A CUSTOMIZED FORCE-DIRECTED LAYOUT ALGORITHM WITH GENETIC ALGORITHM TECHNIQUES FOR BIOLOGICAL GRAPHS WHOSE VERTICES HAVE ENZYME COMMISSION ATTRIBUTES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FIRAT AKSOYDAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

A CUSTOMIZED FORCE-DIRECTED LAYOUT ALGORITHM WITH GENETIC ALGORITHM TECHNIQUES FOR BIOLOGICAL GRAPHS WHOSE VERTICES HAVE ENZYME COMMISSION ATTRIBUTES

submitted by **FIRAT AKSOYDAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Mehmet Volkan Atalay
Supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Tolga Can
Computer Engineering Department, METU _____

Prof. Dr. Mehmet Volkan Atalay
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Tunca Doğan
Institute of Informatics, Hacettepe University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Fırat Aksoydan

Signature :

ABSTRACT

A CUSTOMIZED FORCE-DIRECTED LAYOUT ALGORITHM WITH GENETIC ALGORITHM TECHNIQUES FOR BIOLOGICAL GRAPHS WHOSE VERTICES HAVE ENZYME COMMISSION ATTRIBUTES

Aksoydan, Fırat
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Mehmet Volkan Atalay

September 2019, 84 pages

A pathway can be visualized as a graph whose layout is drawn by a force-directed algorithm. In our previous study, we have described EClerize which is a customized and improved Kamada Kawai force-directed algorithm in order to visualize pathways that contain nodes with attributes as EC numbers. EClerize creates clusters of vertices with enzymes that belong to the same EC class. Here, we make use of genetic algorithm (GA) to obtain a global optimum solution for EClerize and we integrate undirected graph layout drawing with GA. To provide diversity, 5 techniques in mutation phase and for crossover 2 techniques are employed. In mutation, vertices of a selected graph are moved randomly within a limited area or selected edges/vertices are exchanged according the routines of a technique. In crossover, the operation of exchanging vertices is performed between two selected graphs. In each iteration, fitness values of individuals are calculated by 6 different fitness measurements ranging from edge crossing number to drawing area. Overall relative fitness values are used to choose parent individuals. We have applied this method to 3 pathways and the results are better than those of the base study.

Keywords: Graph Visualization, Clustering, Force-directed Algorithm, Enzyme Commission Numbers, Genetic Algorithm

ÖZ

ENZİMLERİ TEMSİL EDEN DÜGÜMLERE SAHİP ÇİZGİLER IÇIN GENETİK ALGORİTMA İLE ÖZELLEŞTİRİLMİŞ KUVVET YÖNELİMLİ YERLEŞİM ALGORITMASI

Aksoydan, Fırat
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Mehmet Volkan Atalay

Eylül 2019 , 84 sayfa

Biyolojik ağlar, kuvvet yönelimli algoritmalar tarafından yaratılan çizgeler aracılığıyla görselleştirilebilir. Önceki çalışmamızda, EClerize'ı aynı enzim sınıfına sahip olan düğümleri içeren biyolojik ağları görselleştirmek için kullanılan özelleştirilmiş ve geliştirilmiş bir Kamada-Kawai (bir kuvvet yönelimli algoritma) örneği olarak tanımlamıştık. EClerize, aynı enzim sınıfına ait enzimler aracılığıyla enzim kümeleri oluşturur. Burada, EClerize aracılığıyla global optimum bir çözüm elde etmek için genetik algoritmayı kullanıyoruz ve yönsüz çizge çizimini genetik algoritma ile birleştiriyoruz. Çeşitliliği sağlamak için, mutasyon aşamasında 5 teknik ve çaprazlama için 2 teknik kullanılmaktadır. Mutasyonda, seçilen çizgenin köşeleri sınırlı bir alanda rastgele hareket eder veya seçilen kenarlar/köşeler bir tekniğin rutinlerine göre değiştirilir. Çaprazlamada, köşeleri değiştirme işlemi seçilen iki çizge arasında gerçekleştirilir. Her bir yineleme sonunda, bireylerin uygunluk değerleri, çakışan kenar sayısından çizim alanına kadar 6 farklı uygunluk ölçümü ile hesaplanır. Bireylerin sahip olduğu genel uygunluk değerleri, bir sonraki nesli

oluştururken ebeveyn bireyleri seçmek için kullanılır. Bu yöntemi 3 farklı biyolojik ağa uyguladık ve elde edilen sonuçlar baz alınan çalışmanın sonuçlarından daha iyi oldu.

Anahtar Kelimeler: Görselleme, Çizge Görselleme, Kuvvet Yönelimli Çizge Yerleşimi, Enzim Komisyonu Sayıları, Genetik Algoritma

To Beril

For her advice, her patience, and her faith.

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Volkan Atalay for his vital comments and supports at each step of this study. His guidance helped me in all the time of research and writing of this thesis. Studying with him is an elusive chance.

I would like to thank my company ASELSAN Inc. for the possibilities that have provided to me during my whole master.

I would like to thank my team leader Ozan Küsmen for his sincere guidance and valuable advice.

Very special thanks go to my friends Murat Ersoy, Serkan Buhurcu and Ege Engin.

Finally, I want to express my sincere gratitude to my family for their thrust throughout my life.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xxii

CHAPTERS

1 INTRODUCTION 1

2 BACKGROUND INFORMATION AND RELATED WORK 5

2.1 Fundamental Information About Graphs 5

2.2 Force Directed Graph Drawing 8

2.2.1 Kamada Kawai Algorithm 11

2.3 Basics of Clustering 12

2.4 Enzymes and Enzyme Commission Nomenclature 15

2.4.1 Enzyme 15

2.4.2 Enzyme Commission Nomenclature 15

2.5 EClerize 17

2.6 Heuristic Approaches 20
2.7 Basic Genetic Algorithm 21
2.8 Related Studies 26
3 ECLERIZE TYPE GA 33
3.1 Overview 33
3.2 Genetic Algorithm on EClerize Type GA 35
3.2.1 Initial Condition 35
3.2.2 Fitness Calculator 37
3.2.3 Selection 39
3.2.4 Crossover 40
3.2.5 Mutation 44
3.2.6 Termination 49
4 EXPERIMENTAL RESULTS 51
4.1 Dataset 51
4.2 Parameter Setup 53
4.2.1 Population Size ($size$) and Iteration Counter ($counter$) 53
4.2.2 Mutation Settings 62
4.2.3 Crossover Settings 63
4.2.4 Fitness Settings 64
4.3 Results 67
5 CONCLUSION 75
5.1 Summary 75
5.2 Perspectives 78

REFERENCES 79

LIST OF TABLES

TABLES

Table 2.1 Some of the force-directed algorithms with their effective aesthetic criteria and time complexities. 10

Table 2.2 Enzyme commission main classes. 16

Table 2.3 Comparison of previous studies with their approaches and contributions. 26

Table 4.1 Primary information about used datasets. 52

Table 4.2 EC class information about used datasets. 52

Table 4.3 Iteration counter ($counter$), edge crossing numbers and execution time (second) when $size$ = 6 on Signaling by EGFR dataset. 56

Table 4.4 Edge crossing numbers and execution time (second) when $size$ = 6 on Signaling by ERBB2 dataset. 57

