Chapter 2
Genetic Algorithms

- What are genetic algorithms?
- Why use genetic algorithms?
- Fundamentals of genetic algorithms
- Applications

What Are Genetic Algorithms?

- GAs are stochastic search techniques based on the mechanism of natural selection and natural genetics to imitate living beings for solving those "difficult" problems with high complexity and/or undesirable structure.

- **Earliest Predecessors:**

1. Fraser, A., "Simulation of Genetic Systems by Automatic Digital Computers: Part I - Introduction", Australian Journal of Biological Science, Vol. 10, pp. 481-491, 1957.

2. Fraser, A., "Part II - Effects of Linkage on Rates of Advance under Selection", Australian Journal of Biological Science, Vol. 10, pp. 492-499, 1957.

- **Creation of the Field:**

1. Holland, J., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.

2. De Jong, K., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", Ph.D. Dissertation, University of Michigan, Ann Arbor, 1975.

General Structure of Genetic Algorithms

Procedure: Genetic Algorithms

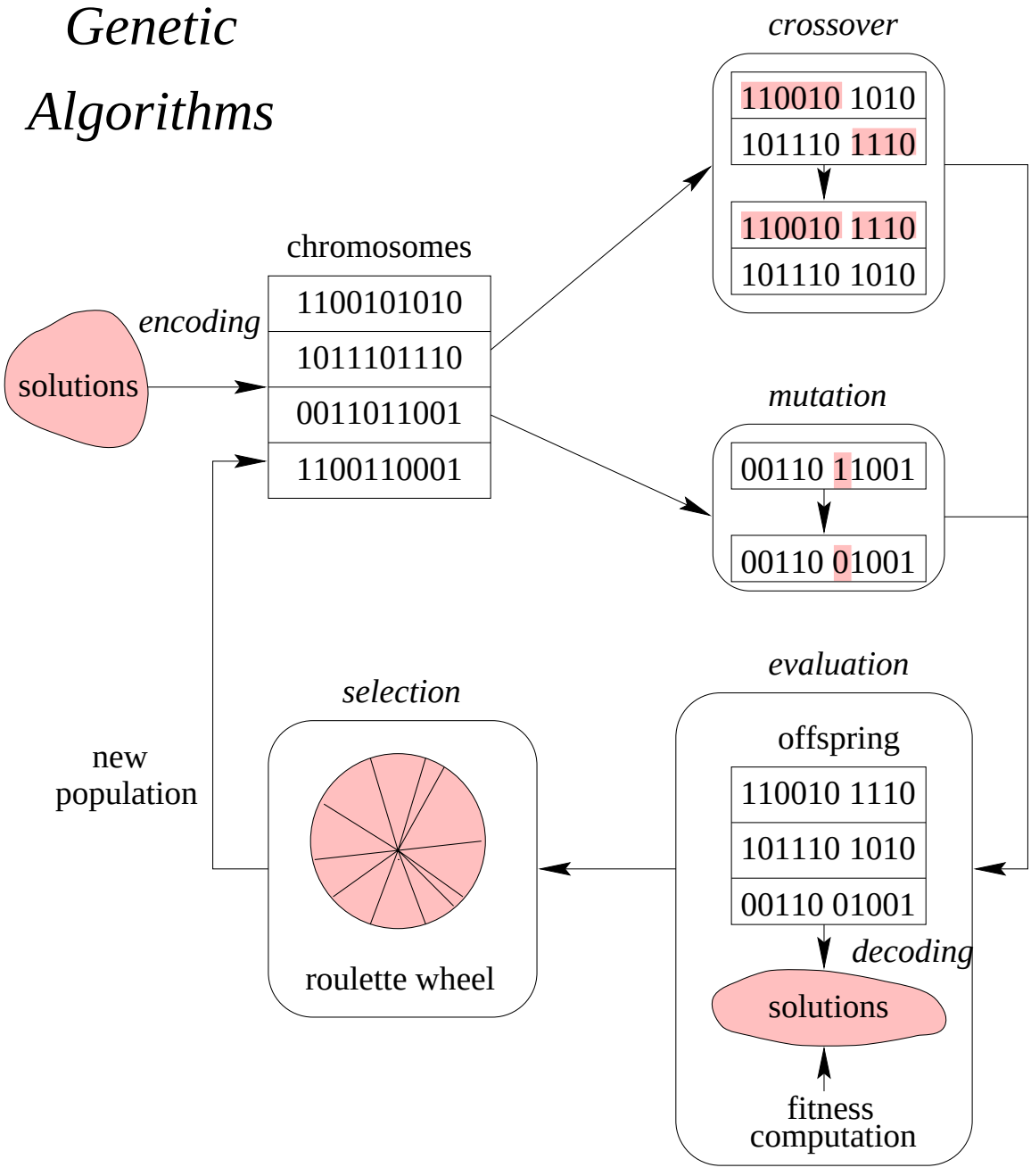
begin

$t \leftarrow 0$;
initialize $P(t)$;
evaluate $P(t)$;
while (not termination condition) **do**
recombine $P(t)$ to yield $C(t)$;
evaluate $C(t)$;
select $P(t+1)$ from $P(t)$ and $C(t)$;
$t \leftarrow t+1$;
end

