# An Evolutionary Approach to the Graph Bandwidth Problem


Maaz Saeed
*Dhanani School of Science and Engineering*
*Habib University*
*Karachi - 75290, Sindh, Pakistan*
*Email: @st.habib.edu.pk*

Muhammad Usaid Rehman
*Dhanani School of Science and Engineering*
*Habib University*
*Karachi - 75290, Sindh, Pakistan*
*Email: mr04302@st.habib.edu.pk*

Maham Shoaib Patel
*Dhanani School of Science and Engineering*
*Habib University*
*Karachi - 75290, Sindh, Pakistan*
*Email: mp04911@st.habib.edu.pk*



***Abstract*—The abstract goes here.**



**Keywords:** Graph Theory, Graph Bandwidth, Evolutionary Algorithms


## 1. Introduction

The graph bandwidth is a well-studied problem in graph theory. It is a combinatorial optimzation problem where the objective is to minimize the maximum distance between two vertices of a graph by finding a suitable labelling $f : V(G) \to \{1, 2, \ldots, n\}$. There are two different forms of the bandwidth problem – the bandwidth problem for graphs and the bandwidth problem for matrices. Both versions of the problem are closely related since the graph version of the problem can be reduced to the matrix version using the adjacency matrix of the graph. The problem can be visualized as placing the vertices of a graph at distinct integer points along the $x$-axis so that the length of the longest edge is minimized.

The inception of the matrix bandwidth problem occurred in the 1950s when structural engineers attempted to analyze steel frameworks by their structural matrices via computerized manipulation. The term 'bandwidth' was birthed as the engineers had endeavoured to discover a matrix in which all the non-zero elements lay withing a narrow 'band'. The inspiration for this came from operations such as inversion or finding determinant in as little time as possible.

In 1962, similar to this approach, L.H Harper, and A.W Hales conceived the bandwidth, and bandwidth sum. They used edge differences to represent single errors in a 6-bit picture code, in a hypercube, where it's vertices were words of the code [1]. Some time after this, R.R Kohrfage initialized his work on the graph bandwidth problem [2]. Finally, F. Harary published the problem, as we know it today, officially [3].

### 1.1. Formal Definition

In more formal terms, the mapping $f$ is defined as $f : V(G) \to \{1, 2, \ldots, n\}$, where $n = |V|$. This is also called a *proper numbering* of the graph $G$. [4] Therefore, we can think of these mappings as essentially labelling or numbering of the vertices. The bandwidth of a numbering $f$ is defined as:

$$B_f(G) = \max_{uv \in E(G)} \{|f(u) - f(v)|\} \tag{1}$$

The bandwidth of the graph $G$ is given by the bandwidth of the best possible numbering:

$$B(G) = \min\{B_f(G) : f \text{ is a numbering of } G\} \tag{2}$$

Bandwidths can be computed using any integer mapping, however, to make our implementation easier, we will be restricting ourselves to working with proper numberings only.

There exist several known mathematical bounds that relate the bandwidth of a graph to various graph theoretic properties. For example, it is a simple exercise to prove that

$$B(G) = n - 1$$

where $G$ is a complete graph of the form $K_n$. Similarly, there are also bounds relating the chromatic number and diameter with the bandwidth of the graph. [2]

### 1.2. NP-Completeness of the bandwidth problem

The bandwidth problem itself is a special case of the quadratic bottleneck assignment problem, which is known to be NP-hard. It is also known that the bandwidth problem is NP-hard to approximate which makes it impossible to find an $O(1)$-approximation algorithm for the bandwidth problem.

Historically, there have been endeavours to decrease (see [5], - [6]), or minimize (see [7], [8]) the bandwidth of large, spares matrices, effectively, by permuating rows and columns. This has been translated to graph theory by Harary (see [9]). Papadimitriou proved that the minimization of the bandwidth of a matrix is an NP-complete problem. [10]

There are several heuristic algorithms for the bandwidth problems such as the Cuthill-McKee algorithm, and the

Gibbs-Poole-Stockmeyer algorithm. Heuristic approaches are of interest to us since they will give us an idea ad to how we can design appropriate meta-heuristic techniques to solve the bandwidth problem.

## 2. Preliminary Concepts

### 2.1. Evolutionary Algorithms

Evolutionary Algorithms (also known as Genetic Algorithms) is a term referring to a family of algorithms based on the evolution we see around us, in nature. By mimicking learning, natural selection, reproduction, we can produce solutions for various search and optimization problems. This concept of evolving algorithms enables us to bypass the setbacks of traditional search / optimization algorithms.

**2.1.1. Darwinian Evolution.** Evolutionary Algorithms are derived of a simplified Darwinian evolution. The principles of such a cycle can be as such:

1) **Variation** - Individual members of a given population may have differing attributes from one another, for example, physical appearance.
2) **Inheritance** - Offspring resemble their parents to certain extents. In this manner, traits are passed down from one generation to another; unrelated individuals are less likely to have common traits, as compared to them with their family trees.
3) **Selection** - Nature follows 'survival of the fittest' ideology. Individuals that are better able to locate and make use of resources compared to their peers, are more likely to survive in their respective environments.