Table 4.5 Edge crossing numbers and execution time (second) when $size$ = 6 on Visual Phototransduction dataset. 58

Table 4.6 Execution results from different values of the parameters $size$ and $counter$ on the datasets Signaling by EGFR, Signaling by ERBB2 and Visual Phototransduction. 61

Table 4.7 Recommended rates of mutation techniques used in EClerize Type GA. 62

Table 4.8 Recommended crossover rates of EClerize Type GA. 63

Table 4.9 Recommended fitness coefficients of EClerize Type GA. 64

Table 4.10 Comparison of different active fitness criteria on Signaling by EGFR dataset. 65

Table 4.11 A comparison between result of our study and base study on Signaling by EGFR dataset. 67

Table 4.12 A comparison between result of our study and base study on Signaling by ERBB2 dataset. 68

Table 4.13 A Comparison Between Result of Our Study and Base Study on Visual Phototransduction dataset. 68

LIST OF FIGURES

FIGURES

Figure 2.1 Illustration of different drawings of the same graph. 7

Figure 2.2 Illustration of forces on a given graph [2]. 9

Figure 2.3 Circular layout (left), Fruchterman-Rheingold layout (middle), and Kamada-Kawai Layout (right) of a small world network [3]. 11

Figure 2.4 Illustration of inter-cluster and intra-cluster distances on the same graph. 13

Figure 2.5 Illustration of tree structure of Enzyme Commission Numbers. Levels correspond to class number, subclass number, sub-subclass number and serial number respectively [37]. 16

Figure 2.6 Graph with the addition of virtual edges by EClerize algorithm [37]. 18

Figure 2.7 An example initial population of the GA. I1 indicates an individual. An individual is represented by a chromosome. A chromosome consists of genes. A group of chromosomes (individuals) generates the population. 22

Figure 2.8 An example of the fitness values of the population. The fitness value of each individual is calculated by, comparing each gene of the selected individual with the corresponding gene of the perfect chromosome. 22

Figure 2.9 An illustration of crossover. I1 is the first parent and I2 is the second parent. The 'Crossover Point' is a tool to separate chromosomes into gene sections. I5 and I6 are children. I5 consists of the first part of I2 and the first part of I1, I6 consists second part of I1 and the first part of I2. 23

Figure 2.10 Illustration of mutation technique. a) The chromosome before mutation. b) After the mutation of the given chromosome, one gene has been changed, which is colored with red. 24

Figure 2.11 Layout of Barreto and Barbosa's algorithm in action [9]. 28

Figure 2.12 Rectangular crossover from TimGA [63]. 29

Figure 3.1 Illustration of rectangular crossover technique. In this figure, the edges are not shown to avoid complexity. a) First parent graph. b) Second parent graph. c) First child graph. d) Second child graph. . . . 41

Figure 3.2 Illustration of three vertices crossover. a) First parent graph. b) Second parent graph. c)First child graph. d) Second child graph. 42

Figure 3.3 An illustration of the tiny vertex mutation. In this mutation technique, a random vertex is selected and moved horizontally or vertically with a random distance on the graph. a) Original graph. b) After the mutation on X-axis. c) After the mutation on Y-axis. 45

Figure 3.4 An illustration of the single vertex mutation. In this mutation technique, a random vertex is selected and moved both horizontally and vertically in a limited random range inside the drawing area. a) Original graph. b) The graph after the mutation of the selected vertices. 46

Figure 3.5 An illustration of the swap of vertices mutation. In this mutation technique, two randomly selected vertices' positions are swapped. a) Original graph. b) The graph after the mutation of the selected vertices. 46

Figure 3.6 An illustration of the edge-based mutation. In this mutation technique, a random edge is selected and moved both horizontally and vertically by keeping its length and angle unchanged. a) Original graph. b) The graph after the mutation of the selected edge with its connected vertices. 47

Figure 3.7 An illustration of the two edge-based mutation. In this technique, a random vertex who has at least 2 edges is selected, then these two edges are moved to the new locations by keeping edge lengths and angles the same. a) Original graph. b) The graph after the two edge-based mutation. 48

Figure 4.1 Relation between parameter $size$ and edge crossing number when $counter$ = 1 on Signaling by EGFR dataset. 54

Figure 4.2 Relation between parameter $size$ and edge crossing number when $counter$ = 1 on Signaling by ERBB2 dataset. 55

Figure 4.3 Relation between parameter $size$ and edge crossing number when $counter$ = 1 on Visual Phototransduction dataset. 55

Figure 4.4 Number of edge crossings and execution time (second) when $size$ = 6 on Signaling by EGFR dataset. 57

Figure 4.5 Edge crossing numbers and execution time (second) when $size$ = 6 on Signaling by ERBB2 dataset. 58

Figure 4.6 Edge crossing numbers and execution time (second) when $size$ = 6 on Visual Phototransduction dataset. 59

Figure 4.7 An illustration of different active fitness criteria on Signaling by EGFR dataset. a) Edge crossing is the only active criterion. b) Edge length deviation is the only active criterion. c) Inter-cluster distance is the only active criterion d) Minimum node distance sum is the only active criterion. 66

Figure 4.8 Different Layouts of Signaling by EGFR dataset. (A) Layout by EClerize Type GA with parameters c = 3 and S = 6. (B) Layout by Native EClerize with its default parameters. 69

Figure 4.9 Different Layouts of Signaling by ERBB2 dataset. (A) Layout by EClerize Type GA with parameters c = 3 and S = 6. (B) Layout by Native EClerize with its default parameters. 70

Figure 4.10 Different Layouts of Signaling by Visual Phototransduction dataset. (A) Layout by EClerize Type GA with parameters c = 3 and S = 6. (B) Layout by Native EClerize with its default parameters. 71

Figure 4.11 Zoomed visuals from the layouts of Signaling by EGFR, Signaling by ERBB2 and Visual Phototransduction datasets by EClerize Type GA and Native EClerize. a) Layout by EClerize Type GA on Signaling by EGFR dataset. b) Layout by Native EClerize on Signaling by EGFR dataset. c) Layout by EClerize Type GA on Signaling by ERBB2 dataset. d) Layout by Native EClerize on Signaling by ERBB2 dataset. e) Layout by EClerize Type GA on Visual Phototransduction dataset. f) Layout by Native EClerize on Visual Phototransduction dataset. 73

List of Algorithms

1 EClerize Algorithm 19
2 EClerize Type GA 34
3 Initial Condition 37
4 Fitness Calculator 38
5 Selection 39
6 Crossover 43
7 Mutation 48
8 Termination 50

LIST OF ABBREVIATIONS

GA	Genetic Algorithm
KK	Kamada-Kawai
DH	Davidson-Harel
FR	Fruchterman-Reingold
EC	Enzyme Commission
MM	Multilevel Modelling

CHAPTER 1

INTRODUCTION

Graphs can be used to represent various domain-specific information. By using vertices to symbolize the data items and edges to represent the relation between these items, various type of information can be represented by the graphs. Drawing graph layouts in an aesthetically pleasing way is an effective method for representing the information to the users and drawing graphs in a nice-looking form is a long-standing problem for computer scientists. Regardless of the which aesthetic criteria are used, the problem usually contains various computationally challenging subproblems; to overcome these problems, several heuristic optimization algorithms are required.

