Ανάλυση Κοινωνικών Δικτύων (Social Network Analysis)

Γενετικοί Αλγόριθμοι & Εφαρμογές στην Ανάλυση Κοινωνικών Δικτύων

Συμεών Παπαβασιλείου (papavass@mail.ntua.gr)
Βασίλειος Καρυώτης (vassilis@netmode.ntua.gr)

17Ιανουαρίου, 2017

Topics

1. **Genetic Algorithms**
    1. Definition
    2. Description
    3. Examples
    4. Analysis

2. **Applications of Genetic Algorithms in Social Network Analysis**
    1. Application of Genetic Algorithms in Computing Shortest Paths
    2. Application of Genetic Algorithms in Community Detection

Genetic Algorithms (1)

Definition: *Genetic* Algorithms are **global adaptive heuristic search and optimization techniques** modeled from natural selection, genetic and evolution

-Introduced in 1970 by James Holland

- Part of evolutionary computing; Darwin's theory for the survival of the fittest. It mimics natural evolution process for producing better results

- Intelligent exploitation of a random search used to solve optimization problems
  - Optimization problem: objective function, unknown variables, constraints

- Competition over individuals that results in the case that the fittest individuals dominate over the weaker ones

Genetic Algorithms (2)

A typical genetic algorithm requires:

1. a **genetic representation** of the solution domain (consisting of individuals or **chromosomes**)
   - usually binary code applied.

2. a **fitness function** to evaluate the solution domain (i.e. each individual)

3. definition **of selection, crossover and mutation operators** since new off springs are generated /evolved from the chromosomes using them

4. terminating condition of the iterations of the algorithm

Representation - Chromosome

- Population of chromosomes of size *m*
- Each member of the population (chromosome) is a string of length *n*
  - *binary code representation*: $S = \{0, 1\}^n$
  - other representation forms will be examined/applied in the sequel
- Chromosome: candidate solution
- Generation of the population at time *t*:

$$\mathcal{B}_t = (b_{1,t}, b_{2,t}, \ldots, b_{m,t}).$$

- First generation: chosen randomly
- New generation (time *t+1*): after selection, mutation, crossover at the population at time *t*

Fitness Function

- Assigns a "*figure of merit* " for each chromosome (How close the solutions are to required solution)
- Choose an appropriate fitness function for each problem (more in examples that follow)
- Chromosomes with higher fitness value are chosen for next generation
- After each round of testing we delete the 'n' worst solutions, and to breed 'n' new ones from the best design solutions using operators like mutation and crossover

Selection

- Component that guides the algorithm to find the solution by preferring individuals with high fitness over low-fitted ones
- Usually includes randomness
- Usually applied schema:
  - Probability to choose a certain individual is proportional to its fitness
  - It can be regarded as a random experiment with

$$P[b_{j,t} \text{ is selected}] = \frac{f(b_{j,t})}{\sum_{k=1}^{m} f(b_{k,t})}.$$

Positive fitness function f

  - Non-decreasing transformation $\varphi : \mathbb{R} \to \mathbb{R}^+$

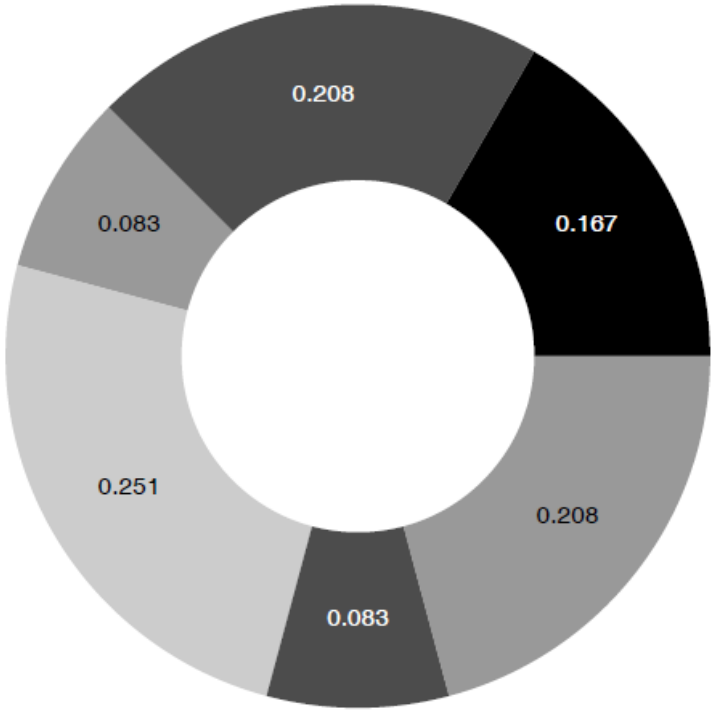
$$P[b_{j,t} \text{ is selected}] = \frac{\varphi(f(b_{j,t}))}{\sum_{k=1}^{m} \varphi(f(b_{k,t}))}$$

For non-positive fitness function f

Elitism: the best solutions carry on to the next generation

Selection – Algorithmic Description

- Generalized roulette game: in a roulette game, the slots are not equally wide, i.e. the different outcomes can occur with different probabilities.



Select an individual – chromosome as in the scheme of the previous slide.