In accordance with these principals, results that we obtain from our algorithm may or may not resemble those of previous iterations. With careful manipulation of parameters, solutions produced by our code can be ranked higher or lower than others.

**2.1.2. Analogies.** Where Darwinian evolution maintains a population of individual solutions, genetic algorithms maintain **individuals** - a population of candidate solutions [11]. The theory behind these algorithms is that solutions are produced, and improved upon, by iteratively re-producing newer generations of solutions.

The various components of an evolutionary algorithm are as follows:

1) **Genotype** - In nature, genotypes are collections of genes. When two individuals procreate, a mixture of genes from both, will make up the chromosomes of the offspring. In code, these **chromosomes** can be expressed, for example, as strings in binary.
2) **Population** - Population refers to the collection of chromosomes. At any given moment, the algorithm will maintain a population of individuals - candidate solutions for the problem that is being attempted to be solved. In a nutshell, it is the current generation, which will be replaced by the next generation of offspring.
3) **Fitness Function** - a function used to evaluate individuals in a given population. Individuals that produce better results, will be more favoured when it comes to selection for breeding of newer generations. As this cycle runs, individuals display continuous improvement until a satisfactory solution to our problem is found, at which point we can terminate the operation.
4) **Selection** - After individuals are evaluated and awarded a *fitness value*, the best among them are chosen to breed and produce the newer generation. It is important to note that individuals with lower scores are still selected, but with lower probabilities, so as to not cause extinction of their respective attributes.
5) **Crossover** - refers to the mixing of chromosomes of the two parent individuals that were paired in the selection process to produce two new chromosomes (offspring). This process is also known as recombination.
6) **Mutation** - fulfills the purpose of periodically (at random; not in a set pattern) refreshing the population. This introduction of new patterns in the chromosomes encourages the algorithm to search in unexplored areas, rather than just exploiting what it has already chartered. The mutation may occur as random changes in chromosomes, for example, in binary string representation, a single bit may be switched.

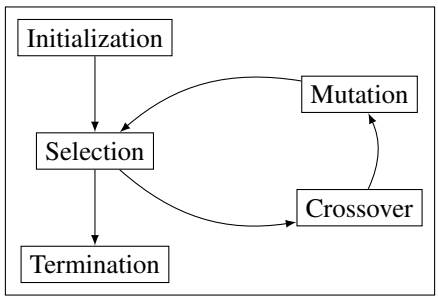


Figure 1. Flowchart of Evolutionary Algorithms

## 3. Implementation Details

### 3.1. Population Represenation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing

vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

# 4. Experimental Analysis

## 4.1. Testing Methods

## 4.2. Results

# 5. Conclusion

The conclusion goes here.

# References


[1] L. H. Harper, "Optimal assignments of numbers to vertices," *Journal of the Society for Industrial and Applied Mathematics*, vol. 12, no. 1, pp. 131–135, 1964.

[2] P. Z. Chinn, J. Chvátalová, A. K. Dewdney, and N. E. Gibbs, "The bandwidth problem for graphs and matrices—a survey," *Journal of Graph Theory*, vol. 6, no. 3, p. 223–254, 1982.

[3] E. M. Fels, "Fiedler, m. (ed.): Theory of graphs and its applications, proceedings of the symposium held in smolenice in june 1963. publishing house of the czechoslovak academy of sciences, prag 1964. 234 s., preis kč 26,50," *Biometrische Zeitschrift*, vol. 8, no. 4, pp. 286–286, 1966.

[4] J. R. Lee, *Graph Bandwidth*, pp. 866–869. New York, NY: Springer New York, 2016.

[5] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 1969 24th National Conference*, ACM '69, (New York, NY, USA), p. 157–172, Association for Computing Machinery, 1969.

[6] T. R. P., "Sparse matrices," vol. 99, no. 3, pp. 66–74, 1973.

[7] K. Y. Cheng, "Minimizing the bandwidth of sparse symmetric matrices," *Computing*, vol. 11, no. 2, pp. 103–110, 1973.

[8] K. Y. Cheng, "Note on minimizing the bandwidth of sparse symmetric matrices," *Computing*, vol. 11, no. 2, pp. 27–30, 1973.

[9] "References," in *Sparse Matrices* (R. P. Tewarson, ed.), vol. 99 of *Mathematics in Science and Engineering*, pp. 141–151, Elsevier, 1973.

[10] C. H. Papadimitriou, "The np-completeness of the bandwidth minimization problem," *Computing*, vol. 16, no. 3, p. 263–270, 1976.

[11] W. E., *Hands-on Genetic Algorithms with Python*. UK: Packt Publishing Ltd, 1 ed., 2020.