Biological graphs which consist of components such as genes, proteins, and enzymes; have great importance in bioinformatics. Some of the biological graphs contain vertices that represent enzyme structure. In these type of graphs, there could be some vertices which have clustering information dedicated to Enzyme Commission (EC) number, which is the numerical classification of an enzyme. With the help of these EC numbers, clustering can be conducted. The vertices that belong to the same EC class are members of the same cluster, proportional with the distance in the distance tree. EClerize [37], which is the study on which we have made our improvements, handles vertices having the same EC class as if they are in the same cluster. EClerize tries to minimize the distance between the vertices of the same EC cluster and to maximize the distance between centroids of these clusters; by this way, more readable layouts can be drawn keeping the aesthetically pleasing nature of force directed-layout algorithms. EClerize is based on Kamada-Kawai (KK) algorithm, which is a force-directed graph layout algorithm.

KK algorithm is efficient on enhancing undirected graph layouts in terms of aesthetic measurements such as number of edge crossings and drawing area. On the other hand, KK algorithm has a disadvantage of getting stuck into local optimums. This manuscript presents improvements over an existing study, EClerize. Here, in our study EClerize Type GA, our purpose is to avoid the local optima and obtain global optimum solutions for EClerize during graph drawing. By use of a well-known heuristic technique, Genetic Algorithm (GA), we integrate undirected graph layout drawing with GA. We have used the previous study EClerize as a fine-tuner on GA. The genetic algorithm draws strength from the diversity for providing global optima, and the mutation and the crossover are the most important resources of the diversity. In our study, 5 techniques in mutation phase and 2 techniques in crossover phase are employed. In mutation, vertices of a selected graph are moved randomly within a limited area or selected edges/vertices are exchanged according to the routines of the selected mutation technique. In the crossover, the operation of exchanging vertices is performed between two selected graphs. In our study, the aesthetic criteria of undirected graphs are served as fitness measurements of GA. In each iteration, to measure how well graphs are drawn, fitness values of graphs are calculated by 6 different fitness measurements ranging from the number of edge crossings to the size of the drawing area. Overall relative fitness values are used to choose parent individuals.

We have applied our study to 3 pathways and the results are better than those of the base study EClerize with respect to certain some measurable fitness criteria with a reasonable longer execution time. From the perspective of global optimum, as genetic algorithm promises, now we can reach better results to draw biological graphs whose vertices are associated with Enzyme Commission attributes.

From the perspective of the graph drawing studies, to the best of our knowledge, our work is the first one with the implementation of a GA, a force-directed algorithm and some clustering techniques altogether. There are a number of studies which combines GA with some force-directed algorithms, however, none of them considers clustering.

The rest of the thesis is as follows. In Chapter 2, a review of previous studies, and

the background information to understand the work in this thesis are given. The topics covered in this chapter can be divided into three main parts. The first part provides information for understanding EClerize. The second part is reserved for EClerize itself. The last part also explains the background information which has been used in our method. Besides, there is an additional section which is reserved for an extensive survey. In Chapter 3, our study EClerize Type GA is described in detail. The technique used to improve the base study is described. In sub-sections of this chapter, all these steps are explained in detail. In Chapter 4, experimental results on several datasets are given. Experimental results obtained from these datasets are presented briefly in this chapter. Finally, in Chapter 5, the thesis is concluded with a summary of the study and some possible future work suggestions.

CHAPTER 2

BACKGROUND INFORMATION AND RELATED WORK

Our study is based on the combination of various domain-specific techniques such as genetic algorithm and enzyme commission information. In this section, the background information is explained in chronological order. First, the information which forms the basis of EClerize is given, and then EClerize is explained. Afterward, the background information of our study is described briefly. Finally, there is an additional section which is reserved for an extensive survey.

2.1 Fundamental Information About Graphs

A graph is a structural model which consists of vertices and edges that represent relations [34]. In a graph: $G = (V, E)$ where V is the set of vertices and E is the set of edges; vertices designate objects, while edges represent relations between the objects [36].

Graphs have been used for various type of applications; such as social networks, bioinformatics, network maps, network visualization, state machines. Graphs come in different types according to their usage area and available data. The major graph types are as follows [7].

- Simple graph or multigraph: In a simple graph, each edge connects two distinct vertices and any pair of vertices can have only one edge. A multigraph can be defined as a graph which contains multiple edges for any selected pair of vertices.

• Undirected or directed graphs: An undirected graph has edges that do not have a direction; that is the edges are unidirectional. A directed graph has edges with direction and edges are usually represented by arrows pointing in the direction that the graph can be traversed [15].

• Weighted or unweighted graphs: An unweighted graph has edges that do not have a cost or weight associated with it, whereas a weighted graph has edges with assigned numerical values [60].

• Labeled or unlabeled graphs : A graph in which labels (which are generally numbers) are assigned to vertices is called labeled graph. If a graph has vertices which do not have labels this graph can be called as unlabeled graph.

The data sets on which we run our studies have simple, non-directional, unweighted and unlabeled graphs. An undirected graph is a graph where all the edges are bidirectional, which means the order of the vertices of an edge is not important. In an undirected graph G, suppose that $e = \{u, v\}$ is an edge of G [36]:

-vertices u and v are its ends,

-u and v are called adjacent,

-e is called as incident with u and v.

According to Helen [53], there are seven aesthetic criteria for measuring how well-structured a graph is. Two different edges cross in a graph drawing if their geometric representations intersect; in a well-structured graph ***edge crossings*** are minimized. In a graph, edge bends are straight-line drawings of a divided graph where every edge has extra vertices joined with it [47], in a well-structured graph the ***edge bends*** are minimized. A planar graph has edges which join only at their endpoints, in a ***planar graph*** there isn't any edge crossing [65]. A graph is symmetric for a line if reflecting the graph over that line represents the same graph, in a well-structured graph the ***symmetry*** is maximized. Intra-cluster distance is the distance between the members of any selected cluster and inter-cluster distance is the distance between clusters. In a nice-looking graph, the ***inter-cluster distance*** is maximized while the ***intra-cluster distance*** is minimized. In mathematics, the orthogonality

is the state or quality of being right-angled or perpendicular; in a well-structured graph ***orthogonality*** of the vertices is maximized.

In Figure 2.1, there are two different drawings of the same graph. In terms of the above given criteria, there are differences between the two drawings. For example, in each graph, the number of vertices, the number of edges and the degree of these vertices are same; but due to different drawing techniques, the number of edge crossings are different.

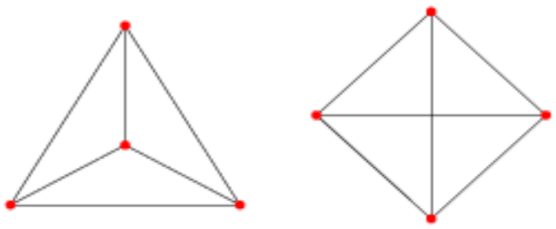


Figure 2.1: Illustration of different drawings of the same graph.

