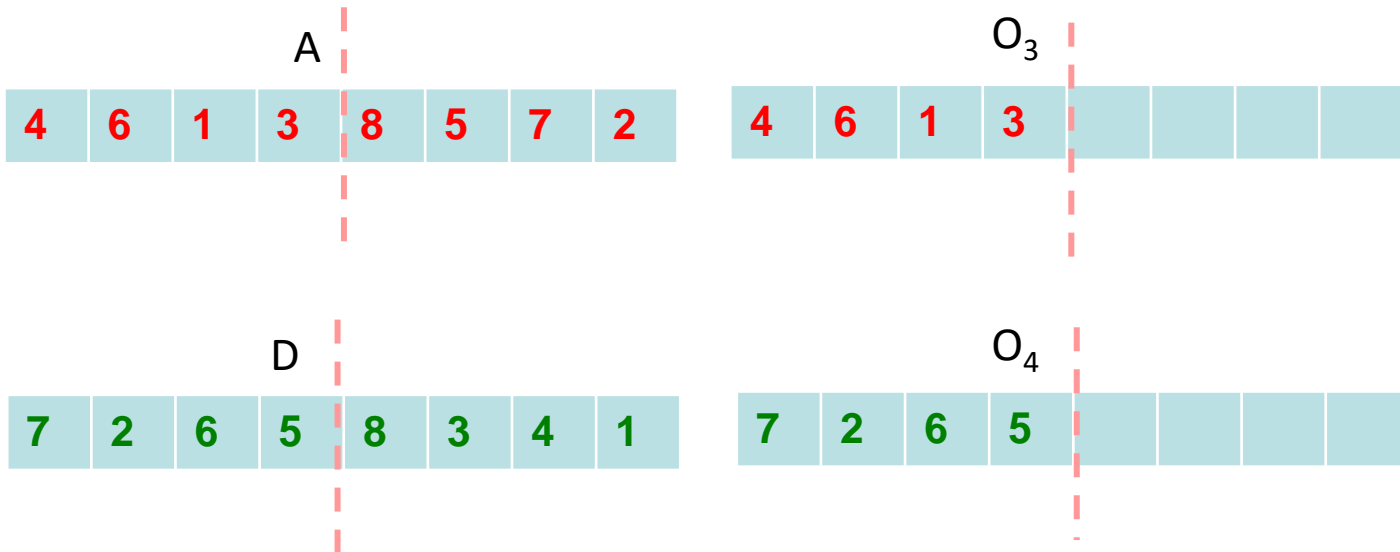
第五步：父代解通过交叉操作产生子代解

- 假设交叉概率 $p_c=0.8$ ，生成随机数 $r=0.3 < p_c$ ，执行交叉操作
- 执行单点交叉，假设交叉位置 $k=3$



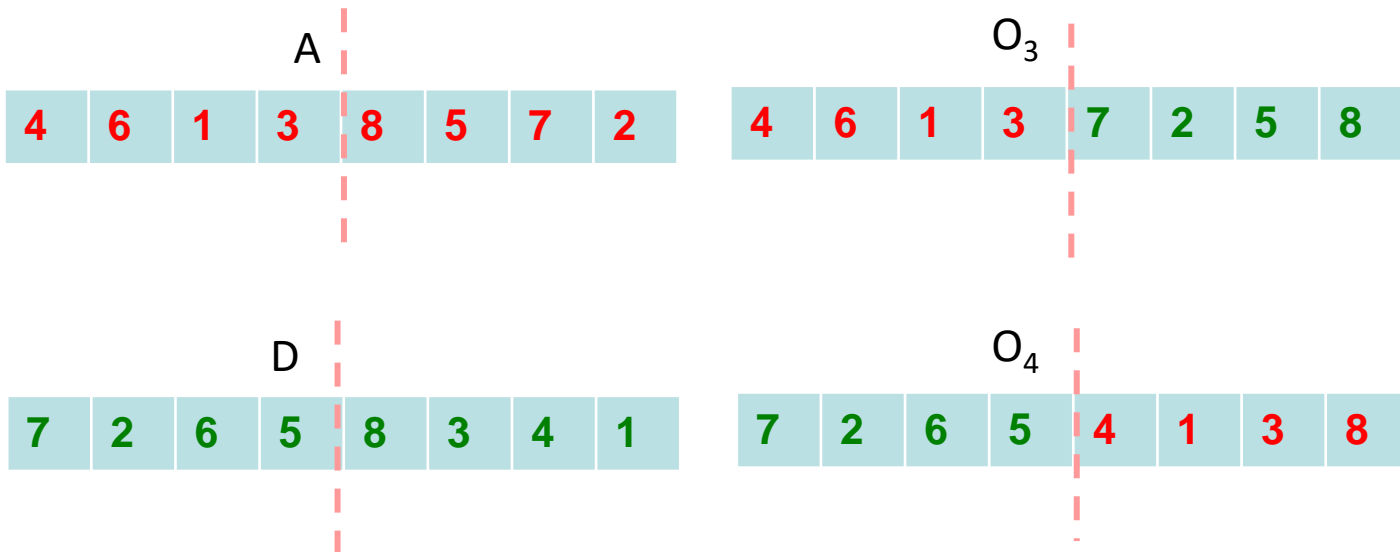
第五步：父代解通过交叉操作产生子代解

- 假设交叉概率 $p_c=0.8$ ，生成随机数 $r=0.1 < p_c$ ，执行交叉操作
- 执行单点交叉，假设交叉位置 $k=4$



第五步：父代解通过交叉操作产生子代解

- 假设交叉概率 $p_c=0.8$ ，生成随机数 $r=0.1 < p_c$ ，执行交叉操作
- 执行单点交叉，假设交叉位置 $k=4$



第六步：子代解进行变异生成新的子代解

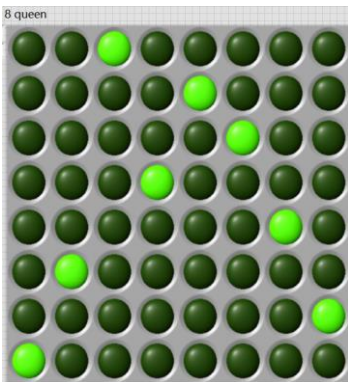
➤ 假设变异概率 $p_m=0.3$

	Offspring	Random number for mutation	New offspring
O_1	[4 6 1 8 2 3 5 7]	0.1 执行交换变异, 假设位置为1和4	[8 6 1 4 2 3 5 7]
O_2	[4 8 6 1 3 5 7 2]	0.2 执行交换变异, 假设位置为3和7	[4 8 7 1 3 5 6 2]
O_3	[4 6 1 3 7 2 5 8]	0.4 不执行交换变异	[4 6 1 3 7 2 5 8]
O_4	[7 2 6 5 4 1 3 8]	0.2 执行交换变异, 假设位置为2和6	[7 1 6 5 4 2 3 8]

第七步：幸存者策略产生新的种群

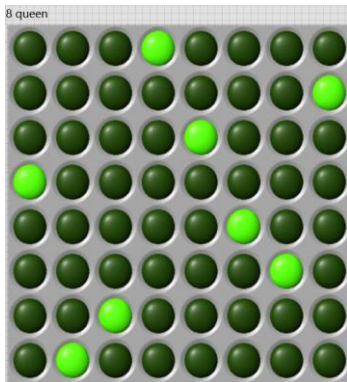
➤ 计算子代解适应度函数值 $f(n)=1/(1+n)$

O_1



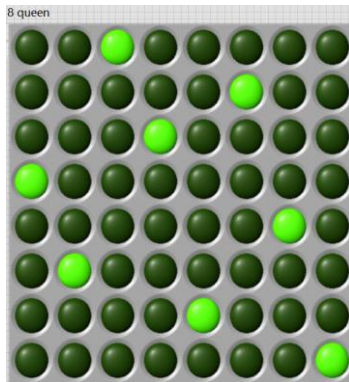
[8 6 1 4 2 3 5 7]

O_2



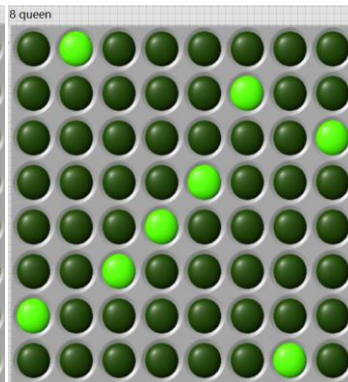
[4 8 7 1 3 5 6 2]

O_3



[4 6 1 3 7 2 5 8]

O_4



[7 1 6 5 4 2 3 8]

$$f(n)=1/9=0.111$$

$$f(n)=1/11=0.091$$

$$f(n)=1/7=0.143$$

$$f(n)=1/13=0.077$$

第七步：幸存者策略产生新的种群

➤ 合并所有的解，挑选最佳的4个解构成下一代种群

	染色体	适应度函数值
A	[4 6 1 3 8 5 7 2]	0.2
B	[6 4 1 2 7 8 5 3]	0.059
C	[4 8 6 1 2 3 5 7]	0.077
D	[7 2 6 5 8 3 4 1]	0.059

	染色体	适应度函数值
O ₁	[8 6 1 4 2 3 5 7]	0.111
O ₂	[4 8 7 1 3 5 6 2]	0.091
O ₃	[4 6 1 3 7 2 5 8]	0.143
O ₄	[7 1 6 5 4 2 3 8]	0.077

下一代种群

	染色体	适应度函数值
A	[4 6 1 3 8 5 7 2]	0.2
O ₁	[8 6 1 4 2 3 5 7]	0.111
O ₂	[4 8 7 1 3 5 6 2]	0.091
O ₃	[4 6 1 3 7 2 5 8]	0.143

-
- 顺序编码遗传算法
 - 举例说明顺序编码遗传算法的工作机理
 - 遗传算法参数经验设置
 - 遗传算法的优点和局限

遗传算法参数经验设置

- 种群规模N：影响着算法的搜索能力和运行效率
1. 若N设置较大，一次进化所覆盖的模式较多，可以保证群体的多样性，从而提高算法的搜索能力，但是由于群体中染色体的个数较多，势必增加算法的计算量，降低了算法的运行效率
 2. 若N设置较小，虽然降低了计算量，但是同时降低了每次进化群体保护更多更好染色体的能力
 3. N的设置一般为20~100

遗传算法参数经验设置

- 染色体的长度 L ：影响算法的计算量和交叉变异操作的效果
 1. L 的设置跟优化问题密切相关，一般由问题定义的解的形式和选择的编码方法决定
 2. 对于二进制编码方法，染色体的长度 L 根据解的取值范围和规定精度要求选择大小
 3. 对于浮点数编码方法，染色体的长度 L 跟问题定义的解的维数 D 相同

遗传算法参数经验设置

➤ 基因的取值范围 R ： R 视采用的染色体编码方案而定

1. 对于二进制编码方法， $R=\{0, 1\}$ ，而对于浮点数编码方法， R 与优化问题定义的解每一维变量的取值范围相同

➤ 交叉概率 P_c ：决定了进化过程种群参加交叉的染色体平均数目 $P_c \times N$

1. P_c 的取值一般为 $0.4 \sim 0.99$
2. 也可采用自适应的方法调整算法运行过程中的 P_c 值

遗传算法参数经验设置

- 变异概率 P_m ：增加群体进化的多样性，决定了进化过程中群体发生变异的基因平均个数
1. P_m 的值不宜过大。因为变异对已找到的较优解具有一定的破坏作用，如果的值 P_m 太大，可能会导致算法目前所处的较好的搜索状态倒退回原来较差的情况
 2. P_m 的取值一般为0.0001~0.1
 3. 也可采用自适应的方法调整算法运行过程中的 P_m 值

遗传算法参数经验设置

➤ 适应值评价：影响算法对种群的选择，恰当的评估函数应该能够对染色体的优劣做出合适的区分，保证选择机制的有效性，从而提高群体的进化能力

1. 评估函数的设置同优化问题的求解目标有关
2. 评估函数应满足较优染色体的适应值较大的规定
3. 为了更好地提高选择的效能，可以对评估函数做出一定的修正
4. 目前主要的评估函数修正方法有：
 - a) 线性变换
 - b) 乘幂变换
 - c) 指数变换等

遗传算法参数经验设置

➤终止条件：决定算法何时停止运行，输出找到的最优解

1. 采用何种终止条件，跟具体问题的应用有关
2. 可以使算法在达到最大进化代数时停止，最大进化代数一般可设置为100~1000，根据具体问题可对该建议值作相应的修改
3. 也可以通过考察找到的当前最优解的情况来控制算法的停止。
 - a. 当目前进化过程算法找到的最优解达到一定的误差要求，则算法可以停止。
 - b. 误差范围的设置同样跟具体的优化问题相关。
 - c. 或者是算法在持续很长的一段进化时间内所找到的最优解没有得到改善时，算法可以停止

-
- 顺序编码遗传算法
 - 举例说明顺序编码遗传算法的工作机理
 - 遗传算法参数经验设置
 - 遗传算法的优点和局限

遗传算法的优点

- 有指导搜索：适应度函数是遗传算法搜索的目标函数，在适应度函数的驱动下，遗传算法不是盲目搜索，也不用穷举，而是一代代向最优解逼近
- 自适应搜索：遗传算法借助选择、交叉、变异等进化操作，体现的是自然选择的规律，不需要添加其他附加条件，种群的品质就能不断改进
- 并行式搜索：遗传算法每一步都是针对一组个体同时进行的，而不是只对单个个体。因此，遗传算法是一种多点式齐头并行算法

遗传算法的优点

- 黑箱式结构：遗传算法只研究输入与输出的关系，并不深究造成这种关系的原因
- 通用性强：传统的优化算法需要将所要解决的问题用数学式表达出来，而且要求数学函数的一阶或二阶导数存在。而遗传算法只需要某种编码方式表达问题，然后根据适应度来区分个体优劣，其余的进化操作都是统一的
- 鲁棒性强：遗传算法吸收自然生物系统“适者生存”的进化原理，从而使它获得了很好的鲁棒性，能够适应复杂空间并能自动进行寻优搜索
- 由于遗传算法的整体搜索策略和优化搜索方法在计算时不依赖于梯度信息和其他辅助知识，只需要影响搜索方向的适应度函数，因此遗传算法成为了一种求解复杂系统问题的通用框架

遗传算法的局限

- 编码的不规范性和不准确性
- 遗传算法的效率通常低于传统的优化方法
- 遗传算法容易出现局部收敛的情况
- 遗传算法在收敛精度、可信度、计算复杂性等方面缺乏有效的定量分析方法
- 目前针对遗传算法的改进主要针对：遗传算法收敛速度慢、局部寻优能力差、产生最优解精度低等缺点进行改进，以改善其性能

进化优化算法

基于仿生和种群的计算机智能方法



差分进化算法
(Differential Evolution)

➤ 引言

➤ 差分进化算法理论和特点

➤ 差分进化算法的基本操作

➤ 举例说明差分进化算法的详细工作机制

➤ 差分进化算法的关键参数设置

➤ 小结

差分进化算法的起源与发展

- 美国学者Rainer Storn和Kenneth Price于1995年提出的一种模拟“优胜劣汰、适者生存”的自然进化法则的仿生智能算法
- 最初设想是解决Chebyshev切比雪夫多项式问题，后来发现较之其他进化算法，差分进化算法在解决复杂的全局优化问题方面的性能更加突出，过程也更为简单，受控参数少，被视为仿生智能计算产生以来在算法结构方面取得的重大进展



Rainer Storn (赖纳.斯托恩)



Kenneth Price (肯尼斯.普莱斯)

差分进化算法的起源与发展

Journal of Global Optimization 11: 341–359, 1997.
© 1997 Kluwer Academic Publishers. Printed in the Netherlands.

