Lecture 2: Evolutionary Search Operators

CSE5012: Evolutionary Computation and Its Applications

Xin Yao

CSE, SUSTech

20 Feb 2023

Summary of the Previous Lecture

- **Why Natural Computation?**
- **What is Evolutionary Computation?**
- **Different Types of Evolutionary Algorithms**
- **Major Areas in Evolutionary Computation**

Recall: Main Steps of Evolutionary Algorithms

1. **Initialise the population at random**
2. **REPEAT**
   - 2.1 **Evaluate fitness of individuals in the population (μ is the population size)**
   - 2.2 **Compute the selection probability for each individual**
   - 2.3 **REPEAT**
     - 2.3.1 **Select two individuals as parents according to the probabilities in Step 2.2**
     - 2.3.2 **Crossover the two individuals in Step 2.3.1 with a crossover rate**
       /* After this step, we have two individuals */
     - 2.3.3 **Mutate the two individuals in Step 2.3.2 with a mutation rate**

     UNTIL we have obtained μ new individuals
   - 2.4 **Use the μ new individuals to replace the previous population**

   UNTIL stopping criteria are met

   /* Output the best individual in the population */

In this lecture, we will focus on the crossover and mutation operators.

Outline

Outline of This Lecture

Recombination/Crossover

- **Pick up two parents to generate two offspring.**
- **Crossover rate: The probability of applying crossover.**
- **Aim to keep and recombine good building blocks of the parents.**
  (But how do we know which part is good?)

Illustrative example of Lecture 1:

Parent 1	**1**	**1**	**0**	**0**	**0**
Parent 2	**0**	**1**	**1**	**0**	**1**
Offspring 1	**1**	**1**	**0**	**0**	**1**
Offspring 2	**0**	**1**	**1**	**0**	**0**

Recombination/Crossover Operators

- **Recombination for discrete representation**
  - **One-point crossover / k-point crossover ($k > 1$)**
  - **Uniform crossover**
  - ...

- **Recombination for real-valued representation (mainly 2 categories):**
  - **Discrete recombination does not change actual (gene) values.**
    Very similar to the crossover operators on binary strings.,
  - **Intermediate recombination does change actual (gene) values.**
    Usually based on some kind of average/mixture among multiple parents.
  - ...

One-point Crossover

Example: choose a random crossover point at 3

Parent 1	0	1	0	1	0	0	1	0	0	0
Parent 2	0	1	1	1	0	1	1	0	1	1
Offspring 1	0	1	0	1	0	1	1	0	1	1
Offspring 2	0	1	1	1	0	0	1	0	0	0

k-point Crossover ($k > 1$)

Example: $k = 2$, choose two random crossover points at 3 and 6

Parent 1	0	1	0	1	0	0	1	0	0	0
Parent 2	0	1	1	1	0	1	1	0	1	1
Offspring 1	0	1	0	1	0	1	1	0	0	0
Offspring 2	0	1	1	1	0	0	1	0	1	1

Example: $k = 3$, choose three random crossover points at 3, 6, 8

Parent 1	0	1	0	1	0	0	1	0	0	0
Parent 2	0	1	1	1	0	1	1	0	1	1
Offspring 1	0	1	0	1	0	1	1	0	1	1
Offspring 2	0	1	1	1	0	0	1	0	0	0

Uniform Crossover

Example: at every point, generate a random number $\in [0, 1]$ with crossover rate 0.5

Gene Index	1	2	3	4	5	6	7	8	9	10
Random	0.31	0.63	0.58	0.07	0.29	0.42	0.59	0.63	0.13	0.73

Parent 1	0	1	0	1	0	0	1	0	0	0
Parent 2	0	1	1	1	0	1	1	0	1	1
Offspring 1	0	1	0	1	0	1	1	0	1	0
Offspring 2	0	1	1	1	0	0	1	0	0	1

Multi-point Discrete Recombination

A Type of Discrete Recombination

- **Similar to that for the binary representation.**
- ***[Example] two parents with a random crossover point at 1:***

Parent 1	**1.0**	**2.4**	**0.8**	**7.6**
Parent 2	**0.2**	**1.1**	**1.6**	**3.4**
Offspring 1	**1.0**	**1.1**	**1.6**	**3.4**
Offspring 2	**0.2**	**2.4**	**0.8**	**7.6**

Global Discrete Recombination

A Type of Discrete Recombination

- **Similar to uniform crossover for the binary representation.**
- **On each dimension, randomly select a parent from the population and take its gene of the dimension.**
- **Selection on each dimension is independent.**

Parent 1	1.0	2.4	0.8	7.6
Parent 2	0.2	1.1	1.6	3.4
Offspring	0.2	2.4	0.8	3.4

Intermediate Recombination

- **Discrete Recombination does not change actual (gene) values.**

$\rightarrow$ Create new values using intermediate recombination (usually based on some kind of average/mixture among multiple parents).