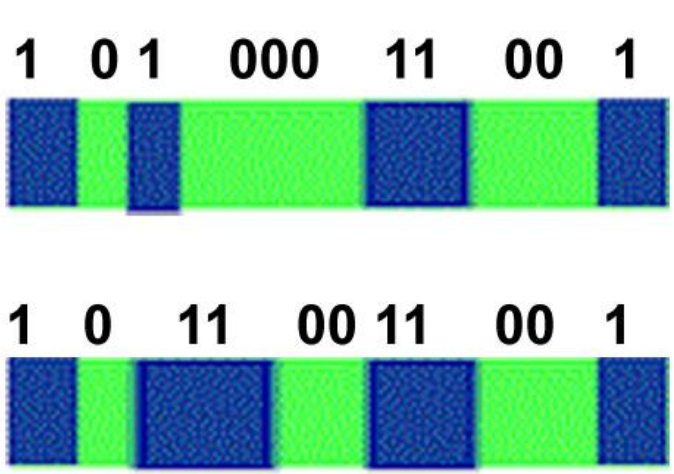
$x := \text{Random}[0, 1];$
$i := 1$

WHILE $i < m \;\&\; x < \sum_{j=1}^{i} f(b_{j,t}) / \sum_{j=1}^{m} f(b_{j,t})$ **DO**
  $i := i + 1;$

select $b_{i,t}$;

Mutation

- **Mutation** is an operator used to maintain genetic diversity from one generation of a population of chromosomes to the next generation
- Implemented through **Bit Flipping**
- **Real reproduction:** probability that a certain gene is mutated is almost equal for all genes
- Mutation Probability p_M



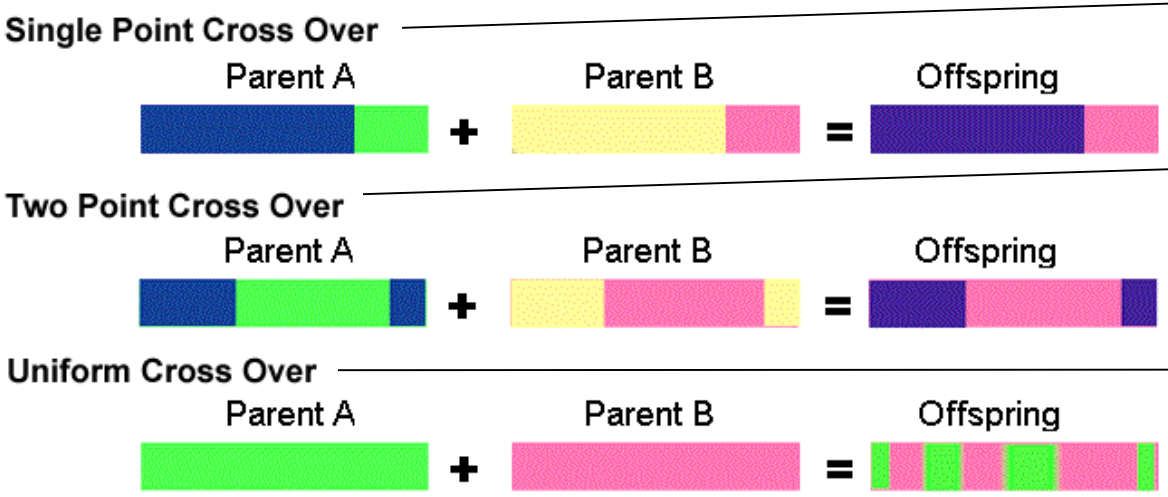
Fitness(decimal value)

163

179

Crossover

- Crossover is a process of taking more than one parent solutions and producing a child solution from them. Exchanges genes from chromosomes between parents
- Biological property that good parents produce well-performing children or even better
- **Three Steps**
    1. Selects a pair of two individuals
    2. A cross site is selected at random along the string length
    3. String bits after the crossover site are exchanged
- Crossover Probability: p_c
- Types:



Swap after the cross site

Every second section is swapped

Decide randomly for each position if it will be swapped

Description of the Algorithm

1. **[Start]** Generate random population of *m* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating the following steps until the new population is complete
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome)
    4. **[Accepting]** Place new offspring in a new population
    5. **[Replace]** Use new generated population for a further run of algorithm
4. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
5. **[Loop] Go to step 2**

Algorithm (1)

$t := 0;$
Create initial population $\mathcal{B}_0 = (b_{1,0}, \ldots, b_{m,0});$

Initial Generation

WHILE *stopping condition not fulfilled* **DO**
BEGIN

(∗ proportional selection ∗)

Selection – chromosomes may be selected more than once

FOR $i := 1$ **TO** m **DO**
BEGIN
$x :=$ Random$[0, 1];$

$k := 1;$
WHILE $k < m$ & $x < \sum_{j=1}^{k} f(b_{j,t}) / \sum_{j=1}^{m} f(b_{j,t})$ **DO**
$k := k + 1;$

$b_{i,t+1} := b_{k,t}$
END

Algorithm (2)

```
(* one-point crossover *)

    FOR i := 1 TO m − 1 STEP 2 DO
    BEGIN
        IF Random[0, 1] ≤ p_C THEN
        BEGIN
            pos := Random{1, . . . , n − 1};

            FOR k := pos + 1 TO n DO
            BEGIN
                aux := b_{i,t+1}[k];
                b_{i,t+1}[k] := b_{i+1,t+1}[k];
                b_{i+1,t+1}[k] := aux
            END
        END
    END

(* mutation *)

    FOR i := 1 TO m DO
        FOR k := 1 TO n DO
            IF Random[0, 1] < p_M THEN
                invert b_{i,t+1}[k];

    t := t + 1
END
```

