THE BINARY GENETIC ALGORITHM

DR H.K. LAM

Department of Engineering
King's College London
Office S2.14, Strand Building, Strand Campus
Email: hak-keung.lam@kcl.ac.uk
https://nms.kcl.ac.uk/hk.lam

Nature-Inspired Learning Algorithms (7CCSMBIM)

Outline

Learning Aims and Objectives

Aims

- To understand the process of the binary genetic algorithms.
- To apply the binary genetic algorithm to optimisation problems.
- To know the limitations of the binary genetic algorithms.

Objectives

- To study how the binary genetic algorithm works in details.
- To consider a number of applications and formulate as minimisation problems.

Problem and Difficulties

Problem and Difficulties

Problem Statement:

- **Task**: minimisation of a cost function
- Gradient information is not required
- Function evaluation

Difficulties:

- non-convexity
- multi-modality

- non-smoothness
- discontinuity
- dimensionality

Introduction

The Genetic Algorithm

- Genetic algorithm (GA) is a technique to solve problems which need optimisation.
- GA is a subclass of Evolutionary Computing

- GA is based on Charles Darwin's theory of evolution
- History of GA
  - Evolutionary Computing evolved in the 1960's.
  - GA were proposed by John Hollan in the middle of 1970's.

Darwin's Theory of Evolution [1]

- An offspring has many of the characteristics of its parents, which implies that the population is stable.
- There are variations in characteristics between individuals that can be passed from one generation to the next.

- Only a small percentage of the offspring produced survive to adulthood.
- Which of the offspring survive depends on their inherited characteristics.

[1] Sue Ellen Haupt, ValliappaLakshmanan, CarenMarzban, AntonelloPasini, and John K. Williams. Environmental Science Models and Artificial Intelligence. Artificial Intelligence Methods in the Environmental Sciences. Springer Science (3-14, 103-126), 2009.

The Binary Genetic Algorithm

- Biological Metaphor - Natural Selection
  - Genetics and Evolution - gene, chromosome, allele, genotype, phenotype, mitosis, meiosis, gamete, crossover, mutation, $\cdots$

- Components of Binary Genetic Algorithm
  - Variable encoding and decoding, fitness function, population, selection, mating mutation, offspring and convergence, $\cdots$

KING'S College LONDON
University of London

Advantages:

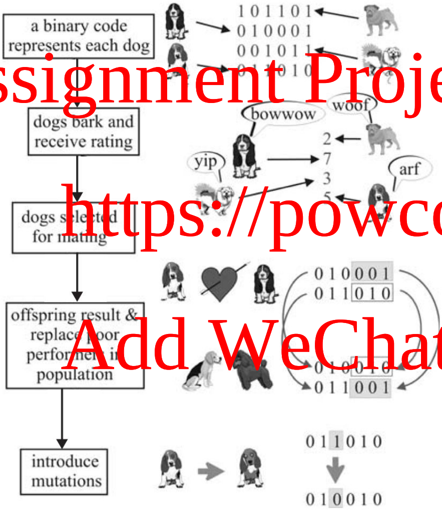
- Optimise with continuous or discrete variables.
- Derivative information is not required.
- Able to deal with a large number of decision variables.
- Optimise decision variables with extremely complex cost function.
- Is less likely trapped in local optimum.
- Tends to search for global optimum.

Notation

- N_{var} : number of decision variables of the chromosome
- N_{bits} : total number of bits of the chromosome
- N_{pop} : population size
- $N_{pop} \times N_{bits}$: total number of bits of the population
- X_{rate} : selection rate in the step of natural selection
- $N_{keep} = N_{pop} \times X_{rate}$: number of chromosomes that are kept for each generation
- $N_{pop} - N_{keep}$: number of chromosomes to be discarded
- x_{lo}: lower bound of variable x
- x_{hi}: upper bound of variable x
- P_n : probability of the n^{th} chromosome in the mating pool of N_{keep} to be chosen
- c_n : cost of the n^{th} chromosome
- C_n : normalised cost of the n^{th} chromosome
- μ : mutation rate (or probability of mutation)

