# Genetic Algorithms for Solving Shortest Path Problems - A review

MIGUEL ALONSO MEDAVILLA[1] AND FRANCESCO RAGAINI[1]

[1] University of Copenhagen

March 5, 2025

## 1. INTRODUCTION

The shortest path problem is a fundamental issue that appears in the study of network systems and is the base of applications in routing and communications [1]. The chosen paper [1], suggests the use of genetic algorithms as a possible alternative to previously researched methods. Genetic algorithms use the concept of natural selection [2] to find solutions to optimization and search problems by repeated crossover and mutation techniques. The results shown in the study reflect an encouraging perspective and route towards the development of new techniques for the completion of shortest path-related problems.

## 2. REVIEW

### Shortest path definition

An undirected graph or network $G = (V, E)$ can be defined by a set of nodes $V$ and edges $E$, connecting the nodes. Each one of these edges has a nonnegative number $w_{ij}$ assigned, named cost or weight. This may represent a physical concept such as time, distance, or inclination between the nodes $v_i$ and $v_j$. A path can then be defined as a sequence of edges connecting two nodes, namely $v_i$ and $v_j$ (note that a path can also be defined as a sequence of nodes).
The shortest path will be defined as the path between two given nodes having minimum total cost. This can be described mathematically by defining an indicator variable, $x_{ij}$, as

$$x_{ij} = \begin{cases} 1, \text{if edge } (i,j) \text{ is included in the path} \\ 0, \text{otherwise.} \end{cases} \quad \textbf{(1)}$$

Such that the shortest path is a path connecting node 1 and node n following,

$$min \sum_i \sum_j w_{ij} x_{ij}. \quad \textbf{(2)}$$

Other constraints to the problem include that any node other than 1 and n has either 0 or 2 nonzero edges and 1 and n are the endpoints of the path.

### Path building

The general idea behind path building is to assign to every node (labeled from 1 to $N_{node}$) a random priority (also from 1 to $N_{node}$) without repetition and then, starting from the first node, going into the connected node with the highest priority, without repetition. With this general idea is possible to define the Algorithm 1 to build a random path that is going from the first to the last point.

**Algorithm 1.** Path building Algorithm

```
$priorities \leftarrow$ list of integers from 1 to $N_{node}$
$priorities \leftarrow permutation(priorities)$
$path\_list[0] \leftarrow 1$                    ▷ where 1 is the first node
$actual\_node \leftarrow 1$
$i \leftarrow 1$
while $actual\_node \neq N_{nodes}$ do
    $possible\_nodes \leftarrow$ list of nodes connected to $actual\_node$
    $next\_node \leftarrow$ node with max priority in $possible\_nodes$
    $path\_list[i] \leftarrow next\_node$
    $actual\_node \leftarrow next\_node$
    $i \leftarrow i + 1$
```

The paper [1] shows that the paths built with 1 have three different properties:

1. **Non redundancy**: at most case there is a one to one mapping between path and permutation of priorities.
2. **Legality** : any permutation of priorities corrisponds to a path.
3. **Completeness** : any path can be written with a permutation of priorities.

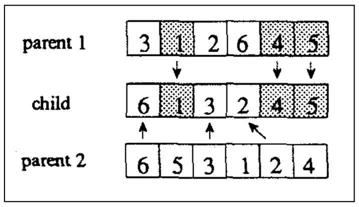

**Fig. 1.** Crossover operator. Taken from [1].

### Genetic algorithm

Throughout the reference paper and other scientific output, terminology from genetics is used to refer to different concepts of this algorithm. In this case, the studied population of ¨chromosomes¨ refers to the population of paths created at the beginning of the implementation. This population may undergo a crossover following a crossover ratio and/or a mutation (following a mutation ratio). This will form a new generation. The process will continue until convergence or a maximum generation values is reached.

The genetic algorithm used by Gen et al. [1] makes use of the crossover operator proposed in an earlier publication by Syswerda [3].
This function works as a crossover operator with a repairing procedure to account for the singularity of the nodes in the path (nodes are restricted to appear once in the path). In other words, the algorithm takes some genes of "parent 1" at random, transfers them to the child, and fills the missing pieces with genes from the other parent by a left-to-right scan (see Figure 1 for reference).