Arithmetic Recombination

A Commonly Used Intermediate Recombination Operator

- **Given**
  - d **is the individual/chromosome length,**
  - **two parents** $\mathbf{x}_1$ **and** $\mathbf{x}_2$**, and**
  - **a weight** $\alpha \in [0, 1]$**:**

$$Offspring\ 1 : x'_i = \alpha x_{1i} + (1 - \alpha)x_{2i}, \quad i \in \{1, \ldots, d\}$$

$$Offspring\ 2 : x''_i = \alpha x_{2i} + (1 - \alpha)x_{1i}, \quad i \in \{1, \ldots, d\}$$

- **If** $\alpha = 0.5$**, the two offspring/children are identical.**

Arithmetic Recombination

- ***[Example] two parents with*** $\alpha = 0.4$

Parent 1	**1.0**	**2.4**	**0.8**	**7.6**
Parent 2	**0.2**	**1.1**	**1.6**	**3.4**

$$0.4 * 1.0 + (1 - 0.4) * 0.2 = 0.52, \qquad 0.4 * 0.2 + (1 - 0.4) * 1.0 = 0.68$$
$$0.4 * 2.4 + (1 - 0.4) * 1.1 = 1.62, \qquad 0.4 * 1.1 + (1 - 0.4) * 2.4 = 1.88$$
$$0.4 * 0.8 + (1 - 0.4) * 1.6 = 1.28, \qquad 0.4 * 1.6 + (1 - 0.4) * 0.8 = 1.12$$
$$0.4 * 7.6 + (1 - 0.4) * 3.4 = 5.08, \qquad 0.4 * 3.4 + (1 - 0.4) * 7.6 = 5.92$$

Offspring 1	**0.52**	**1.62**	**1.28**	**5.08**
Offspring 2	**0.68**	**1.88**	**1.12**	**5.92**

Simple Arithmetic Recombination

Combining Arithmetic Recombination with One-Point Recombination

- **Given**
  - **two parents $\mathbf{x}_1$ and $\mathbf{x}_2$,**
  - **a weight $\alpha \in [0, 1]$, and**
  - **a randomly selected point k:**

$$
\begin{aligned}
Offspring\ 1: \quad & x'_i = x_{1i}, \quad i \in \{1, \ldots, k\} \\
& x'_i = \alpha x_{1i} + (1 - \alpha) x_{2i}, \quad i \in \{k + 1, \ldots, d\} \\
Offspring\ 2: \quad & x''_i = x_{2i}, \quad i \in \{1, \ldots, k\} \\
& x''_i = \alpha x_{2i} + (1 - \alpha) x_{1i}, \quad i \in \{k + 1, \ldots, d\}
\end{aligned}
$$

- **If $\alpha = 0.5$, the part after the k^{th} gene of the two offspring are identical.**

Simple Arithmetic Recombination

- ***[Example 1] two parents with $\alpha = 0.4$ and a random point 1***

Parent 1	1.0	2.4	0.8	7.6
Parent 2	0.2	1.1	1.6	3.4
Offspring 1	1.0	1.62	1.28	5.08
Offspring 2	0.2	1.88	1.12	5.92

- ***[Example 2] two parents with $\alpha = 0.5$ and a random point 2***

Parent 1	1.0	2.4	0.8	7.6
Parent 2	0.2	1.1	1.6	3.4
Offspring 1	1.0	2.4	1.2	5.5
Offspring 2	0.2	1.1	1.2	5.5

Single Arithmetic Recombination

- **Given**
  - **two parents $\mathbf{x}_1$ and $\mathbf{x}_2$,**
  - **a weight $\alpha \in [0, 1]$, and**
  - **a randomly selected point k:**

$$
\begin{aligned}
Offspring\ 1: \quad & x'_i = x_{1i}, \quad i \in \{1, \ldots, d\} \textbf{ and } i \neq k \\
& x'_k = \alpha x_{1k} + (1 - \alpha) x_{2k} \\
Offspring\ 2: \quad & x''_i = x_{2i}, \quad i \in \{1, \ldots, d\} \textbf{ and } i \neq k \\
& x''_k = \alpha x_{2k} + (1 - \alpha) x_{1k}
\end{aligned}
$$