341

Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces

RAINER STORN
*Siemens AG, ZFE T SN2, Otto-Hahn Ring 6, D-81739 Muenchen, Germany.
(e-mail: rainer.storn@mchp.siemens.de)*

KENNETH PRICE
836 Owl Circle, Vacaville, CA 95687, U.S.A. (email: kprice@solano.comunity.net)

(Received: 20 March 1996; accepted: 19 November 1996)

Abstract. A new heuristic approach for minimizing possibly nonlinear and non-differentiable continuous space functions is presented. By means of an extensive testbed it is demonstrated that the new method converges faster and with more certainty than many other acclaimed global optimization methods. The new method requires few control variables, is robust, easy to use, and lends itself very well to parallel computation.

Key words: Stochastic optimization, nonlinear optimization, global optimization, genetic algorithm, evolution strategy.

1. Introduction

Problems which involve global optimization over continuous spaces are ubiquitous throughout the scientific community. In general, the task is to optimize certain properties of a system by pertinently choosing the system parameters. For convenience, a system's parameters are usually represented as a vector. The standard approach to an optimization problem begins by designing an objective function that can model the problem's objectives while incorporating any constraints. Especially in the circuit design community, methods are in use which do not need an objective function but operate with so-called regions of acceptability: Brayton *et al.* (1981), Lueder (1990), Storn (1995). Although these methods can make formulating a problem simpler, they are usually inferior to techniques which make use of an objective function. Consequently, we will only concern ourselves with optimization methods that use an objective function. In most cases, the objective function defines the optimization problem as a minimization task. To this end, the following investigation is further restricted to minimization problems. For such problems, the objective function is more accurately called a "cost" function.

When the cost function is nonlinear and non-differentiable, direct search approaches are the methods of choice. The best known of these are the algorithms

Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces

来自 Springer | ♥ 喜欢 0 | 阅读量: 3077

作者: R Storn, K Price

摘要: A new heuristic approach for minimizing possibly nonlinear and non-differentiable continuous space functions is presented. By means of an extensive testbed it is demonstrated that the new method converges faster and with more certainty than many other acclaimed global optimization methods. The new method requires few control variables, is robust, easy to use, and lends itself very well to parallel computation.

关键词: stochastic optimization nonlinear optimization global optimization genetic algorithm evolution strategy

DOI: 10.1023/A:1008202821328

被引量: 2.1万

年份: 1997

☆ 收藏

<> 引用

批量引用

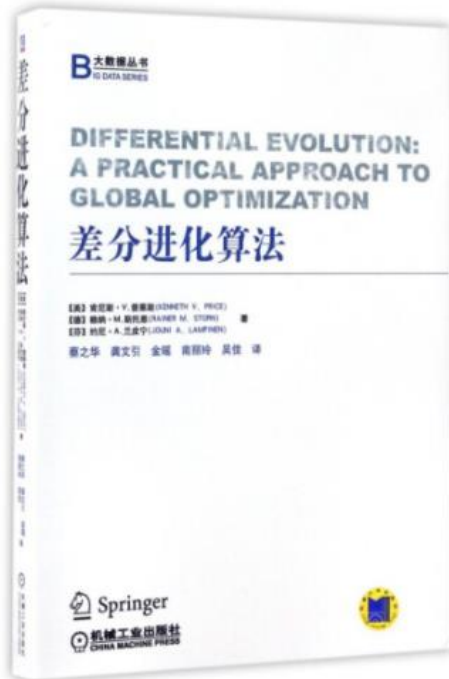
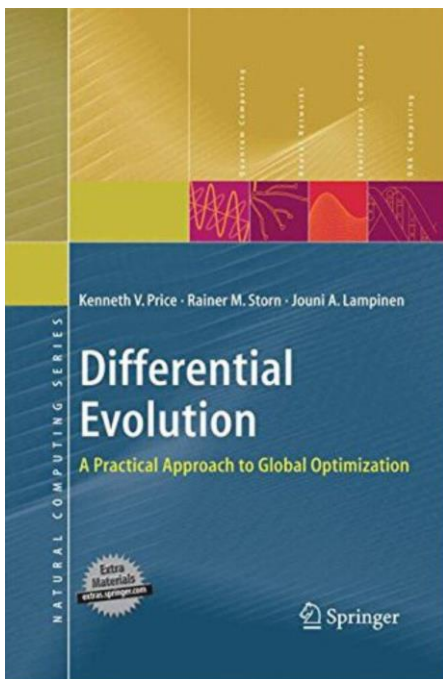
🔔 报错

🗨️ 分享

➤ 基于群体智能理论的优化算法，是通过群体内个体间的合作与竞争而产生的智能优化搜索算法

差分进化算法的起源与发展

- 2004年12月，Price、Storn和Lampinen共同编写出版了学术著作Differential Evolution: A Practical Approach to Global Optimization，该书已成为差分进化算法的经典之作



肯尼斯·V.普莱斯，1974年于美国伦斯勒理工学院获得物理学学士学位。在迁往旧金山之前，他主要在纽约Teledyne-Gurley Scientific Instrument公司担任主管。现在他居住在加州瓦卡维尔市。1994年，他发表了“遗传退火”算法，并指导Rainer Storn博士攻克切比雪夫多项式拟合问题。他提出的“差分变异”被证明是解决切比雪夫问题和其他难题的核心。

赖纳·M.斯托恩博士是差分进化算法的发明人之一。目前是IEEE高级会员，罗德施瓦兹公司平台软件研发部总监。曾先后担任西门子公司城市网络研发小组组长、伯克利国际计算机科学研究所博士后、英飞凌公司ADSL项目总监、罗德施瓦兹公司SDR项目总监。

因差分进化算法的贡献，Kenneth V. Price和Rainer Storn被授予IEEE计算智能学会“先驱奖”。

约尼·A.兰皮宁是芬兰瓦萨大学计算机科学学院教授，是差分进化算法的早期重要贡献者。

差分进化算法的起源与发展

- 根据“物竞天择，适者生存”的自然进化规律，建立了功能强大的智能算法，影响力比较大的主要有遗传算法、进化策略、进化规划和遗传规划。其中，遗传算法是进化计算的杰出代表，也是当今影响最为广泛的仿生智能计算方法之一
- 差分进化算法则以其突出的全局优化性能成为进化计算家族的后起之秀
- 1996年在日本召开的首届IEEE国际进化计算竞赛（International Competition on Evolutionary Optimization, ICEO）中，差分进化算法表现突出，在所有参数算法中取得第三名的优异成绩，前两名让位于非进化类算法，但这两种非进化类算法只在求解某一问题时比较优秀，不具备普遍性，而在参赛的所有进化算法中，差分进化算法被证明是最优秀的

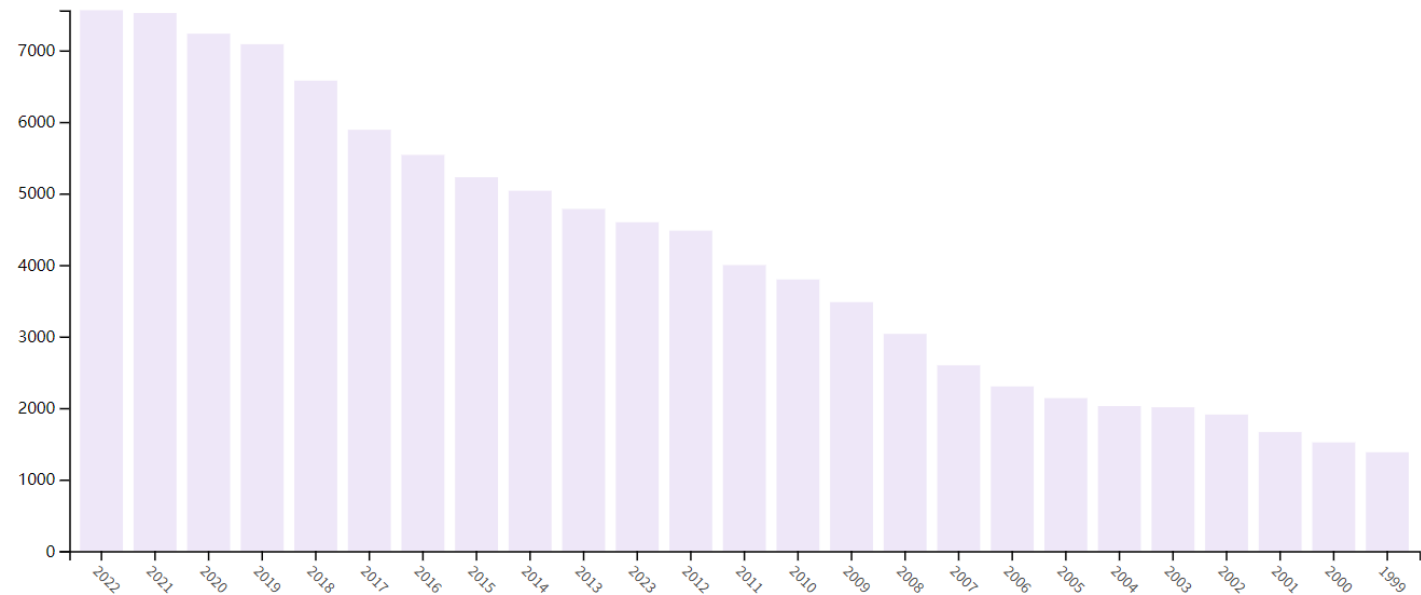
差分进化算法的起源与发展

- Vestertrom等在其论文中将差分进化算法与粒子群算法用于解决34个广泛应用的数值Benchmark问题，对两种算法的性能进行了深入的比较研究
- 实验结果表明，差分进化算法的收敛速度及稳定性要明显优于粒子群算法，同样也优于其他仿生智能计算方法

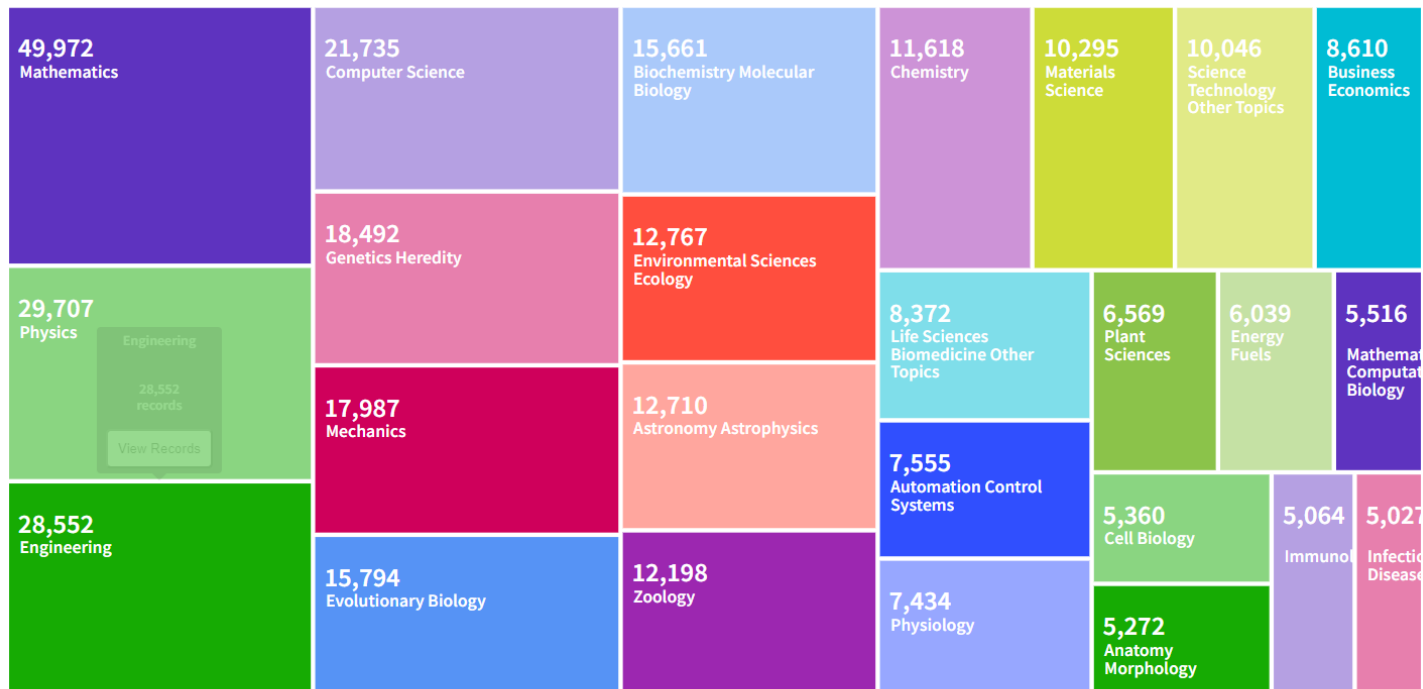
与现有的几个进化算法相比，差分进化实施起来更简单更直接…… 对其他领域的从业者来说，重要的是编程要简单，因为他们可能不是编程专家……