Graph data usually does not have location information; they mostly have the knowledge on paired vertices. Graph layout algorithms desire to locate vertices in the appropriate coordinates to represent the graph aesthetically. On the other hand, regardless of the aesthetic criteria, graph layout algorithms need to take into account several computational sub-problems [34], the most important ones are execution time and memory consumption. There are several techniques to minimize these problems. Hierarchical layout, grid layout, circular layout, tree layout, orthogonal layouts, and force-directed are some of the popular graph layout algorithms [29]. These techniques are also used for visualizing various entities in bioinformatics. Inoue and Shimozono [39] describe one of the significant studies on visualizing large-scale biochemical network maps by using a grid layout algorithm. The study by Kojima et. al [44] utilizes a similar graph drawing technique for biological networks. The work done by Wegner and Kummer [69] plays a vital role for drawing complex biochemical reaction networks using a hierarchical graph layout along with a circular graph layout algorithm. Becker and Rojas [11] have conducted one

of the most comprehensive study in this field in which they have used circular, hierarchic and force-directed graph layout algorithms altogether for drawing pathways. Tsay et al. [66] propose an improved heuristic graph drawing technique for drawing complex pathways; in their study, the complex pathways are defined as hierarchical structures in which a pathway is divided into subparts and arranged hierarchically as a tree. The study by Gu et. al [32] provides a circular layout graph drawing technique for different types of biological data.

The execution of the previously given aesthetic criteria is quite costly in terms of time. Some of the complexities for checking previously given aesthetic criteria are given below:

- minimizing edge crossings is NP-hard [30]
- testing planarity costs linear time [38]
- testing upward planarity is NP-hard [10]
- minimizing bends [22] in planar orthogonal drawing:
  - NP-hard in general [31]
  - polynomial time for a fixed embedding [16], [62]

2.2 Force Directed Graph Drawing

There are different specific algorithmic strategies for drawing graphs and force-directed algorithms are essential for drawing undirected graphs. These algorithms are also known as spring embedder algorithms that simulate a system of forces acting on the vertices and edges. The simulated physical system tries to find a minimum energy state, where the vertices perform repulsive forces and edges perform attraction forces [43].

Force-directed graph drawing algorithm is first performed on graphs by Eades [18]. In this technique, vertices symbolize charged particles and repulse each other while edges symbolize springs which perform attraction forces to connected vertices,

which is shown in Figure 2.2. Eades' algorithm takes edge lengths as uniform magnitudes and connects non-adjacent vertices with additional springs with infinite magnitudes and keeps the symmetry of the graph.

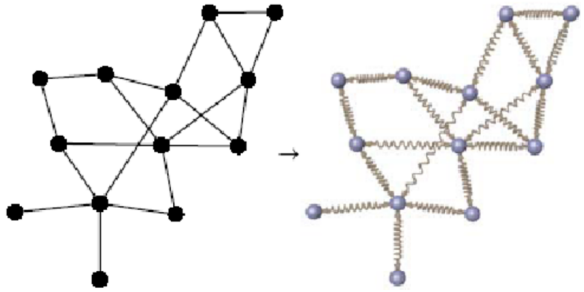


Figure 2.2: Illustration of forces on a given graph [2].

A spring embedder simulates vertices as electrically charged particles on a plane that are connected by springs, they attract each other. If they are not connected, they repel each other. A force-directed algorithm works iteratively. In each iteration, all forces on each vertex v are calculated. Each incident edge $e = \{u, v\}$ attracts the vertex v with the force $f(\overrightarrow{u, v})$ in the direction of u according to "Hooke's Law". At the same time, each virtual (disconnected) edge $e = \{u, v\}$ repels the vertex v away from u. After computing all forces on each vertex (particles), all of the forces are summed up; the particles move on the plane with respect to the calculated final force on them. Then the spring embedder algorithm passes into the next iteration until a termination condition is reached. Termination criterion could be a limit on an iteration counter or a minimum energy constraint on the system's energy state.

Force-directed algorithms have various advantages as indicated below.

-Simplicity: Its implementation and understanding are easy. Its simple logic is easier than other graph drawing techniques.

-Good-quality results: Produced graphs by force-directed techniques are successful in terms of aesthetic criteria.

-Flexibility: It's easy to adapt and extend by reorganizing energy function based on

newly selected criteria.

-Interactivity: User can see any intermediate result on graph interactively. This also enables stopping during execution on an acceptable intermediate result.

After the studies conducted by Eades, new approaches have been proposed to improve this force-directed technique. The study by Davidson and Harel [17] avoided the local minima with the help of simulated annealing. An important improvement is provided by Fruchterman and Rheingold [26] by placing neighboring vertices by using a system of forces similar to that of subatomic particles and celestial bodies. Kamada and Kawai also proposed a different technique [42], to merge graph theoretic distance with the spring algorithm. Some of these force-directed techniques with their aesthetic criteria considerations and their time complexities are given Table 2.1. As shown in the table, different algorithms consider varied aesthetic criteria and this circumstance causes different execution times. In Figure 2.3, we can see the different aesthetic considerations of these algorithms from the obtained results.

Table 2.1: Some of the force-directed algorithms with their effective aesthetic criteria and time complexities.

Algorithm	Effective Aesthetic Criteria	Time Complexity
Kamada-Kawai	Minimum edge crossings, symmetry	$O(\|V\|^3)$
Davidson-Harel	Uniform edge length	$O(\|V\|^2\|E\|)$
Fructerman-Reingold	Symmetry, uniform edge length	$O(\|V\|^2 + \|E\|)$

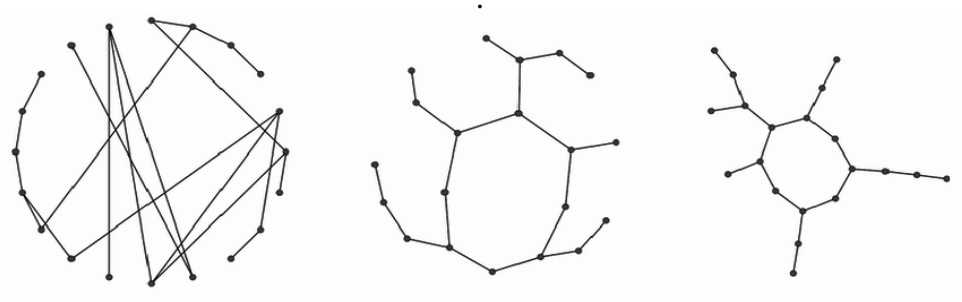


Figure 2.3: Circular layout (left), Fruchterman-Rheingold layout (middle), and Kamada-Kawai Layout (right) of a small world network [3].

2.2.1 Kamada Kawai Algorithm

Kamada Kawai (KK) algorithm is a force-directed drawing technique [42]. The main concern of the KK algorithm is producing graphs with good symmetry with relatively small number of edge-crossings. Different than the algorithms that were applied before them, the term "ideal distance" was created, which is a concept assuming that the ideal distance is proportional to the shortest path between the two edges.

In KK algorithm, a virtual dynamic system has been offered. In this new system, every two vertices are linked by a “spring” of ideal length. If there are n vertices in the graph, these vertices can be represented as $v_1, v_2, v_3, ..., v_n \in V$.