Single Arithmetic Recombination

- **Given**
  - **two parents $\mathbf{x}_1$ and $\mathbf{x}_2$,**
  - **a weight $\alpha \in [0, 1]$, and**
  - **a randomly selected point k:**

$$
\begin{aligned}
Offspring\ 1: \quad & x'_i = x_{1i}, \quad i \in \{1, \ldots, d\} \textbf{ and } i \neq k \\
& x'_k = \alpha x_{1k} + (1 - \alpha) x_{2k} \\
Offspring\ 2: \quad & x''_i = x_{2i}, \quad i \in \{1, \ldots, d\} \textbf{ and } i \neq k \\
& x''_k = \alpha x_{2k} + (1 - \alpha) x_{1k}
\end{aligned}
$$

- ***[Example] two parents with $\alpha = 0.4$ and a random point 1***

Parent 1	**1.0**	**2.4**	**0.8**	**7.6**
Parent 2	**0.2**	**1.1**	**1.6**	**3.4**
Offspring 1	**1.0**	**1.62**	**0.8**	**7.6**
Offspring 2	**0.2**	**1.88**	**1.6**	**3.4**

Other Recombination Operators for Real-valued Representation I

Heuristic Recombination **Assume $\mathbf{x_2}$ is no worse than $\mathbf{x_1}$.**

$$\mathbf{x}' = \alpha(\mathbf{x_2} - \mathbf{x_1}) + \mathbf{x_2},$$

where α is a uniformly distributed random number $\in [0, 1]$.

Simplex Recombination **Randomly select a group (> 2) of parents. Assume $\mathbf{x_b}$ is the best individual and $\mathbf{x_w}$ is the worst in the group. Compute the centroid, $\mathbf{c}$, of the group without $\mathbf{x_w}$. Let the following $\mathbf{x}'$ replace $\mathbf{x_w}$.**

$$\mathbf{x}' = \mathbf{c} + (\mathbf{x_b} - \mathbf{x_w}).$$

Other Recombination Operators for Real-valued Representation II

Geometric Recombination **Can be generalised to multiple parents.**

$$\mathbf{x}' = (\sqrt{x_{11}x_{21}}, \sqrt{x_{12}x_{22}}, ...)$$

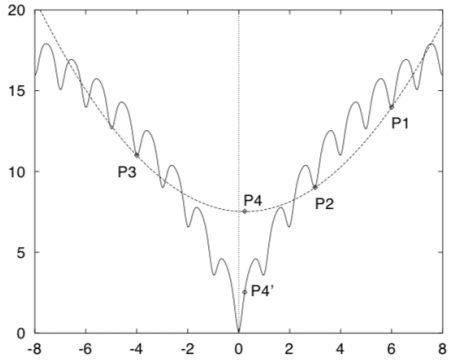
Quadratic Recombination **Let $x_{i,j}$ be the j-th component of the vectors $\mathbf{x}_i$, $\forall i \in \{1, 2, 3\}$, $j \in \{1, \cdots, d\}$, where d is the dimensionality. We approximate the position of P_4 using the quadratic interpolation method as follow:**

$$x_{4,j} = \frac{1}{2} \cdot \frac{(x_{2,j}^2 - x_{3,j}^2)f(\mathbf{x}_1) + (x_{3,j}^2 - x_{1,j}^2)f(\mathbf{x}_2) + (x_{1,j}^2 - x_{2,j}^2)f(\mathbf{x}_3)}{(x_{2,j} - x_{3,j})f(\mathbf{x}_1) + (x_{3,j} - x_{1,j})f(\mathbf{x}_2) + (x_{1,j} - x_{2,j})f(\mathbf{x}_3)}.$$

What Does Quadratic Recombination Mean?

$$x_{4,j} = \frac{1}{2} \cdot \frac{(x_{2,j}^2 - x_{3,j}^2) f(\mathbf{x}_1) + (x_{3,j}^2 - x_{1,j}^2) f(\mathbf{x}_2) + (x_{1,j}^2 - x_{2,j}^2) f(\mathbf{x}_3)}{(x_{2,j} - x_{3,j}) f(\mathbf{x}_1) + (x_{3,j} - x_{1,j}) f(\mathbf{x}_2) + (x_{1,j} - x_{2,j}) f(\mathbf{x}_3)}.$$

Note that we are minimising "fitness" here.