达斯, 苏甘坦, 科埃略 (S. Das, P. Suganthan, C. Coello Coello) [Das et al., 2011]

差分进化算法



差分进化算法



The areas on the chart are not strictly proportional to the values of each entry

➤引言

➤差分进化算法理论和特点

➤差分进化算法的基本操作

➤举例说明差分进化算法的详细工作机制

➤差分进化算法的关键参数设置

➤小结

差分进化算法的数学模型

- 作为一种基于群体进化的仿生智能计算方法
- 差分进化算法具有记忆个体最优解和种群内部信息共享的特点，即通过种群内个体间的合作与竞争来实现对优化问题的求解
- 算法本质上可看做一种基于实数编码的、具有保优思想的贪婪遗传算法

差分进化算法的数学模型

设种群规模为 N_p ，可行解空间的维数为 D ，

用 $X(t)$ 来表示进化到第 t 代时的种群。

首先，在问题的可行解空间内随机产生初始种群

$$X(0) = \{X_1^0, X_2^0, \dots, X_{N_p}^0\}$$

其中 $X_i^0 = [X_{i,1}^0, X_{i,2}^0, \dots, X_{i,D}^0]$ 用于表征第 i 个个体解

个体的各个分量可按下式产生：

$$X_{i,j}^0 = X_{j,\min} + rand(X_{j,\max} - X_{j,\min})$$

式中， $X_{j,\max}$ 和 $X_{j,\min}$ 分别为解空间第 j 维的上下界

差分进化算法的工作过程

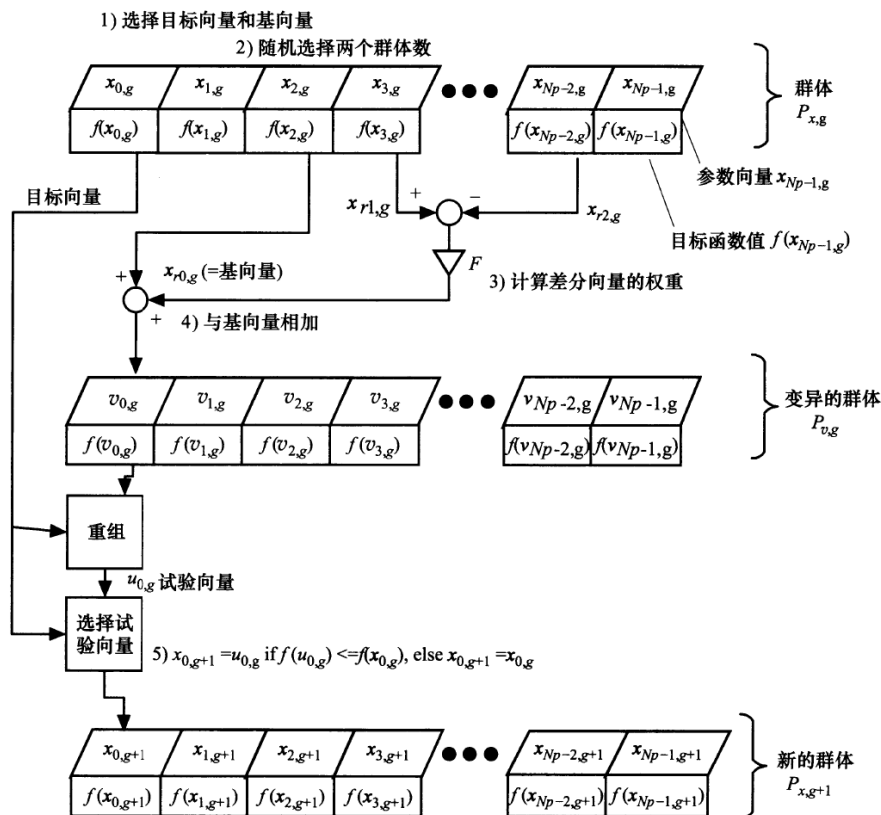
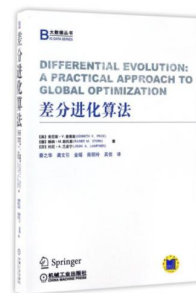


图 2.4 DE 生成和测试循环的流程图



差分进化算法的基本操作

- 每个解称为目标向量
- 差分进化算法的基本操作包括变异、交叉及选择三种操作
- 目标向量通过下面变异及重组/交叉的方式生成试验向量

目标向量 (X) $\xrightarrow{\text{变异}}$ 变异向量(V) $\xrightarrow{\text{重组 / 交叉}}$ 试验向量(U)

- 在生成所有的试验向量之后选择更好的解
- 在目标向量和试验向量之间执行贪婪选择 (greedy selection)

差分进化算法的基本操作

目标向量 (X) $\xrightarrow{\text{变异}}$ 变异向量(V) $\xrightarrow{\text{重组 / 交叉}}$ 试验向量(U)

- 在计算迭代中，算法随机选择两个不同的个体向量相减产生差分向量，然后将差分向量赋予权值后与另一随机选出的基向量相加，从而生成变异个体（变异向量）；
- 变异个体与目标个体进行参数混合交叉，得到交叉个体（试验向量）；
- 然后对试验向量与原目标向量进行一对一的选择，择优生成新一代的种群

差分进化算法的原理

- 差分进化算法的基本思想源于遗传算法，同其他进化算法一样也是对候选解的种群进行操作，而不是对一个单一解
- DE算法利用实数参数向量作为每一代的种群，它的自参考种群繁殖方案与其他进化算法不同
- DE算法给予父代所有个体以平等的机会进入下一代，不歧视劣质个体

选择“0”向量作为目标
向量（父代向量）

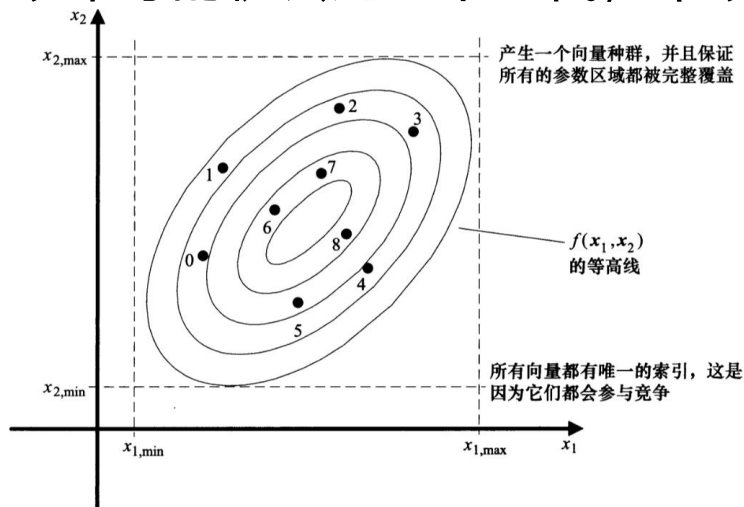


图 1.24 初始化差分进化的种群

差分进化算法的原理

- DE算法的关键思想与传统进化方法不同：传统方法是用预先确定的概率分布函数决定向量扰动；而DE算法的自组织程序利用种群中两个随机选择的不同向量来干扰一个现有向量，种群中的每个向量都要进行干扰
- DE算法利用一个向量种群，其中种群向量的随机扰动可独立进行，因此是并行的

随机选择目标向量（“0”向量）之外的两个向量（例如，选择“6”向量和“8”向量），产生差分向量

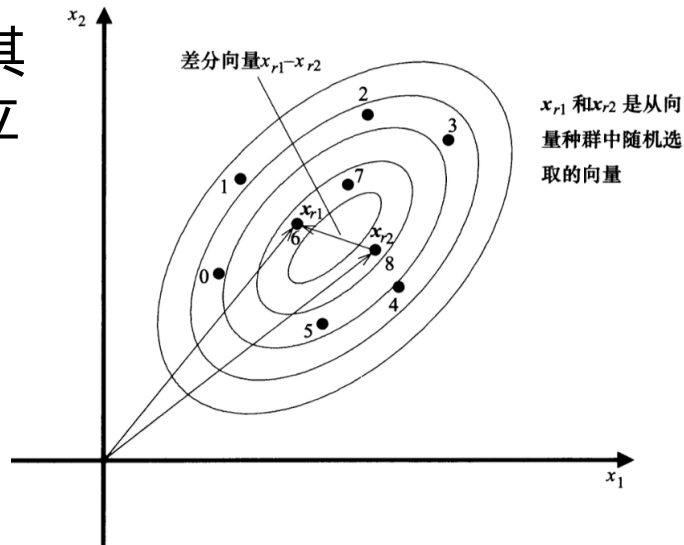


图 1.25 生成扰动： $x_{r1} - x_{r2}$

差分进化算法的原理

- DE算法是通过把种群中两个个体之间的加权差向量加到第三个个体上来产生新参数向量，这一操作称为“变异”
- 然后将变异向量的参数与另外预先决定的目标向量的参数按照一定的规则混合起来产生子个体，这一操作称为“交叉”

随机选择基向量（“4”向量）与加权差分向量生成变异向量 u_0

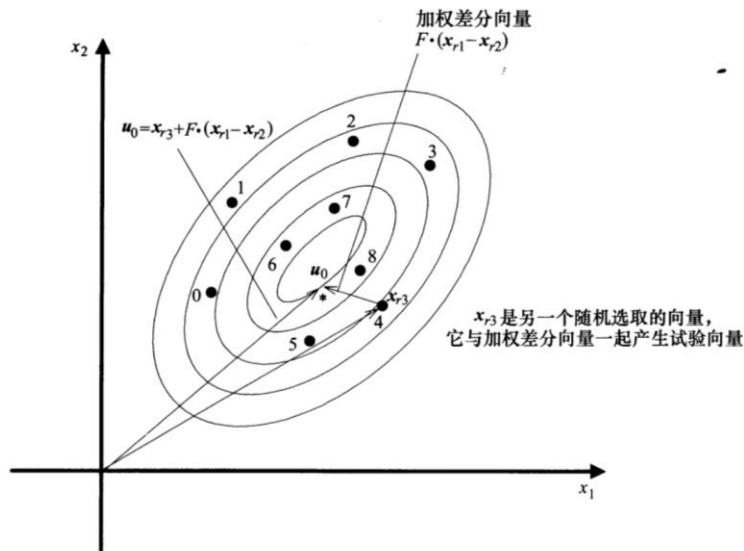


图 1.26 变异

差分进化算法的原理

- 新产生的子个体只有当它比种群中的目标个体优良时才对其进行替换，这一操作称为“选择”
- DE算法的选择操作是在完成变异、交叉之后由父代个体与新产生的候选个体——对应地进行竞争，优胜劣汰，使得子代个体总是等于或优于父代个体

目标向量（“0”向量）与试验向量 u_0 执行贪婪选择

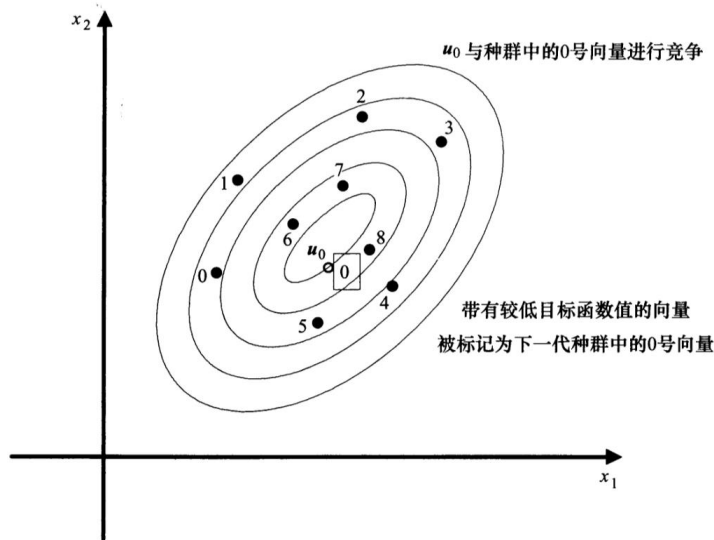


图 1.27 选择（因为 u_0 具有更低的函数值，故在下一代中使用它替换索引为 0 的向量）

差分进化算法的原理

- 差分进化是一种随机的启发式搜索算法，简单易用，有较强的稳健性和全局寻优能力。它从数学角度看是一种随机搜索算法，从工程角度看是一种自适应的迭代寻优过程
- 除了具有较好的收敛性外，差分进化算法非常易于理解与执行，它只包含不多的几个控制参数，并且在整个迭代过程中，这些参数的值可以保持不变
- 差分进化算法把一定比例的多个个体的差分信息作为个体的扰动量，使得算法在跳跃距离和搜索方向上具有自适应性

差分进化算法的原理

- 在进化的早期，因为种群中个体的差异性较大，使得扰动量较大，从而使得算法能够在较大范围内搜索，具有较强的探索能力
- 到了进化的后期，当算法趋向于收敛时，种群中个体的差异较小，算法在个体附近搜索，这使得算法具有较强的局部开发能力
- 正是由于差分进化算法具有向种群个体学习的能力，使得其拥有其他进化算法无法比拟的性能

差分进化算法的特点

1. 结构简单，容易使用。差分进化算法主要通过差分变异算子来进行遗传操作，由于该算子只涉及向量的加减运算，因此很容易实现；该算法采用概率转移规则，不需要确定性的规则。此外，差分进化算法的控制参数少，这些参数对算法性能的影响已经得到一定的研究，并得出了一些指导性的建议，因而可以方便使用人员根据问题选择较优的参数设置
2. 性能优越。差分进化算法具有较好的可靠性、高效性和稳健性，对于大空间、非线性和不可求导的连续问题，其求解效率比其他进化方法好，而且很多学者还在对差分进化算法继续改良，以不断提高其性能

差分进化算法的特点

3. 自适应性。差分进化算法的差分变异算子可以是固定常数，也可以具有变步步长和搜索方向自适应的能力，根据不同目标函数进行自动调整，从而提高搜索质量
4. 差分进化算法具有内在的并行性，可协同搜索，具有利用个体局部信息和群体全局信息指导算法进一步搜索的能力。同样精度要求下，差分进化算法具有更快的收敛速度
5. 算法通用，可直接对结构对象进行操作，不依赖于问题信息，不存在对目标函数的限定。差分进化算法操作十分简单，易于编程实现，尤其利于求解高维的函数优化问题