end

Recombination: to yield offspring

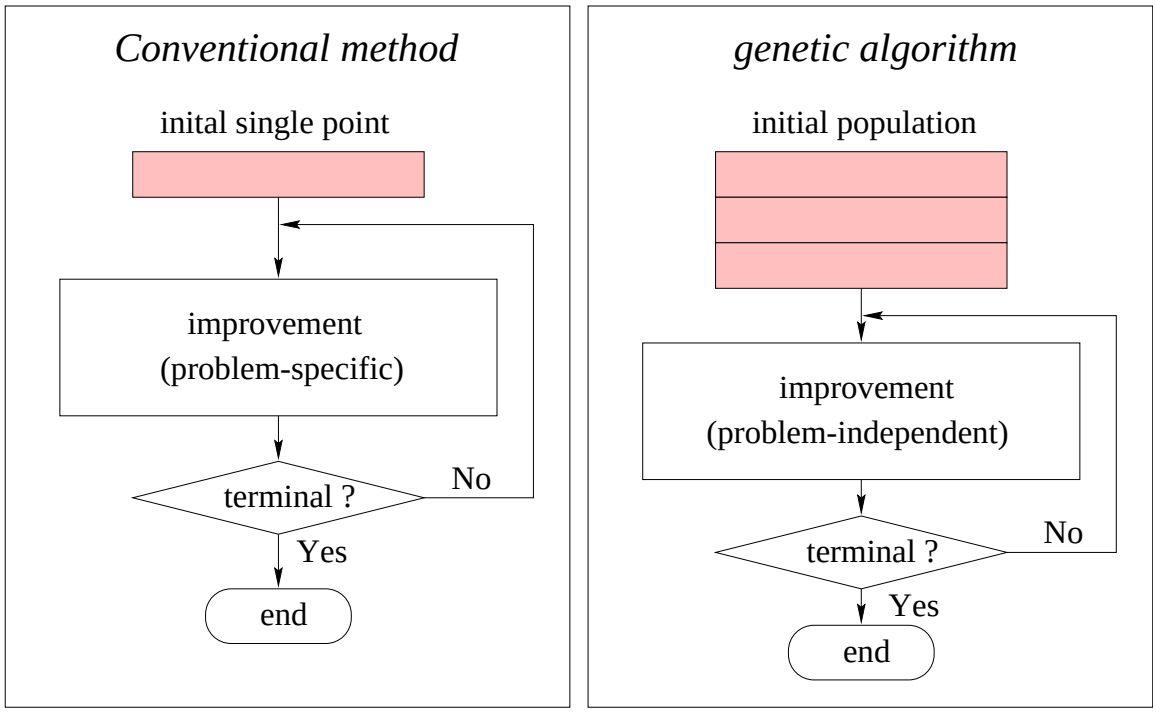
1. *Genetic operations*: crossover and mutation
2. *Evolution operations*: selection


Genetic
Algorithms
crossover
110010 1010
101110 1110
110010 1110
101110 1010
chromosomes
encoding
solutions
1100101010
1011101110
0011011001
1100110001
mutation
00110 11001
00110 01001
evaluation
selection
offspring
new
population
110010 1110
101110 1010
00110 01001
decoding
solutions
roulette wheel
fitness
computation

Terminology of GAs

Genetic Algorithms	Explanation
Chromosome (string, individual)	Solution (coding)
Genes (bits)	Part of solution
Locus	Position of gene
Alleles	Values of gene
Phenotype	Decoded solution
Genotype	Encoded solution

GAs vs. Conventional Approach



- **Nature of GAs**

1. Genetic algorithms work with a coding of solution set, not the solutions themselves.

2. Genetic algorithms search from a population of solutions, not a single solution.

3. Genetic algorithms use payoff information (fitness function), not derivatives or other auxiliary knowledge.

4. Genetic algorithms use probability transition rules, not deterministic rules.

5. Genetic algorithms exploit the best solution while exploring the search space.

- **Major Advantages of GAs**

1. Genetic algorithms do not have much mathematical requirements about the optimization problems.

2. The ergodicity of evolution operators makes genetic algorithms very effective at performing global search (in probability).