In their system there is always a spring between two vertices, whether or not two vertices are connected. k_{ij} is the strength of springs between v_i and v_j, d_{ij} is the shortest path between given vertices and K is the constant. k_{ij} is given by:

$$k_{ij} = \frac{K}{{d_{ij}}^2} \tag{21}$$

l_{ij} is the desired length between v_i and v_j, and L is the convenient length of a single edge in the plane. l_{ij} is given by:

$$l_{ij} = L \times d_{dj} \tag{22}$$

The main purpose of this force-directed algorithm is to locate the vertices so that the energy function reaches the minimum value. Energy function of KK is formulated as:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} (\left| v_i - v_j \right| - l_{ij})^2 \tag{23}$$

2.3 Basics of Clustering

In a cluster, members are similar to each other and different from other clusters members. Clustering helps us to discover undetected relationships in a group of dataset.

A good clustering should present the following features:

- dealing with noise;
- handling outliers;
- conforming intra-cluster similarity and inter-cluster dissimilarity;
- providing scalability;

To measure how a cluster conforms above-given features, there are certain metrics [14, 28]. Inter-cluster distance and intra-cluster distances are the most important ones. Intra-cluster distance is the distance between the members of any selected cluster and inter-cluster distance is the distance between clusters. Inter-cluster and intra-cluster distances are shown in Figure 2.4.

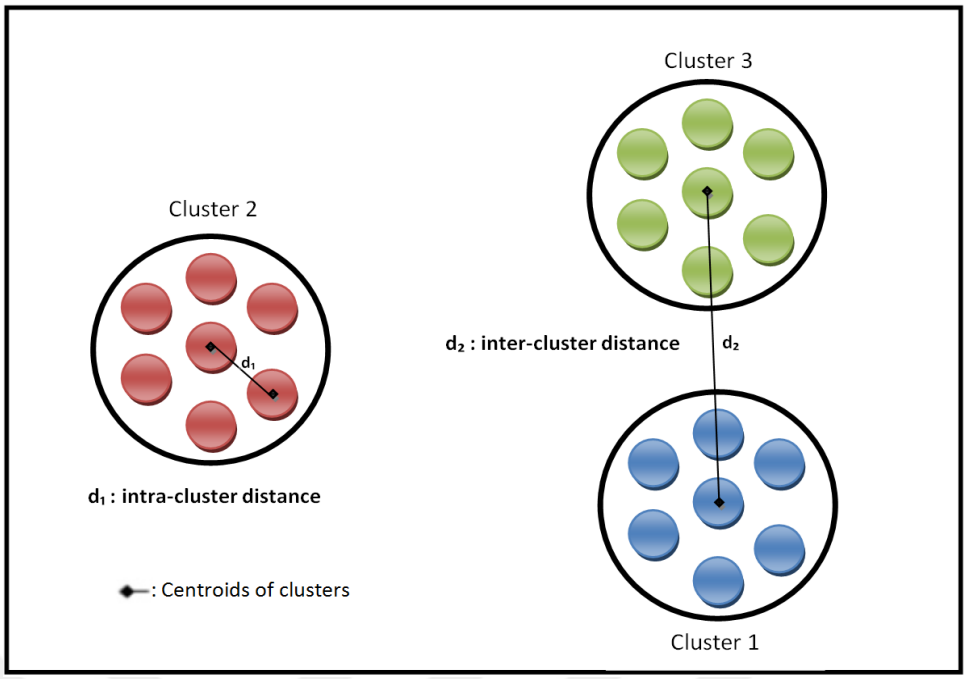


Figure 2.4: Illustration of inter-cluster and intra-cluster distances on the same graph.

To measure distances between two clusters there exist the following techniques.

In Equation 24, Equation 25, Equation 26 and Equation 29, $d(x,y)$ represents the distance between vertices $x \in X$ and $y \in Y$ and, X and Y are two clusters. In Equation 25, Equation 28, and Equation 210, X and Y are two clusters; n_X is size of the cluster X and n_Y is size of the cluster Y.

• In single linkage, the minimum distance between vertices is considered; the formula of single linkage is given in Equation 24.

$$D_{(single-linkage)}(x,y) = \min_{x \in X, y \in Y} d(x,y) \tag{24}$$

• In average linkage, the distance between the two sets is defined as the average distance between all pairs of possible objects having an object of each pair belonging to a separate set. Average linkage is given in Equation 25.

$$D_{(average-linkage)}(x,y) = \frac{1}{n_X n_Y} \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} d(x,y) \tag{25}$$

• In complete linkage, the distance between two clusters is defined as the furthest distance between to vertices in each cluster. The formula of complete linkage is given in Equation 26.

$$D_{(complete-linkage)}(x,y) = \max_{x \in X, y \in Y} d(x,y) \tag{26}$$

• In centroid linkage, the distance between group centers is measured. In equation, c_x represents center vertex of cluster X and c_y represents center vertex of cluster Y. The centroid linkage distance is used in our study to measure inter-cluster similarity. The formula of centroid linkage is given in Equation 27.

$$D_{(centroid-linkage)}(x,y) = d(c_x - c_y) \tag{27}$$

To measure the distances of a cluster's members between each other, intra-cluster techniques are used. Some of them are as follows:

• In average diameter, the average distance between all members in a cluster. The average diameter is used in this thesis to measure intra-cluster similarity. The formula of average diameter is given in Equation 28.

$$D_{(average-diameter)}(x) = \frac{1}{n_X(n_X-1)} \sum_{x=1}^{n_X} \sum_{y=1}^{n_X} d(x,y) \tag{28}$$

• Complete diameter is the distance between two furthest vertices in a cluster. The formula of complete diameter is given in Equation 29.

$$D_{(complete-diameter)}(x) = \max_{x \in X, y \in X} d(x,y) \tag{29}$$

• In centroid diameter, the average distance between all vertices in a cluster and the cluster's centre. In Equation 210, $c_x \in X$ is the centroid of the cluster X. The centroid diameter can be formulated as:

$$D_{(centroid-diameter)}(x) = \frac{1}{n_X} \sum_{x=1}^{n_X} d(c_x - x) \tag{210}$$

2.4 Enzymes and Enzyme Commission Nomenclature

2.4.1 Enzyme

Enzymes are biological molecules (typically proteins) that significantly speed up biochemical reactions in living organisms that take place within cells [52]. They are critical for life and they serve a wide variety of vital missions in the body, from helping speed up chemical reactions in the human body to aiding in digestion and metabolism.

Enzymes have different roles in biological operations. For example, they help the body to break down larger complex molecules into smaller molecules, participates in DNA replication and destroys toxins in the body.

2.4.2 Enzyme Commission Nomenclature

The Enzyme Commission number is a number appointed to enzymes about their chemical functions [20]. The enzymes which have similar EC number, catalyze a similar reaction.

There are six main classes of enzymes known as Enzyme Commission Codes [52], which are given in Table 2.2:

Table 2.2: Enzyme commission main classes.

