EVOLUTIONARY COMPUTATION AND MULTIOBJECTIVE OPTIMIZATION

Gary G. Yen, *FIEEE*, *FIET, FIAPR*

gyen@okstate.edu

***Regents Professor*, Oklahoma State University**

四川大学高端外籍讲座教授

Summer Course at

四川大学计算机学院

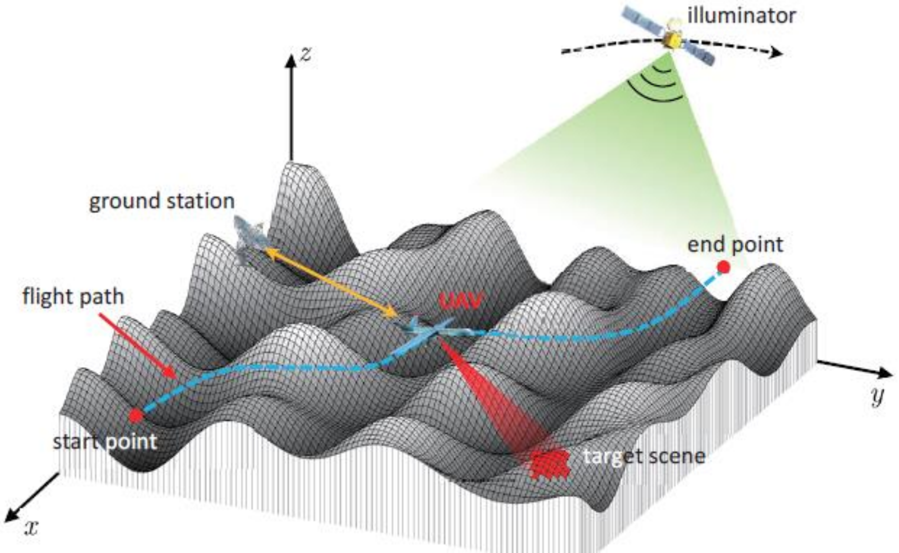
Day Three of EIGHT, June 30, 2022

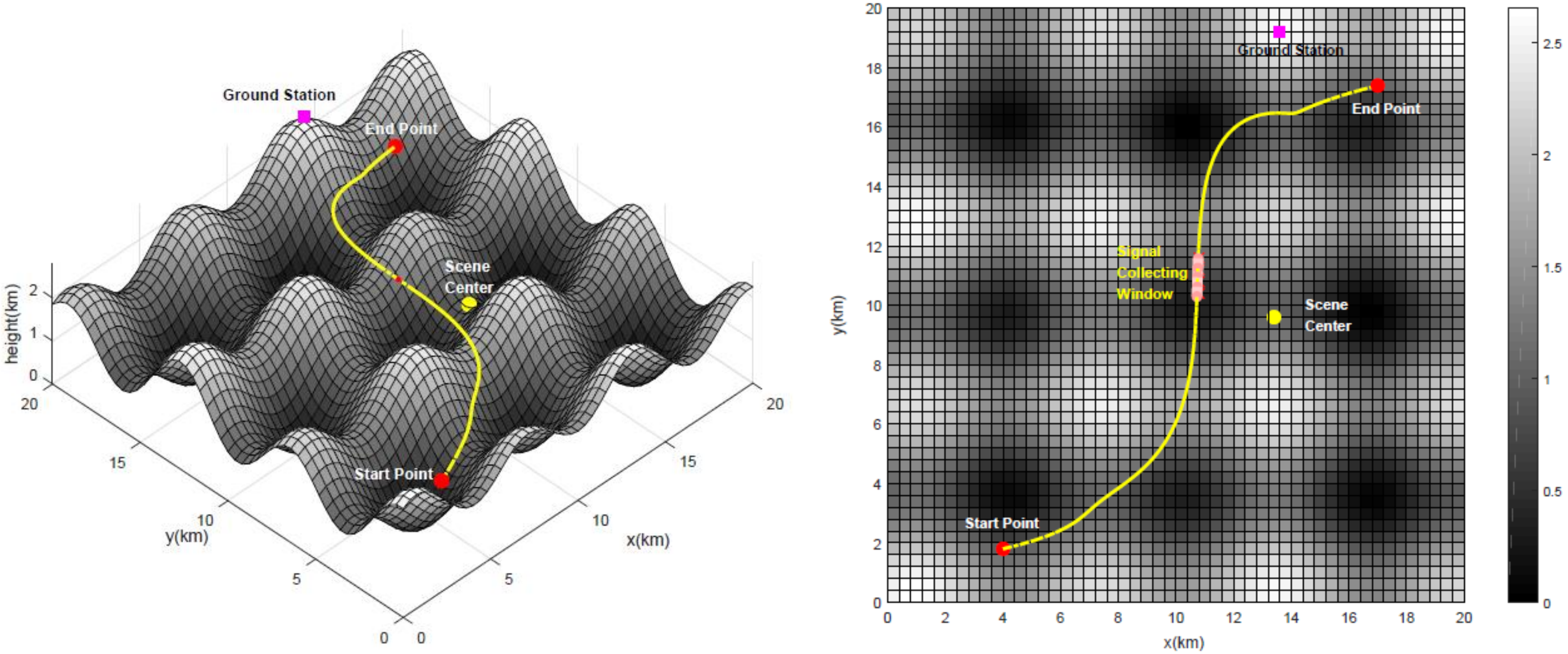
Case Study 3:
UAV Path Planning

- **Motivation:** Due to the superior flexibility and cost-efficiency, unmanned aerial vehicle (UAV) is becoming an indispensable platform for advanced remote sensing applications, such as disaster relief, transportation/agriculture surveillance.

- **Approach:** The performance and implementation of the energy-efficient passive UAV radar imaging system is investigated. Equipped with a synthetic aperture radar (SAR) receiver, the UAV platform passively reuses the backscattered signal of an external illuminator, and achieves SAR imaging and data communication.
- Firstly, we present the system concept and block diagram. Then, the imaging performance and feasibility are analyzed for the typical illuminators. A set of mission performance evaluators is established to comprehensively assess the capability of the system, including UAV navigation, passive SAR imaging and communication.

- An energy-efficient path planning method is then presented to generate a UAV path with optimized mission performance. Numerical simulations are conducted to verify the effectiveness of the performance evaluators and path planning method. Finally, a real implementation of the system is given. The imaging results of an airborne passive SAR experiment using external illuminator further validates the imaging capability and performance analysis of the system.

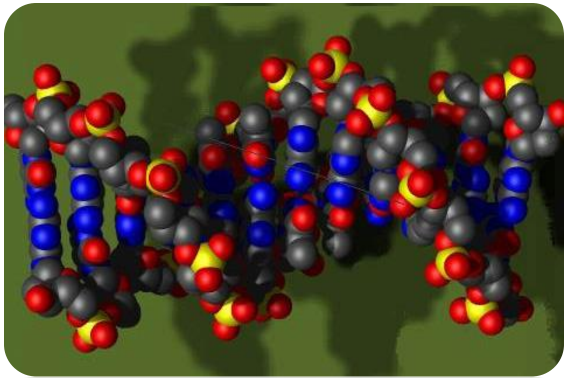




"Mission planning for energy-efficient passive UAV radar imaging system based on sub-stage division collaborative search," Sun Z., Yen G.G., Wu J., Ren H., An H., and Yang J., ***IEEE Transactions on Cybernetics***, 2022, in early access.

5. GENETIC ALGORITHMS- Part 2

遗传算法



Design Parameters

1. Design a representation
2. Decide how to initialize a population
3. Design a mechanism to map the phenotype to genotype and vice versa
4. Design a way of evaluating an individual
5. Design suitable mutation operators
6. Design suitable crossover operators
7. Decide how to select parents for crossover
8. Decide how to select individuals to be replaced
9. Decide when to stop the algorithm
10. Decide how to manage the population

1. Designing a Representation

- We have to come up with a method of representing an individual as a genotype.
- The way we choose to do it must be *relevant* to the problem that we are trying to solve.
- When choosing a representation, we have to bear in mind how the genotypes will be evaluated and what the genetic operators might be.

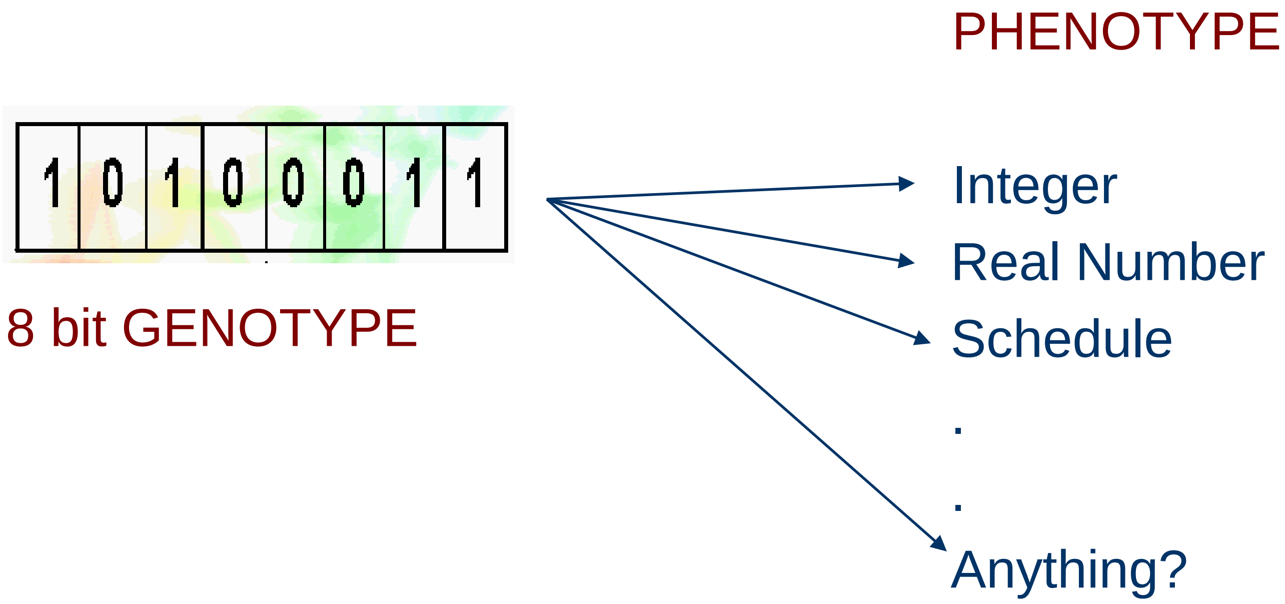
Binary Valued Representation

- A chromosome should in some way contain information about solution that it represents. The most used way of encoding is a binary string. A chromosome then could look like this:

Chromosome 1	1101 1001 0011 0110
Chromosome 2	1101 1110 0001 1110

- Each chromosome is represented by a binary string. Each bit in the string can represent some characteristics of the solution. Another possibility is that the whole string can represent a number

Examples



PHENOTYPE

1	0	1	0	0	0	1	1

8 bit GENOTYPE

Integer

Real Number

Schedule

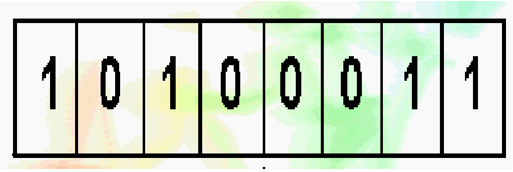
.