➤引言

➤差分进化算法理论和特点

➤差分进化算法的基本操作

➤举例说明差分进化算法的详细工作机制

➤差分进化算法的关键参数设置

➤小结

差分进化算法的操作

1. 初始化
2. 变异
3. 重组 (均匀交叉, 指数交叉)
4. 选择
5. 边界条件的处理

初始化

差分进化算法利用 N_p 个维数为 D 的实数值参数向量，将它们作为每一代的种群

每个个体表示为： $x_{i,G}(i = 1, 2, \dots, N_p)$

式中： i 表示个体在种群中的序列； G 表示进化代数；

N_p 表示种群规模，在进化优化过程中 N_p 保持不变

初始化

在差分进化算法中，一般假定所有随机初始化种群均符合均匀概率分布。

设参数变量的界限为 $x_j(L) < x_j < x_j(U)$ ，则

$$x_{ji,0} = rand[0, 1] \cdot (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$$

$$(i = 1, 2, \dots, N_p; \quad j = 1, 2, \dots, D)$$

式中 $rand[0, 1]$ 表示在 $[0, 1]$ 之间产生的均匀随机数

如果可以预先得到问题的初步解，则初始种群也可以通过初步解加入正态分布随机偏差来产生，这样可以提高优化效果

差分进化算法的操作

1. 初始化
2. 变异
3. 重组 (均匀交叉, 指数交叉)
4. 选择
5. 边界条件的处理

基于变异的差分向量

- * 变异操作是差分进化算法与遗传算法最主要的区别
- * 在差分进化算法中，变异个体的生成过程中用到了父代种群中多个个体的线性组合
- * 最基本的变异成分是父代个体的差分向量

基于变异的差分向量

对于父代群中任意的一个目标向量 X_i 而言，差分进化算法按下面公式生成变异向量

目标向量(X_i)的变异向量 (V_i) 由下式产生

$$V_i = X_{r_1} + F(X_{r_2} - X_{r_3})$$

F 缩放因子，通常是处于0到2之间的常数

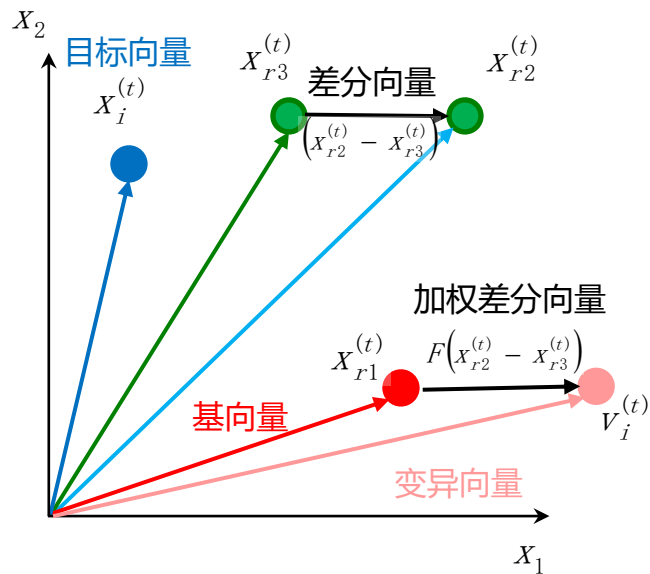
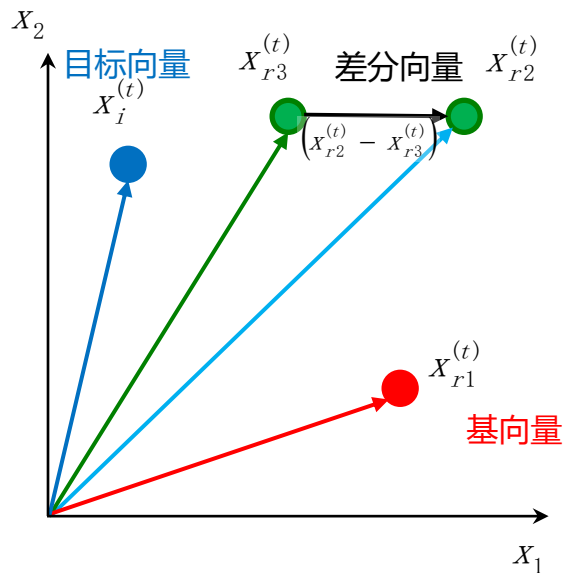
$r_1, r_2, r_3 \in \{1, 2, 3, \dots, N_p\}$ 并且 $r_1 \neq r_2 \neq r_3 \neq i$

X_{r_1} 为基向量

目标向量不参与变异操作

变异操作中总共涉及4个向量，因此 $N_p \geq 4$

变异的图形表示



差分进化的变异策略 (DE/X/Y/Z)

- DE: Differential Evolution 差分进化算法以线性差分策略为主要特征
- Price和Storn先后提出了10余种不同的策略来生成差分向量以实现算法的变异操作, 可用符号DE/X/Y/Z来区分
- X: 表示基向量选择, 为rand、best或current, 分别表示在种群中随机 (rand) 选择个体 x_r , 选择种群当前最优 (best) 个体 x_{best} 或者当前 (current) 目标个体 x_c
- Y: 表示变异操作中需要使用的差分向量个数
- Z: 交叉操作的概率分布, 当Z为bin时表示概率分布满足二项分布形式, 当Z为exp时表示满足指数分布形式

变异策略

DE/rand/1/bin

变异向量的表达式

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$

最少种群数量 N_p

4

差分进化的变异策略 (DE/x/y/z)

以二项式分布交叉 (*bin*) 为例，差分进化算法的7种扩展模式如下：

变异策略	变异向量的表达式	最少种群数量 N_p
<i>DE/rand/1/bin</i>	$V = X_{r_1} + F(X_{r_2} - X_{r_3})$	4
<i>DE/rand/2/bin</i>	$V = X_{r_1} + \lambda(X_{r_2} - X_{r_3}) + F(X_{r_4} - X_{r_5})$	6
<i>DE/best/1/bin</i>	$V = X_{best} + F(X_{r_1} - X_{r_2})$	3
<i>DE/best/2/bin</i>	$V = X_{best} + \lambda(X_{r_1} - X_{r_2}) + F(X_{r_3} - X_{r_4})$	5
<i>DE/rand - to - best/2/bin</i>	$V = X_{r_1} + \lambda(X_{best} - X_{r_2}) + F(X_{r_3} - X_{r_4})$	5
<i>DE/current - to - rand/2/bin</i>	$V = X_i + \lambda(X_{r_1} - X_i) + F(X_{r_2} - X_{r_3})$	4
<i>DE/current - to - best/2/bin</i>	$V = X_i + \lambda(X_{best} - X_i) + F(X_{r_2} - X_{r_3})$	3

- DE/rand/1/bin和DE/current-to-best/2/bin简写为DE1和DE2，是目前使用最广、应用最为成功的差分策略。
- DE1有利于保持种群的多样性，DE2则强调算法的收敛速度。

差分进化算法的操作

1. 初始化
2. 变异
3. 重组 (均匀交叉, 指数交叉)
4. 选择
5. 边界条件的处理

重组：二项式（均匀）交叉

差分进化算法重组 / 交叉操作的目的是通过变异向量 V_i 和目标向量 X_i 各维分量的随机重组以提高种群个体的多样性。

试验向量可以通过以下方式产生

$$u^j = \begin{cases} v^j & \text{如果 } r \leq p_c \text{ 或者 } j = \delta \\ x^j & \text{如果 } r > p_c \text{ 并且 } j \neq \delta \end{cases}$$

p_c 交叉概率

δ 随机选择的向量中变量的位置 $\delta \in \{1, 2, 3, \dots, D\}$

r 在0和1之间的随机数

u^j 试验向量的第 j 个变量

v^j 变异向量的第 j 个变量

x^j 目标向量的第 j 个变量

重组：二项式（均匀）交叉

- * δ 的作用是保证至少一个变量来自于变异向量
- * 交叉的概率 p_c 通常比较大，高交叉概率 p_c 意味着更多的变量来自于变异向量
- * 突变等位基因的遗传数量应该遵循二项式分布，其原因在于：等位基因的源是由数量有限且具有恒定概率结果的独立实验所确定的

均匀交叉的示例

目标向量 X

$x^1 = 12$
 $x^2 = 43$
 $x^3 = 52$
 $x^4 = 87$
 $x^5 = 96$
 $x^6 = 32$
 $x^7 = 64$
 $x^8 = 29$

变异向量 V

$v^1 = 23$
 $v^2 = 15$
 $v^3 = 49$
 $v^4 = 71$
 $v^5 = 67$
 $v^6 = 51$
 $v^7 = 35$
 $v^8 = 99$

试验向量可以通过以下方式产生

$$u^j = \begin{cases} v^j & \text{如果 } r \leq p_c \text{ 或者 } j = \delta \\ x^j & \text{如果 } r > p_c \text{ 并且 } j \neq \delta \end{cases}$$

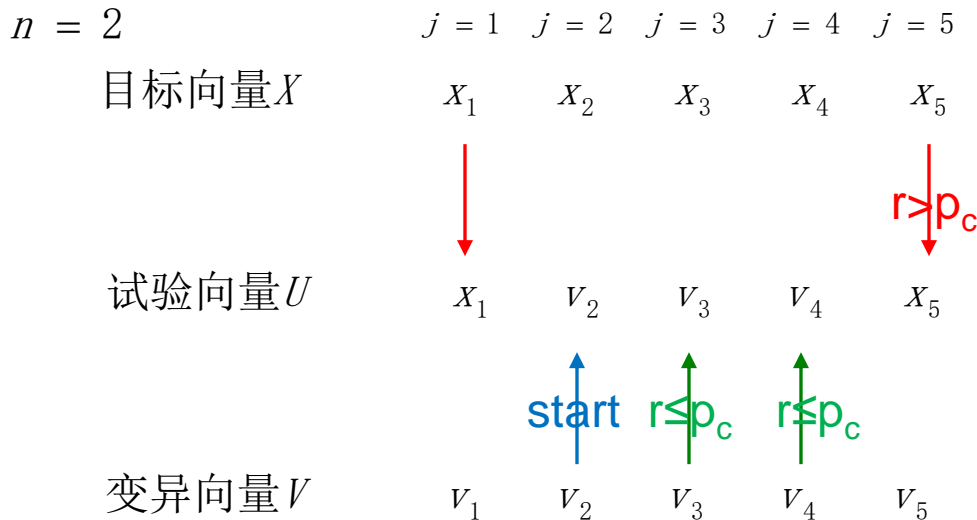
$$\delta = 4, p_c = 0.8$$

试验向量 U

$r = 0.50$	$r < p_c \ \& \ j \neq \delta$	$\rightarrow v^1 = 23$	u^1
$r = 0.91$	$r > p_c \ \& \ j \neq \delta$	$\rightarrow x^2 = 43$	u^2
$r = 0.79$	$r < p_c \ \& \ j \neq \delta$	$\rightarrow v^3 = 49$	u^3
$r = 0.90$	$r > p_c \ \& \ j = \delta$	$\rightarrow v^4 = 71$	u^4
$r = 0.25$	$r < p_c \ \& \ j \neq \delta$	$\rightarrow v^5 = 67$	u^5
$r = 0.85$	$r > p_c \ \& \ j \neq \delta$	$\rightarrow x^6 = 32$	u^6
$r = 0.41$	$r < p_c \ \& \ j \neq \delta$	$\rightarrow v^7 = 35$	u^7
$r = 0.98$	$r > p_c \ \& \ j \neq \delta$	$\rightarrow x^8 = 29$	u^8

指数交叉

- 在 $1 \sim D$ 区间中随机选择整数 n
- 把变异向量 V 中的第 n 个变量拷贝到试验向量 U 中的第 n 个变量
- 在随后的变量操作中，产生 $0 \sim 1$ 的随机数，直到 $r > p_c$
- 如果 $r \leq p_c$ ，从变异向量 V 中将该变量拷贝到试验向量 U
- 如果 $r > p_c$ ，将目标向量 X 中剩余的变量拷贝到试验向量 U



指数交叉的示例1

$$n = 3, \quad p_c = 0.8$$

$j = 1$ $j = 2$ $j = 3$ $j = 4$ $j = 5$ $j = 6$ $j = 7$ $j = 8$

目标向量 X 2 8 7 9 6 3 5 4

试验向量 U u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8

变异向量 V 1 2 4 8 7 0 9 2

指数交叉的示例1

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	u_4	u_5	u_6	u_7	u_8
			↑ start					
变异向量 V	1	2	4	8	7	0	9	2

指数交叉的示例1

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	u_4	u_5	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

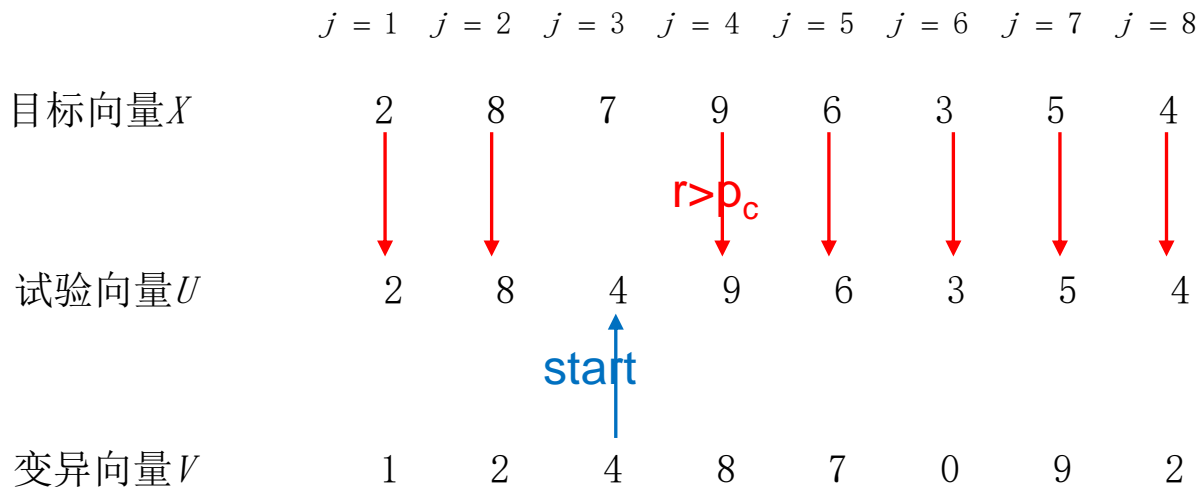
start



$$r = 0.86$$

指数交叉的示例1

$$n = 3, \quad p_c = 0.8$$



$$r = 0.86$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

$j = 1$ $j = 2$ $j = 3$ $j = 4$ $j = 5$ $j = 6$ $j = 7$ $j = 8$

目标向量 X 2 8 7 9 6 3 5 4

试验向量 U u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8

变异向量 V 1 2 4 8 7 0 9 2

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	u_4	u_5	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