The Binary Genetic Algorithm

KING'S College LONDON
University of London



Initial population with random members

Rating

Selection

Reproduction

Mutation

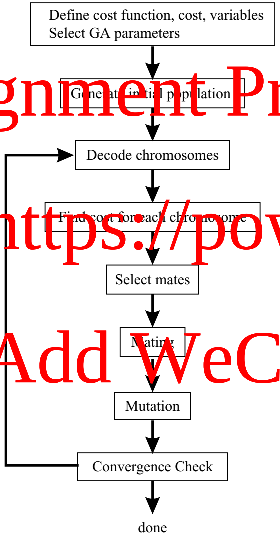
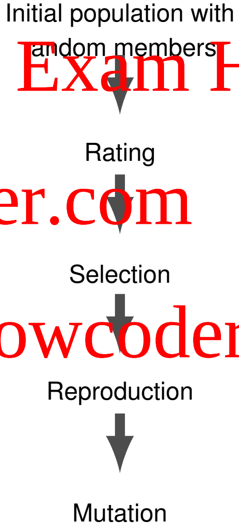


Figure 1: Flowchart of a binary genetic algorithm.



Binary number conversion:

Example: Convert 25.3125 to binary

The integer part: 25

$$25/2 \longrightarrow \boxed{1}$$
$$12/2 \longrightarrow \boxed{0}$$
$$6/2 \longrightarrow \boxed{0}$$
$$3/2 \longrightarrow \boxed{1}$$
$$1/2 \longrightarrow \boxed{1}$$

The fractional part: 0.3125

$$0.3125 \times 2 = 0.625 \longrightarrow \boxed{0}$$
$$0.625 \times 2 = 1.25 \longrightarrow \boxed{1}$$
$$0.25 \times 2 = 0.5 \longrightarrow \boxed{0}$$
$$0.5 \times 2 = 1 \longrightarrow \boxed{1}$$

$$25.3125_{10} = 11001.0101_2$$

Binary to Decimal:

$$(\mathbf{1} \times 2^4 + \mathbf{1} \times 2^3 + \mathbf{0} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0).(\mathbf{0} \times 2^{-1} + \mathbf{1} \times 2^{-2} + \mathbf{0} \times 2^{-3} + \mathbf{1} \times 2^{-4})$$

Given a number $x \in [x_{lo}, x_{hi}]$, how many bits (m) are required to achieve precision of d decimal places?

$$\frac{x_{hi} - x_{lo}}{10^{-d}} \leq 2^m - 1$$

Example: $x \in [25, 100]$, precision: 2 decimal places

$\frac{100-25}{10^{-2}} \leq 2^m - 1 \Rightarrow 7501 \leq 2^m \Rightarrow m = 12.8729 \approx 13 bits$

$0000000000000 \rightarrow 25 + 0 \times \frac{100-25}{2^{13}-1} = 25$

$0000000000001 \rightarrow 25 + 1 \times \frac{100-25}{2^{13}-1} = 25.0092$

$0000000000010 \rightarrow 25 + 2 \times \frac{100-25}{2^{13}-1} = 25.0183$

$$\boxed{\textbf{Decoding: } x = x_{lo} + decimal(1001 \cdots 001_2) \frac{x_{hi} - x_{lo}}{2^m - 1}}$$

- The optimisation/decision variables are represented by *chromosome*.

$$chromosome = [p_1, p_2, \cdots, p_{N_{var}}]$$

- Each gene (p_i, i = 1, 2, $\ldots$, N_{var}) is coded by m_i bits.
- Total number of bits per chromosome: $N_{bits} = \sum_{i=1}^{N_{var}} m_i$
- The cost is evaluated by a cost (fitness) function.

$$cost = f(chromosome) = f(p_1, p_2, \cdots, p_{N_{var}})$$

- Genotype: The bit string representation of the chromosome
- Phenotype: The decision variables. The genotype can be mapped to phenotype through decoding or vice versa through encoding.
- Allele: the value of a single bit in the chromosome

Example: Consider an optimisation problem with decision variables of p_1, p_2, $\cdots$, $p_{N_{var}}$.

- Phenotype: $[p_1, p_2, \cdots, p_{N_{var}}]$
- When each gene is represented by 10 bits,

Chromosome: $\left[\underbrace{1100110011}_{gene_1(p_1)}\underbrace{0111110000}_{gene_2(p_2)}\cdots\underbrace{1111001111}_{gene_{N_{var}}(p_{N_{var}})}\right]$

Genotype: $1100110011\ 0111110000\ \cdots\ 1111001111$

Allele: the allele of the first bit from the left is "1".

Population

- The GA starts with a group of chromosomes known as the population.

- The population has N_{pop}.
- A population represented by a $N_{pop} \times N_{bits}$ matrix filled with random 0s and 1s.

- **Purposes:** Population collects a group of potential solutions, which will be evolved to improve their quality in each generation.