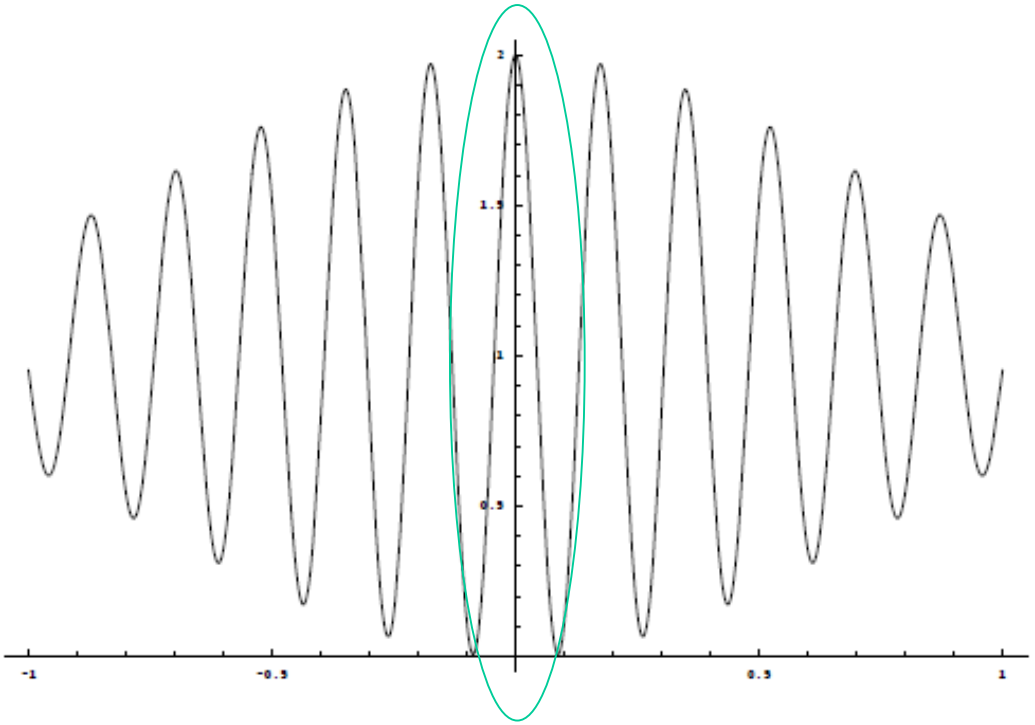
Crossover

Mutation

Example (1)

Finding the global maximum of the function:

$$\begin{aligned} f_2 : [-1,1] &\longrightarrow \mathbb{R} \\ x &\longmapsto 1 + e^{-x^2} \cdot \cos(36x) \end{aligned}$$



Example (2)

Create a uniform grid of 2^n bits

$$c_{n,[a,b]}: \quad [a,b] \longrightarrow \{0,1\}^n$$
$$x \longmapsto \mathrm{bin}_n\left(\mathrm{round}\left((2^n-1)\cdot\tfrac{x-a}{b-a}\right)\right)$$

Converts a number to its binary representation of length *n*.

Decoding function: $\tilde{c}_{n,[a,b]}: \quad \{0,1\}^n \longrightarrow [a,b]$

$$s \longmapsto a + \mathrm{bin}_n^{-1}(s)\cdot\tfrac{b-a}{2^n-1}.$$

- not bijective since information is lost due to the rounding operation

Applying the above coding scheme to the interval $[-1,1]$ with $n=16$, we get a maximum accuracy of the solution of

$$\frac{1}{2}\cdot\frac{2}{2^{16}-1} \approx 1.52\cdot 10^{-5}.$$

Example (3)

$m = 6$, $p_C = 1$, and $p_M = 0.005$.

```
Generation   1 max. fitness 1.9836 at -0.0050
#0        0111111101010001 fitness: 1.98
#1        1101111100101011 fitness: 0.96
#2        0111111101011011 fitness: 1.98
#3        1001011000011110 fitness: 1.97
#4        1001101100101011 fitness: 1.20
#5        1100111110011110 fitness: 0.37
Average Fitness: 1.41

...

Generation  52 max. fitness 2.0000 at 0.0000
#0        0111111101111011 fitness: 1.99
#1        0111111101111011 fitness: 1.99
#2        0111111101111011 fitness: 1.99
#3        0111111111111111 fitness: 2.00
#4        0111111101111011 fitness: 1.99
#5        0111111101111011 fitness: 1.99
Average Fitness: 1.99
```

$52 \times 6 = 312$

evaluations of the fitness function

$2^{16} = 65536$

Size of the search space

Decoding the best string

$$-1 + 32767 \cdot \frac{1 - (-1)}{65535} = -1 + 0.9999847 = -0.0000153.$$

Analysis of Genetic Algorithms

- This is not a hard analysis, i.e. provides only information on the expected behavior of the algorithm and not on its strict convergence properties
- Inner structure of genetic algorithms ➡
  - complicated as the transition of one generation to the next consists of probabilistic operations (three)
  - there are several variants of these probabilistic operations

Schema Definition

Definition. A string $H = (h_1, \ldots, h_n)$ over the alphabet $\{0, 1, *\}$ is called a (binary) *schema* of length n. An $h_i \neq *$ is called a *specification* of H, an $h_i = *$ is called *wildcard*.