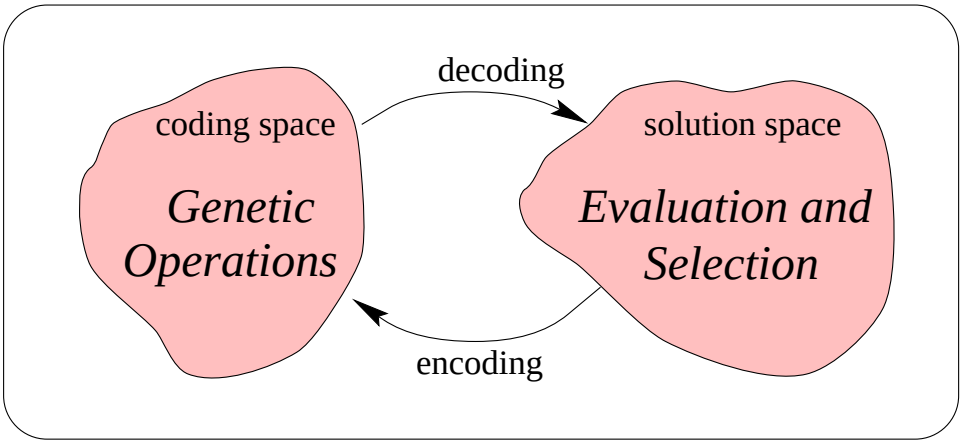
3. Genetic algorithms provide us a great flexibility to hybridize with domain-dependent heuristics to make an efficient implementation for a specific problem.

- **Key Components of GAs**

1. Encoding / Decoding
2. Crossover / Mutation
3. Selection

Encoding / Decoding

- How to encode a solution of the problem into a chromosome is a key to success.



- **Issues**:

  1. The feasibility of a chromosome
  2. The legality of a chromosome
  3. The uniqueness of mapping

- **Coding Schemes**

1. Binary coding
2. Integer coding
3. Real number coding
4. Other specially designed codings

Examples

1. Optimization Problems

$$\max f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$$

$$-3.0 \leq x_1 \leq 12.1$$

$$4.1 \leq x_2 \leq 5.8$$

(Binary Coding)
Let the domain of variable x_j is $[a_j, b_j]$ and the required precision is five places after the decimal point. The precision requirement implies that the range of domain of each variable should be divided into at least $(b_j - a_j) \times 10^5$ size ranges. The required bits (denoted with m_j) for a variable is calculated as follows:

$$2^{m_j - 1} < (b_j - a_j) \times 10^5 \leq 2^{m_j} - 1$$

The mapping from a binary string to a real number for variable x_j is straight forward and completed as follows:

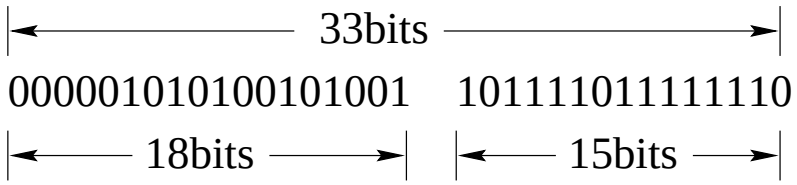
$$x_j = a_j + decimal(substring_j) \times \frac{b_j - a_j}{2^{m_j} - 1}$$

where $decimal(substring_j)$ represents the decimal value of $substring_j$ for decision variable x_j.

In this example,

$$
\begin{aligned}
&(12.1 - (-3.0)) \times 10,000 = 151,000 \\
&2^{17} < 151,000 \leq 2^{18}, \qquad m_1 = 18 \\
&(5.8 - 4.1) \times 10,000 = 17,000 \\
&2^{14} < 17,000 \leq 2^{15}, \qquad m_2 = 15 \\
&m = m_1 + m_2 = 18 + 15 = 33
\end{aligned}
$$

The total length of a chromosome is 33 bits which can be represented as follows:



The corresponding values for variables x_1 and x_2 are given below:

	Binary Number	Decimal Number
x_1	000001010100101001	5417
x_2	101111011111110	24318

$$
\begin{aligned}
x_1 &= -3.0 + 5417 \times \tfrac{12.1-(-3.0)}{2^{18}-1} = -2.687969 \\
x_2 &= 4.1 + 24318 \times \tfrac{5.8-4.1}{2^{15}-1} = 5.361653
\end{aligned}
$$

2. Word-Matching Problem

- The *word-matching problem* tries to evolve an expression of "to be or not to be" from the randomly-generated lists of letters.

- Since there are 26 possible letters for each of 13 locations in the list, the probability that we get the correct phrase in a pure random way is $(1/26)^{13} = 4.03038 \times 10^{-19}$, which is about two chances out of a billion.

(Integer Coding)

- We use a list of ASCII integers to encode the string of letters.

- The lowercase letters in ASCII are represented by numbers in the range [97, 122] in the decimal number system.

- Fox example, the string of letters *tobeornottobe* becomes

$$[116, 111, 98, 101, 111, 114, 110, 111, 116, 116, 111, 98, 101]$$

Genetic Operations: Crossover/Mutation

- The genetic operations mimic the process of heredity of genes to create new offspring at each generation.

- Crossover operates on two chromosomes at a time and generates offspring by combining both chromosomes' features.

- The *crossover rate* (denoted by p_c) is defined as the ratio of the number of offspring produced in each generation to the population size (usually denoted by pop_size).