EC	NAME
EC 1	Oxidoreductases
EC 2	Transferases
EC 3	Hydrolases
EC 4	Lyases
EC 5	Isomerases
EC 6	Ligases

Enzyme codes are composed of four numbers and separated by dots. This classification starts with the letter "EC" and followed by the numbers. The first letter is reserved for main class info, second is for the subclass, third is for sub-subclass and last section is reserved for its serial number. The tree structure of enzyme commission numbers is given in Figure 2.5

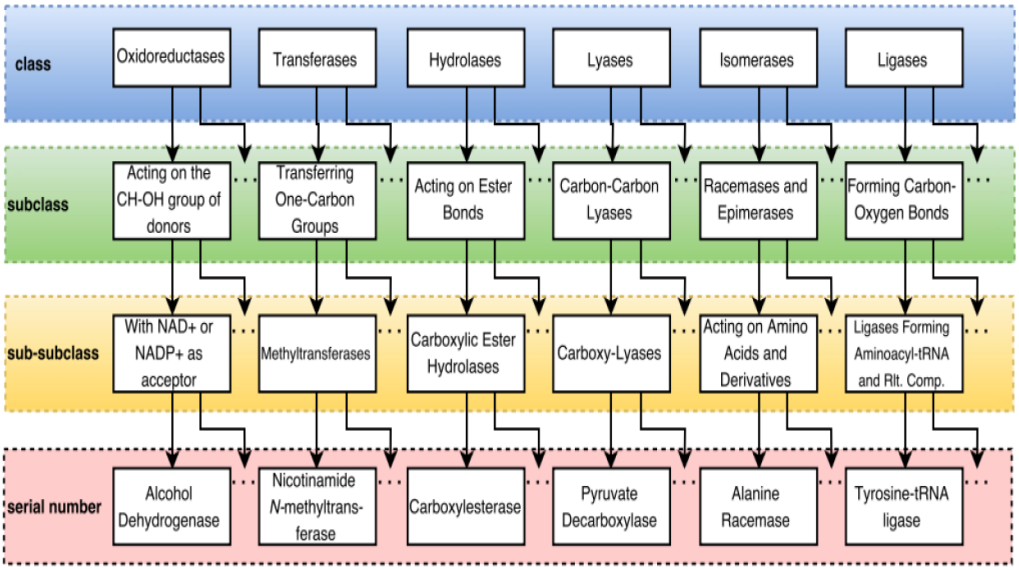


Figure 2.5: Illustration of tree structure of Enzyme Commission Numbers. Levels correspond to class number, subclass number, sub-subclass number and serial number respectively [37].

For example, the interpretation of the enzyme EC 3.4.11.4" [1] is as follows:

-The first number "3" shows us the main class which is "Hydrolases".
-The second number "4" represents us (with previous 3.4) the subclass information which is "Acting on the aldehyde or oxo group of donors".
-The third number "11" represents us (with previous numbers 3.4.11) the subsubclass information which hydrolyzes that cleave off the amino-terminal amino acid from a polypeptide".
-The fourth number "4" represents us (with previous numbers 3.4.11.4) the serial number which is "cleave off the amino-terminal end from a tripeptide".

2.5 EClerize

EClerize [37] is a study on biological graphs that represent pathways in which the vertices are identified with Enzyme Commission (EC) numbers. It is inspired by GOlorize [49] which uses Gene Ontology annotations to draw network visualization. In gene ontology, there are three categories: Biological process, molecular function, and cellular component; GOlorize uses the information of a gene's Gene Ontology (GO) categories to emphasize the biological function genes [49].

EClerize is based on KK algorithm. In addition to KK Algorithm, EClerize uses EC number of vertices for clustering. The vertices which are in the same EC class are treated to be in the same cluster as same cluster members and they are drawn closer to each other. To connect these vertices, the algorithm adds virtual edges between these vertices and strengths of springs of these edges are calculated according to the EC number, which also shows the similarity of enzymes in a cluster. Vertices which do not have EC numbers are treated as regular vertices like they are treated in basic Kamada-Kawai layout algorithm. In Figure 2.6, virtual edges added by algorithm are drawn with red.

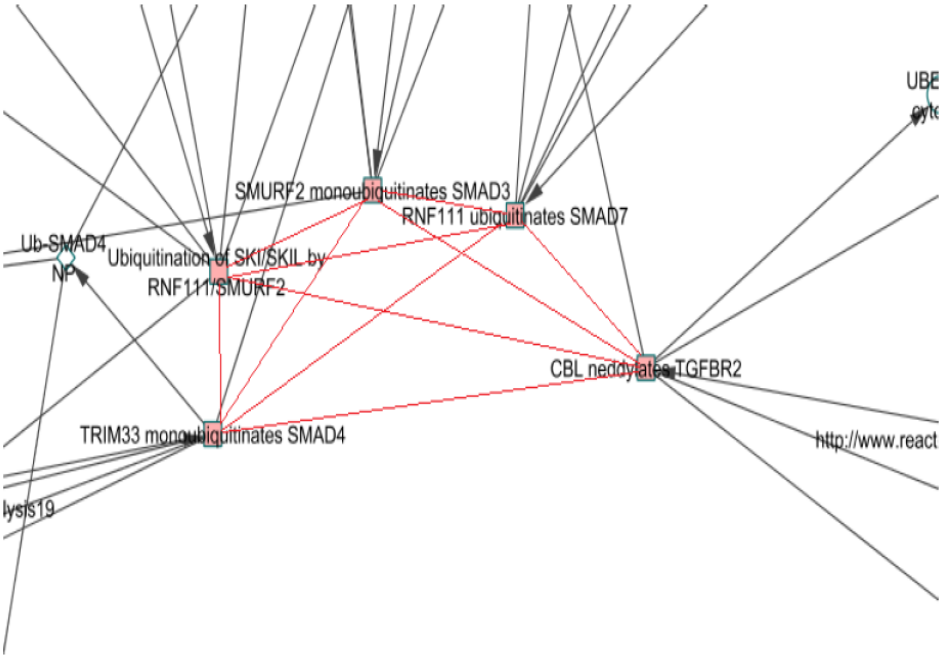


Figure 2.6: Graph with the addition of virtual edges by EClerize algorithm [37].

In EClerize algorithm, the motivation is to minimize intra-cluster distance in any cluster while maximizing the inter-cluster distance between all clusters. There are three parameters which affect drawing graph by EClerize:

- the list of vertices which are associated with an EC number;
- a parameter S to define Strengths of Virtual Edges in KK algorithm;
- a constant c which adjusts inter cluster distance.

The pseudocode of EClerize is given in Algorithm 1.

Algorithm 1 EClerize Algorithm

INPUT: A Graph $G = (V, E)$ List of vertices $V_{ec} = (v \in V)$, v has EC number
constant S, strength of spring of EC edges
constant c, distance factor for clusters

OUTPUT: A nice layout of G

1: Add_Virtual_Edges(G, V_{ec})
2: $dist \leftarrow$ Compute_Distance_Matrix(G)
3: $lengths \leftarrow$ Compute_Pairwase_Ideal_Lengths($G, dist$)
4: $strn \leftarrow$ Compute_Pairwase_Spring_Strengths($G, dist$)
5: Apply_Spring_Model($G, V_{ec}, S, strn$)
6: Layout_Algorithm_Model($G, null, S, strn$)
7: **if** $c \neq 0$ **then**
8: Increase_Inter_Cluster_Distance(G, V_{ec}, c)
9: Layout_Algorithm_Model($G, V_{ec}, S, strn$)
10: **end if**

First, virtual edges are added between the vertices which have the same EC class. Then the distance matrix of the graph is computed. By using this distance matrix, ideal lengths and ideal strengths are calculated. In Step 5, the forces of springs of vertices with EC numbers are adjusted for the ideal strength between vertices by considering EC info. Strengths of vertices are directly proportional to how close they are according to EC. In Step 6 the spring layout algorithm is applied with re-organized values of spring strengths. If the parameter for adjusting inter-cluster distance is not zero, first clusters are moved away from each other and then Step 6 is repeated with a difference, this time vertices which are members of a cluster are not moved.