↑
start

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	u_4	u_5	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

↑
start

$$r = 0.7$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	u_5	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

Diagram illustrating the exponential crossover process:

- A blue arrow labeled "start" points from the value 4 in the V row (at $j=3$) to the value 4 in the U row (at $j=3$).
- A green arrow labeled " $r \leq p_c$ " points from the value 8 in the V row (at $j=4$) to the value 8 in the U row (at $j=4$).

$$r = 0.7$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	u_5	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

Diagram illustrating the exponential crossover process:

- A blue arrow labeled "start" points from the value 4 in the V row (at $j=3$) to the value 4 in the U row (at $j=3$).
- A green arrow labeled " $r \leq p_c$ " points from the value 8 in the V row (at $j=4$) to the value 8 in the U row (at $j=4$).

$$r = 0.3$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	7	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

Diagram illustrating the exponential crossover process:

- A blue arrow labeled "start" points from $V_3 = 4$ to $U_3 = 4$.
- Two green arrows labeled " $r \leq p_c$ " point from $V_4 = 8$ to $U_4 = 8$ and from $V_5 = 7$ to $U_5 = 7$.

$$r = 0.3$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	7	u_6	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

Diagram illustrating the exponential crossover process:

- A blue arrow labeled "start" points from $V_3 = 4$ to $U_3 = 4$.
- A green arrow labeled " $r \leq p_c$ " points from $V_4 = 8$ to $U_4 = 8$.
- A green arrow labeled " $r \leq p_c$ " points from $V_5 = 7$ to $U_5 = 7$.

$$r = 0.4$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	7	0	u_7	u_8
			start	$r \leq p_c$	$r \leq p_c$	$r \leq p_c$		
变异向量 V	1	2	4	8	7	0	9	2

$$r = 0.4$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	7	0	u_7	u_8
			start	$r \leq p_c$	$r \leq p_c$	$r \leq p_c$		
变异向量 V	1	2	4	8	7	0	9	2

$$r = 0.5$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	7	0	9	u_8
			start	$r \leq p_c$	$r \leq p_c$	$r \leq p_c$	$r \leq p_c$	
变异向量 V	1	2	4	8	7	0	9	2

$$r = 0.5$$

指数交叉的示例2

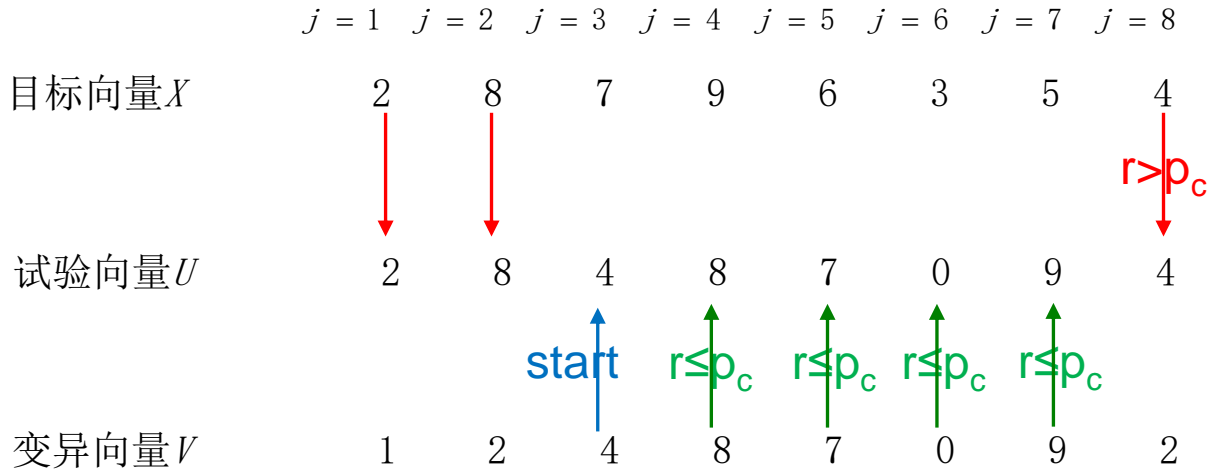
$$n = 3, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	4	8	7	0	9	u_8
			start	$r \leq p_c$	$r \leq p_c$	$r \leq p_c$	$r \leq p_c$	
变异向量 V	1	2	4	8	7	0	9	2

$$r = 0.9$$

指数交叉的示例2

$$n = 3, \quad p_c = 0.8$$



$$r = 0.9$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

$j = 1$ $j = 2$ $j = 3$ $j = 4$ $j = 5$ $j = 6$ $j = 7$ $j = 8$


目标向量 X 2 8 7 9 6 3 5 4

试验向量 U u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8

变异向量 V 1 2 4 8 7 0 9 2

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	u_3	u_4	u_5	0	u_7	u_8
								
变异向量 V	1	2	4	8	7	0	9	2

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	u_3	u_4	u_5	0	u_7	u_8
变异向量 V	1	2	4	8	7	0	9	2

↑
start

$$r = 0.6$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	u_3	u_4	u_5	0	9	u_8
变异向量 V	1	2	4	8	7	0	9	2

start $r \leq p_c$

$$r = 0.6$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	u_3	u_4	u_5	0	9	u_8
变异向量 V	1	2	4	8	7	0	9	2

start $r \leq p_c$

$$r = 0.2$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	u_3	u_4	u_5	0	9	2
						start	$r \leq p_c$	$r \leq p_c$
变异向量 V	1	2	4	8	7	0	9	2

$$r = 0.2$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	u_1	u_2	u_3	u_4	u_5	0	9	2
						start	$r \leq p_c$	$r \leq p_c$
变异向量 V	1	2	4	8	7	0	9	2

$$r = 0.7$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	1	u_2	u_3	u_4	u_5	0	9	2
变异向量 V	1	2	4	8	7	0	9	2

$r \leq p_c$

start

$r \leq p_c$

$r \leq p_c$

$$r = 0.7$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	1	u_2	u_3	u_4	u_5	0	9	2
变异向量 V	1	2	4	8	7	0	9	2

$r \leq p_c$

start

$r \leq p_c$

$r \leq p_c$

$$r = 0.9$$

指数交叉的示例3

$$n = 6, \quad p_c = 0.8$$

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
目标向量 X	2	8	7	9	6	3	5	4
试验向量 U	1	8	7	9	6	0	9	2
变异向量 V	1	2	4	8	7	0	9	2

Diagram illustrating the exponential crossover process for $n = 6$ and $p_c = 0.8$. The process involves three rows: Target Vector X , Trial Vector U , and Mutation Vector V , indexed by j from 1 to 8.

Arrows indicate the crossover operation:

- Red arrows (downward) from X to U for $j = 2, 3, 4, 5$ are labeled $r > p_c$.
- Green arrows (upward) from V to U for $j = 1, 7, 8$ are labeled $r \leq p_c$.
- A blue arrow (upward) from V to U for $j = 6$ is labeled "start".

$$r = 0.9$$

差分进化算法的操作

1. 初始化
2. 变异
3. 重组 (均匀交叉, 指数交叉)
4. 选择
5. 边界条件的处理

选择

- 选择：决定试验向量 U 是否能够进入种群
- 计算所有试验向量 U 的目标函数值 f_U
- 一种方式是目标向量 X_i 和当前试验向量 U_i 进行比较，进行贪婪选择，更新种群

$$\left. \begin{array}{l} X_i = U_i \\ f_i = f_{U_i} \end{array} \right\} \text{如果 } f_{U_i} < f_i$$

X 和 f 保持不变如果 $f_{U_i} > f_i$

- 另一种方式是生成所有的试验向量之后，在所有目标向量 X 和所有试验向量 U 中进行贪婪选择操作（greedy selection），选择最好的 N_p 个向量更新种群

差分进化算法的操作

1. 初始化
2. 变异
3. 重组 (均匀交叉, 指数交叉)
4. 选择
5. 边界条件的处理

边界条件的处理

在有边界约束的问题中，必须保证产生新个体的参数值位于问题的可行域中

1.一个简单方法是将不符合边界约束的新个体在可行域随机产生的参数向量代替，即：若 $u_{ji,G+1} < x_j^{(L)}$ 或 $u_{ji,G+1} > x_j^{(U)}$ ，那么

$$u_{ji,G+1} = rand[0, 1] \cdot (x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$$

$(i = 1, 2, \dots, N_p; \quad j = 1, 2, \dots, D)$

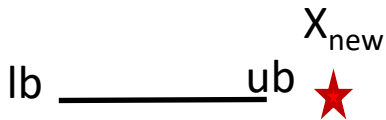
2.另外一个方法是进行边界吸收处理，即将超过边界约束的个体值设置为临近的边界值

保证新解(试验向量 u)在取值范围内



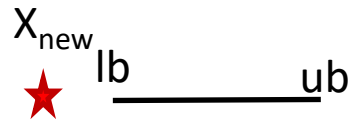
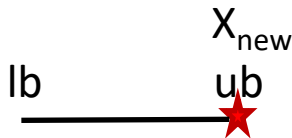
x_{new} 在取值范围内

不需要修改任何值



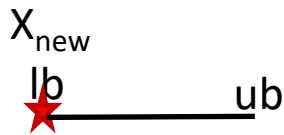
x_{new} 违反取值范围的上限

将 x_{new} 移到取值范围的上限

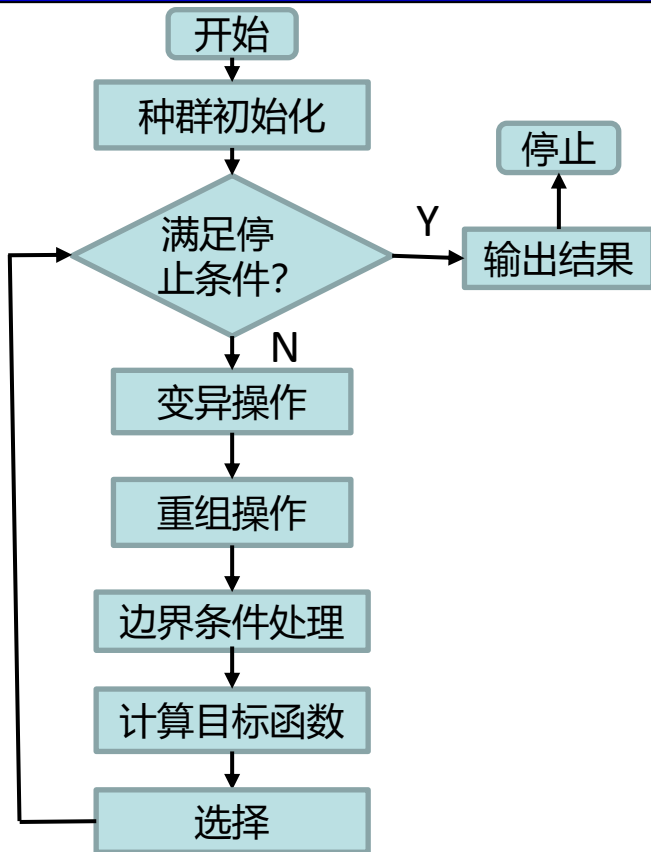


x_{new} 违反取值范围的下限

将 x_{new} 移到取值范围的下限



差分进化算法流程



差分进化算法采用实数编码、基于差分的简单变异操作和“一对一”的竞争生存策略，具体步骤如下：

Step1.确定差分进化算法的控制参数和所要采用的具体策略。差分进化算法的控制参数包括：种群数量、变异算子、交叉算子、最大进化代数、终止条件等

Step2.随机产生初始种群，进化代数 $k = 1$

Step3.对初始种群进行评价，即计算初始种群中每个个体的目标函数值

Step4.判断是否达到终止条件或达到最大进化代数：若是，则进化终止，将此时的最佳个体作为解输出；否则，继续下一步操作

Step5.进行变异操作和交叉操作，对边界条件进行处理，得到临时种群

Step6.对临时种群进行评价，计算临时种群中每个个体的目标函数值

Step7.对临时种群中的个体和原始种群中对应的个体，进行“一对一”的选择操作，得到新种群

Step8.进化代数 $k = k + 1$ ，转到Step4

差分进化算法的工作过程

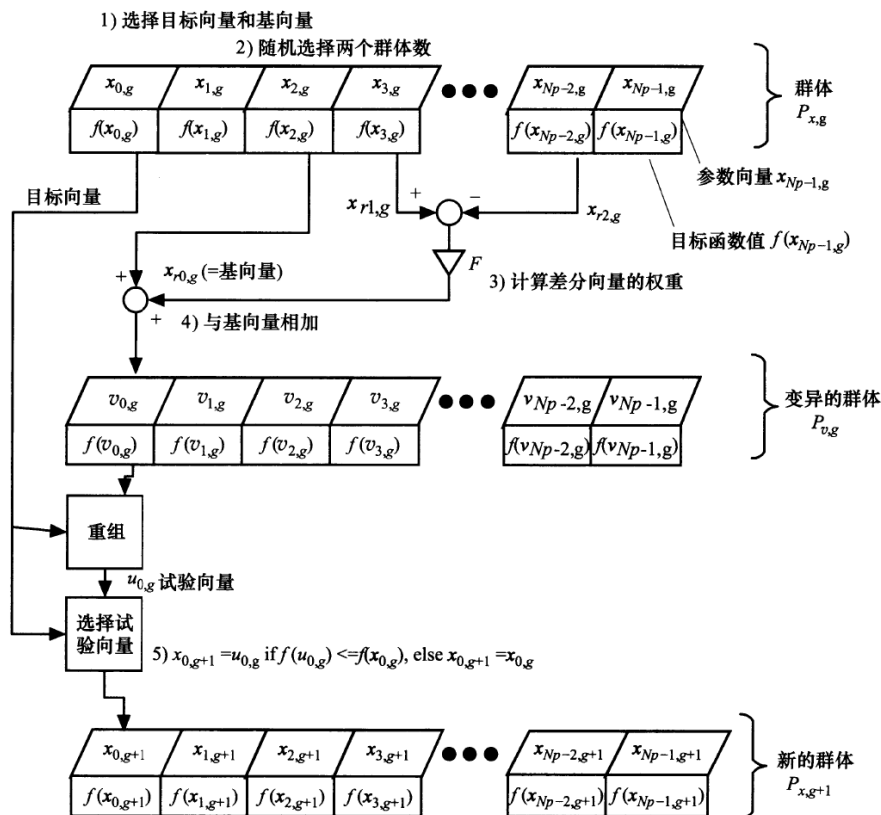
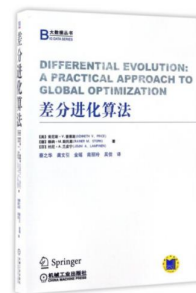