KING'S College LONDON
University of London

Example: A cost function: $cost = f(x, y)$ with 7 bits in each gene.

$$chromosome = \left[\underbrace{1100011}_{x}\underbrace{0011001}_{y}\right]$$

Chromosome	Cost
00101111000110	-12359
11100101100100	-11872
00110010001100	-13477
00101111001000	-12363
11001111111011	-11631
01000101111011	-12097
11101100000001	-12588
01001101110011	-11860

Table 1: Example initial population

- **Two approaches**: X_{rate} and Thresholding.

- **Purposes**: Determine who should survive and who should die. The stronger ones will survive and the weaker ones will die, i.e., “Survival of the fittest” from Darwinian evolutionary theory.

Natural Selection: X_{rate}

Natural Selection using X_{rate}

- Survival of the fittest: Only the best are selected to survive.
- Natural selection occurs each generation or iteration of the algorithm.
- The selection rate, X_{rate}, is the fraction of N_{pop} that survives.

- The number of chromosomes that are kept (for each generation):

  $N_{keep} = X_{rate} N_{pop}$.

- The **top** N_{keep} will be kept in each generation.
- The **bottom** $N_{pop} - N_{keep}$ chromosomes will be discarded and replaced by new offspring.

King's College London
University of London

Example: $N_{pop} = 8$ and $X_{rate} = 50\%$. $N_{keep} = X_{rate}N_{pop} = 0.5 \times 8 = 4$.

Chromosome	Cost
00110010001100	-13477
11101100000001	-12588
00101111001000	-12363
00101111000110	-12359
01000101111011	-12097
11100101100100	-11872
01001101110011	-11860
11001111111011	-11631

Table 2: Ranked population. The upper four will be kept and the lower four will be discarded and replaced.

Natural Selection using Thresholding:

- All chromosomes that have a cost **lower** than a pre-defined threshold survive.
- Chromosomes with a cost **higher** than the threshold will be discarded and replaced.

- If no chromosomes survive, a whole new population will be generated.
- The threshold can be changed in each generation.

Advantage over X_{rate} natural selection:

- Less computationally expensive as population does not have to be sorted.

Selection

Selection:

- **Four approaches:**
  1) Pairing from top to bottom
  2) Random pairing
  3) Weighted random pairing
     - 3.1) Rank weighting
     - 3.2) Cost weighting
  4) Tournament selection

- **Purposes:** Determine who should reproduce offspring.

Selection: Two chromosomes are selected from the mating pool of N_{Keep} chromosomes to produce two new offspring. Selection will take place until $N_{pop} - N_{keep}$ offspring are born to replace the discarded chromosomes.

1) **Pairing from top to bottom:** Start at the top of the list and pair the two chromosomes at a time until the top N_{keep} chromosomes are selected for mating.

   Property: Simple and easy to implement.

2) **Random pairing:** A uniform random number generator to select chromosomes.

   Property: All chromosomes (in the mating pool of N_{Keep}) have chance to mate. Introduce diversity to the population resulting in higher chance of producing offspring of quality.

3) **Weighted Random Paring (roulette wheel weighting):** The probabilities assigned to the chromosomes in the mating pool are inversely proportional to their cost.

- **Property:** A chromosome with the lowest cost has the greatest probability of mating, while the chromosome with the highest cost has the lowest probability of mating.
- **Two techniques:** Rank weighting and cost weighting

King's College London — University of London

3.1) **Rank weighting (roulette wheel weighting):**

$$P_n = \frac{N_{keep} - n + 1}{\sum_{n=1}^{N_{keep}} n}$$

where P_n: probability of the n^{th} chromosome in the mating pool of N_{keep} to be chosen.

Example: $N_{keep} = 4$

n	Chromosome	Cost	P_n	$\sum_{i=1}^{n} P_n$
1	00110010001100	-13477	0.4	0.4
2	11101100000001	-12588	0.3	0.7
3	00101111001000	-12363	0.2	0.9
4	00101111000110	-12359	0.1	1.0

Table 3: Probability table for rank weighting.

3.1) **Rank weighting (roulette wheel weighting):**

Properties:

- It is problem independent.
- Small populations have a high probability of selecting the same chromosome.
- The probabilities only have to be calculated once $\Rightarrow$ less computationally expensive.