Example: schema **11 represents strings 1111, 0011, 1011 and 0111. Alternatively, 1011 and 1101 contain common schemata 1***, ***1 and 1**1.

In general, for a string of length l, there are 3^l schemata.

The Schema Theorem (1)

1. A string $S = (s_1, \ldots, s_n)$ over the alphabet $\{0, 1\}$ *fulfills* the schema $H = (h_1, \ldots, h_n)$ if and only if it matches H is all non-wildcard positions:

$$\forall i \in \{j \mid h_j \neq *\} : \; s_i = h_i$$

According to the discussion above, we write $S \in H$.

2. The *number of specifications* of a schema H is called *order* and denoted as

$$\mathcal{O}(H) = |\{i \in \{1, \ldots, n\} | h_i \neq *\}|.$$

3. The distance between the first and the last specification

$$\delta(H) = \max\{i | h_i \neq *\} - \min\{i | h_i \neq *\}$$

is called the *defining length* of a schema H.

The Schema Theorem (2)

1. The number of individuals which fulfill H at time step t are denoted as

$$r_{H,t} = |\mathcal{B}_t \cap H| .$$

2. The expression $\bar{f}(t)$ refers to the observed average fitness at time t:

$$\bar{f}(t) = \frac{1}{m} \sum_{i=1}^{m} f(b_{i,t})$$

3. The term $\bar{f}(H,t)$ stands for the observed average fitness of schema H in time step t:

$$\bar{f}(H,t) = \frac{1}{r_{H,t}} \sum_{i \in \{j | b_{j,t} \in H\}} f(b_{i,t})$$

The Schema Theorem (3)

Theorem (Schema Theorem—Holland 1975). *Assuming we consider a genetic algorithm of type 2.5, the following inequality holds for every schema H:*

$$\mathsf{E}[r_{H,t+1}] \geq r_{H,t} \cdot \frac{\bar{f}(H,t)}{\bar{f}(t)} \cdot \left(1 - p_{\mathbf{C}} \cdot \frac{\delta(H)}{n-1}\right) \cdot (1 - p_{\mathbf{M}})^{\mathcal{O}(H)}$$

Comments:

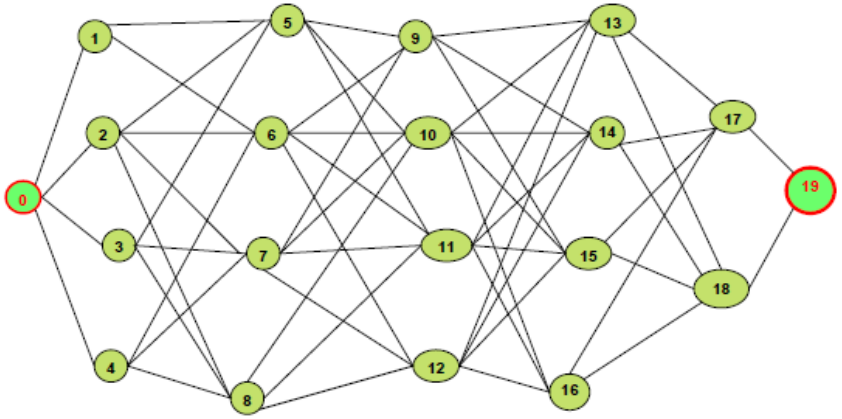
- Above average fitness, short, low-order schema will have a large survival probability. They will grow at least exponentially in the population

- **Building Block Hypothesis**

Short, highly fit, low order schemata are called building blocks; it is thought that they represent partial solutions to the problem and that processing them will build up the full solution

Application in Finding Shortest Paths (1)

- **20** Nodes
- **62** Links



- We chose **20** chromosomes as initial population through random selection. A chromosome represents a path!
- **Genes** are the node numbers through which candidate path passes through (not binary here!)

0	2	3	7	9	11	12	17	18	20
0	4	5	7	10	11	13	15	16	20

Application in Finding Shortest Paths (2)