图 2.4 DE 生成和测试循环的流程图



可视化DE：差分向量分布

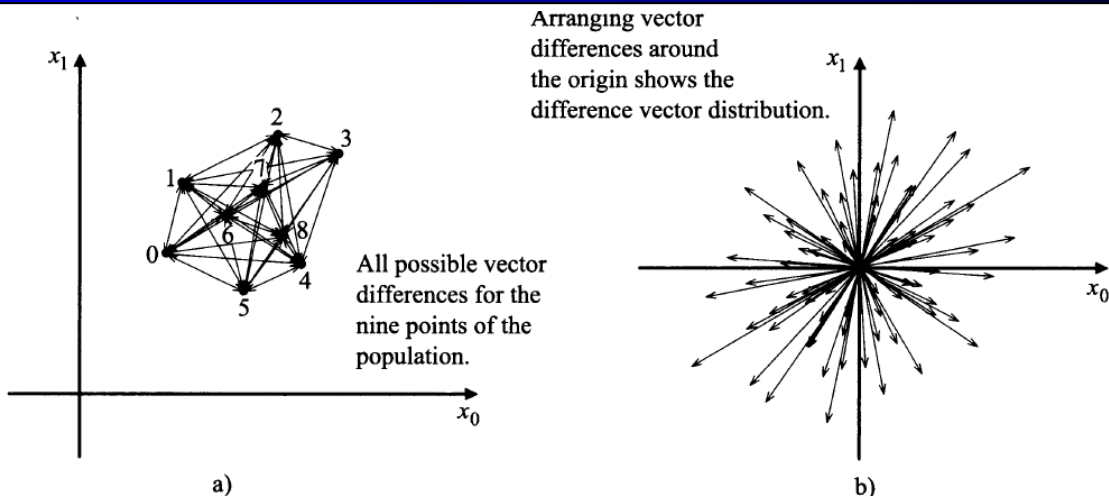


图 2.5 9 个向量（图 a）和它们相应的分布（图 b）

a) 差分向量 b) 差分向量的分布

- 图2.5a显示了9个向量所有可能两两组合形成的差分向量
- 如果将差分向量放到一个共同的起点就能够更清晰地显示出它们的分布（图2.5b）
- 因为所有的差分向量都是一个与其相逆的对应向量以及一个相等的被选择机会，所以它们分布的平均值为零

可视化DE：差分向量分布

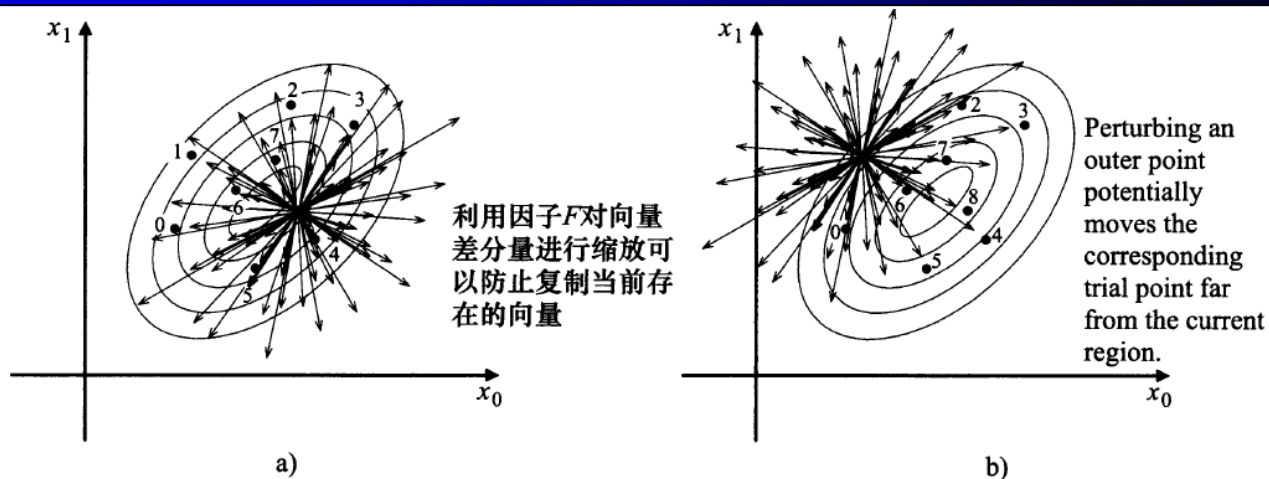


图 2.6 缩放的效果（图 a）和较大向量差分量的效果（图 b）

a) 缩放的效果 b) 较大向量差分量的效果

- 对向量差分分量进行缩放可以保证试验向量不会复制当前存在的点
- 缩放操作还能够在局部搜索和全局搜索之间变换搜索的重点
- 图2.6b显示了在包含向量数目较多的差分向量分布中，较长的差分向量将会降低整个种群陷入局部最优情况的可能性

可视化DE：搜索过程

$$f(x_1, x_2) = 3(1 - x_1)^2 \cdot \exp(x_1^2 + (x_2 + 1)^2) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \cdot \exp(x_1^2 + x_2^2) - \frac{1}{3} \cdot \exp((x_1 + 1)^2 + x_2^2)。$$

(1.16)

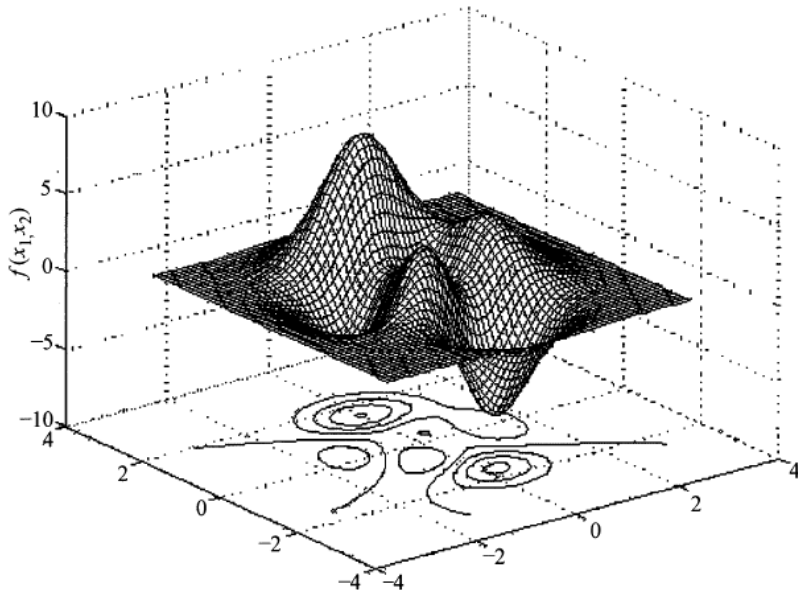


图 1.13 式 (1.16) 定义的“峰”函数是多峰函数

可视化DE：搜索过程

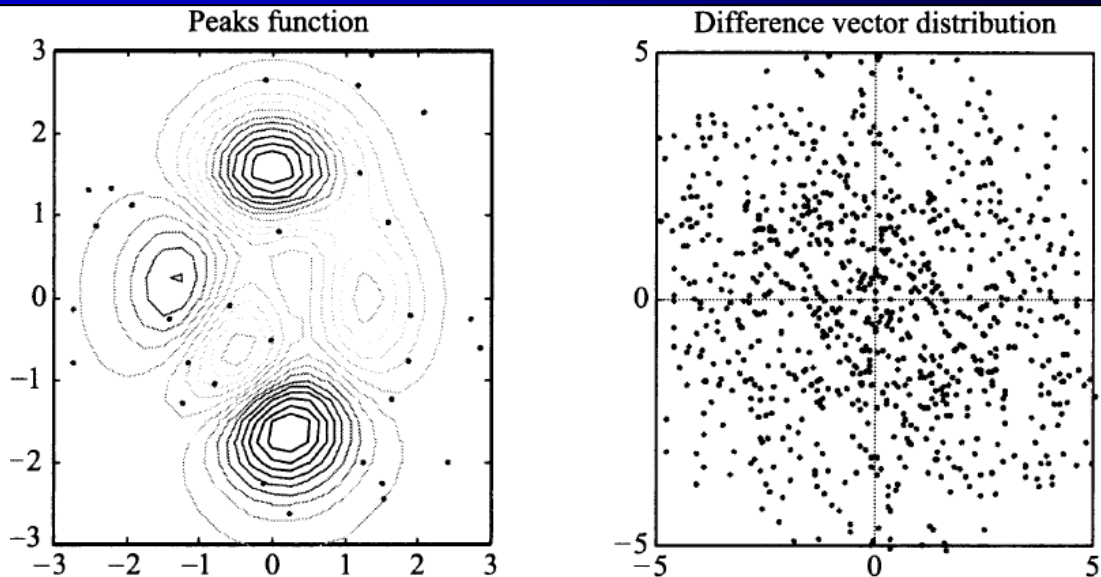


图 2.7 第 1 代：DE 种群和差分向量的分布

- 差分向量所提供的最大优势之一是进化步长大小和搜索方向自动适应了目标函数的地形
- 为了表现出更加清晰直观的效果，差分向量分布图只显示了差分向量的末端

可视化DE：搜索过程

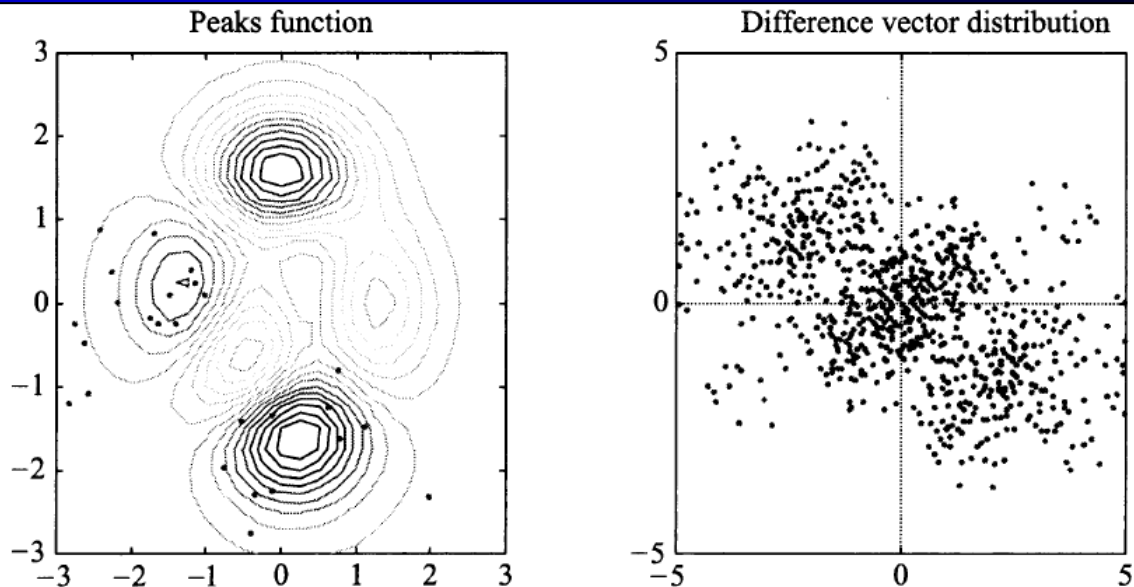


图 2.8 第 6 代：种群在两个主要最小值周围的结合

- 随着进化的进行，种群将在最小值周围聚集
- 在这个阶段，差分分布如同函数本身一样是多峰的。它不仅包含适应在盆地内搜索的步长，也包含了能够在盆地和外部之间传送向量的更大的步长