3.2) **Cost weighting (roulette wheel weighting):**

Normalised Cost: $C_n = c_n - c_{N_{keep+1}}$

Probability: $P_n = \left| \dfrac{C_n}{\sum_{m=1}^{N_{keep}} C_m} \right|$

Example: $N_{keep} = 4$

n	Chromosome	$C_n = c_n - c_{N_{keep+1}}$	P_n	$\sum_{i=1}^{n} P_n$
1	00110010001100	$-13477 + 12097 = -1380$	0.575	0.575
2	11101100000001	$-12588 + 12097 = -491$	0.205	0.780
3	00101111001000	$-12363 + 12097 = -266$	0.111	0.891
4	00101111000110	$-12359 + 12097 = -262$	0.109	1.000

Table 4: Probability table for cost weighting.

3.2) **Cost weighting (roulette wheel weighting):**

Properties:

- It is cost function dependent.
- It tends to weight the top chromosome more when there is a large spread in the cost between the top and bottom chromosome.
- It tends to weight the chromosomes evenly when all the chromosomes have approximately the same cost.
- The probabilities have to be calculated each generation $\Rightarrow$ computationally expensive.

Remark for Normalisation: Different scaling functions can be used. For example, a more general form of scaling function could be

$$C_n = ac_n + b$$

where a and b are scalars to be chosen that can be constants or functions.

In our example, we choose $a = 1$ and $b = -c_{N_{keep+1}}$ so that

$$C_n = c_n - c_{N_{keep+1}}.$$

4) **Tournament selection:**

- Randomly pick a small subset of chromosomes (two or three) from the mating pool in the N_{keep}, and the chromosome with the lowest cost in the subset becomes a parent.

Properties:

- It is problem independent.
- It works best for larger population sizes because sorting becomes time-consuming for large populations $\Rightarrow$ less computationally expensive.

- Chromosomes of good quality (with lower cost) have higher chance to be chosen.

Crossover:

- **Two approaches:**
  1) Single-point crossover
  2) Double-point crossover
  3) Uniform crossover

- **Purposes:** Create offspring from the parents selected in the selection process (by exchanging information).

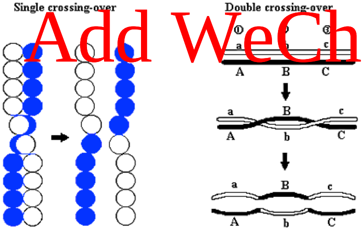
1) **Single-point crossover**:

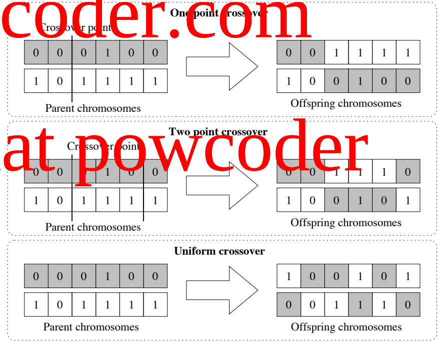
Step 1: A crossover point is randomly selected between the first and last bits of the parents' chromosomes.

Step 2: Generate two offspring by swapping the chromosomes from the crossover point between two parents.

Step 3: Replace any two chromosomes to be discarded in the pool of $N_{pop} - N_{keep}$ in the population.

Step 4: Repeat Steps 1 to 3 for the next two parents until the pool of $N_{pop} - N_{keep}$ is replaced.





1) **Single-point crossover**:

Example: $N_{keep} = 4$

Chromosome	Family	Binary String
3	ma(1)	*00101111001000*
2	pa(1)	11101100000001
5	$offspring_1$	*00101*100000001
6	$offspring_2$	11101*111001000*
3	ma(2)	*00101111001000*
4	pa(2)	00101111000110
7	$offspring_3$	*001011110000*110
8	$offspring_4$	0010111100*1000*

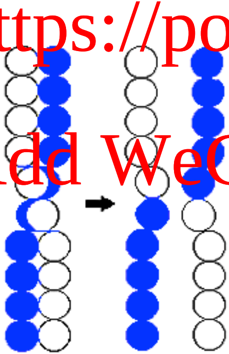
Table 5: Pairing and mating process of single-point crossover.

Crossover

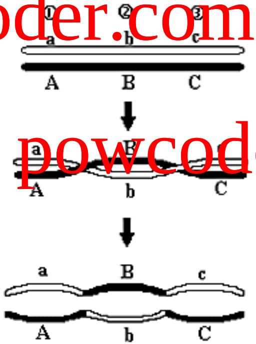
2) **Double-point crossover**: The segments in between two randomly generated crossover points are swapped between parents.

3) **Uniform crossover**: Bits are randomly chosen for swapping between parents.