- A higher crossover rate allows exploration of more of the solution space at the cost of computations.

- Effectiveness and feasibility issues (repairing work).

- Mutation is a background operator which produces spontaneous random changes in various chromosomes.

  (a) replacing the genes lost from the population during the selection process so that they can be tried in a new context.

  (b) or providing the genes that were not present in the initial population.

- The *mutation rate* (denoted by p_m) is defined as the percentage of the total number of genes in the population.

  (a) If it is too low, many genes that would have been useful are never tried out;

  (b) If it is too high, there will be much random perturbation to learn from the history of the search.

- Effectiveness and feasibility issues.

Evolution Operation: Selection

- The principle behind genetic algorithms is essentially Darwinian natural selection. It directs a genetic algorithm search toward promising regions in the search space.

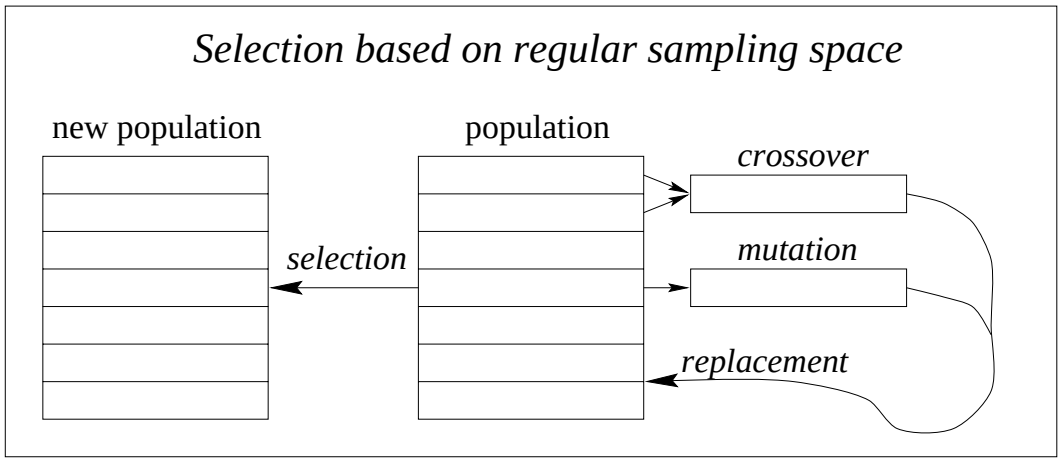
- Basic Issues:

  1. Sampling space
  2. Sampling mechanism
  3. Selection probability

1. Sampling Space

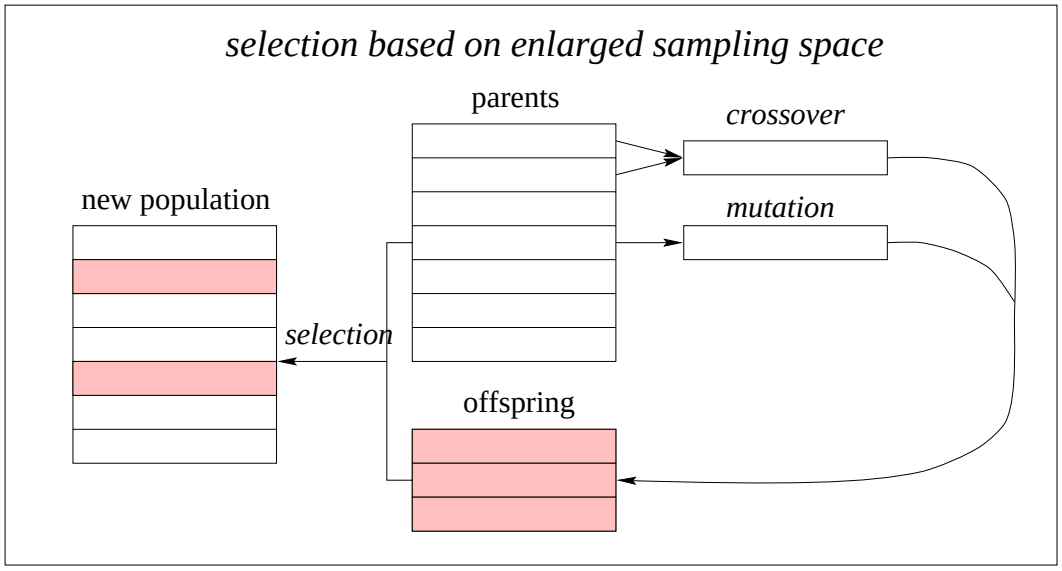
(a) Regular Sampling Space:

Parents are replaced by their offspring soon after they give birth.



(b) Enlarged Sampling Space:

Both parents and offspring have the same chance of competing for survival.


selection based on enlarged sampling space
parents
crossover
new population
mutation
selection
offspring

2. Sampling Mechanism

- Sampling mechanism concerns the problem of how to select chromosomes from sampling space.