可视化DE：搜索过程

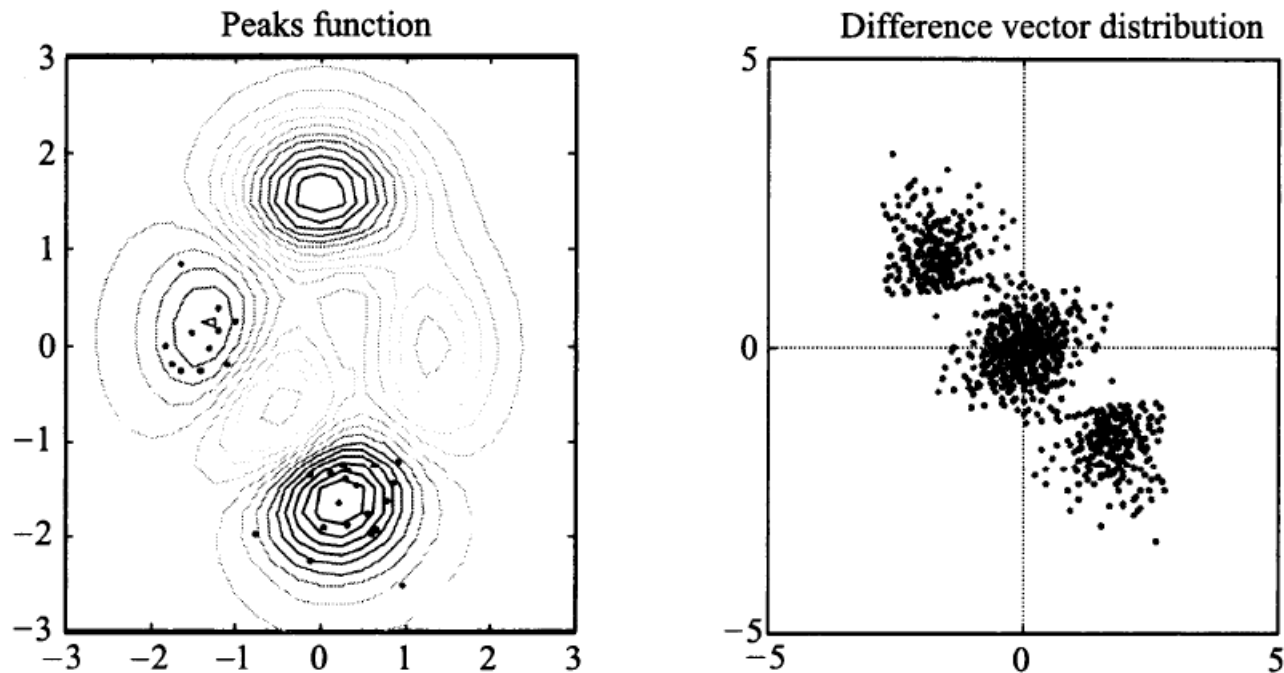


图 2.9 第 12 代：差向量分布包含三个主要的云团（一个是局部搜索，另两个是在两个主要最小值间的传输）

可视化DE：搜索过程

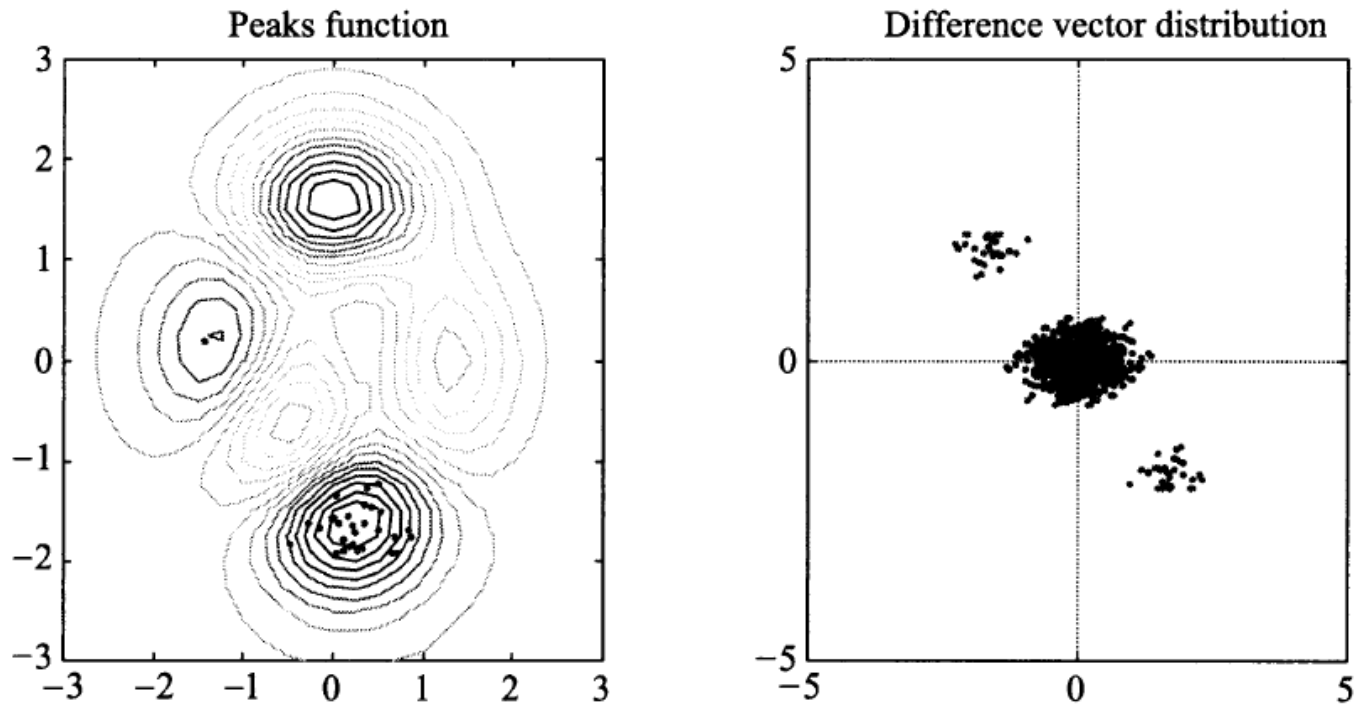


图 2.10 第 16 代：种群已经集中于主要的最小值

可视化DE：搜索过程

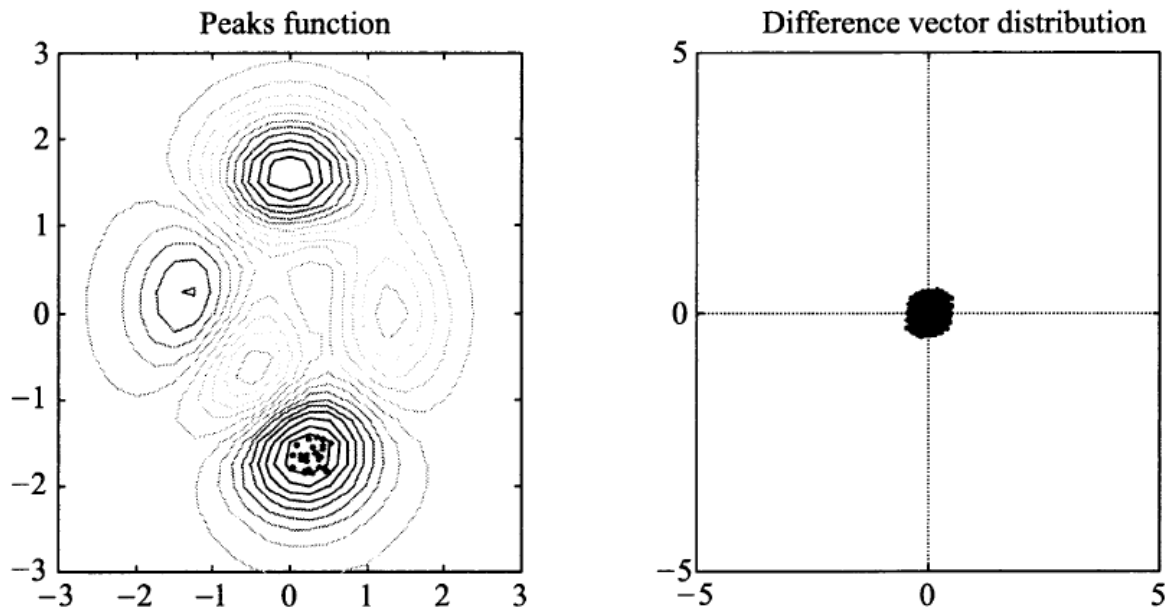


图 2.11 第 20 代：即将收敛（差向量为了细致的局部搜索而自动变短）

- 一旦种群进入最优盆地，那么差分向量的分布就变成了单峰的形式，同时步长也表现出了适合局部搜索的规模和方向

可视化DE：搜索过程

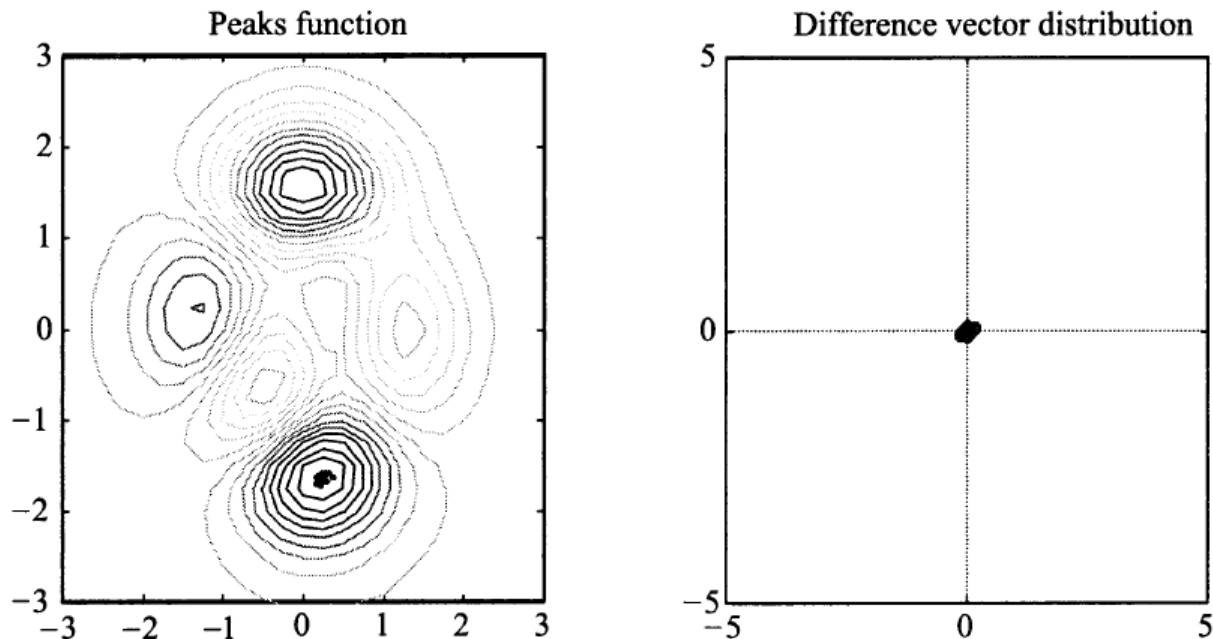


图 2.12 第 26 代：种群基本收敛

可视化DE：搜索过程

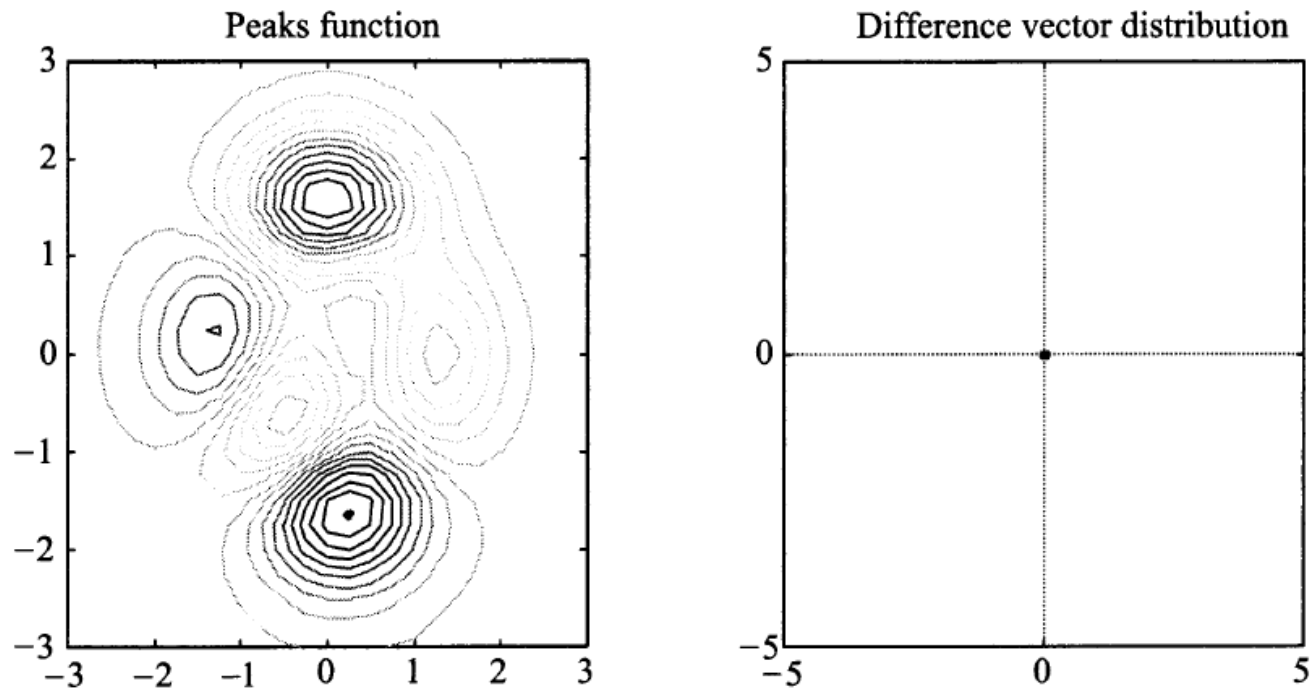


图 2.13 第 34 代：DE 找到全局最优解

差分进化算法的伪代码

Input : Fitness function, lb, ub, N_p , T, F, p_c

1. Initialize a random population (P)

2. Evaluate fitness(f) of P

for $t = 1$ to T

for $i = 1$ to N_p

Generate the donor vector(V_i) using mutation

Perform crossover to generate offspring(U)

end

for $i = 1$ to N_p

Bound U_i

Evaluate the fitness(f_{U_i}) of U_i

Perform greedy selection using f_{U_i} and f_i to update P

end

end

目标向量(X_i)的合成向量 (V) 由下式产生

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$

试验向量 U 可以通过以下方式产生

$$u^j = \begin{cases} v^j & \text{如果 } r \leq p_c \text{ 或者 } j = \delta \\ x^j & \text{如果 } r > p_c \text{ 并且 } j \neq \delta \end{cases}$$

$$\begin{aligned} X_i &= U_i \\ f_i &= f_{U_i} \end{aligned} \left\{ \begin{array}{l} \text{如果 } f_{U_i} < f_i \\ \text{X和f保持不变如果 } f_{U_i} > f_i \end{array} \right.$$

差分进化算法的伪代码

算法 12.1 最小化 n 元函数 $f(\mathbf{x})$ 的简单差分进化算法. 此算法被称为经典差分进化, 或 DE/rand/1/bin.