Cost Matrix

(contains cost of each link)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	10000	52	61	8	16	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
1	52	10000	10000	10000	10000	78	41	6	92	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
2	61	10000	10000	10000	10000	84	63	2	99	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
3	8	10000	10000	10000	10000	71	48	223	73	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
4	16	10000	10000	10000	10000	63	55	44	88	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
5	10000	78	84	71	63	10000	10000	10000	10000	11	22	33	44	10000	10000	10000	10000	10000	10000	10000
6	10000	41	63	48	55	10000	10000	10000	10000	21	32	43	54	10000	10000	10000	10000	10000	10000	10000
7	10000	6	2	223	44	10000	10000	10000	10000	74	85	96	14	10000	10000	10000	10000	10000	10000	10000
8	10000	92	99	73	88	10000	10000	10000	10000	46	64	75	35	10000	10000	10000	10000	10000	10000	10000
9	10000	10000	10000	10000	10000	11	21	74	46	10000	10000	10000	10000	66	55	44	11	10000	10000	10000
10	10000	10000	10000	10000	10000	22	32	85	64	10000	10000	10000	10000	91	97	73	19	10000	10000	10000
11	10000	10000	10000	10000	10000	33	43	96	75	10000	10000	10000	10000	45	65	25	85	10000	10000	10000
12	10000	10000	10000	10000	10000	44	54	14	35	10000	10000	10000	10000	73	37	87	16	10000	10000	10000
13	10000	10000	10000	10000	10000	10000	10000	10000	10000	66	91	45	73	10000	10000	10000	10000	86	84	10000
14	10000	10000	10000	10000	10000	10000	10000	10000	10000	55	97	65	37	10000	10000	10000	10000	74	76	10000
15	10000	10000	10000	10000	10000	10000	10000	10000	10000	44	73	25	87	10000	10000	10000	10000	2	6	10000
16	10000	10000	10000	10000	10000	10000	10000	10000	10000	11	19	85	16	10000	10000	10000	10000	7	9	10000
17	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	86	74	2	7	10000	10000	52
18	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	84	76	6	9	10000	10000	25
19	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000	52	25	10000

Application in Finding Shortest Paths (3)

Fitness Function

- Sum of weights between nodes (i.e. of links) of paths are taken as the fitness of the chromosome
- So the candidate chromosomes (paths) with lowest value (i.e. shortest paths) are the most fit ones

$$F = \begin{cases} \dfrac{1}{\sum_{i=1}^{N-1} C_i\,(g_i, g_{i+1})} \\ \\ 0 \end{cases}$$

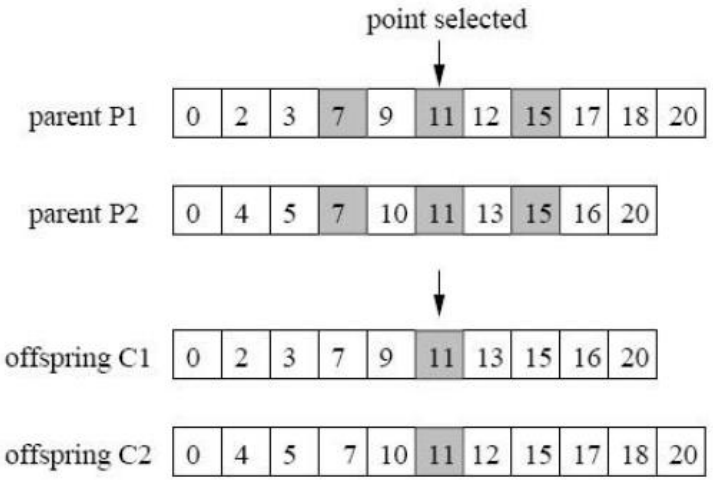
Cost matrix elements

Sequential nodes on the path

Application in Finding Shortest Paths (4)

Crossover

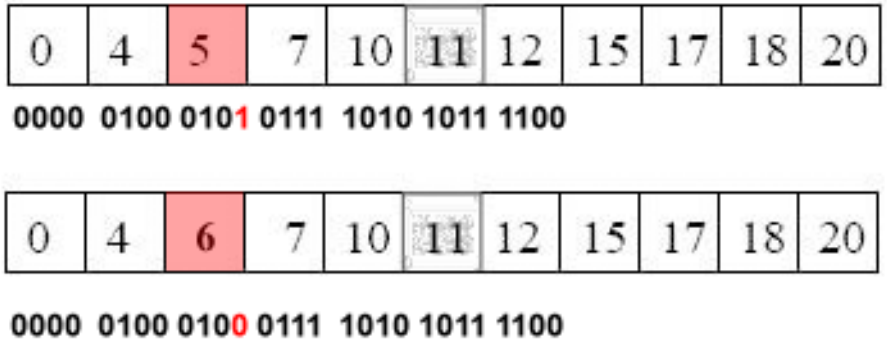
- The crossover function takes two parents to mate
- It looks for the common points in the parents
- The common nodes are where these two paths intersect
- **Crossover Probability=0.99**



Application in Finding Shortest Paths (5)

Mutation

- A very low mutation probability is chosen since flipping of many bits may lead to wrong paths
- **Mutation Probability =0.01**
- **Correction Function** to validate paths after mutation



Terminating Condition

- **No change** in population fitness and stall generation (Convergence)
- Reached **N** generations
- Chromosome has attained some **predefined fitness level**