- Basic approaches:

  (a) Stochastic sampling

  (b) Deterministic sampling

  (c) Mixed sampling

(a) Stochastic Sampling

- The selection phase determines the actual number of copies that each chromosome will receive based on its survival probability.

- Roulette Wheel Selection

  f_k: fitness of chromosome k
  p_k: selection probability

$$p_k = \frac{f_k}{\sum_{j=1}^{pop\_size} f_j}$$

- Then we can make a wheel according to these probabilities. The selection process is based on spinning the roulette wheel *pop_size* times. Each time, we select a single chromosome for the new population.

- Prohibition of duplicated chromosomes

(b) Deterministic Sampling

- This approach usually selects the best *pop_size* chromosomes from the sampling space.

- Truncation Selection:

  T% best chromosomes are selected and each one received nearly 100/T copies.

- Block Selection

(c) Mixed Sampling

- This approach contains both random and deterministic features simutaneously.

- Tournament Selection:

  This method randomly chooses a set of chromosomes and picks out the best one from the set for reproduction.

- Stochastic Tournament Selection:

  In this method, selection probabilities are calculated normally and successive pairs of chromosomes are drawn using roulette wheel selection. After drawing a pair, the chromosome with higher fitness is inserted in the new population. The process continues until the population is full.

3. Selection Probability

- Scaling and Ranking:

  - Scaling method maps raw objective function values to some positive real values, and the survival probability for each chromosome is determined according to these values.

  - Ranking method ignores the actual objective function values and uses a ranking of chromosomes instead to determine survival probability.

- Fitness Scaling

$$f'_k = g_k(f_k)$$

  where $g(\bullet)$ transforms the raw fitness into scaled fitness.

  – To maintain a reasonable differential between relative fitness ratings of chromosomes.

  – To prevent a too-rapid takeover by some super chromosomes in order to meet the requirement to limit competition early on, but to stimulate it later.

Examples

1. Linear Scaling

$$f'_k = a \times f_k + b$$

2. Sigma Truncation

$$f'_k = f_k - (\bar{f} - c \times \sigma)$$

3. Power Scaling

$$f'_k = f_k^{\alpha}$$

4. Logarithmic Scaling

$$f'_k = b - \log(f_k)$$

5. Boltzmann Scaling

$$f'_k = e^{f_k/T}$$

- **Ranking Selection:**
  Sort the population from the best to the worst and assign the selection probability of each chromosome according to the ranking but not its raw fitness.

- **Linear Ranking:**

$$p_k = \text{selection probability for the } k\text{-}th \text{ ranked chromosome}$$
$$= q - (k-1) \times \gamma$$

where q = probability for the best one

$$\gamma = \frac{q - q_0}{pop\_size - 1}$$

where q_0 = probability for the worst one

- **Exponential Ranking:**

$$p_k = q(1-q)^{k-1}$$

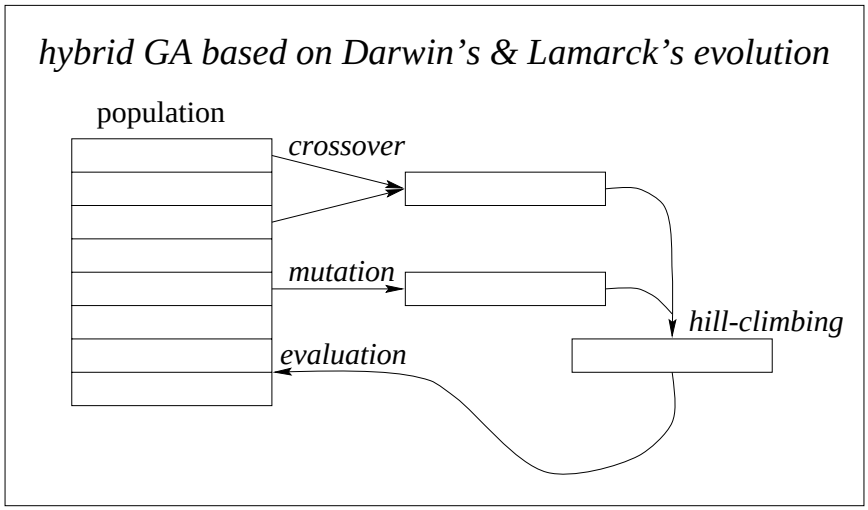
or

$$p_k = q^{k-1}$$

Hybrid Genetic Algorithms

- GAs + local optimization heuristics
- Lamarckian Evolution:

  Environmental changes throughout an organism's life cause structural changes that are transmitted to offspring.