F = 步长参数, $F \in [0.4, 0.9]$

c = 交叉率, $c \in [0.1, 1]$

初始化候选解种群 $\{\mathbf{x}_i\}$, $i \in [1, N]$

While not (终止准则)

For 每一个个体 \mathbf{x}_i , $i \in [1, N]$

$r_1 \leftarrow$ 随机整数 $\in [1, N] : r_1 \neq i$

$r_2 \leftarrow$ 随机整数 $\in [1, N] : r_2 \notin \{i, r_1\}$

$r_3 \leftarrow$ 随机整数 $\in [1, N] : r_3 \notin \{i, r_1, r_2\}$

$\mathbf{v}_i \leftarrow \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$ (变异向量)

$\mathcal{J}_r \leftarrow$ 随机整数, $\mathcal{J}_r \in [1, n]$

For 每一维 $j \in [1, n]$

$r_{cj} \leftarrow$ 随机数, $r_{cj} \in [0, 1]$

If $(r_{cj} < c)$ or $(j = \mathcal{J}_r)$ then

$u_{ij} \leftarrow v_{ij}$

else

$u_{ij} \leftarrow x_{ij}$

End if

下一维

下一个个体

For 每一个种群指标 $i \in [1, N]$

If $f(\mathbf{u}_i) < f(\mathbf{x}_i)$ then $\mathbf{x}_i \leftarrow \mathbf{u}_i$ End if

下一个种群指标

下一代



-
- 引言
 - 差分进化算法理论和特点
 - 差分进化算法的基本操作
 - 举例说明差分进化算法的详细工作机制
 - 差分进化算法的关键参数设置
 - 小结

示例：Sphere function

$$\min \quad f(x) = \sum_{i=1}^4 x_i^2 \quad 0 \leq x_i \leq 10 \quad i = 1, 2, 3, 4$$

决策变量： x_1, x_2, x_3, x_4 $f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2$

第一步：确定种群规模, 迭代次数, 交叉概率, 缩放系数

$$N_p = 5 \quad T = 10 \quad p_c = 0.8 \quad F = 0.85$$

示例：Sphere function

第二步：在决策变量的取值范围内生成随机解

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 1 & 2 & 8 & 3 \end{bmatrix}$$

$$f = \begin{bmatrix} 81 \\ 140 \\ 35 \\ 102 \\ 78 \end{bmatrix}$$

差分进化的工作机制：第一个解

第三步：在1和 N_p 之间生成3个随机整数

选取第一个解为目标向量， $r_1 \neq r_2 \neq r_3 \neq 1$

假设 $r_1 = 4$ $r_2 = 2$ $r_3 = 3$

$$p_c = 0.8$$

$$F = 0.85$$

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 1 & 2 & 8 & 3 \end{bmatrix}$$

$$f = \begin{bmatrix} 81 \\ 140 \\ 35 \\ 102 \\ 78 \end{bmatrix}$$

目标向量(X_i)的变异向量 (V) 由下式产生

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$

差分进化的工作机制：第一个解

第四步：确定变异向量 V

$$\begin{aligned} V_1 &= X_4 + F(X_2 - X_3) = \begin{bmatrix} 2 & 1 & 4 & 9 \end{bmatrix} + 0.85 \times (\begin{bmatrix} 3 & 1 & 9 & 7 \end{bmatrix} - \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix}) \\ &= \begin{bmatrix} 2 & 1 & 4 & 9 \end{bmatrix} + \begin{bmatrix} 2.55 & -1.7 & 6.8 & 1.7 \end{bmatrix} = \begin{bmatrix} 4.55 & -0.7 & 10.8 & 10.7 \end{bmatrix} \end{aligned}$$

$$p_c = 0.8 \quad F = 0.85$$

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 1 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 81 \\ 140 \\ 35 \\ 102 \\ 78 \end{bmatrix}$$

目标向量(X_i)的变异向量 (V) 由下式产生

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$

差分进化的工作机制：第一个解

第四步：确定变异向量 V

$$\begin{aligned} V_1 &= X_4 + F(X_2 - X_3) = \begin{bmatrix} 2 & 1 & 4 & 9 \end{bmatrix} + 0.85 \times (\begin{bmatrix} 3 & 1 & 9 & 7 \end{bmatrix} - \begin{bmatrix} 0 & 3 & 1 & 5 \end{bmatrix}) \\ &= \begin{bmatrix} 2 & 1 & 4 & 9 \end{bmatrix} + \begin{bmatrix} 2.55 & -1.7 & 6.8 & 1.7 \end{bmatrix} = \begin{bmatrix} 4.55 & -0.7 & 10.8 & 10.7 \end{bmatrix} \end{aligned}$$

第五步：生成 D 个随机数

假设 $r = \begin{bmatrix} 0.3 & 0.9 & 0.2 & 0.6 \end{bmatrix}$

在1和 D 之间生成随机数 $\delta = 1$

$$p_c = 0.8 \quad F = 0.85$$

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 1 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 81 \\ 140 \\ 35 \\ 102 \\ 78 \end{bmatrix}$$

目标向量(X_i)的变异向量 (V) 由下式产生

$$V = X_{r_1} + F(X_{r_2} - X_{r_3})$$

差分进化的工作机制：第一个解

第六步：确定试验向量 U

j	目标向量 X	变异向量 V	r	$r < P_c$	$\delta \neq j$	$\delta = j$	试验向量 U
1	4	4.55	0.3	√	×	√	4.55
2	0	-0.7	0.9	×	√	×	0
3	1	10.8	0.2	√	√	×	10.8
4	8	10.7	0.6	√	√	×	10.7

第七步：确定试验向量 U 是否满足变量范围要求

$$U_1 = [4.55 \quad 0 \quad 10.8 \quad 10.7]$$

$$\Rightarrow U_1 = [4.55 \quad 0 \quad 10 \quad 10]$$

$$p_c = 0.8 \quad F = 0.85 \quad \delta = 1$$

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 1 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 81 \\ 140 \\ 35 \\ 102 \\ 78 \end{bmatrix}$$

试验向量可以通过以下方式产生

$$u^j = \begin{cases} v^j & \text{如果 } r \leq p_c \text{ 或者 } j = \delta \\ x^j & \text{如果 } r > p_c \text{ 并且 } j \neq \delta \end{cases}$$

$$0 \leq x_i \leq 10$$

$$\begin{aligned} x &= lb & \text{如果} & \quad x < lb \\ x &= ub & \text{如果} & \quad x > ub \end{aligned}$$

差分进化的工作机制：第一次迭代

第一次迭代的 r_1, r_2, r_3, r, δ

目标向量(X_i)的变异向量 (V) 由下式产生
 $V = X_{r_1} + F(X_{r_2} - X_{r_3})$

$$p_c = 0.8 \qquad F = 0.85$$

i	目标向量 X	r ₁	r ₂	r ₃	变异向量 V	r	δ	试验向量 U
1	[4 0 1 8]	4	2	3	[4.55 -0.7 10.8 10.7]	[0.3 0.9 0.2 0.6]	1	[4.55 0 10 10]
2	[3 1 9 7]	5	1	3	[4.4 -0.55 8 5.55]	[0.3 0.2 0.6 0.4]	4	[4.4 0 8 5.55]
3	[0 3 1 5]	4	2	1	[1.15 1.85 10.8 8.15]	[0.2 0.5 0.4 0.3]	4	[1.15 1.85 10 8.15]
4	[2 1 4 9]	5	3	2	[-1.55 3.7 1.2 1.3]	[0.8 0.3 0.6 0.2]	1	[0 3.7 1.2 1.3]
5	[1 2 8 3]	2	4	1	[1.3 1.85 11.55 7.85]	[0.7 0.5 0.9 0.2]	3	[1.3 1.85 8 7.85]

第八步：确定试验向量 U 的适应度函数值

$$U = \begin{bmatrix} 4.55 & 0 & 10 & 10 \\ 4.4 & 0 & 8 & 5.55 \\ 1.15 & 1.85 & 10 & 8.15 \\ 0 & 3.7 & 1.2 & 1.3 \\ 1.3 & 1.85 & 8 & 7.85 \end{bmatrix} \qquad f_U = \begin{bmatrix} 220.70 \\ 114.16 \\ 171.17 \\ 16.82 \\ 130.73 \end{bmatrix}$$

试验向量可以通过以下方式产生

$$u^j = \begin{cases} v^j & \text{如果 } r \leq p_c \text{ 或者 } j = \delta \\ x^j & \text{如果 } r > p_c \text{ 并且 } j \neq \delta \end{cases}$$

$$f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2$$

差分进化的工作机制：第一次迭代的贪婪选择

第九步：每次迭代，执行贪婪选择方法并更新种群

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 3 & 1 & 9 & 7 \\ 0 & 3 & 1 & 5 \\ 2 & 1 & 4 & 9 \\ 1 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 81 \\ 140 \\ 35 \\ 102 \\ 78 \end{bmatrix} \quad U = \begin{bmatrix} 4.55 & 0 & 10 & 10 \\ 4.4 & 0 & 8 & 5.55 \\ 1.15 & 1.85 & 10 & 8.15 \\ 0 & 3.7 & 1.2 & 1.3 \\ 1.3 & 1.85 & 8 & 7.85 \end{bmatrix} \quad f_U = \begin{bmatrix} 220.70 \\ 114.16 \\ 171.17 \\ 16.82 \\ 130.73 \end{bmatrix}$$

下次迭代的种群为

$$P = \begin{bmatrix} 4 & 0 & 1 & 8 \\ 4.4 & 0 & 8 & 5.55 \\ 0 & 3 & 1 & 5 \\ 0 & 3.7 & 1.2 & 1.3 \\ 1 & 2 & 8 & 3 \end{bmatrix} \quad f = \begin{bmatrix} 81 \\ 114.16 \\ 35 \\ 16.82 \\ 78 \end{bmatrix}$$

-
- 引言
 - 差分进化算法理论和特点
 - 差分进化算法的基本操作
 - 举例说明差分进化算法的详细工作机制
 - 差分进化算法的关键参数设置
 - 小结

差分进化算法的关键参数设置

➤ 控制参数对一个全局优化算法的影响很大，差分进化算法的控制变量选择也有一些经验规则

1. 种群数量 N_p ：一般情况下，种群的规模 N_p 越大，其中的个体就越多，种群的多样性就越好，寻优的能力也就越强，但也因此增加了计算的难度。所以， N_p 不能无限取大。根据经验，种群数量 N_p 的合理选择为 $5D \sim 10D$ ，必须满足 $N_p \geq 4$ ，以确保差分进化算法具有足够的不同的变异向量

差分进化算法的关键参数设置

2. 变异算子 F : 变异算子 $F \in [0, 2]$ 是一个实常数因数, 它决定偏差向量的放大比例

- 变异算子 F 过小, 则可能造成算法“早熟”。随着 F 值的增大, 防止算法陷入局部最优的能力增强, 但当 $F > 1$ 时, 想要算法快速收敛到最优值会变得十分不易; 这是由于当差分向量的扰动大于两个个体之间的距离时, 种群的收敛性会变得很差
- 目前的研究表明, F 小于0.4和大于1的值仅偶尔有效, $F=0.5$ 通常是一个较好的初始选择。若种群过早收敛, 那么 F 或 N_p 应该增大

差分进化算法的关键参数设置

3. 交叉算子CR: 交叉算子CR是一个范围在 $[0, 1]$ 内的实数, 它控制着一个试验向量参数来自于随机选择的变异向量而不是原来向量的概率。交叉算子CR越大, 发生交叉的可能性就越大。较大的CR通常会加速收敛, 为了看看是否可能获得一个快速解, 可以先尝试 $CR = 0.9$ 或 $CR = 1.0$
4. 最大进化代数G: 表示差分进化算法运行结束条件的一个参数, 表示差分进化算法运行到指定的进化代数之后就停止运行, 并将当前群体中的最佳个体作为所求问题的最优解输出。一般, G的取值范围为100~500

差分进化算法的关键参数设置

5. 终止条件：除最大进化代数可作为差分进化算法的终止条件外，还可以增加其他判定准则。一般当目标函数值小于阈值时程序终止，阈值常选 10^{-6}

上述参数中， F ， CR 与 N_p 一样，在搜索过程中是常数，一般 F 和 CR 影响搜索过程的收敛速度和稳健性，它们的优化值不仅依赖于目标函数的特性，还与 N_p 有关。通常可通过对不同值做一些试验之后，利用试验和结果误差找到 F 、 CR 和 N_p 的合适值。

➤引言

➤差分进化算法理论和特点

➤差分进化算法的基本操作

➤举例说明差分进化算法的详细工作机制

➤差分进化算法的关键参数设置

➤小结

差分进化算法

- 差分进化算法保留了基于种群的全局搜索策略，采用实数编码、基于差分的简单变异操作和“一对一”的竞争生存策略，降低了进化计算操作的复杂性
- 差分进化算法特有的记忆能力使其可以动态跟踪当前的搜索情况，以调整其搜索策略，具有较强的全局收敛能力和稳健性，且不需要借助问题的特征信息，适用于求解一些利用常规数学规划方法很难求解的复杂优化问题

差分进化算法

- 差分进化算法作为一种高效的并行搜索算法，对其进行理论和应用研究具有重要的学术意义和工程价值
- 差分进化算法已经成功应用于系统工程、控制工程、统计学求解等领域的众多优化问题
- 随着差分进化算法在众多领域的成功应用，事实证明：DE不仅能更快、更稳定地收敛到问题的全局最优解，而以其具有易理解、易执行、鲁棒性强等特点，表明它比其他算法更加有效。DE算法已逐渐成为智能优化界中的一类充满朝气蓬勃发展的优化方法

差分进化算法的研究方向

➤ 差分进化算法是一种新兴的算法，已有的研究和应用成果都证明了其有效性和广阔的发展前景，但由于人们对其研究刚刚开始，远没有像遗传算法那样已经具有良好的理论基础、系统的分析方法和广泛的应用基础，因此还有待进一步的研究。

1. 算法理论方法目的研究。差分进化算法虽然在实际应用中被证明是有效的，但是对算法的收敛性、收敛速度、参数选取等方面还缺乏理论分析，还需要进一步的数学证明

差分进化算法的研究方向

2. 参数选择和优化。种群数量、变异算子、交叉算子等参数选择对差分进化的性能有重要影响，如何选择、优化和调整参数，使算法既能避免早熟又能较快收敛，对研究和应用有着重要的意义
3. 算法的改进。由于实际问题的多样性和复杂性，单靠一种算法的机制是不能满足需要的。因此，应注重高效的差分进化算法的开发，在分析算法优劣的基础上，改进状态更新公式，并有效地均衡全局和局部搜索，如何将其他算法和差分进化算法结合，构造出更加高效的混合差分进化算法仍是当前算法改进的热点

差分进化算法的研究方向

4. 算法的应用研究。基本差分进化算法主要适用于连续空间的函数优化问题，如何改变其搜索及变异机制使其用于离散空间的优化，特别是组合优化问题，将是差分进化算法研究的热点之一。此外，差分进化算法在多目标、噪声环境、动态等复杂优化问题中的应用还有待进一步拓宽