2.6 Heuristic Approaches

An local optimal solution is a solution for which no better feasible answer can be found in the immediate neighborhood of the given solution. In mathematics and computer science, a local optima is the best solution to a problem within a small neighborhood of possible solutions. This concept is in contrast to the global optima, which is the optimal solution when every possible solution is considered.

A globally optimal solution is a feasible solution with an objective value that is better than all other possible solutions. There's no general way to know that the found local optimum result is global. However, in some cases the fitness of the global optima or some bound on its value may be known, so one can check the optimality of the found solutions by just inspecting its fitness value, these cases are quite rare and only applicable on convex problems.

In real-world problems, the case of convexity is not frequently seen; the search space is also huge for expecting that found local optima is the global optima. Especially for NP-complete problems, there is no way to exhaustively check every candidate (local optimum) solution for being sure whether it is the best or not. There are some techniques which can be applied to solve computational problems in general; one of them is use of heuristics.

A heuristic is a technique or a strategy that provides solving problems faster or for finding a valid answer when classic methods are not enough to find an approximate solution. There are some cases where heuristic is useful. For some problems, there is no time to find and verify the found solution is the most optimal solution; on the other side, on some problems the incomplete information for the solution is the biggest problem. In both cases, by applying some arbitrary choices, heuristics find not too bad solutions which can be called as good enough. There is no prescribed instructions or formulas about the execution of heuristics. On the contrary, according to the type of problem, heuristics can be used with various innovative and initiative ways to explore solutions.

There are various algorithms which are heuristic. In our study we use Genetic Algorithm (GA), and GA is described in Section 2.7.

2.7 Basic Genetic Algorithm

Genetic Algorithm (GA) is a branch of evolutionary computing. In evolutionary computing, routines depend on a heuristic random search. GA is inspired from nature based on evolution. The process of evolution by natural selection, which has been observed in nature, forms the basis of GA. The natural way of evolution is the basis of steps of GA.

Different kinds of discrete optimization problems can be solved with the help of GA [56]. There are various types of problems, from control optimization [57] to social-network visualization. GAs are used to reach 'nearly the best' answers for optimization and search problems [48]. For many problems, the main purpose of using GA is finding a solution which is a close approximation of the optimal solution [59].

In GA, a population consists set of individuals and an individual is a solution to the problem. In GA, the fitness score of an individual shows how much an individual is close to the optimal solution or it's a comparison for the chosen individual with other individuals in the same population.

In this section, for better understanding of the GA, the steps of the GA are explained by visualizing common genetic concepts. GA has some steps which form the basis of this heuristic and GA applies these steps in the given order:

• [Start] The initial condition of GA. In this step, the population is created from given dataset by considering the parameter of population size. A sample initial population for a toy example composed of genes whose elements are bits is given in Figure 2.7. In this figure, an individual is represented by a chromosome which consists of genes. The population consists of a group of chromosomes (individuals).

• [Fitness] Evaluation of the result for each individual according to some desired requirements. In our study, the aesthetic criteria of undirected graphs are used for the fitness of GA. A sample illustration on the finesses of individuals is given in Figure 2.8.

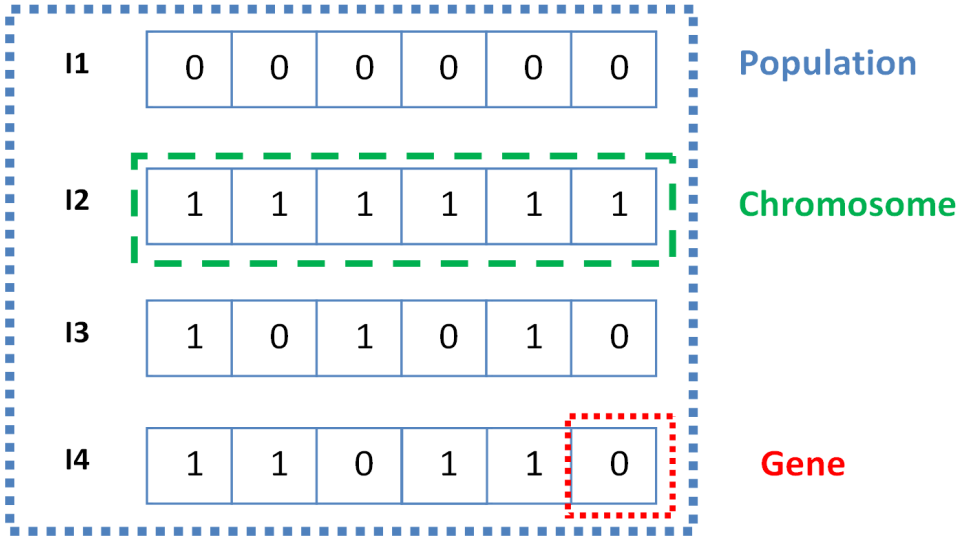


Figure 2.7: An example initial population of the GA. I1 indicates an individual. An individual is represented by a chromosome. A chromosome consists of genes. A group of chromosomes (individuals) generates the population.

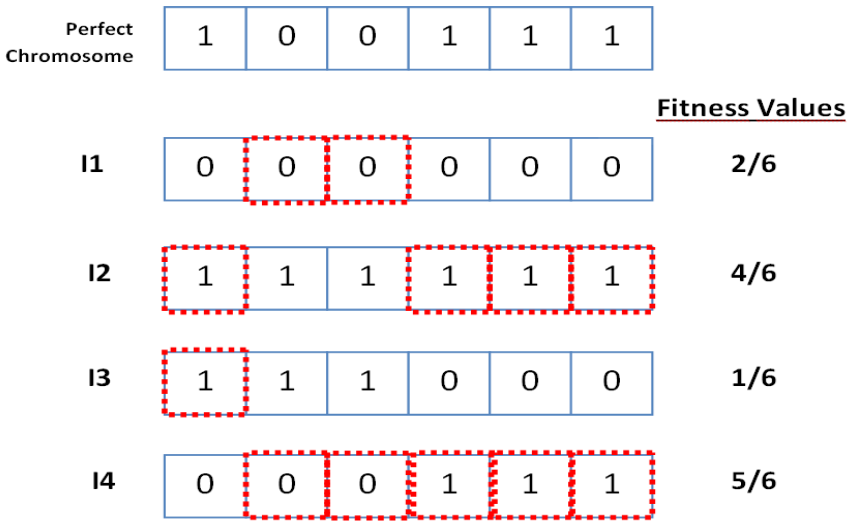


Figure 2.8: An example of the fitness values of the population. The fitness value of each individual is calculated by, comparing each gene of the selected individual with the corresponding gene of the perfect chromosome.

• [Selection] In GA, the selection part performs a critical role in the creation of new generations. In this phase, current generation's individuals are selected, to be used for the creation of a new generation, with a randomized technique proportional with their fitness value. In the selection phase, the individual who has better fitness score has a higher chance to be selected. The selection has a significant role in improving our populations by discarding bad individuals and only keeping the best individuals who have better fitness values in the population. In selection, the elitism is also used. Elitism is passing of the best individual from the current generation to the next generation. This technique guarantees that there isn't any generation whose best individual is less fit than the its previous generation, elitism blocks devolution in GA.