Single crossing-over



Double crossing-over



Mutations:

- **Purposes:** Random mutations alter a certain percentage of the bits in the list of chromosomes. It allows the GA to explore a cost surface by introducing new information.

Mutation process:

Step 1: Choose the mutation rate (or probability of mutation) μ (0 to 1).

Step 2: Determine the number of bits to be mutated:

$$\#mutation = \mu(N_{pop} - 1)N_{bits} \text{ (elitism is implemented)}$$

Step 3: Flip the chosen bits.

Remark: If elitism is NOT implemented,

$$\#mutation = \mu N_{pop} N_{bits}.$$

- **Mutations**:

Example: $N_{keep} = 4$, $\mu = 0.2$, $\#mutation = 0.2(8-1)14 = 19.6 \approx 20$

Population after Mating	Population after Mutations	New Cost
00110010001100	00110010001100	-13477
11101100000001	11101100000001	-12588
00101111001000	00101111010000	-12415
00101111000110	00001011000111	-13482
00101100000001	00101000000001	-13171
11101111001000	11110111010010	-12146
00101111000110	00100111101000	-12716
00101111001000	00110111001000	-12103

Table 6: Mutating the population.

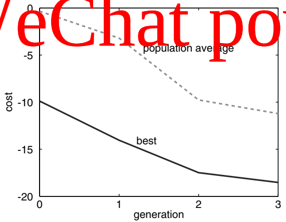
Chromosome	Cost
00001011000111	-13482
00110010001100	-13477
00101000000001	-13171
00100111001000	-12716
11101100000001	-12588
00101111010000	-12415
11110111010010	-12146
00110111001000	-12103

Table 7: New ranked population at the start of the second generation.

KING'S College LONDON
University of London

Stopping Criteria:

- Whether an acceptance solution is reached.
- A set number of iterations is exceeded.
- No changes on the chromosomes.
- No changes on the cost.
- Population statistics on mean and minimum cost.



Convergence

Question: Find the maximum percentage of the possible solution being searched after 3 generations

Total number of possible solutions: $128 \times 128 = 16384$

maximum number of solutions searched at the 3rd generation:

$$\underbrace{8}_{\text{initial population}} + \underbrace{7}_{\substack{\text{max cost evaluations per generation} \\ \text{(elitism is implemented)}}} \times \underbrace{3}_{\text{generations}} = 29$$

$\frac{29}{128 \times 128} \times 100 = 0.18\%$ of the solution space has been searched.

Performance Evaluation

Statistics

- mean, standard deviation, the worst and the best of the cost of multiple runs

Convergence rate

Benchmark functions

- Functions of different properties

Function 1:

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2, -5.12 \le x_i \le 5.12$$

minimum: $\mathbf{x}^* = 0, f_1(\mathbf{x}^*) = 0$

Function 2:

$$f_2(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right), -2.048 \le x_i \le 2.048$$

minimum: $\mathbf{x}^* = 0$, $f_2(\mathbf{x}^*) = 0$

Function 3:

$$f_3(\mathbf{x}) = 6n + \sum_{i=1}^{n} floor(x_i), -5.12 \le x_i \le 5.12$$

minimum: $x_i^* = [5, 5.12]$, $f_3(\mathbf{x}^*) = 0$

KING'S College LONDON
University of London

Function 4:

$$f_4(\mathbf{x}) = \sum_{i=1}^{i} i x_i^4 + Gauss(0,1), \; -1.28 \le x_i \le 1.28$$

minimum: $\mathbf{x}^* = 0, f_4(\mathbf{x}^*) = 0$

Function 5:

$$f_5(\mathbf{x}) = \frac{1}{k} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}, -65.356 \le x_i \le 65.356$$

maximum: $x_i^* = -32, f_5(\mathbf{x}^*) \approx 1$ with $k = 500$

$$a_{ij} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & 16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 \\ 32 & 32 & 32 & 32 & 32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 \end{bmatrix}$$

Function 6:

$$f_6(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10), -5.12 \le x_i \le 5.12$$

minimum: $\mathbf{x}^* = 0, f_6(\mathbf{x}^*) = 0$

Binary GA Example by Hand

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \le x,y,z \le 5$. Each variable is represented by 2 bits.