To further resemble a genetic process, a mutation operator was also used which selected two positions at random and swapped their contents as seen in Figure 2.

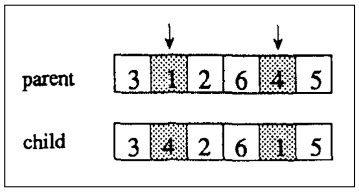

**Fig. 2.** Mutation operator. Taken from [1].

Before the application of the aforementioned operators, a fraction of the population has to be chosen for its genetic recombination and consequential creation of the new generation. Each one of the paths can be assigned a fitness value which is proportional to the inverse of the sum of the weights of its constituting edges. This fitness will then define the probability of each path being chosen. The preferred method used in the bibliography was the *roulette wheel* approach, in which every path is assigned a probability proportional to:

$$p = \frac{f_i}{\sum_{j=1}^{N} f_j}, \tag{3}$$

where $f$ is the fitness, $i$ refers to the studied path, $j$ to the other paths and N is the number of paths or population [2].
Furthermore, the *elitist way* was used to ensure the fittest path passes on to the next generation. This procedure automatically writes the fittest path to the next generation if the previous roulette wheel did not.

### Results

In order to test the algorithm, the paper proposes three different experiments, with problems of increasing complexity, the experimental setting is reported in Table 1. For all of them, the best

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

**Table 1.** Experimental setup for the problems

path is found using the Floyd-Warshall [4] algorithm and the test is repeated for 400 runs using different seeds for the random generator. For every problem, the frequency of successful runs (runs where the best path was found) is counted, together with the best, the average, and the worst result. As reported in Table 2, for the first two experiments, the algorithm is able to find the best path at least for 98% of the runs. On the other hand, for the last one the successful frequency is 64%, for this reason a more accurate study on number three is conducted. The article presents a new setup and focus on the study of the impact of population and maximum generation on the frequency of the best result. In this setup population and maximum generation are fixed to 10 and 200 (when not specified differently) crossover ratio is 0.4, mutation ratio 0.2, and the runs are 200. The results of these trials can be seen in Tables 3 and 4. The former shows data for a fixed population size with a changing maximum generation while the latter uses a fixed maximum generation for different population sizes.

| Problem | Optimal | Best | Worst | Average | Frequency |
|---|---|---|---|---|---|
| 1 | 10 | 10 | 10 | 10 | 100% |
| 2 | 205 | 205 | 220 | 205.15 | 98% |
| 3 | 2708 | 2708 | 3040 | 2745.53 | 64% |

**Table 2.** Results for the genetic algorithm for different problems

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

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

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

**Table 4.** Study for the population size in problem 3 with Maximum Generation fixed to 200

## 3. CONCLUSION

The paper studies the possible usage of genetic algorithms to solve shortest path problems , acknowledging the higher efficiency of other methods such as Floyd-Warshall. According to the authors the results are encouraging,in fact this algorithm is performing rapidly and precisely to find the best path. Evidence suggests that this approach may be used to solve more complex and difficult problems. Research on a bigger network, at for example city scale, could further prove the effectiveness of the algorithm.
In this case, the higher number of nodes would raise several concerns. For example, the random choosing of priorities of the nodes when building the path could very easily lead to inefficient redundancy or no-way-out cases. This could be fixed by changing the constraints and allowing repeating nodes plus prioritizing nodes that are closer to the end point. Unfortunately, these changes lead to other complications when framing the genetic algorithm, related to the crossover and mutation operators.

## REFERENCES

1. M. Gen, R. Cheng, and D. Wang, (2002).
2. A. Petrowski and S. Ben-Hamida, *Evolutionary algorithms* (Iste, Ltd. ; Hoboken, Nj, London, 2017).
3. G. Syswerda, p. 332–349 (1991).
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms* (MIT Press, 2009).