Application in Finding Shortest Paths (6)

```
Individual-0   Fitness: 222  path: 0 - 1 - 6 - 12 - 16 - 17 - 19 -
Individual-1   Fitness: 1025 path: 0 - 4 - 5 - 9 - 15 - 17 - 16 - 12 - 14 - 11 - 8 - 1 - 6 - 3 - 7 - 10 - 13 - 18 - 19
Individual-2   Fitness: 231  path: 0 - 3 - 5 - 11 - 16 - 18 - 19 -
Individual-3   Fitness: 879  path: 0 - 1 - 8 - 12 - 14 - 17 - 13 - 11 - 6 - 2 - 7 - 3 - 5 - 9 - 16 - 18 - 19 -
Individual-4   Fitness: 728  path: 0 - 3 - 7 - 11 - 13 - 10 - 5 - 2 - 6 - 9 - 15 - 18 - 19 -
Individual-5   Fitness: 574  path: 0 - 2 - 7 - 9 - 13 - 10 - 8 - 12 - 14 - 18 - 16 - 17 - 19 -
Individual-6   Fitness: 390  path: 0 - 3 - 5 - 1 - 6 - 12 - 14 - 18 - 19 -
Individual-7   Fitness: 901  path: 0 - 4 - 6 - 1 - 5 - 12 - 16 - 10 - 15 - 17 - 14 - 11 - 7 - 2 - 8 - 9 - 13 - 18 - 19
Individual-8   Fitness: 755  path: 0 - 3 - 6 - 1 - 5 - 9 - 13 - 12 - 7 - 4 - 8 - 11 - 14 - 18 - 16 - 17 - 19 -
Individual-9   Fitness: 466  path: 0 - 2 - 5 - 9 - 8 - 10 - 13 - 18 - 19 -
Individual-10  Fitness: 364  path: 0 - 4 - 8 - 12 - 13 - 18 - 16 - 17 - 19 -
Individual-11  Fitness: 210  path: 0 - 1 - 7 - 12 - 14 - 18 - 19 -
Individual-12  Fitness: 380  path: 0 - 3 - 7 - 9 - 15 - 18 - 19 -
Individual-13  Fitness: 281  path: 0 - 2 - 8 - 9 - 15 - 18 - 19 -
Individual-14  Fitness: 792  path: 0 - 2 - 8 - 11 - 15 - 18 - 14 - 10 - 6 - 1 - 5 - 12 - 7 - 9 - 16 - 17 - 19 -
Individual-15  Fitness: 531  path: 0 - 4 - 7 - 1 - 8 - 11 - 6 - 10 - 14 - 17 - 19 -
Individual-16  Fitness: 954  path: 0 - 2 - 8 - 4 - 5 - 12 - 7 - 11 - 13 - 18 - 16 - 9 - 15 - 10 - 14 - 17 - 19 -
Individual-17  Fitness: 700  path: 0 - 3 - 7 - 12 - 6 - 11 - 13 - 18 - 14 - 9 - 15 - 17 - 19 -
Individual-18  Fitness: 243  path: 0 - 3 - 6 - 11 - 16 - 17 - 19 -
Individual-20  Fitness: 219  path: 0 - 1 - 5 - 11 - 15 - 18 - 19 -
```

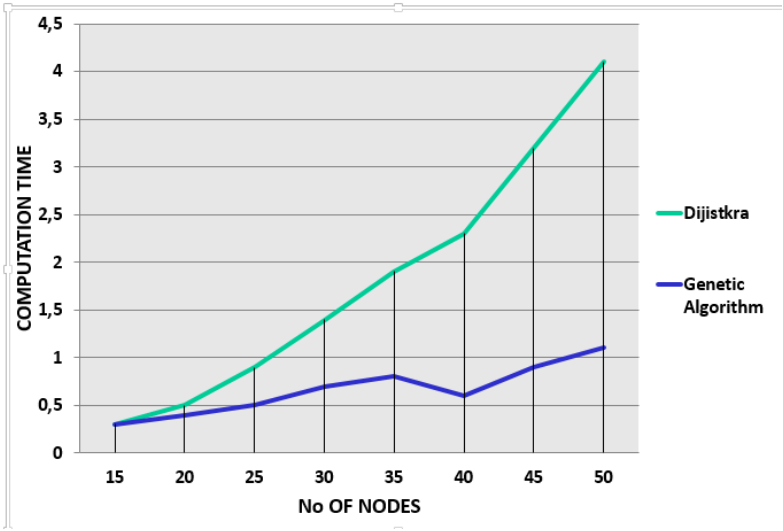
First 20 Individuals in Initial Population

Fitness Value ranged from 147 to 1138 , Average value 687

50 Such generations were produced.

Application in Finding Shortest Paths (7)

- After few generations the average fitness of Individuals reached an optimum value of **137**
- Since then the paths were repeatedly modified by crossover and mutation