Chromosome: $[x,y,z]$

After encoding: $\left[\underbrace{XX}_{x}\underbrace{XX}_{y}\underbrace{XX}_{z}\right]$

Encoding\Decoding $\{2,3,4,5\} \leftrightarrow \{00,01,10,11\}$

Decoding: $x = x_{lo} + decimal(1001\cdots001_2)\frac{x_{hi}-x_{lo}}{2^m-1}$

With $x_{lo} = 2$, $x_{hi} = 5$ and $m = 2$,

$00 \rightarrow 2 + decimal(00) \times \frac{5-2}{2^2-1} = 2$

$01 \rightarrow 2 + decimal(01) \times \frac{5-2}{2^2-1} = 3$

$10 \rightarrow 2 + decimal(10) \times \frac{5-2}{2^2-1} = 4$

$11 \rightarrow 2 + decimal(11) \times \frac{5-2}{2^2-1} = 5$

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \leq x, y, z \leq 5$. Each variable is represented by 2 bits.

Step 1: Population initialised with population size = 4

n	Chromosome	Decoded x, y, z	Cost
1	101110	4, 5, 4	-24
2	000011	2, 2, 5	9
3	001100	2, 5, 2	-12
4	110100	5, 3, 2	-19

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \leq x, y, z \leq 5$. Each variable is represented by 2 bits.

Step 2: Ranked population and natural selection with $N_{keep} = 3$

n	Chromosome	Decoded x, y, z	Cost
1	101110	4, 5, 4	-24
2	110100	5, 3, 2	-19
3	001100	2, 5, 2	-12
4	000011	2, 2, 5	9

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \le x,y,z \le 5$. Each variable is represented by 2 bits.

Step 3: Selection with cost weighting (roulette wheel weighting)

n	Chromosome	Decoded x, y, z	Cost	$C_n = c_n - c_{N_{keep+1}}$	P_n	$\sum^{n}_{=} P_n$
1	101110	4, 5, 4	-24	$-24-9=-33$	0.4024	0.4024
2	110100	5, 3, 2	-19	$-19-9=-28$	0.3415	0.7439
3	001100	2, 5, 2	-12	$-12-9=-21$	0.2561	1.0000
4	000011	2, 2, 5	9			

$$P_n = \left|\frac{C_n}{\sum_{m=1}^{N_{keep}} C_m}\right| = \left|\frac{C_n}{\sum_{m=1}^{3} C_m}\right| = \left|\frac{C_n}{-33-28-21}\right| = \left|\frac{C_n}{-82}\right|$$

$$P_1 = \left|\frac{-33}{-82}\right|, P_2 = \left|\frac{-28}{-82}\right|; P_3 = \left|\frac{-21}{-82}\right|$$

- Generate two random numbers: 0.9649, 0.2785

What happen if the same chromosome is chosen?

Binary GA example by hand

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \le x, y, z \le 5$. Each variable is represented by 2 bits

Step 4: Crossover with single-point crossover technique

p_3: 001100

p_1: 101110

• Generate randomly a crossover point: 2

offspring$_1$: 001110

offspring$_2$: 101100

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \leq x,y,z \leq 5$. Each variable is represented by 2 bits.

Step 5: Mutation

Ranked population for mutation with $\mu = 0.2$;

$\#mutation = 0.2(4-1)6 = 3.6 \approx 4$

row = [2 2 3 4]; column = [4 5 2 5].

n	Chromosome	Chromosome after mutation	Decoded x, y, z	Cost
1	101110	101110	4, 5, 4	-24
2	110100	110010	5, 2, 4	-3
3	001100	011100	3, 5, 2	-21
4	001110	001100	2, 5, 2	-12

Binary GA example by hand

Example: Consider a function $f(x,y,z) = x - 2xy + 3z$ to be minimised, where $2 \leq x, y, z \leq 5$. Each variable is represented by 2 bits.

- The 1^{st} generation is done.
- **Ranked population for next generation.**

n	Chromosome	Decoded x, y, z	Cost
1	101110	4, 5, 4	-24
2	011100	3, 5, 2	-21
3	001100	2, 5, 2	-12
4	110010	5, 2, 4	-3

- Repeat steps 1 to 5 until stopping criteria have been met.

Why do GAs work? Schema Theorem

- **Binary** genetic algorithms are considered.

Schema Theorem (Michalewicz, 1992)

"Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm."

- **A scheme:** a template representing a subset of binary strings using symbols '0', '1' and $\star$ (don't care symbol).

Example:

Schema $\star$1100 matches 2 strings: {**0**1100, **1**1100}.

Schema $\star$110$\star$ matches 4 strings: {**0**110**0**, **0**110**1**, **1**110**0**, **1**110**1**}.