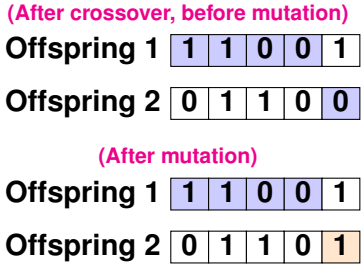


Outline of This Lecture

Mutation

- **Change values of gene(s) at random.**
- **Mutation rate: Note the *difference* between per bit (gene) and per chromosome (individual) mutation rates.**
- **Be careful with the randomised part when implementing.**

Illustrative example of Lecture 1:



Mutation Operators

- **For discrete representation:**
  - **Bit-flipping**
  - **Random bit assignment**
  - **Swap mutation**
  - **Inverse mutation**
  - ...

- **For real-valued representation (mainly two categories):**
  - **Uniform mutation**
  - **Nonuniform mutation**

Bit-flipping

- **One-bit flipping / One-bit mutation:**
  Flip one of the bits uniformly at random, e.g.,

Gene index	1	2	3	4	5	6	7	8	9	10
Random index	3									
Before mutation	0	1	0	1	0	1	1	0	1	1
After mutation	0	1	1	1	0	1	1	0	1	1

- **Multi-bit flipping:**
  Select multiple bits and flip their values, e.g.,

Gene index	1	2	3	4	5	6	7	8	9	10
Random indices	3,	6,	7	(e.g. 3-bit flipping)						
Before mutation	0	1	0	1	0	1	1	0	1	1
After mutation	0	1	1	1	0	0	0	0	1	1

Bitwise mutation

- **At every point, generate a random number $\in [0, 1]$ with mutation rate p (usually $p = 1/d$)**

Gene	1	2	3	4	5	6	7	8	9	10
Random	0.03	0.31	0.07	0.58	0.79	0.52	0.19	0.13	0.93	0.23

Before mutation	0	1	0	1	0	1	1	0	1	1
After mutation	1	1	1	1	0	1	1	0	1	1

Random mutation

- **Extension of bit-flipping:**
  binary representation $\rightarrow$ integer representation.

- **Example:**
  - **An integer representation with values $\in \{0, 1, 2, 3, 4\}$**
  - **At every point, generate a random number $\in [0, 1]$ with mutation rate 0.1, mutate to another possible value uniformly at random.**

Gene index	1	2	3	4	5	6	7	8	9	10
Random for selecting gene	0.03	0.31	0.07	0.58	0.79	0.52	0.19	0.13	0.93	0.23
Random for selecting value	0.21	-	0.72	-	-	-	-	-	-	-

Before mutation	0	1	3	2	0	1	2	2	4	1
After mutation	1	1	2	2	0	1	2	2	4	1

[Exercises]

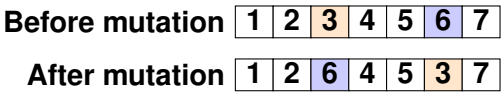
1. **Search by yourself to find out what is random bit assignment?**

2. **Implement one-bit flipping and bitwise mutation, then compare them using the objective function of Lab 1.**

Swap mutation

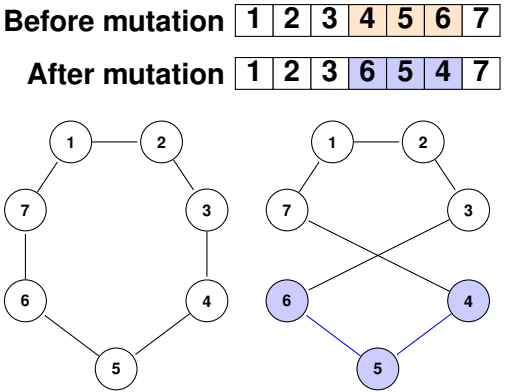
- **Swap the values of two genes selected uniformly at random.**

- **Example:**



Inversion mutation

- **Invert the order of a subset of string.**
- **Example:**



Uniform Mutation

- **Similar to uniform mutation for the binary representation:**

$$x'_i = UniformRandom(LowerBound_i, UpperBound_i),\ \forall i \in \{1, \ldots, d\}$$