.

Anything?

WHAT CAN IT REPRESENT ???

- Phenotype can be an integer number



1	0	1	0	0	0	1	1

$$=1*2^7+0*2^6+1*2^5+0*2^4+0*2^3+0*2^2+1*2^1+1*2^0$$

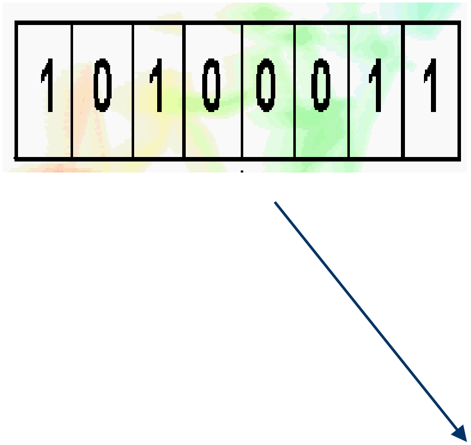
$$= 128+32+2+1 = 163$$

- Phenotype can be a real number

  e.g. a number between 2.5 and 20.5 using 8 binary digits

1	0	1	0	0	0	1	1

X= 2.5 + (163/256) (20.5-2.5) = 13.9609



1	0	1	0	0	0	1	1

Job	Time step
1	2
2	1
3	2
4	1
5	1
6	1
7	2
8	2

- Phenotype can be a schedule
- e.g. 8 jobs , 2 time steps

Real Valued Representation

- A very natural encoding if the solution we are looking for is a list of real-valued numbers, then encode as a list of real-valued numbers (not as a string of 1’s and 0’s)

- Lots of applications , e.g., parameter optimization

- Individuals are represented as a tuple of *n* real valued numbers:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x_i \in R$$

- The fitness function maps tuples of real numbers to a single real number

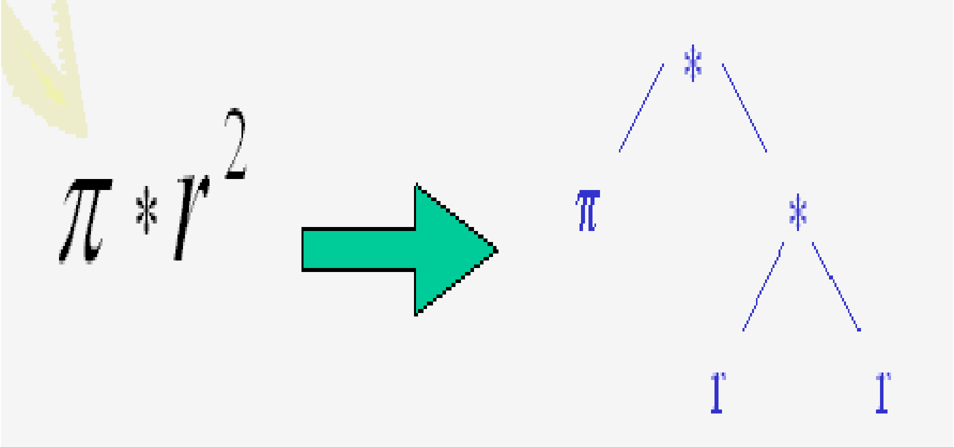
$$f\text{: } R^n \rightarrow R$$

Order Based Representation

- Individuals are represented as permutations
- Used for ordering, sequencing problems
- Famous example: traveling salesman problem where every city gets assigned a unique number from 1 to *n*. A solution could be (5, 4, 2, 1, 3).
- Famous example: *N*-queen problem (or N^2-queen)
- Needs special operators to make sure the individuals stay in *valid* permutations.

Tree Based Representation

- Individuals in the population are trees
- Any expression can be drawn as a tree of function and terminals
  - Functions : sine, cosine, add, sub, if-then-else
  - Terminals : X, Y, 0.456, true, false, sensor 1,



Tree Based: Closure and Sufficiency

- In the tree based representation we need to specify a *function set* and a *terminal set*. It is very desirable that these sets both satisfy closure and sufficiency.
- By *closure* we mean that each of the functions in the function set is able to accept as its arguments any value and data type that may possibly be returned by some other function or terminal.
- By *sufficiency* we mean that there should be a solution in the space of all possible programs constructed from the specified function and terminal sets.

2. Initialization of a Population

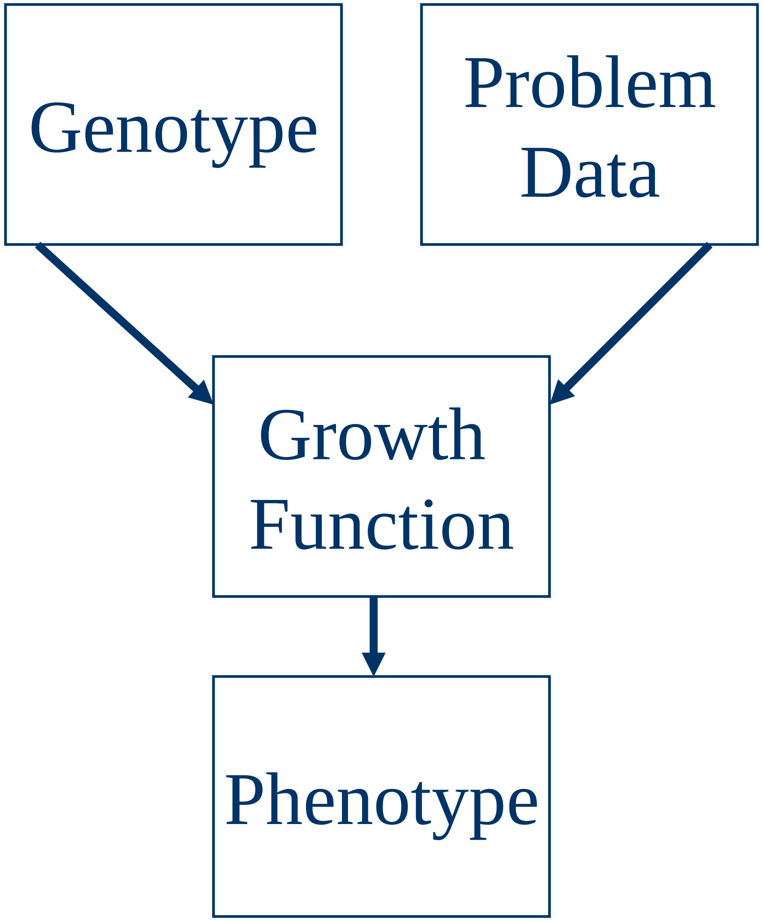
- Usually, at random

- Uniformly on the search space …. *If possible*
  - Binary representation: 0 and 1 with probability of 0.5
  - Real-valued representation: uniformly on a given interval (OK for bounded values only)

- Seed the population with previously known values or those from heuristics. With care:
  - Possible loss of genetic diversity
  - Possible unrecoverable bias

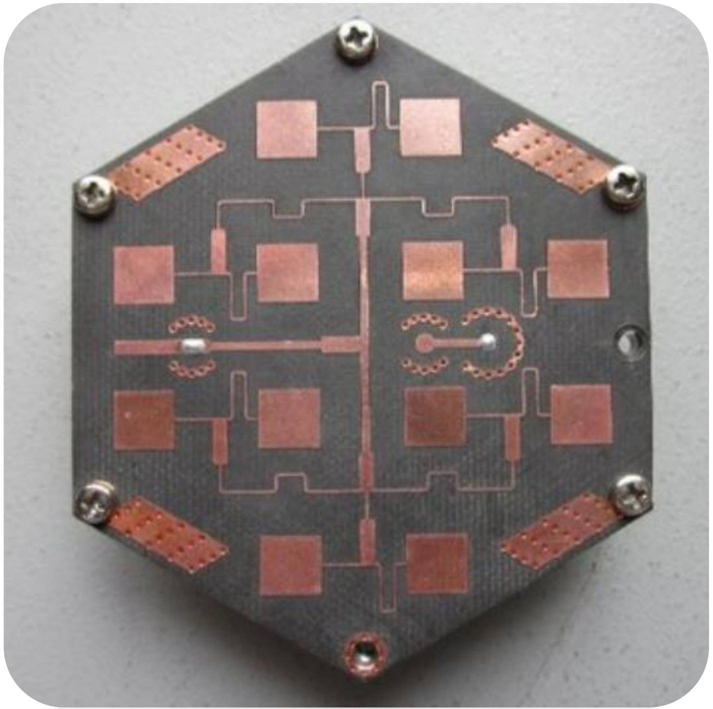
3. Mapping a Phenotype from a Genotype

- Sometimes producing the phenotype from the genotype is an obvious process.
- Other times the genotype might be a set of parameters to some algorithm, which work on the problem data to produce the phenotype.



4. Evaluating an Individual

- By far the most costly step for real applications
  - *Do not re-evaluate unmodified individuals*
- It might be a subroutine, a black-box simulator, or any external process (e.g., robot experiment).
- The effectiveness of the process depends on the choice of the fitness function.
- Your could use approximate fitness, but not far too long (*fitness inheritance* or *fitness approximation*).
- Constraint handling- what if the phenotype breaks some constraint of the problem:
  - Penalize the fitness
  - Specific evolutionary method
- Multiobjective evolutionary optimization gives a set of compromise solutions