Schema $\star\star\star\star\star$ matches 2^5 strings: {00000, 00001, $\cdots$, 11110, 11111}.

Schema 11100 matches 1 string {11100}.

String 11100 is matched by 2^5 schemata: {11100, $\star$1100, 1$\star$100, $\cdots$, $\star\star\star\star 0$, $\star\star\star\star\star$}.

King's College London
University of London

Schema properties:

1. **Order** of the schema S (denoted by $o(S)$): the number of fixed positions ('0' and '1' positions), i.e., the length of the template minus the number of don't care symbols.

2. **Defining length** of the schema S (denoted by $\delta(S)$): the distance between the first and the last fixed positions.

Example:

$S_1 = \star\star\star 001 \star 110 \Rightarrow o(S_1) = 6, \delta(S_1) = 10 - 4 = 6$

$S_2 = \star\star\star\star 00 \star\star 0 \star \Rightarrow o(S_2) = 3, \delta(S_2) = 9 - 5 = 4$

$S_3 = 11101 \star\star 001 \Rightarrow o(S_3) = 8, \delta(S_3) = 10 - 1 = 9$

Why do GAs work? Schema Theorem

Binary GAs:

1. select $P(t)$ based on $P(t-1)$
2. recombine $P(t)$
3. evaluate $P(t)$
4. $t \leftarrow t+1$

Binary GA configurations:

- $N_{keep} = 0$.
- Selection: roulette wheel weighting, **cost weighting**.
- Crossover: single-point crossover (crossover points are allowed any point in between the first and last bits).
- Mutation: uniform mutation (with the probability of mutation p_m for each bit).
- Elitism is not implemented.

$\xi(S,t)$: the number of strings in the population at generation t matched by a schema S.

Example: Consider the following population (**costs are normalised** and thus of all negative):

$$
\begin{aligned}
v_1 &= 100101 \qquad f(v_1) = -15 \\
v_2 &= 011100 \qquad f(v_2) = -22 \\
v_3 &= 110111 \qquad f(v_3) = -6 \\
v_4 &= 000110 \qquad f(v_4) = -1 \\
v_5 &= 110001 \qquad f(v_5) = -8 \\
v_6 &= 110011 \qquad f(v_6) = -3
\end{aligned}
$$

$S_1 = \star\star 01 \star\star, \qquad \xi(S_1,t) = 3$

$f(S,t)$: average cost of all strings in the population matched by the schema S.

Assuming that there are p strings $\{u_1, \cdots, u_p\}$ in the population matched by a schema S at generation t,

$$f(S,t) = \frac{\sum_{i=1}^{p} f(u_i)}{p}$$

Example:

$S_1 = \star\star01\star\star$ is matched by $\{v_1, v_3, v_4\} = \{100101, 110111, 000110\}$

$f(S_1,t) = \frac{f(v_1)+f(v_3)+f(v_4)}{3} = \frac{-15-6-1}{3} = -7.3333$

KING'S College LONDON
University of London

Objective: Investigate the probability of survival of all schemata S $(S_1, \cdots, S_{2^{N_{bits}}})$ in the GA process (selection, crossover, mutation).

Selection: The probability of selecting the string v_i:

$$p_i = \frac{f(v_i)}{F(t)}, i = 1, \cdots, N_{pop}$$

where $F(t) = \sum_{j=1}^{N_{pop}} f(v_j)$.

1. The number of strings matched by schema S: $\xi(S,t)$.
2. The average probability of a string matched by schema S to be selected: $\frac{f(S,t)}{F(t)}$.
3. The number of strings to be selected for recombination: N_{pop} ($N_{keep} = 0$).

King's College London, University of London

$\xi(S, t+1)$: the number of strings matched by S after the selection process.

$$\begin{aligned} \xi(S, t+1) &= \xi(S, t)\frac{f(S,t)}{F(t)}N_{pop} \\ &= \xi(S,t)\frac{f(S,t)}{\overline{F}(t)} \end{aligned}$$

where $\overline{F}(t) = \frac{F(t)}{N_{pop}}$ is the average cost of the population.

$$\begin{aligned} \xi(S, t+1) &= \xi(S, t)\frac{f(S,t)}{\overline{F}(t)} \\ &= \xi(S,t)\frac{\overline{F}(t) + f(S,t) - \overline{F}(t)}{\overline{F}(t)} \\ &= \xi(S,t)(1+\varepsilon(t)) \end{aligned}$$

where $\varepsilon(t) = \frac{f(S,t)) - \overline{F}(t)}{\overline{F}(t)}$.