Procedure: Hybrid Genetic Algorithms

begin

$t \leftarrow 0$;

initialize $P(t)$;

evaluate $P(t)$;

while (not termination condition) **do**

recombine $P(t)$ to yield $C(t)$;

locally climb $C(t)$;

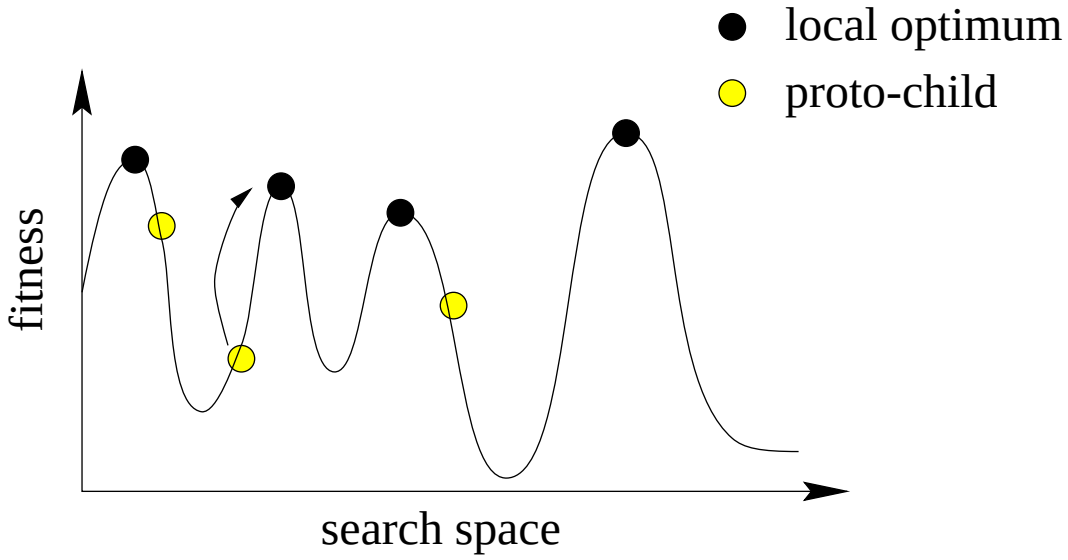
select $P(t+1)$ from $P(t)$ and $C(t)$;

$t \leftarrow t+1$;

end

end

- **Memetic Algorithms**



Application of GAs in ATM

1. Topology design / Network configuration
2. Dynamic routing / Multi-casting
3. Bandwidth allocation / Virtual paths
4. QoS allocation
5. Connection admission control
6. Buffer management
7. Cell multiplexing

GAs for Multicasting

- **Minimum Spanning Tree Problem:**

$$G = (V, E)$$

$$V = \{v_1, v_2, \cdots, v_n\}$$

$$E = \{e_{ij} \mid e_{ij} = (v_i, v_j)\}$$

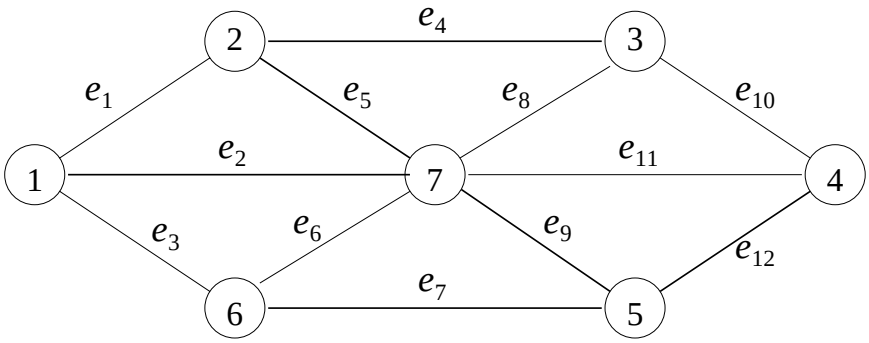
$$W = \{w_{ij} \mid w_{ij} = w(e_{ij}) > 0\}$$

A spanning tree is a minimal set of edges from E that connects all the vertices in V and therefore at least one spanning tree can be found in graph G. The minimum spanning tree, denoted as T^*, is the spanning tree whose total weight of all edges is minimal, i.e.,

$$T^* = \min_{T} \sum_{e_{ij} \in E \cap T} w_{ij}$$

- **Encoding**

(1) Edge encoding



e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
0	1	0	1	1	0	1	0	1	0	0	1

<u>Problems:</u>

- Representation space is too large.

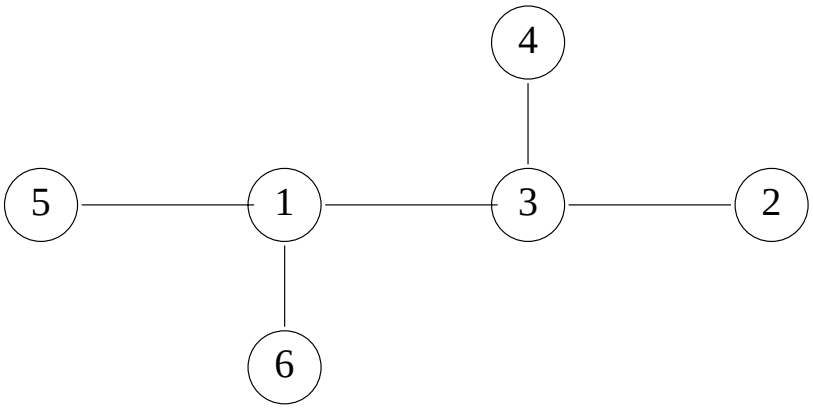
- Probability of having a tree is too low.