- **Potential problem: possibility of diverging too much from an already good solution.**
- ***[Example] One parent with mutation rate*** 0.1 ***and random numbers generated as follows:***

Gene	1	2	3	4	
Interval	$[0.0, 1.0]$	$[1.0, 4.0]$	$[0.5, 2.3]$	$[2.0, 30]$	
Random	0.02	0.63	0.58	0.07	*// mutate if random value* < 0.1
Random	0.30	-	-	4.5	*// randomly generate new value*

Before mutation	1.0	2.4	0.8	7.6
After mutation	0.3	2.4	0.8	4.5

Nonuniform Mutation

Usually, new value $=$ current value $+$ random perturbation (Δ), i.e.,

$$x'_i = x_i + \Delta,\ \forall i \in \{1, \ldots, d\} \text{ and } LB_i \leq x'_i \leq UB_i,$$

where Δ is sampled from a distribution with 0 mean, and a given standard deviation (σ), also called mutation step-size.

Main differences compared to the uniform mutation presented previously:

- **Nonuniform.**
- **$+$ A random perturbation, instead of generating a totally new value.**

Question

Why the perturbation is sampled from a distribution with 0 mean not a mean < 0 or > 0?

Nonuniform Mutation (continued)

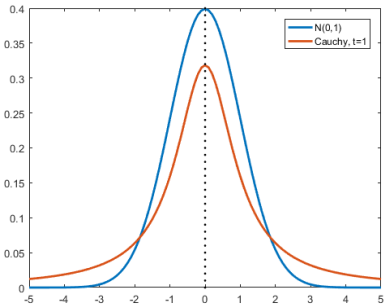
$$x'_i = x_i + \Delta,\ \forall i \in \{1, \ldots, d\} \textbf{ and } LB_i \leq x'_i \leq UB_i,$$

- **Perturbation applied with probability 1 per gene.**
- **Different perturbation techniques:**
  - **Differ in the distribution:**
    - **Gaussian distribution:** $\Delta \sim \mathcal{N}(0, \sigma^2)$
    - **Cauchy distribution:** $\Delta \sim \mathcal{C}(0, t)$
      $\rightarrow$ **A "fatter" tail.** $\rightarrow$ **higher probability of generating larger values.**
  - **Differ in how σ is updated:**
    - **Self-adaptive: demonstrated to be successful for real-valued, binary and integer search space.** $\rightarrow$ **More in the next lecture.**
- **Curtailing the resulting value to locate in $[LB_i, UB_i]$, if not.**

Normal Distribution and Cauchy Distribution

PDF of Gaussian: $f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

PDF of Cauchy: $f(x; x_0, t) = \frac{1}{\pi t\left[1+\left(\frac{x-x_0}{t}\right)^2\right]} = \frac{1}{\pi t}\left[\frac{t^2}{(x-x_0)^2+t^2}\right]$, **where t is the scale parameter which specifies the half-width at half-maximum.**



Outline of This Lecture

Summary

1. **Evolutionary algorithms vary in the**
   - **representation,**
   - **search operators (crossover operators and mutation operators)**
   - **and selection schemes.**

2. **(THIS LECTURE) crossover operators and mutation operators for discrete and real-valued representations.**

3. **Different problems require different search operators and selection schemes. There is no universally best one.**

We will learn selection schemes in the next lecture.

Essential Reading for This Lecture

1. **X. Yao, "Evolutionary computation: A gentle introduction," In *Evolutionary Optimization*, R. Sarker, M. Mohammadian and X. Yao (eds.), Chapter 2, pp.27-53, Kluwer Academic Publishers, Boston, 2002. (ISBN 0-7923-7654-4)**
   `https://link.springer.com/chapter/10.1007/0-306-48041-7_2`
   (You can download the pdf through VPN of our university)

2. **H. G.Beyer & H. P. Schwefel, "Evolution strategies-A comprehensive introduction". *Natural Computing*, 1(1), 3-52, 2002.**

Essential Reading for Next Lecture

1. **T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.), Handbook of Evolutionary Computation, IOP Publ. Co. & Oxford University Press, 1997. Part C.**
   (The part of selection schemes.)

2. **X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, 3(2):82-102, July 1999.**
   `https://www.cs.bham.ac.uk/~xin/papers/published_tec_jul99.pdf`
   (You can also visit IEEEXplore and download the pdf through VPN of our university)