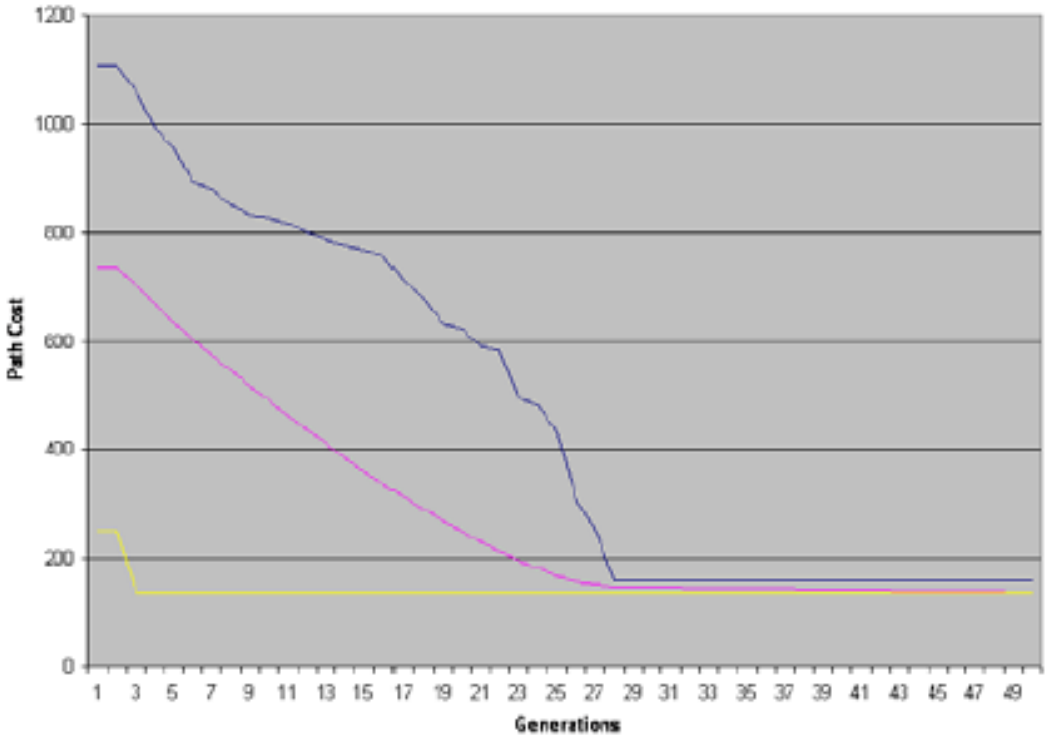


Fig. 6: Average of 30 runs for 50 generations

Application in Community Detection (1)

Social Network: *G=(V,E)*
Represented by an Adjacency Matrix: $A=[a_{ij}]$

Problem:
Find communities in the social network ⟶
Find a partitioning of *A* into sub-matrices

Let $S = (I, J)$ be sub-matrix of A, where I is a subset of the rows $\{I_1, \ldots, I_N\}$ of A, and J is a subset of the columns $\{J_1, \ldots, J_N\}$ of A.

Let a_{iJ} denote the *mean value* of the ith row of the S, and a_{Ij} the mean of the jth column of S.

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, \text{ and } a_{Ij} = \frac{1}{|I|} \sum_{i \in I} a_{ij}$$

The *volume* v_S of a sub-matrix $S = (I, J)$ is the number of 1 entries a_{ij} such that $i \in I$ and $j \in J$, that is $v_S = \sum_{i \in I, j \in J} a_{ij}$.

Application in Community Detection (2)

Given a sub-matrix $S = (I, J)$, the *power mean of S of order r*, denoted as $\mathbf{M}(S)$ is defined as

$$\mathbf{M}(S) = \frac{\sum_{i \in I} (a_{iJ})^r}{|I|}$$

Definition of the fitness function

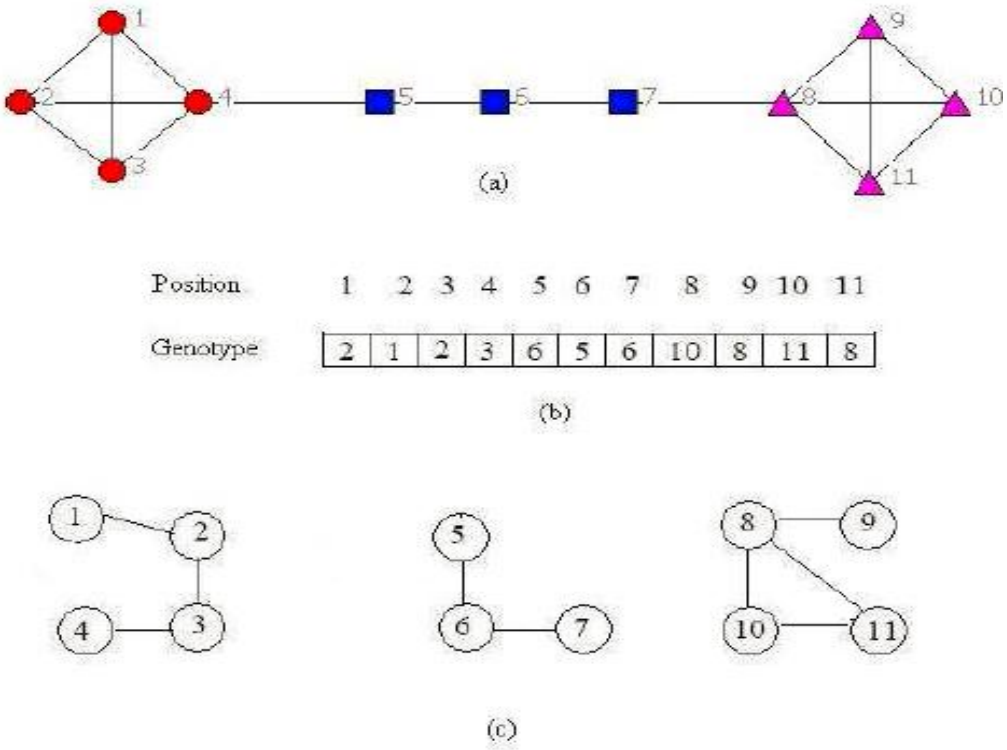
The *score* of S is defined as $Q(S) = \mathbf{M}(S) \times v_S$.
The *community score* of a partitioning $\{S_1, \ldots S_k\}$ of A is defined as

$$\mathcal{CS} = \sum_{i}^{k} Q(S_i)$$

Application in Community Detection (3)

Representation:

- Each individual is a vector of size *N* (number of nodes), where the value *j* at position *i* means a link between nodes *(i,j).*
- Such links should exist in the original network.
- Initialization via “safe” individuals i.e. check and correct if the links do not exist. If the value at position *i* is *k* and the link *(i,k)* does not exist, replace this link with a neighbor of *i*.
- Find communities represented by each individual via finding connected components.

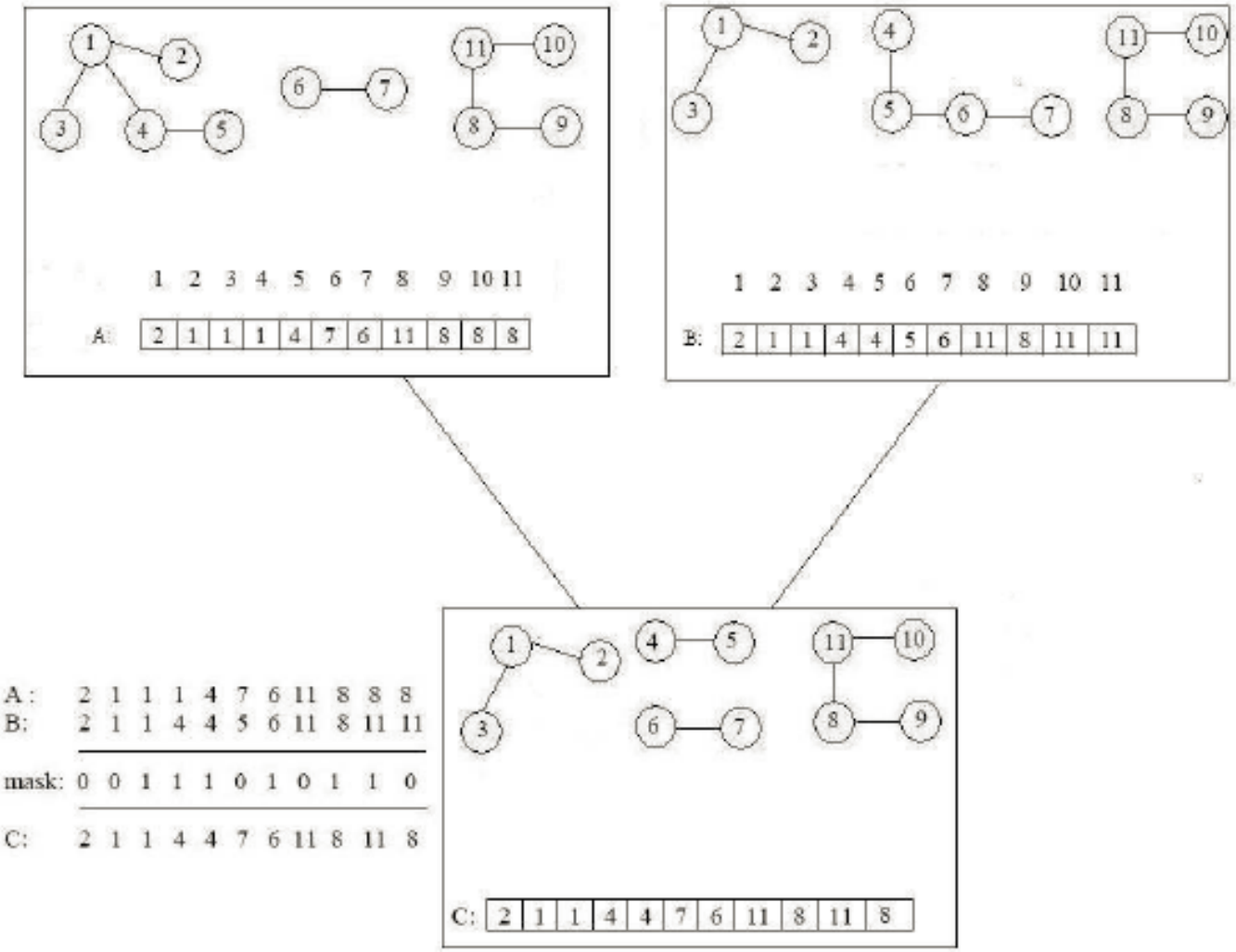


Mutation

- At position *i* only with the neighbors of *i*.

Application in Community Detection (4)

Uniform Crossover: exchange randomly genes of safe parents.



Rules of Thumb on Genetic Algorithms

- "Rules of thumb": results of empiric studies performed on binary encoding only, but they usually work fine . . .

- Crossover rate should be high generally, about 80% − 95% (However some results show that for some problems crossover rate about 60% is the best.)
- Mutation rate should be very low. Best rates seems to be about 0.5% − 1% per bit
- Very big population size usually does not improve performance (in the sense of speed of finding solution). Good population size is about 20 − 30, however sometimes sizes 50 − 100 are reported as the best
- Basic roulette wheel selection can be used. Elitism should be used for sure if you do not use other method for saving the best found solution

Characteristics of Genetic Algorithms

Advantages

1. Can explore search space in parallel
2. Improved performance over larger no of bits in encoding
3. Easy to discover global optimum
4. Solution space is wider

Drawbacks

1. Identifying Fitness Function is not easy
2. Representation of problem in genetic terms is difficult
3. Premature convergence occurs
4. Choosing various parameters like size of population, mutation and crossover probability has no defined rules
5. Hardware implantation difficult