(2) Prüfer number encoding:

Cayley's Theorem: There are n^{n-2} distinct labeled tree on a complete graph with n vertices.

(Prüfer established a one-to-one correspondence between such trees and the set of all strings of $n-2$ digits.)

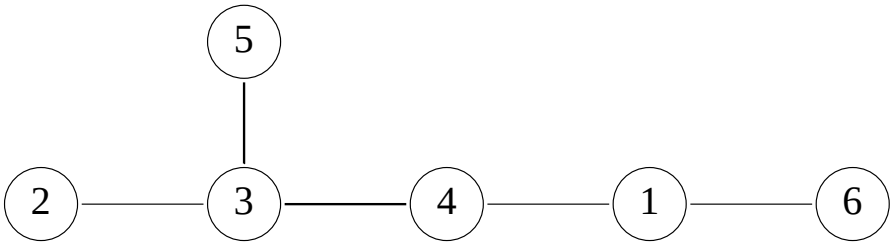
Examples



Tree

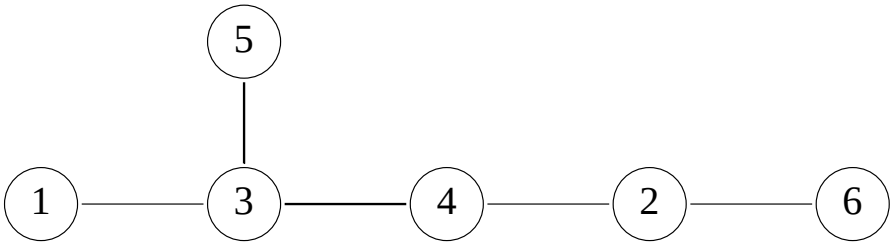
3	3	1	1

Prufer number



3	3	4	1

Prufer number



3	3	4	2

Prufer number

Procedure: Encoding

Step 1. Let vertex i be the smallest labeled leaf vertex in a labeled tree T.

Step 2. Let j be the first digit in the encoding as the vertex j incident to vertex i is uniquely determined. Here we build the encoding by appending digits to the right, and thus the encoding is built and read from left to right.

Step 3. Remove vertex i and the edge from i to j; thus we have a tree with $n - 1$ vertices.

Step 4. Repeat the above steps until one edge is left. We produce a Prüfer number or an encoding with $n - 2$ digits between 1 and n inclusive.

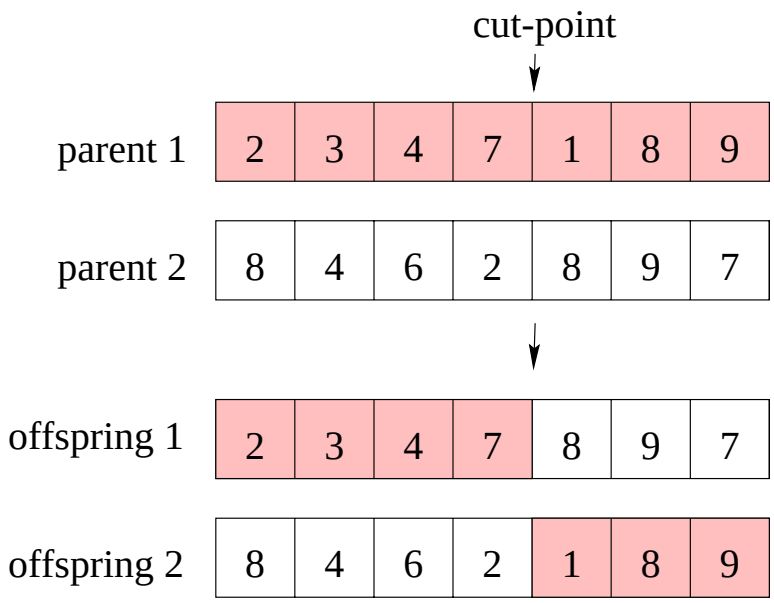
Procedure: Decoding

Step 1. Let P be the original Prüfer number and let $\bar{P}$ be the set of all vertices not included in P. P designates vertices eligible for consideration in building a tree.