Why do GAs work? Schema Theorem

KING'S College LONDON
University of London

$$
\begin{aligned}
\xi(S,t+1) &= \xi(S,t)(1+\varepsilon(t)) \\
&= \underbrace{\xi(S,t-1)(1+\varepsilon(t-1))}_{\xi(S,t)}(1+\varepsilon(t)) \\
&= \underbrace{\xi(S,t-2)(1+\varepsilon(t-2))}_{\xi(S,t-1)}(1+\varepsilon(t-1))(1+\varepsilon(t)) \\
&= \underbrace{\xi(S,t-3)(1+\varepsilon(t-3))}_{\xi(S,t-2)}(1+\varepsilon(t-2))(1+\varepsilon(t-1))(1+\varepsilon(t)) \\
&\ \ \vdots \\
&= \xi(S,1)(1+\varepsilon(1))(1+\varepsilon(2))\cdots(1+\varepsilon(t-1))(1+\varepsilon(t))
\end{aligned}
$$

- $\varepsilon(t) > 0$ for most of t: $\xi(S,t+1)$ is increasing
- $\varepsilon(t) < 0$ for most of t: $\xi(S,t+1)$ is decreasing

Implication: Above average schemata receive increasing number of strings in the next generation; however, below average schemata will die out as t increases.

Recombination - Crossover

Probability of destruction of a schema S: $p_d(S) = \frac{\delta(S)}{N_{bits}-1}$

Probability of schema S survival: $p_s(S) = 1 - \frac{\delta(S)}{N_{bits}-1}$

Example

A string $v_1 = 1101110010$ is matched by 2^{10} schemata.

$S_1 = 1101110010$ $\quad p_d(S_1) = \frac{9}{9} = 1, p_s(S_1) = 0$

$\vdots$

$S_a = \star\star\star111\star\star\star\star$ $\quad p_d(S_a) = \frac{2}{9}, p_s(S_a) = \frac{7}{9}$

$\vdots$

$S_b = 11\star\star\star\star\star\star10$ $\quad p_d(S_b) = \frac{9}{9} = 1, p_s(S_b) = 0$

$\vdots$

$S_{2^{N_{bits}}} = \star\star\star\star\star\star\star\star\star\star$ $\quad p_d(S_{2^{N_{bits}}}) = \frac{0}{9} = 0, p_s(S_{2^{N_{bits}}}) = 1$

KING'S College LONDON
University of London

$v'_1 = \mathbf{1101110010}$

$v'_2 = 0100000011$

Example 1: After crossover

$o^c_1 = \mathbf{110111}|0011$

$o^c_2 = 010000|\mathbf{0010}$

S_a survives but not S_b.

Example 2: After crossover

$o^c_1 = \mathbf{110}|0000011$

$o^c_2 = 010|\mathbf{1110010}$

S_a survives but not S_b.

Example 3: After crossover

$o^c_1 = \mathbf{1101}|000011$

$o^c_2 = 0100|\mathbf{110010}$

Both S_a and S_b cannot survive.

Example 4: After crossover

$o^c_1 = \mathbf{11011}|00011$

$o^c_2 = 01000|\mathbf{10010}$

Both S_a and S_b cannot survive.

KING'S College LONDON
University of London

$$p_s(S) = 1 - \frac{\delta(S)}{N_{bits} - 1}$$

Modification of $p_s(S)$ considering that, e.g., the schema of v_1' and v_2' is the same

$$p_s(S) \geq 1 - \frac{\delta(S)}{N_{bits} - 1}$$

Schema growth equation with the consideration of crossover:

$$\xi(S, t+1) \geq \xi(S, t) \frac{f(S, t)}{\overline{F}(t)} \left(1 - \frac{\delta(S)}{N_{bits} - 1}\right)$$

Mutation: Uniform mutation - each bit will be mutated if a random number $r < p_m$, where $p_m \in [0,1]$ is the probability of mutation.

Probability of a single bit survival (no mutation takes place):

$$1 - p_m$$

Probability of a schema S survival (no mutation takes place in fixed bits):

$$p_s(S) = (1 - p_m)^{o(S)}$$

Schema growth equation with the consideration of crossover and mutation:

$$\xi(S, t+1) \geq \xi(S, t)\frac{f(S, t)}{\overline{F}(t)}\left(1 - \frac{\delta(S)}{N_{bits} - 1}\right)(1 - p_m)^{o(S)}$$

Schema Theorem (Michalewicz, 1992)

"Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm."