Genetic algorithm for shortest path problem

Francesco Ragaini mbn349 Miguel Alonso cdn764

SCIENCE
University of Copenhagen

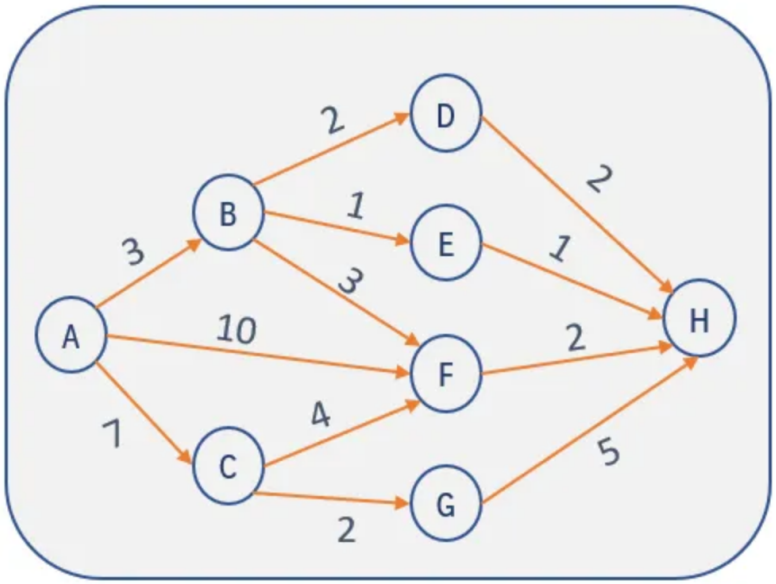
6th of March, 2025

Index

Introduction

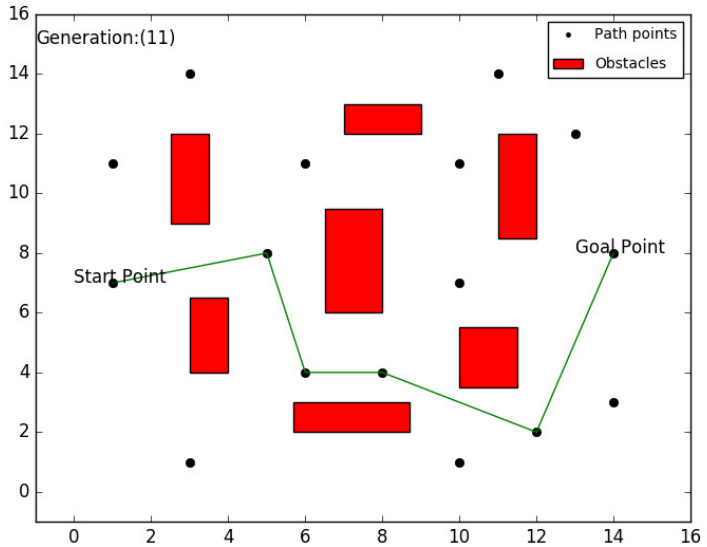
- Paper: M. Gen, R. Cheng, and D. Wang, "Genetic algorithms for solving shortest path problems," Nov. 2002, doi: https://doi.org/10.1109/icec.1997.592343.
- Widespread problem with many applications
- Difficult to solve
- Main problem: from path to chromosome

Dijkstra's algorithm



Source: Wikipedia

Genetics algorithm



Source: Github, username Yaaximus

Shortest path description

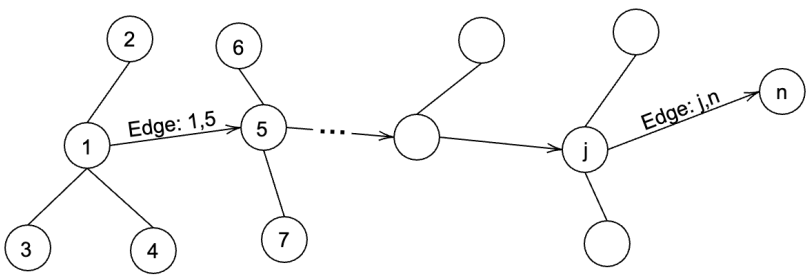


Figure: Shortest path diagram from 1 to n

Shortest path description

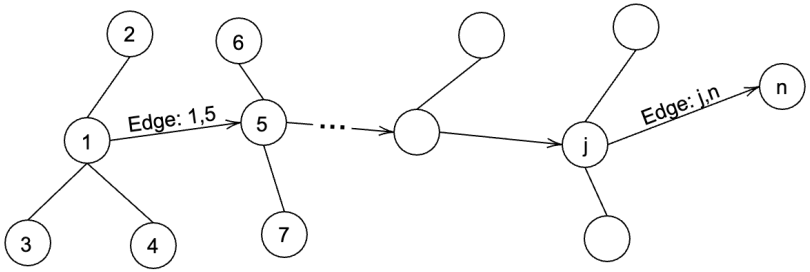


Figure: Shortest path diagram from 1 to n

The mathematical description uses:

$$x_{i,j} = \begin{cases} 1, & \text{edge } (i,j) \text{ in path} \\ 0, & \text{otherwise} \end{cases}$$

Shortest path description

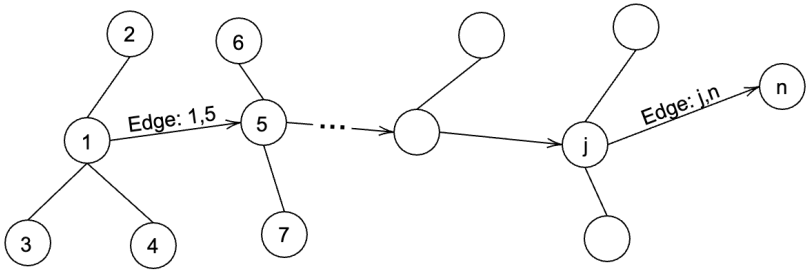


Figure: Shortest path diagram from 1 to n

The mathematical description uses:

$$x_{i,j} = \begin{cases} 1, & \text{edge } (i,j) \text{ in path} \\ 0, & \text{otherwise} \end{cases}$$

$$\min \sum_i \sum_j w_{i,j} x_{i,j}$$

$$\text{s.t.} \quad \sum_i x_{i,j} \leq 2, \quad \forall i \in V$$

Shortest path description

With the following constraints:

$$\sum_{j \neq k} x_{i,j} \geq x_{i,k}, \forall (i,k) \in E,$$

$$\forall i \in (V - \{1, n\})$$

$$\sum_{j} x_{1,j} = \sum_{j} x_{j,n} = 1, \forall i,j \in V$$

$$x_{i,j} = x_{j,i}, \forall (i,j) \in E$$

$$0 \leq x_{i,j} \leq 1, \forall (i,j) \in E$$

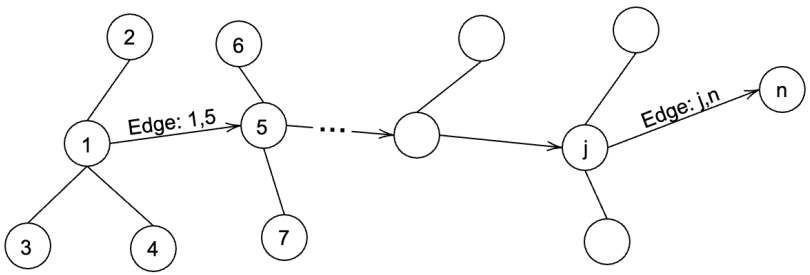


Figure: Shortest path diagram from 1 to n

Path building

Steps:

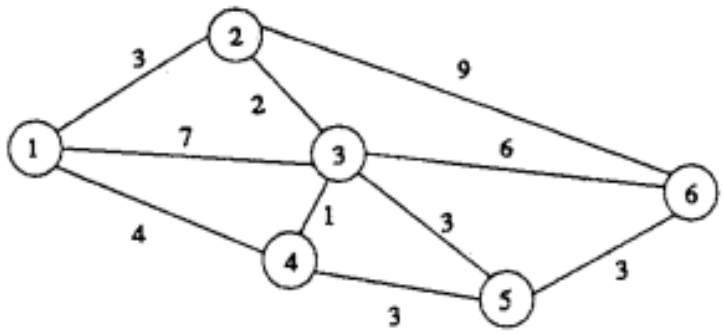
1. Assign random priority to each node.
2. Start from the first.
3. Select the neighbor with the highest priority (no repetitions).
4. Repeat 3 until the last node is found.

Path building

Steps:

1. Assign random priority to each node.
2. Start from the first.
3. Select the neighbor with the highest priority (no repetitions).
4. Repeat 3 until the last node is found.

Example:



Node	1	2	3	4	5	6
Priority	3	5	4	6	2	1

Path building

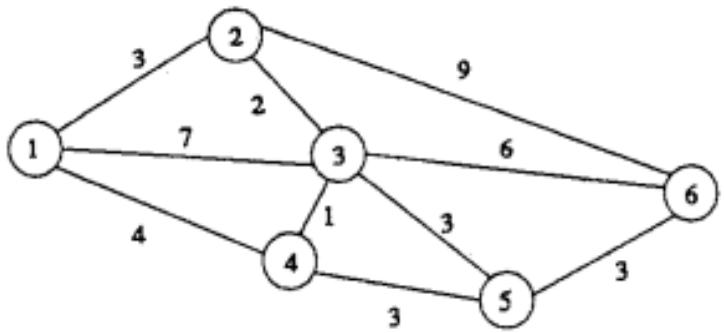
Steps:

1. Assign random priority to each node.
2. Start from the first.
3. Select the neighbor with the highest priority (no repetitions).
4. Repeat 3 until the last node is found.

Properties:

- Non redundancy.
- Completeness.
- Legality.

Example:



Node	1	2	3	4	5	6
Priority	3	5	4	6	2	1

Genetic algorithm

Two main operators to simulate real genetic composition:

Crossover

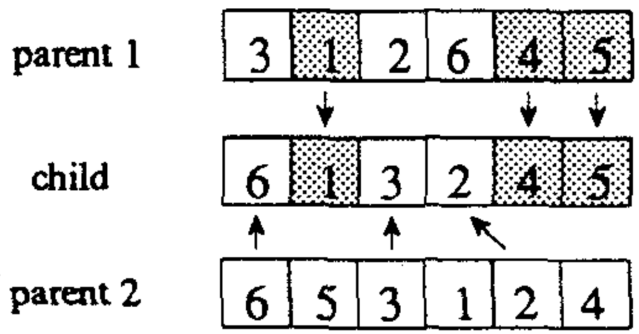
Computes child path:

Genetic algorithm

Two main operators to simulate real genetic composition:

Crossover

Computes child path:

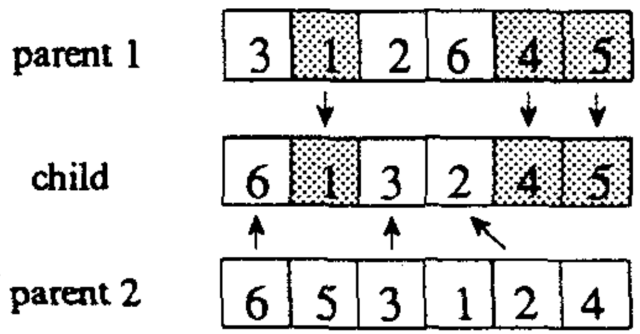


Genetic algorithm

Two main operators to simulate real genetic composition:

Crossover

Computes child path:



Mutation

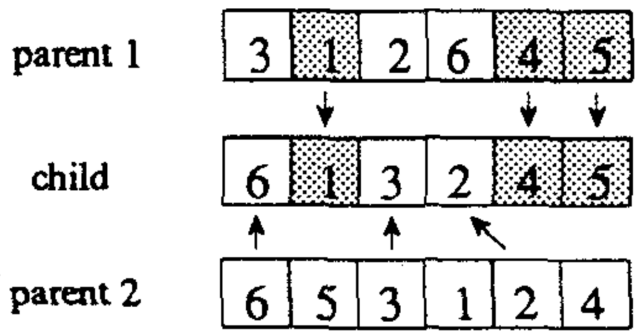
Alters genes to introduce variability:

Genetic algorithm

Two main operators to simulate real genetic composition:

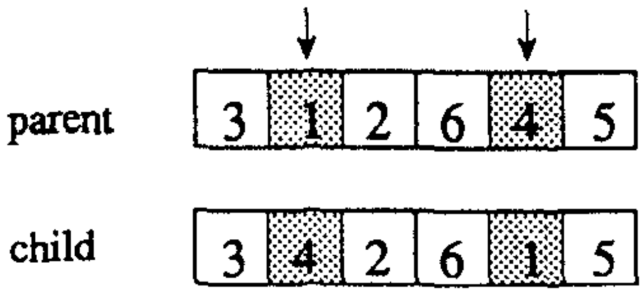
Crossover

Computes child path:



Mutation

Alters genes to introduce variability:



Genetic Algorithm: From Fitness to Probability

Roulette Method

Selection probability based on fitness:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Genetic Algorithm: From Fitness to Probability

Roulette Method

Selection probability based on fitness:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Example:

- Fitness values:
  $f_1 = 10, f_2 = 20, f_3 = 5$
- Probabilities:

$$p_1 = \frac{10}{35}, \quad p_2 = \frac{20}{35}, \quad p_3 = \frac{5}{35}$$

Genetic Algorithm: From Fitness to Probability

Roulette Method

Selection probability based on fitness:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Example:

- Fitness values:
  $f_1 = 10, f_2 = 20, f_3 = 5$
- Probabilities:

$$p_1 = \frac{10}{35}, \quad p_2 = \frac{20}{35}, \quad p_3 = \frac{5}{35}$$

Genetic Algorithm: From Fitness to Probability

Roulette Method

Selection probability based on fitness:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

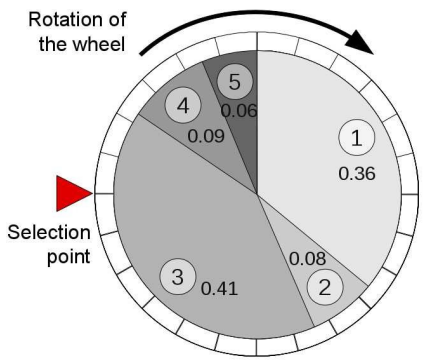


Figure: Source

Genetic Algorithm: From Fitness to Probability

Roulette Method

Selection probability based on fitness:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

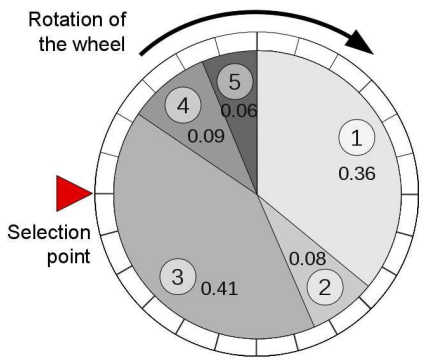


Figure: Source

Elitist Selection

Ensures the best individual passes to the next generation.

Genetic Algorithm: From Fitness to Probability

Roulette Method

Selection probability based on fitness:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

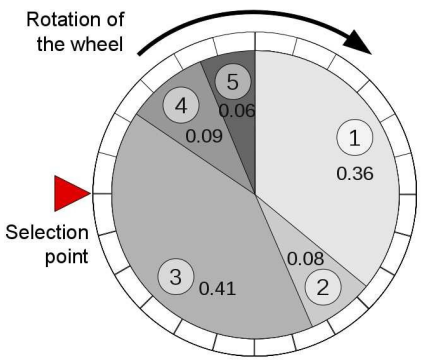


Figure: Source

Elitist Selection

Ensures the best individual passes to the next generation.

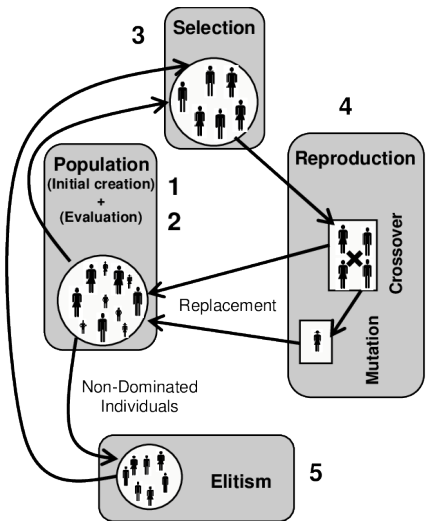


Figure: Source

Experimental results

Setup

Problem	Nodes	Edges	Pop	Max Gen	Crossover	Mutation	Runs
1	6	10	10	50	0.2	0.1	400
2	32	66	20	100	0.4	0.1	400
3	70	211	40	200	0.4	0.2	400

Experimental results

Setup

Problem	Nodes	Edges	Pop	Max Gen	Crossover	Mutation	Runs
1	6	10	10	50	0.2	0.1	400
2	32	66	20	100	0.4	0.1	400
3	70	211	40	200	0.4	0.2	400

Results

Problem	Nodes	Edges	Optimal	Best	Worst	Average	Frequency
1	6	10	10	10	10	10	100%
2	32	66	205	205	220	205.15	98%
3	70	211	2708	2708	3040	2745.53	64%

Focus on number 3

Setup

Problem	Nodes	Edges	Pop	Max Gen	Crossover	Mutation	Runs
3	70	211	10	200	0.4	0.2	200

Focus on number 3

Setup

Problem	Nodes	Edges	Pop	Max Gen	Crossover	Mutation	Runs
3	70	211	10	200	0.4	0.2	200

Population results

Population Size	Frequency
10	21%
20	66%
40	76%
100	92%

Study for the population size in problem 3 with maximum generation fixed to 200

Focus on number 3

Setup

Problem	Nodes	Edges	Pop	Max Gen	Crossover	Mutation	Runs
3	70	211	10	200	0.4	0.2	200

Population results

Population Size	Frequency
10	21%
20	66%
40	76%
100	92%

Study for the population size in problem 3 with maximum generation fixed to 200

Maximum Generation results

Maximum Generation	Frequency
1000	66%
1200	76%
2000	92%
3000	94%

Study for the Maximum Generation in problem 3 with population fixed to 10

Them and us

Main difference is our much higher number of nodes, which complicates the algorithm design!