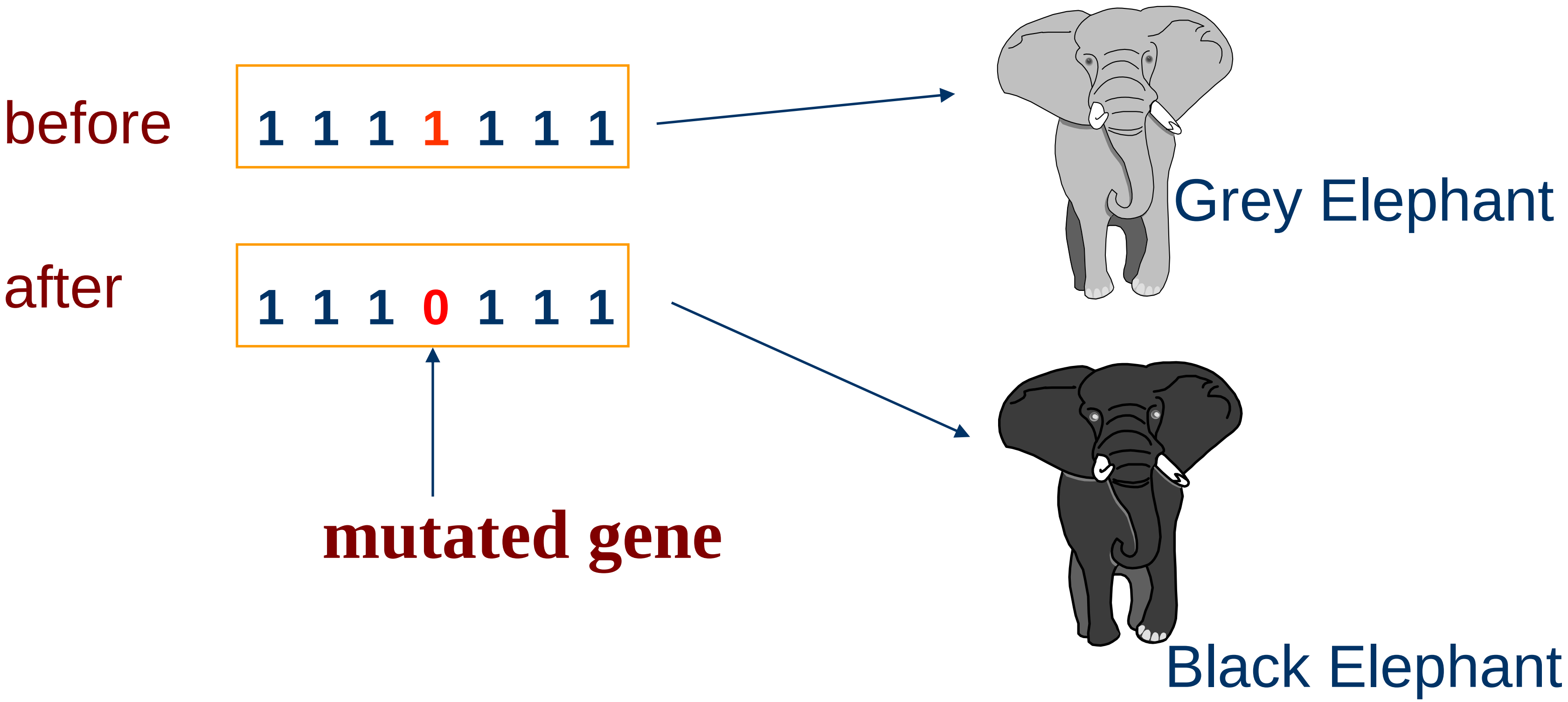
The fitness evaluations of a given geometrical configuration for a 5 by 5 ***microstrip antenna array*** by COMSOL or FEKO would take more than 26 hours to quantify the gain, side lobe level or reflection coefficient.

“5 by 5 microstrip antenna array design by multiobjective differential evolution based on fuzzy performance feedback,” Jariyatantiwait C. and Yen G.G., ***International Journal of Swarm Intelligence Research***, 7(4), 2016, pp. 1-22.

5. Mutation Operators

- The mutation operator should allow every part of the search space to be reached.
- The size of mutation is important and should be controllable.
- Mutation should produce valid chromosomes.

Binary Valued Mutation



before 1 1 1 1 1 1 1

after 1 1 1 0 1 1 1

mutated gene

Grey Elephant

Black Elephant

Mutation probability p_m, for each gene

Real Valued Mutation

- Perturb values by adding some random noise

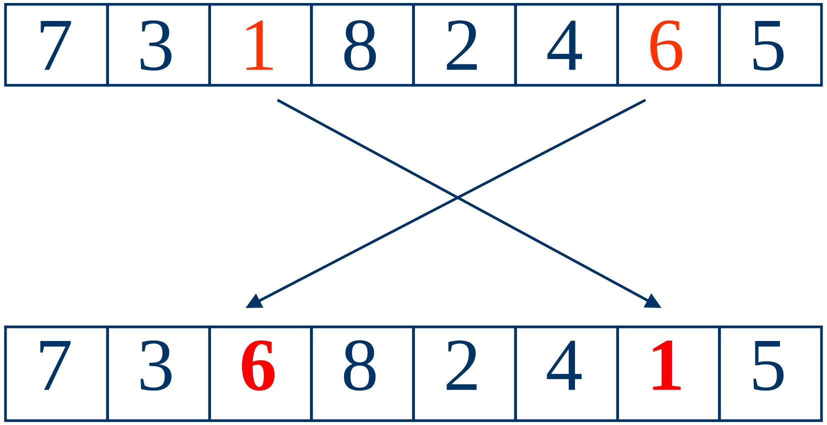
  Often, a Gaussian/normal distribution N(0,σ) is used, where

  - 0 is the mean value
  - σ is the standard deviation

and $x'_i = x_i + N(0,\sigma_i)$ for each parameter

Order Based Mutation

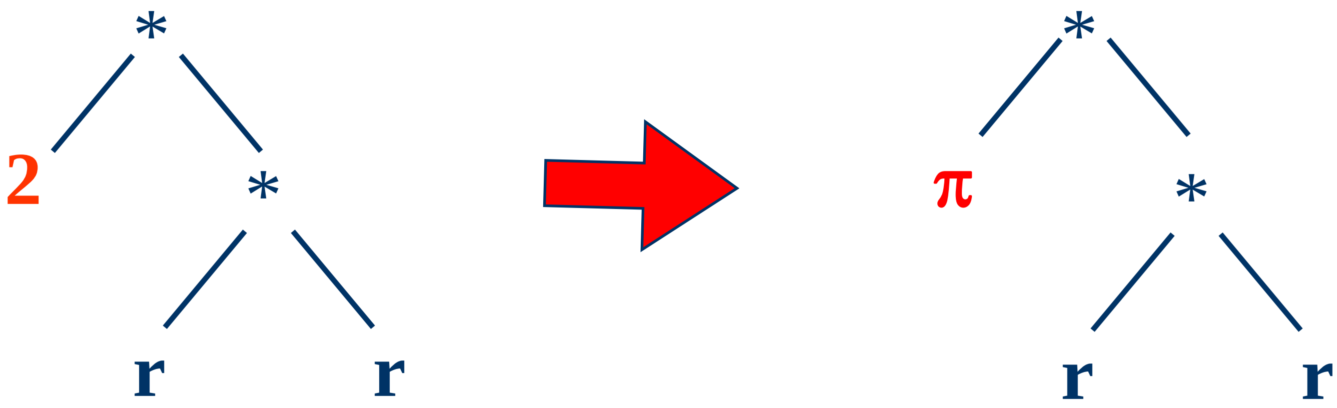
- Randomly select two different genes and swap them.



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

Tree Based Mutation

- Single point mutation selects one node and replaces it with a similar one.



Polynomial Mutation (PM)

For one individual x_t, it is updated by polynomial mutation

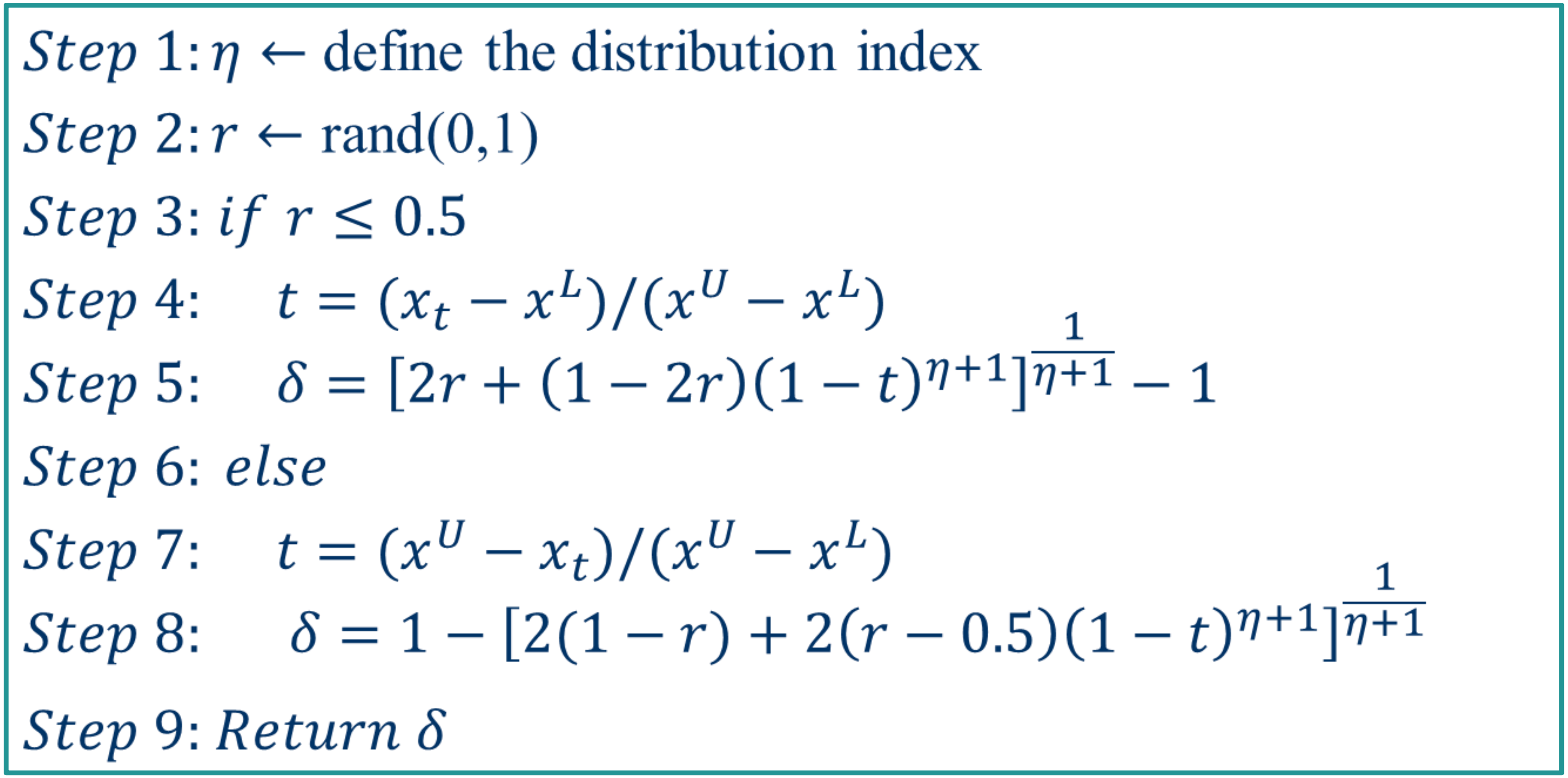
$$x_{t+1} = x_t + \delta(x^U - x^L)$$

where x^U and x^L denote the upper and lower bound of x_t, δ is a scaling factor with the probability

$$P(\delta) = 0.5(\eta + 1)(1 - |\delta|^{\eta})$$

and η is a predefined parameter termed as distribution index

Calculating Scaling Factor δ



```
$Step\ 1{:}\ \eta \leftarrow$ define the distribution index
$Step\ 2{:}\ r \leftarrow \mathrm{rand}(0,1)$
$Step\ 3{:}\ if\ r \leq 0.5$
$Step\ 4{:}$    $t = (x_t - x^L)/(x^U - x^L)$
$Step\ 5{:}$    $\delta = [2r + (1 - 2r)(1 - t)^{\eta+1}]^{\frac{1}{\eta+1}} - 1$
$Step\ 6{:}\ else$
$Step\ 7{:}$    $t = (x^U - x_t)/(x^U - x^L)$
$Step\ 8{:}$    $\delta = 1 - [2(1 - r) + 2(r - 0.5)(1 - t)^{\eta+1}]^{\frac{1}{\eta+1}}$
$Step\ 9{:}\ Return\ \delta$
```