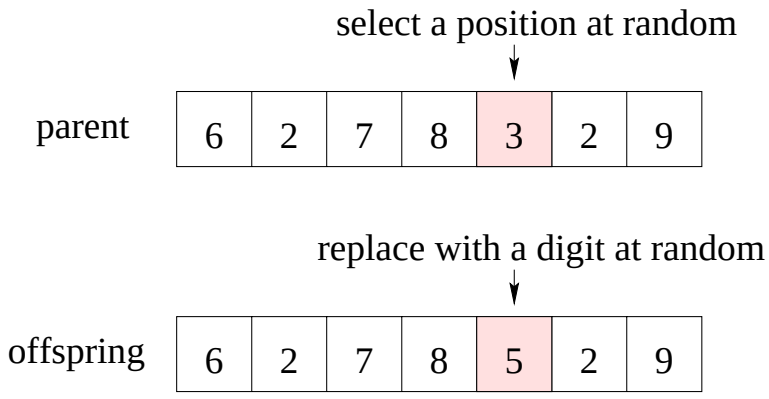
Step 2. Let i be the eligible vertex in $\bar{P}$ with the smallest label. Let j be the leftmost digit of P. Add the edge from i to j into the tree. Remove i from $\bar{P}$ and j from P. If j does not occur anywhere in P, put it into $\bar{P}$. Repeat the process until no digits are left in P.

Step 3. If no digits remain in P, there are exactly two vertices, r and s, in $\bar{P}$ and thus still eligible for consideration. Add edge from r to s into the tree to form a tree with $n-1$ edges.

- **Crossover**



- **Mutation**



- **Evaluation**

  Step 1. Convert a chromosome into a tree.

  Step 2. Calculate the total weight of the tree.

  - Let P be a chromosome, and let $\bar{P}$ be the set of eligible vertices.

Procedure: Evaluation

begin
$T \leftarrow \{\phi\}$;
$eval(T) \leftarrow 0$;
define $\bar{P}$ according to P;
repeat
select the leftmost digit from P, say i;
select the eligible vertex with the smallest label
from $\bar{P}$, say j;
$T \leftarrow T \cup \{e_{ij}\}$;
$eval(T) \leftarrow eval(T) + w_{ij}$;
remove i from P;
remove j from $\bar{P}$;
if i does not occur anywhere in remaining P **then**
put i into P;
end
$k \leftarrow k + 1$;
until $k \leq n - 2$;
$T \leftarrow T \cup \{e_{rs}\}, r, s \in \bar{P}$;
$eval(T) \leftarrow eval(T) + w_{rs}$;
end

- **Selection**

  - a mixed strategy with $(\mu + \lambda)$ selection and *roulette wheel* selection is used.

Procedure: Selection

begin

select μ' best different chromosomes;

if $\mu' < \mu$ **then**

select $\mu - \mu'$ chromosomes by *roulette wheel* selection;

end

end

- **Algorithm**

Procedure: Genetic Algorithm for dc-MST

begin
    $t \leftarrow 0$;
    initialize $P(t)$;
    evaluate $P(t)$;
    while (not termination condition) **do**
        recombine $P(t)$ to yield $C(t)$;
        evaluate $C(t)$;
        select $P(t+1)$ from $P(t)$ and $C(t)$;
        $t \leftarrow t+1$;
    end
end

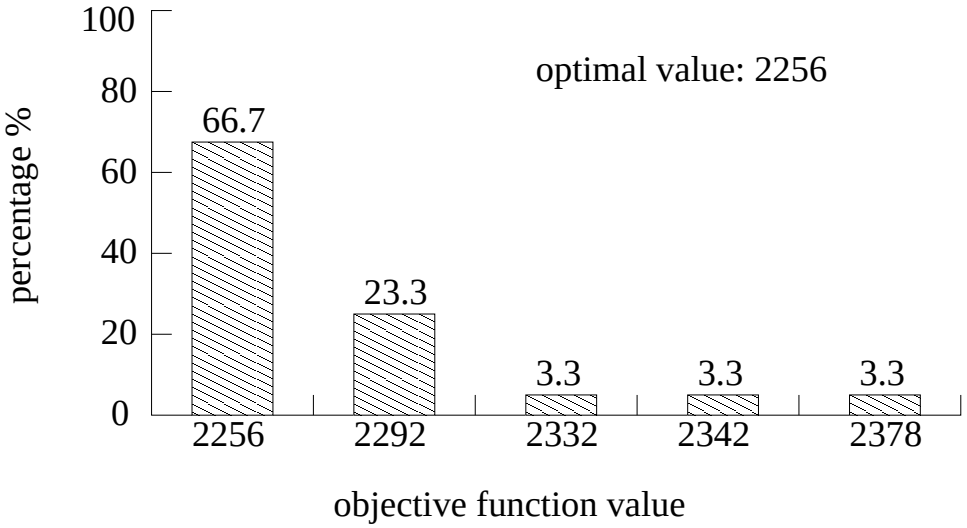
Parameters:

population size $pop\_size = 50$;

crossover probability $p_c = 0.5$;

mutation probability $p_m = 0.01$;

maximum generation $max\_gen = 500$;



- **References**

  http://www.ie.ncsu.edu/fangroup/ga.html