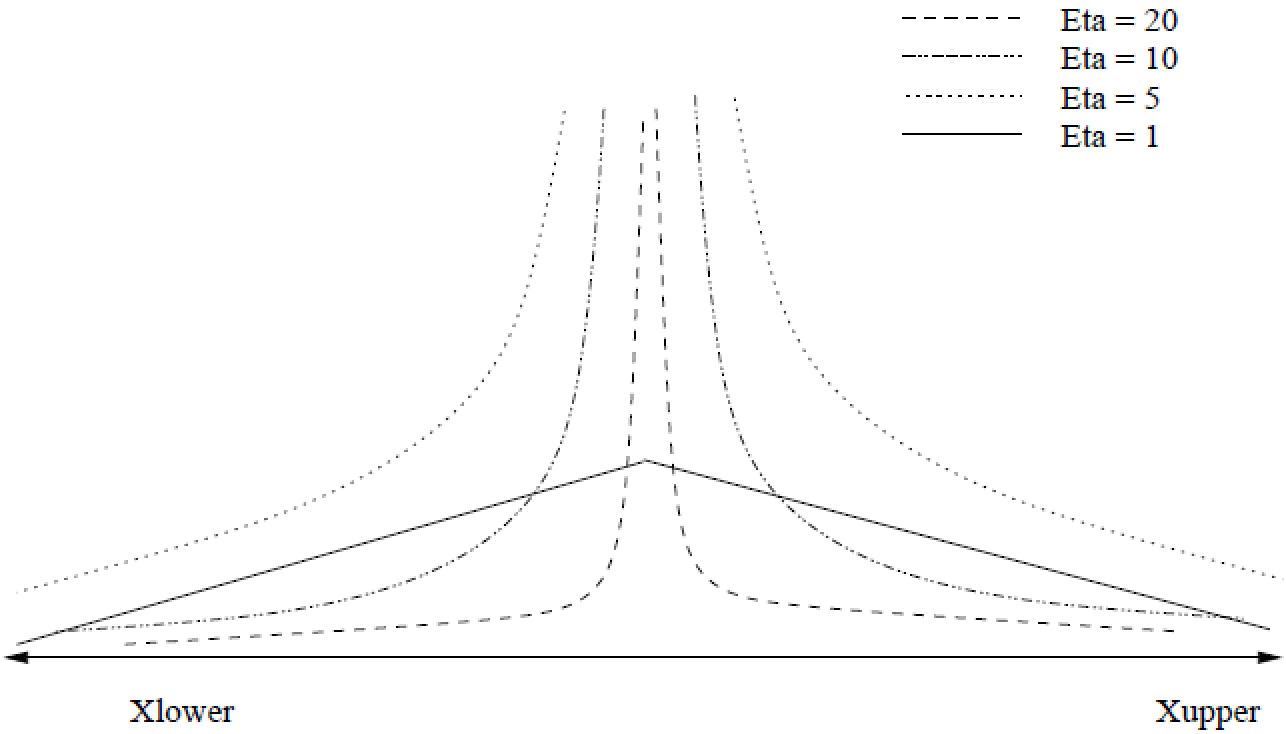
PM Algorithm

```
Step 1: $\eta \leftarrow define\ the\ distribution\ index$
Step 2: $for\ each\ parameter\ x_t^i\ in\ x_t$:
Step 3:     $if\ rand(0,1) < p_m$:
Step 4:         $compute\ \delta\ based\ on\ its\ updated\ algorithm$
Step 5:         $x_{t+1}^i = x_t^i + \delta(x^U - x^L)$
Step 6: $Return\ x_{t+1}$
```

p_m is the probability of mutation and commonly set to $1/n$ in most literatures

Effects of Distribution Index

$\eta \in [20,100]$ is adequate in most problems and commonly set to 20 in most literatures.



The effect of distribution index η on mutated values

6. Recombination/Crossover Operators

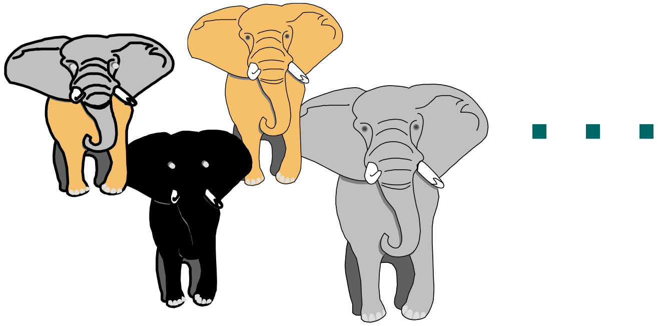
- The child should inherit something from ***both*** parents. If this is not the case then the operator is a mutation operator.

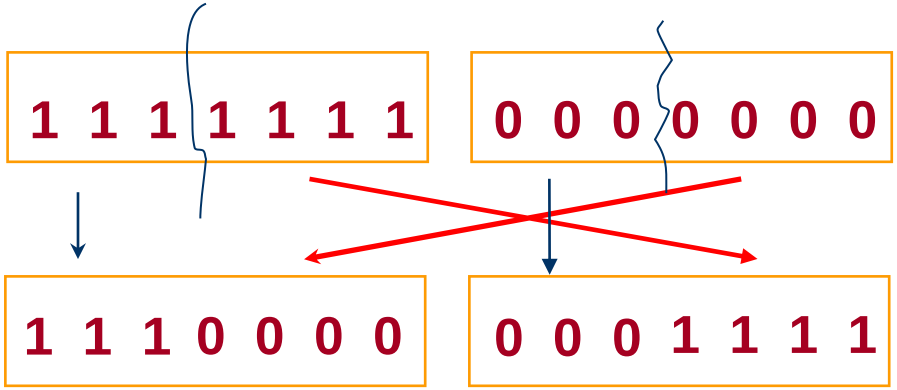
- The recombination/crossover operator should be designed in conjunction with the representation so that recombination is not always catastrophic.

- Recombination should produce valid chromosomes.

Binary Valued Crossover

Whole Population:





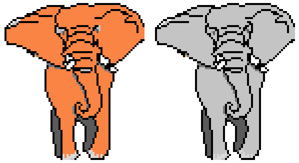
1 1 1 1 1 1 1

0 0 0 0 0 0 0

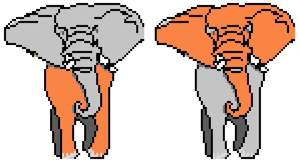
1 1 1 0 0 0 0

0 0 0 1 1 1 1

parents

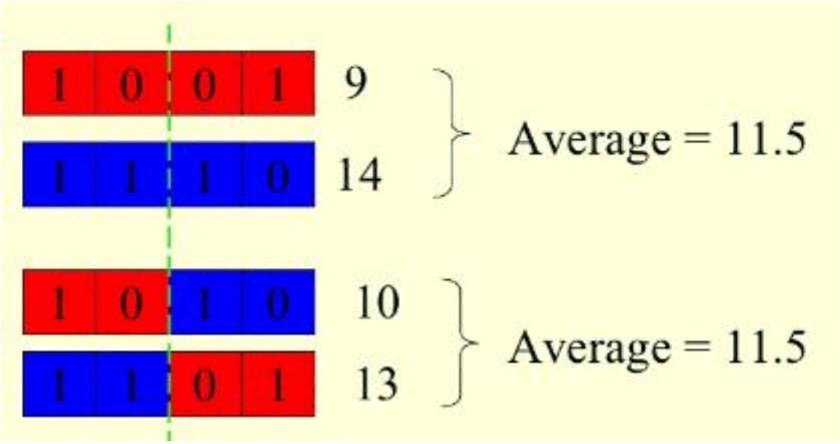


offspring



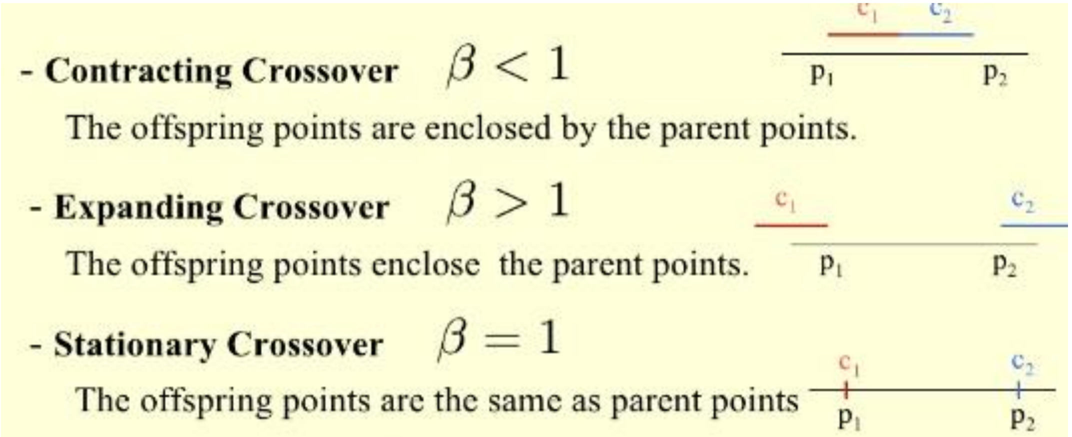
1-point crossover

- Important properties of 1-point crossover
  - The average of the decoded parameter values is the same before and after the crossover



- The probability of occurrence of spread factor $\beta \approx 1$ is more likely than any other value.
- The spread factor β is defined as the ratio of the spread of offspring to that of parents

$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right|$$



- **Contracting Crossover** $\beta < 1$

  The offspring points are enclosed by the parent points.

- **Expanding Crossover** $\beta > 1$

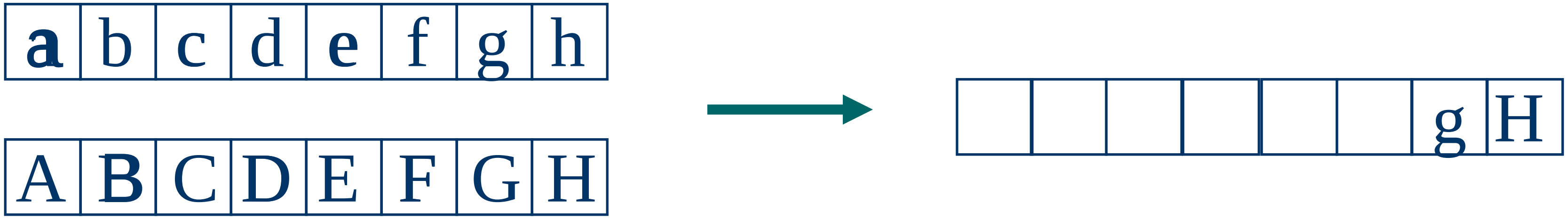
  The offspring points enclose the parent points.

- **Stationary Crossover** $\beta = 1$

  The offspring points are the same as parent points

Real Valued Crossover

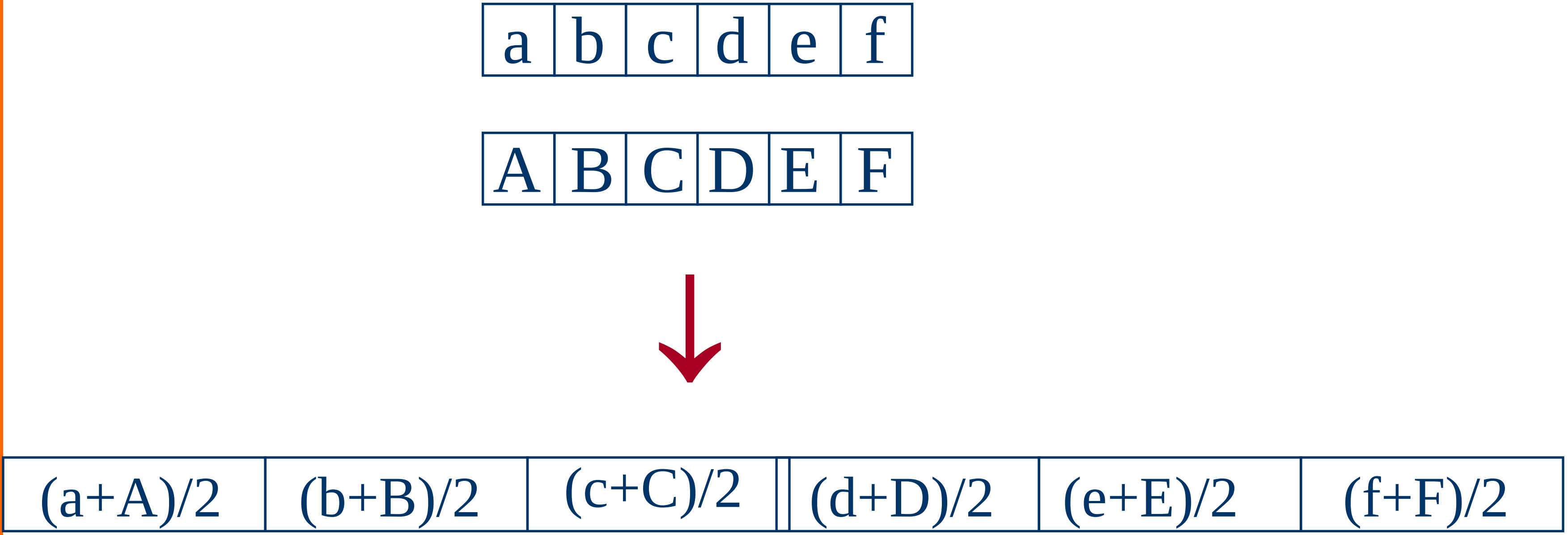
- Uniform crossover: given two parents one child is created as follows



a	b	c	d	**e**	f	g	h
A	**B**	C	D	E	F	G	H

						g	H

- Intermediate recombination (arithmetic crossover): given two parents one child is created as follows



a	b	c	d	e	f

A	B	C	D	E	F

↓

(a+A)/2	(b+B)/2	(c+C)/2	(d+D)/2	(e+E)/2	(f+F)/2

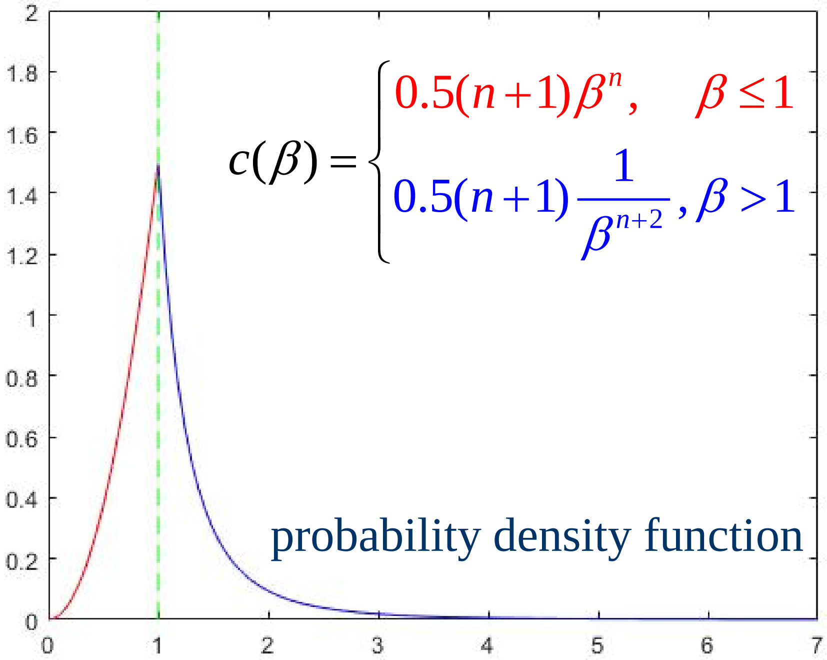
Simulated Binary Crossover (SBX)

1. *Average* Property

$$c_1 = \frac{1}{2}(p_1 + p_2) - \frac{1}{2}\beta(p_2 - p_1)$$

$$c_2 = \frac{1}{2}(p_1 + p_2) + \frac{1}{2}\beta(p_2 - p_1)$$

2. *Spread Factor* Property

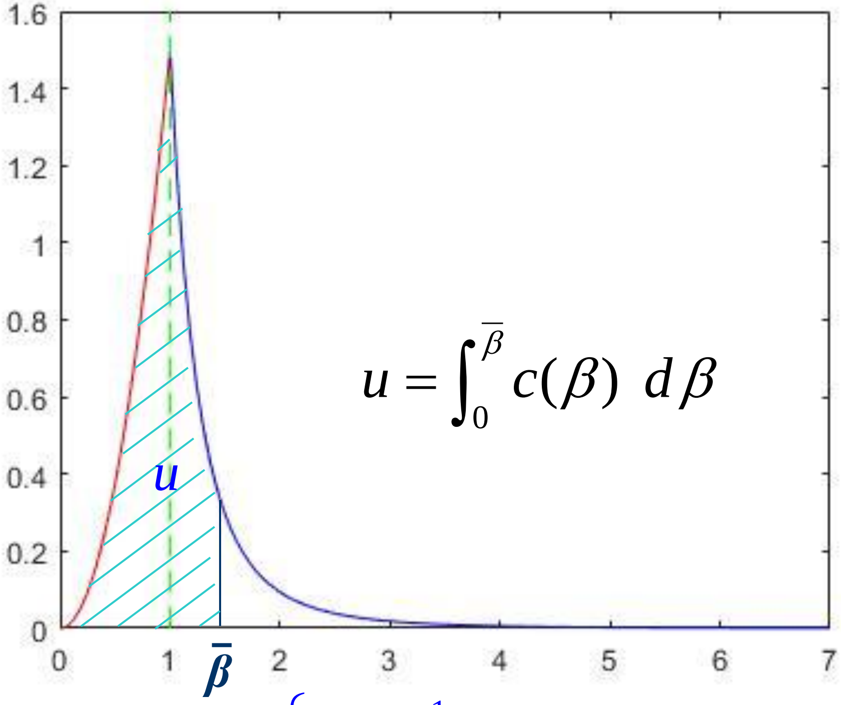


$$c(\beta) = \begin{cases} 0.5(n+1)\beta^n, & \beta \le 1 \\ 0.5(n+1)\dfrac{1}{\beta^{n+2}}, & \beta > 1 \end{cases}$$

probability density function

If we can generate ***β*** randomly which satisfy given probability density function, we can get lots of different children c_1 and c_2.

Using a Probability Density Function to Compute Probability



$$u = \int_0^{\bar{\beta}} c(\beta)\ d\beta$$

$$c(\beta) = \begin{cases} 0.5(n+1)\beta^n, & \beta \le 1 \\ 0.5(n+1)\dfrac{1}{\beta^{n+2}}, & \beta > 1 \end{cases}$$

Probability Density Function(PDF)

$$\beta = \begin{cases} (2u)^{\frac{1}{n+1}}, & if\ u \le 0.5 \\ (\dfrac{1}{2-2u})^{\frac{1}{n+1}}, & if\ u > 0.5 \end{cases}$$

Cumulative Distribution Function (CDF)

1.Select two parents p_1 and p_2

2.Generate a uniform distribution random number $u \in [0,1)$

3.Calculate β

$$\beta = \begin{cases} (2u)^{\frac{1}{n+1}}, & if\ u \leq 0.5 \\ (\frac{1}{2-2u})^{\frac{1}{n+1}}, & if\ u > 0.5 \end{cases}$$

where n is the Spread Factor distribution index

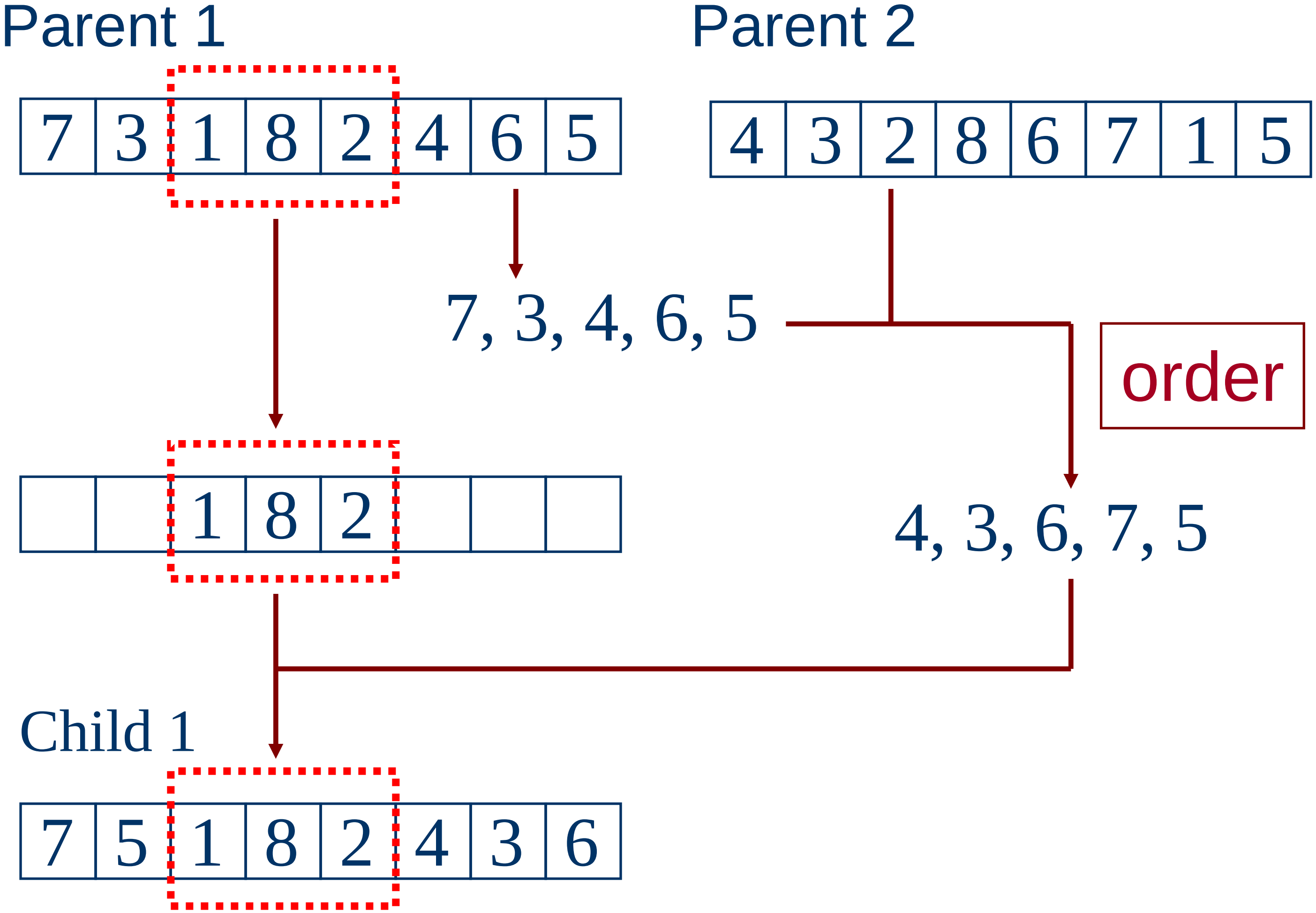
4. Compute offspring as

$$c_1 = \frac{1}{2}(p_1 + p_2) - \frac{1}{2}\beta(p_2 - p_1)$$

$$c_2 = \frac{1}{2}(p_1 + p_2) + \frac{1}{2}\beta(p_2 - p_1)$$

Order Based Crossover

- Choose an arbitrary part from the first parent and copy this to the first child.
- Copy the remaining genes that are not in the copied part to the first child:
  - starting right from the cut point of the copied part
  - using the order of genes from the second parent
  - wrapping around at the end of the chromosome
- Repeat this process with the parent roles reversed.



Parent 1

7	3	1	8	2	4	6	5

Parent 2

4	3	2	8	6	7	1	5

7, 3, 4, 6, 5

order

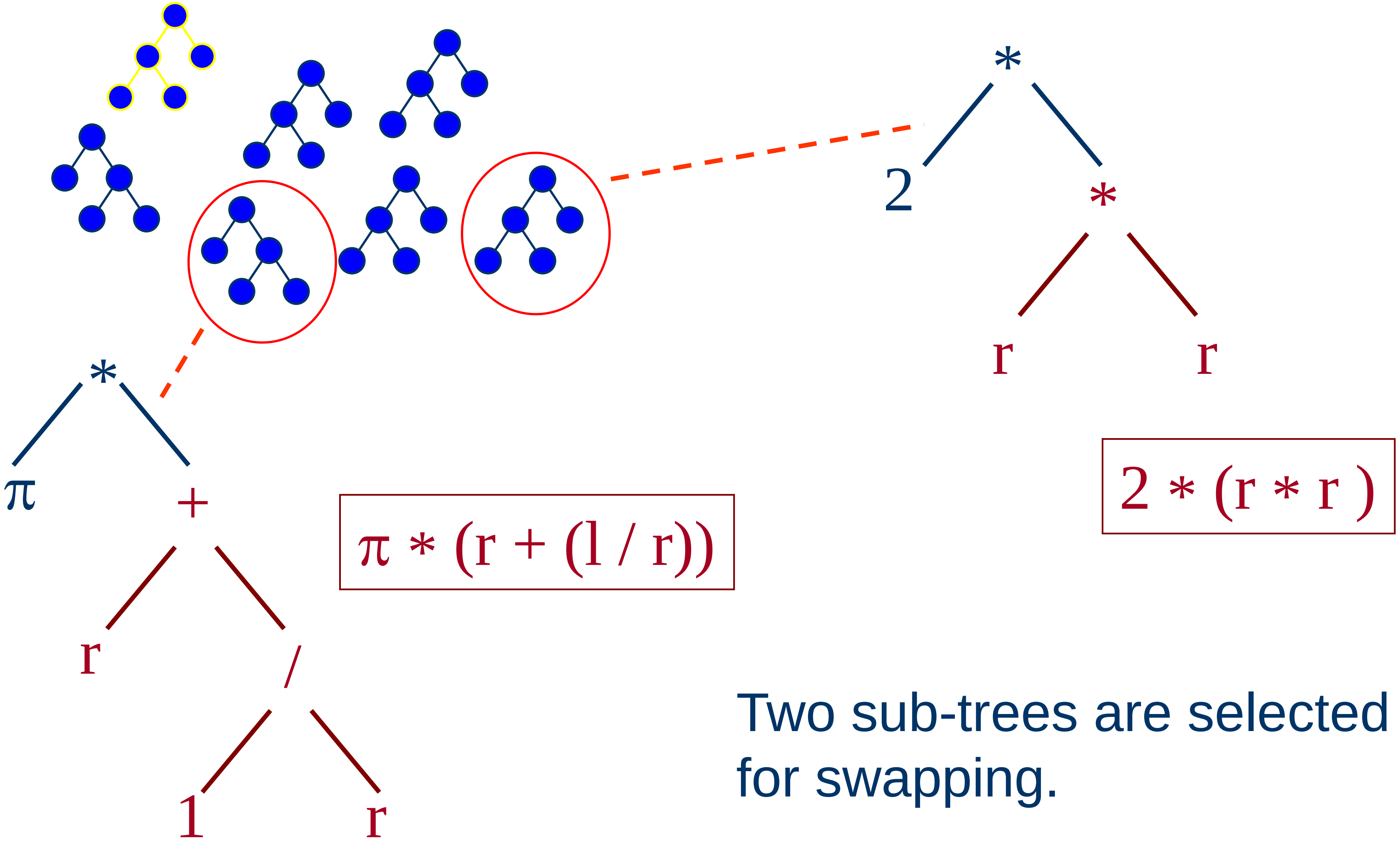
		1	8	2			

4, 3, 6, 7, 5

Child 1

7	5	1	8	2	4	3	6

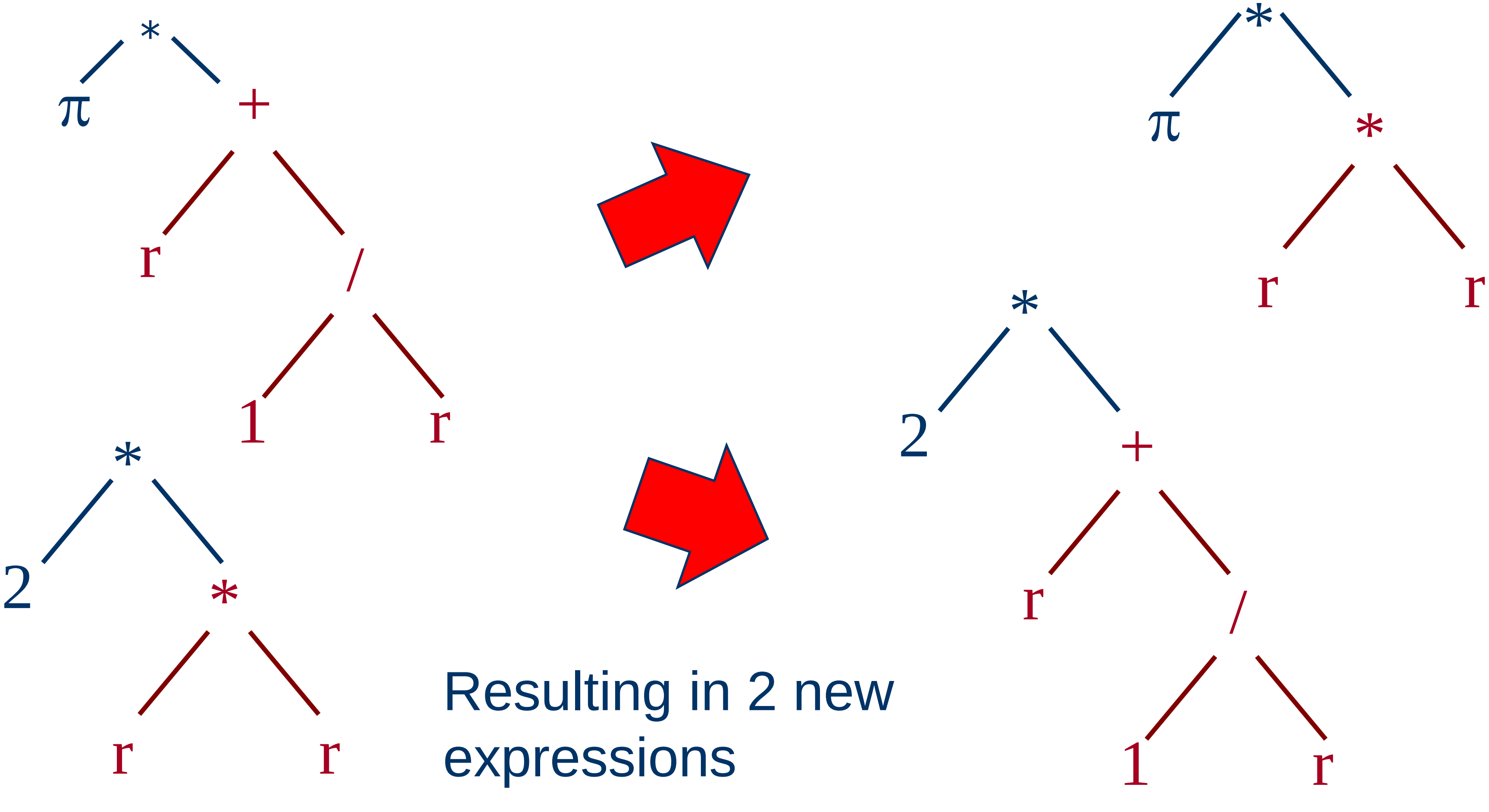
Tree Based Crossover



π * (r + (l / r))

2 * (r * r)

Two sub-trees are selected for swapping.



Resulting in 2 new expressions

7. Selection Strategy

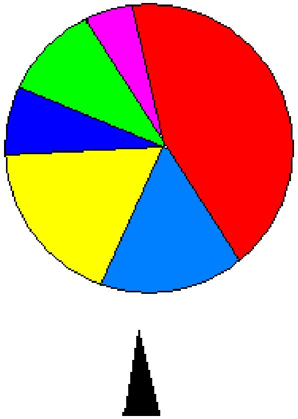
- We want to have some way to ensure that better fitted individuals have a better chance of being chosen for reproduction than poorly fitted ones.
- This will give us selection pressure which will drive the population forward.
- On the other hand, we have to be careful to give less good individuals at least some chance of being parents - they may include some useful genetic material.
- Risk of loss diversity (example from CS-ACM Talk)

Roulette Wheel Selection

- Parents are selected according to their fitness. The better the chromosomes are, the more chances they are to be selected.

- The size of the section in the roulette wheel is proportional to the value of the fitness function of every chromosome - the bigger the value is, the larger the section is.

- Better (fitter) individuals have:
  - more space
  - more chances to be selected



Fitness Proportionate Selection

Disadvantages:

- Danger of premature convergence because outstanding individuals take over the entire population very quickly.
- Low selection pressure when fitness values are near each other.
- Behave differently on transposed versions of the same function.

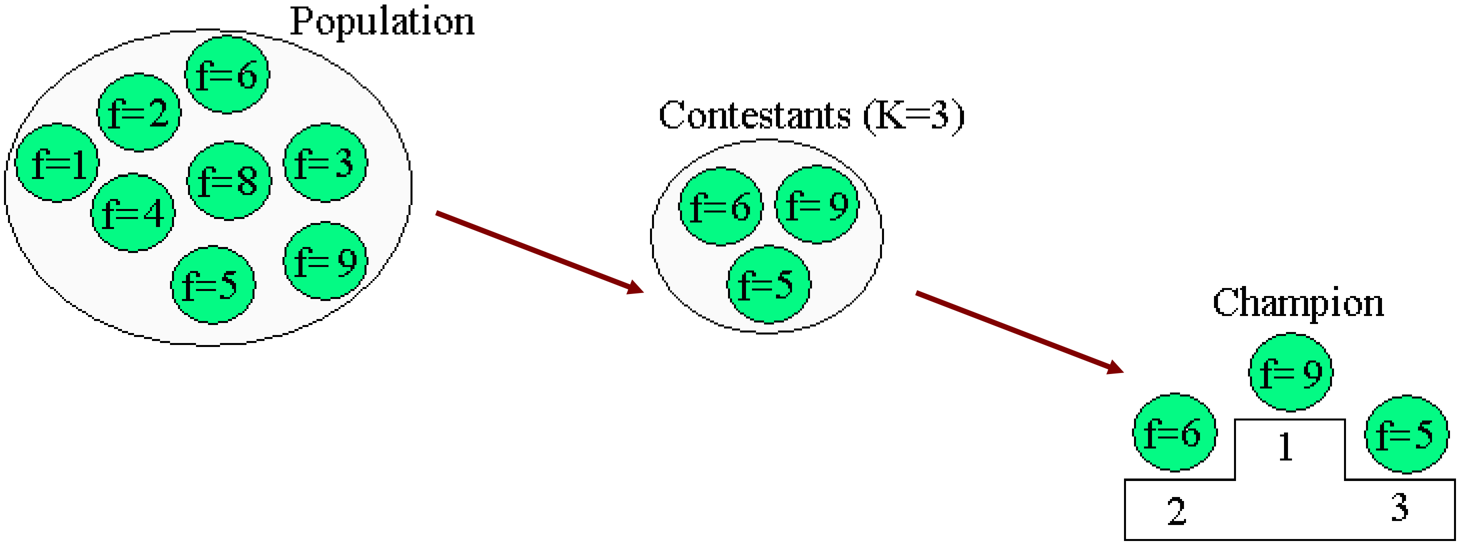
Fitness Scaling

- At the start, a few extraordinary individuals will dominate the evolution process → premature convergence
- Later on, the population average fitness may be closed to the population best fitness → random walk among the mediocre
- a cure for the problem issue
  - Start with the raw fitness function
  - Standardize to ensure
    - Lower fitness is better fitness
    - Optimal fitness equals to 0
  - Adjust to ensure
    - Fitness ranges from 0 to 1
  - Normalize to ensure
    - The sum of the fitness values equals to 1
  - Linear scaling: $f' = af + b$
    - *a and b are chosen such that* $f_{avg} = f'_{avg}$

Tournament Selection

- Select ***k*** random individuals, without replacement

- Take the best
  - ***k*** is called the size of the tournament



Ranked Based Selection

- Individuals are sorted on their fitness value from best to worse. The place in this sorted list is called rank.

- Instead of using the fitness value of an individual, the rank is used by a function to select individuals from this sorted list. The function is biased towards individuals with a high rank (= good fitness).

- Fitness: $f(A) = 5$, $f(B) = 2$, $f(C) = 19$

- Rank: $r(A) = 2$, $r(B) = 3$, $r(C) = 1$

$$h(x) = \text{min} + (\text{max} - \text{min})^* \, (r(x) - 1)/(n - 1)$$

- Function: $h(A) = 3$, $h(B) = 5$, $h(C) = 1$
- Proportion on the roulette wheel: $p(A) = 11.1\%$, $p(B) = 33.3\%$, $p(C) = 55.6\%$

Crossover vs. Mutation

- Crossover
  - modifications depend on the whole population
  - decreasing effects with convergence
  - *exploitation* operator
- Mutation
  - mandatory to escape local optima
  - *exploration* operator
- GA emphasize crossover; while ES and EP emphasize mutation

Exploration vs. Exploitation

Exploration = sample unknown regions

Too much exporation = random search, no convergence

Exploitation = try to improve the best-so-far individuals

Too much expoitation = local search only

convergence to a local optimum

8. Replacement Strategy

- The selection pressure is also affected by the way in which we decide which members of the population to eliminate in order to make space for the new individuals.

- We can use the stochastic selection methods in reverse, or there are some deterministic replacement strategies.

- We can decide never to replace the best in the population: elitism.

Elitism

- Should fitness constantly improves?
  - Re-introduce in the population previous best-so-far (elitism) or
  - Keep best-so-far in a safe place (preservation)

- Theory
  - GA: preserve mandatory
  - ES: no elitism sometimes is better

- Application: avoid user's frustration

9. Stopping Criteria

- The optimum is reached !!!!!
- Limit on CPU resources
- Maximum number of evolution generations
- Maximum number of fitness evaluations
- Limit on the user’s patience
- After some generations without improvement

10. Performance Measurement

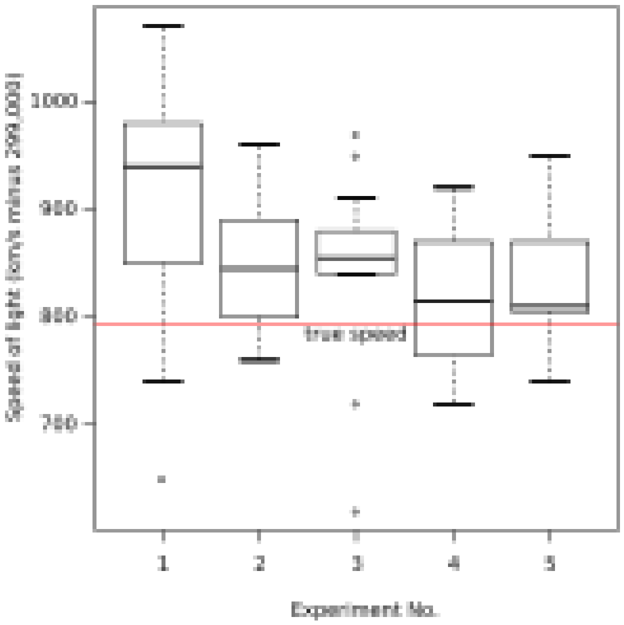
- *Never* draw any conclusion from a single run
  - Use statistical measure (average, median) (Box plot)
  - From a sufficient number of independent runs (30-50 minimum)

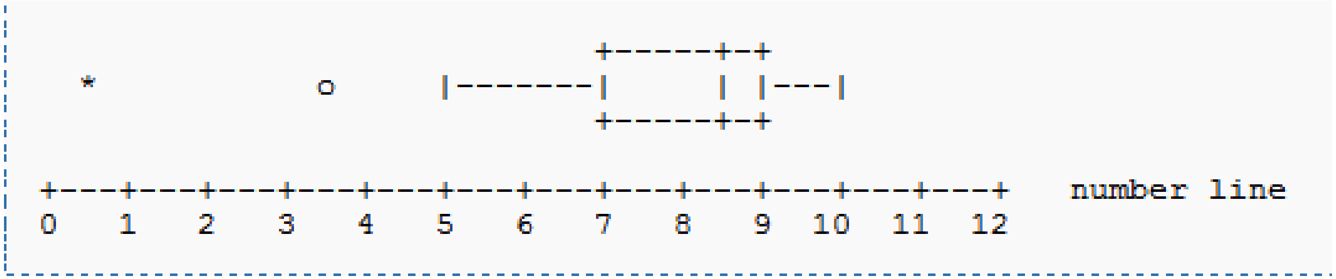
- From the application point of view
  - Design perspective
    - Find *a very good* solution at *least once*
  - Production perspective
    - Find *a good* solution at *almost every run*

- “What you test is what you get”, don’t tune algorithm performance on toy data and expect it to work with real data

Box Plot

- In descriptive statistics, a **boxplot** (also known as a **box-and-whisker diagram** or **plot**) is a convenient way of graphically depicting groups of numerical data through their five-number summaries (the smallest observation, lower quartile (Q1), median (Q2), upper quartile (Q3), and largest observation). A boxplot may also indicate which observations, if any, might be considered outliers. The boxplot was invented in 1977 by the American statistician John Tukey.
- Boxplots can be useful to display differences between populations without making any assumptions of the underlying statistical distribution. The spacings between the different parts of the box help indicate the degree of dispersion (spread) and skewness in the data, and identify outliers. Boxplots can be drawn either horizontally or vertically.





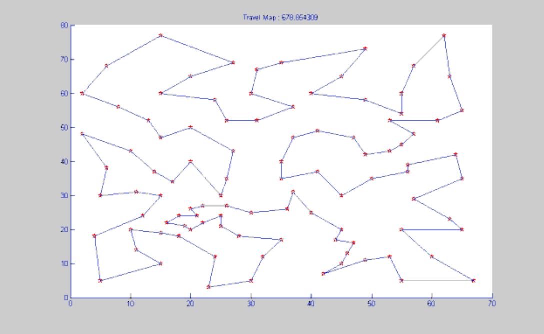
```
                            +-----+-+
  *           o     |-------|     | |---|
                            +-----+-+

+---+---+---+---+---+---+---+---+---+---+---+---+   number line
0   1   2   3   4   5   6   7   8   9  10  11  12
```

For this data set:

- smallest non-outlier observation = 5 (left "whisker")
- lower (first) quartile ($Q1$, $x_{.25}$) = 7
- ***median*** (second quartile) (*Med*, $x_{.5}$) = 8.5
- upper (third) quartile ($Q3$, $x_{.75}$) = 9
- largest non-outlier observation = 10 (right "whisker")
- the value 3.5 is a "mild" outlier, between
- the value 0.5 is an "extreme" outlier,
- the data are skewed to the left (*negatively skewed*)
- the mean value of the data can also be labeled with a point.

Genetic Algorithm for TSP



- Chromosome representation: **(order-based)**
- Population size & initialization: **(50, at random)**
- Fitness function: **(distance traveled)**
- Mutation: **(5%)**
- Recombination: **(order-based)**
- Selection: **(rank-based)**
- Replacement: **(elitism)**
- Stopping criteria: **(number of generations- 200)**

Benchmark Functions

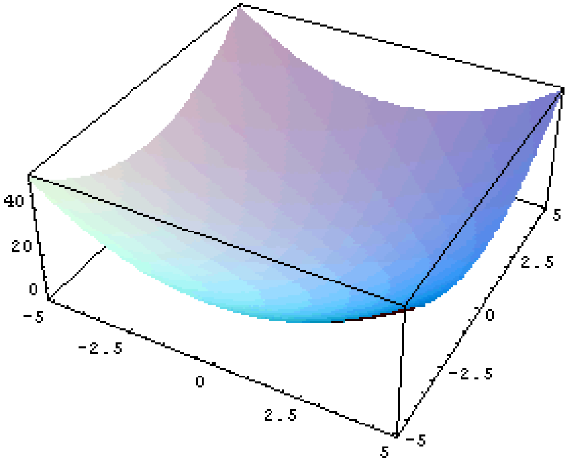
- ***de Jong's Function 1*** (Sphere Model)

  continuous, convex and unimodal

  $F_1(x) = x_1^2 + x_2^2$

  where $-5.12 \leq x_1, x_2 \leq 5.12$

  global minimum of $F_1(x) = 0$ at $x_1 = 0, x_2 = 0$



- ***de Jong's Function 2*** (Rosenbrock's valley)

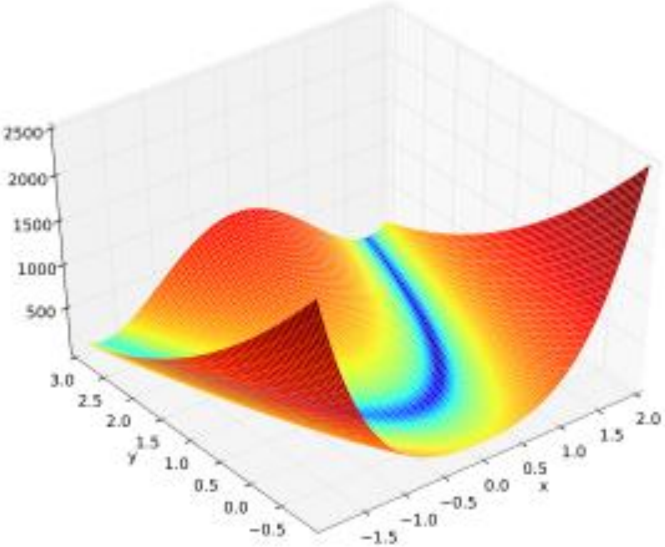
  Rosenbrock's valley, a non convex function, is a classic optimization problem, also known as Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley.

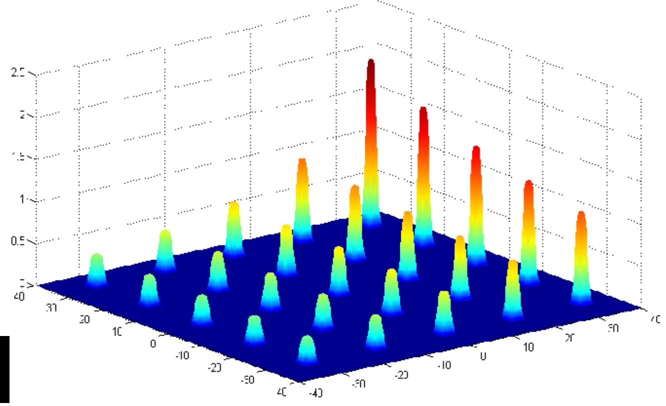
$$F_2(x) = 100 \times (x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$\text{where } -2.048 \le x_1, x_2 \le 2.048$$

global minimum of $F_2(x) = 0$ at

$$x_1 = 1, x_2 = 1$$





- ***de Jong's Function 5*** (Shekel's Foxholes)

This function, a multimodel function, which has many local optimal (i.e., 25). *Note that all 25 peaks have the same base width and the peaks are located at (32, 32), (16, 32), ... , (-32, -32).*

$$F_5(x) = 0.002 + \sum_{j=1}^{25} (1/\, f_j)$$

$$f_j = j + \sum_{i=1}^{2} (x_i - a_{i,j})^6$$

$$a_{ij} = \left[\begin{matrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 \end{matrix}\right.$$

$$\left.\begin{matrix} 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{matrix}\right]$$

with global maximum of $F_5(x) \cong 1$ at $x_1 = -32, x_2 = -32$

Problem Issues in GA

- Population size
- Binary representation vs. real representation
- Population initialization
- Noisy fitness function
- Stochastic fitness function (or dynamic environment)
- Fitness inheritance and fitness approximation
- Selection/ranking strategy
- Crossover/Recombination operator
- Mutation operator
- Replacement strategy
- Stopping criteria
- Elitism strategy
- Benchmark test functions
- Exploration vs. exploitation dilemma
- Constraint handling
- Diversity promotion
- Population management

A Demo by Goran Muric

Genetic algorithm
is used by scientists and engineers to solve various problems

What else GA can do?

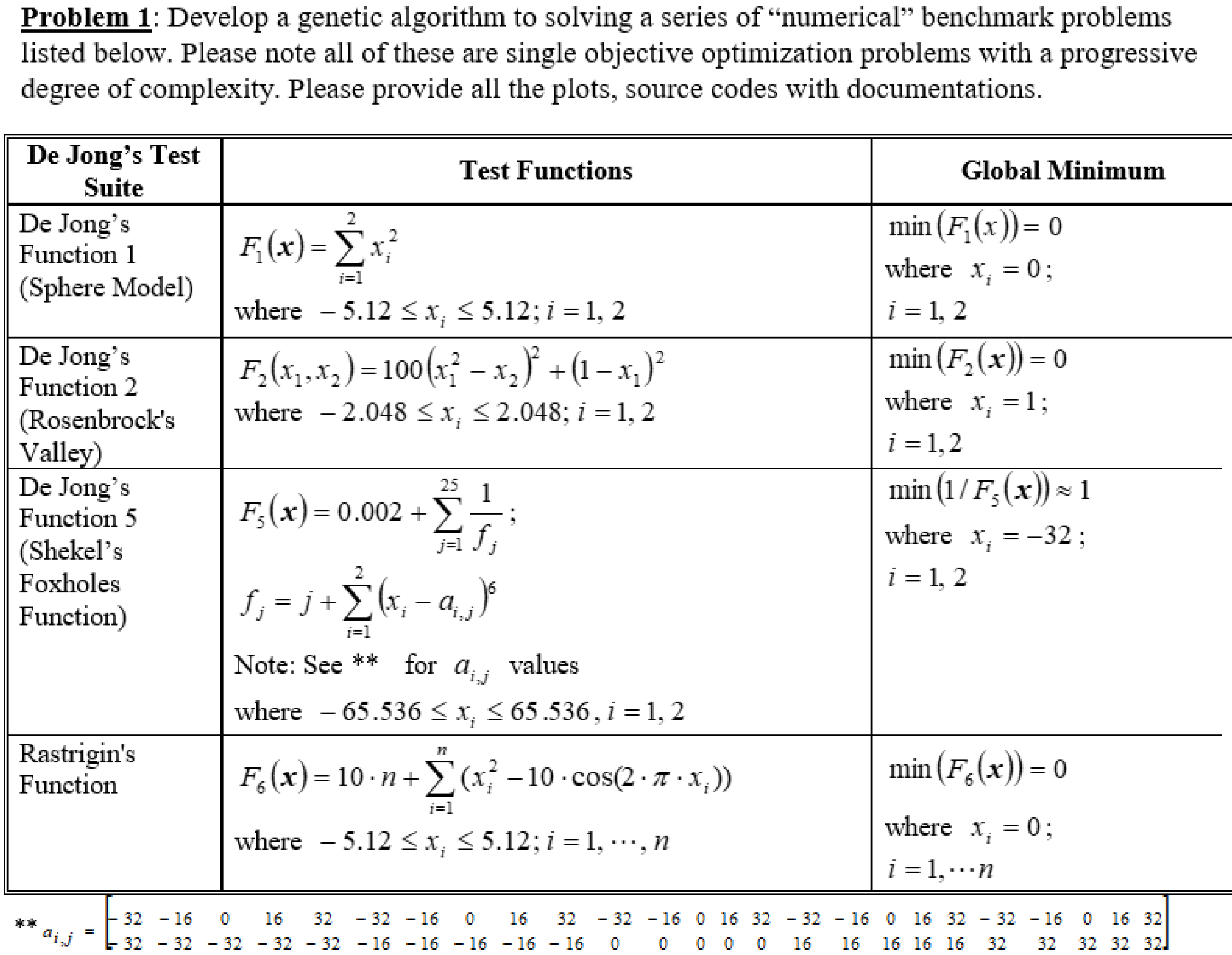
Critical Message Conveyed

With care, *Genetic Algorithm*, an evolution-inspired computational paradigm,

can work well in hard optimization problems.

Homework #4

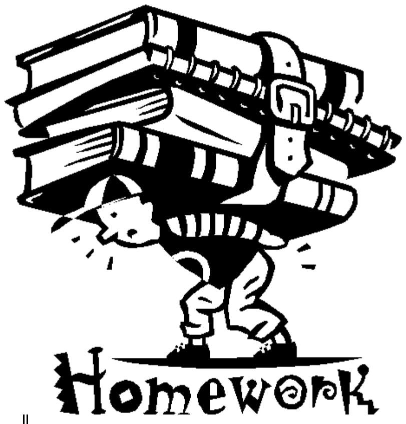
- Due: next Monday on 7/4/22, before noon



Problem 1: Develop a genetic algorithm to solving a series of "numerical" benchmark problems listed below. Please note all of these are single objective optimization problems with a progressive degree of complexity. Please provide all the plots, source codes with documentations.

De Jong's Test Suite	Test Functions	Global Minimum
De Jong's Function 1 (Sphere Model)	$F_1(\boldsymbol{x}) = \sum_{i=1}^{2} x_i^2$ where $-5.12 \le x_i \le 5.12; i = 1, 2$	$\min(F_1(x)) = 0$ where $x_i = 0;$ $i = 1, 2$
De Jong's Function 2 (Rosenbrock's Valley)	$F_2(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$ where $-2.048 \le x_i \le 2.048; i = 1, 2$	$\min(F_2(\boldsymbol{x})) = 0$ where $x_i = 1;$ $i = 1, 2$
De Jong's Function 5 (Shekel's Foxholes Function)	$F_5(\boldsymbol{x}) = 0.002 + \sum_{j=1}^{25} \frac{1}{f_j};$ $f_j = j + \sum_{i=1}^{2} (x_i - a_{i,j})^6$ Note: See ** for $a_{i,j}$ values where $-65.536 \le x_i \le 65.536, i = 1, 2$	$\min(1/F_5(\boldsymbol{x})) \approx 1$ where $x_i = -32;$ $i = 1, 2$
Rastrigin's Function	$F_6(\boldsymbol{x}) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i))$ where $-5.12 \le x_i \le 5.12; i = 1, \cdots, n$	$\min(F_6(\boldsymbol{x})) = 0$ where $x_i = 0;$ $i = 1, \cdots n$

$$** \, a_{i,j} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32 \end{bmatrix}$$



Q&A