• [Crossover] An operation to produce two new offspring from chosen parents. The accustomed technique is producing new offspring from two parents, but even in bioinformatics, some studies show us that more than two parents can be used to generate offspring [21, 64]. In some exceptional cases to produce new offspring, crossover is not performed, new individuals could be a direct copy of parents [8]. An illustration of crossover is given in Figure 2.9.

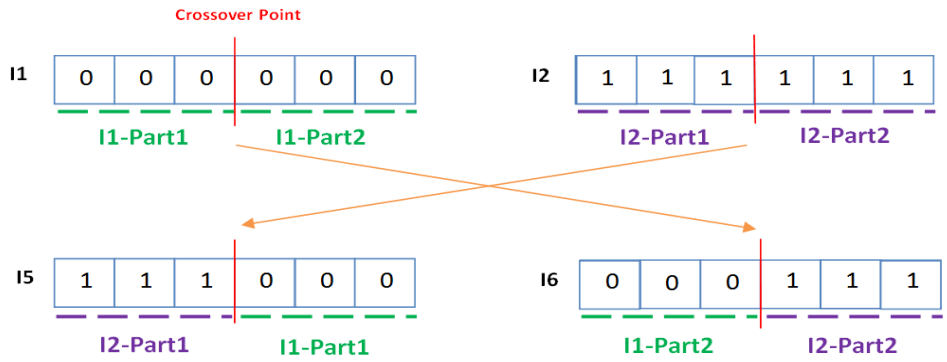


Figure 2.9: An illustration of crossover. I1 is the first parent and I2 is the second parent. The 'Crossover Point' is a tool to separate chromosomes into gene sections. I5 and I6 are children. I5 consists of the first part of I2 and the first part of I1, I6 consists second part of I1 and the first part of I2.

• [Mutation] An operation which provides GA to escape from local minima. With mutation, individuals have minimal changes randomly with predefined mutation techniques. An illustration of mutation is given in second figure.

(a) Before mutation (b) After mutation

Figure 2.10: Illustration of mutation technique. a) The chromosome before mutation. b) After the mutation of the given chromosome, one gene has been changed, which is colored with red.

• [Termination] End condition/conditions to finish the algorithm. There are different conditions to terminate your genetic algorithm:

-Finding an individual which is the best one in the global solution space.

-Exceeding counting or timing limits in the program.

-By using statistical techniques, detecting the state in which there is not enough improvement for individuals to continue.

• [Continue] In case of the termination condition is not satisfied, the system of iteration takes the role to continue GA. By passing mutated new population to Fitness step, the algorithm proceeds to search the best solution.

GA has pretty useful advantages on performed problems:

1. First of all, the implementation of GA is considerably easy. Outline of its steps and order of these steps are similar for most of the problems.

2. GA does not get stuck into local minima. With its indeterministic behaviors, GA can find almost the best solutions.

3. One of the advantages is their parallelism. With this property, GA could be run on larger populations. This also makes it less likely to get converge in local extremes [57].

Besides these advantages, GA has a couple of disadvantages. The main problem of GA is in computational time. Like other heuristic approaches it's not deterministic, which makes it slower than some other methods [41]. Another disadvantage is probable high memory consumption, depending on the individual and size of population supplying required memory can be a challenging task. Non-deterministic nature of GA also makes hard to predict its performance in a stable manner [58]. In conclusion, GAs perform well at effectively searching on problems which have large non-convex solution space and alternative solution technique is brute force.

In the literature, GA is integrated with force-directed algorithms in various forms. One of them is the role of initializer; in this type of usage, GA runs only one time on initial data before the force-directed algorithm is started. The second option is the use of the GA as a post-processor, in this approach after the end of the selected force-directed algorithm, the GA process result data for optimization. The third alternative is the use of the GA as a fine-tuner, at the end of each iteration of the selected force-directed algorithm, the GA performs on the result data for passing the data in a better form to the next iteration of the force-directed algorithm.

2.8 Related Studies

In this section, we explain previous studies on the problem of drawing undirected graphs by using force-directed algorithms with heuristic approaches. In the literature, there are various studies which use genetic algorithm (GA) or simulated annealing (SA) to draw undirected graphs. Some of them also use force-directed techniques along with these heuristic approaches. Comparison of previous studies with their approaches and contributions is given in Table 2.3.

Table 2.3: Comparison of previous studies with their approaches and contributions.

Article	Year	Approach	Contributions
Drawing Graphs Nicely Using Simulated Annealing [17]	1996	SA	First technique for drawing undirected graphs by using a heuristic technique.
Using Genetic Algorithms for Drawing Undirected Graphs [13]	1998	GA, Eades	Eades algorithm is used as a fine-tuner of the GA.
Graph Layout Using a Genetic Algorithm [9]	2000	GA, KK	Energy function of KK used in the fitness function of GA.
TimGA: A Genetic Algorithm for Drawing Undirected Graphs [63]	2001	GA	This is the first article that aims to draw graphs directly by using GA.
Drawing Undirected Graphs with Genetic Algorithms [54]	2005	GA	Usage of inversion in mutation.
Simulated Annealing as Pre-Processing Step for Force-Directed Graph Drawing [24]	2016	FR, SA	SA is used as a pre-processing step for FR algorithm.

Davidson and Harel [17] describe one of the earliest studies on graph drawing by using a heuristic approach. In their study, nice-looking undirected graphs are drawn with the help of the simulated annealing. One of the critical aspects of this study is a flexible selection mechanism for choosing aesthetic criteria. Users can

choose their preferred aesthetic criteria with the use of a configurable interface, which consists of a lot of aesthetic criteria such as the number of edge crossing and size of the drawing area. With this support, for given problems, different solutions are provided considering user choices. However, their study has also certain problems. The long execution time is the most critical problem of this study, so that, execution of the graphs which have a large number of vertices is not recommended by the authors. In addition to the long execution times, the used displacement technique, where vertices are located on a radius, does not allow this study for running on large datasets.

The study by Branke et. al [13] describes a similar graph drawing technique with a different approach. In this study, a force-directed algorithm is used as a fine-tuner of the GA. After each iteration of GA, the force-directed algorithm runs on each individual, then the new generation is generated from these processed ancestors. In this study, Eades's algorithm [18] is also used, which is a well-known force-directed algorithm. Branke et. al have observed a set of important things. They have seen that if they want to draw aesthetically nice graphs, they need to conform some criteria [46]. Force-directed algorithms are not successful enough on global optimization and may depend on the initial layout. GA is a successful algorithm for global optimization on various problems. According to them, besides its ability to find the globally optimal solution, GA is not good at tuning the graph. These observations motivated Branke et. al. to use the GA for graph-drawing with the help of a force-directed algorithm as a fine-tuner. In each iteration, after a new offspring is produced, their study uses Eades' algorithm [18] to fine-tune the new individual. The biggest problem in this study is that running time performance is quite costly. This is an obstacle to the use of graphics with a large number of vertices.

The work done by Barreto and Barbosa [9] plays a vital role in using an evolutionary algorithm along with a force-directed algorithm. In their study, the Kamada-Kawai (KK) algorithm is used with GA; this is a new approach that hasn't been done before. In our study, in addition to these techniques clustering algorithms are also used for visualizing given data in a form of clustered vertices. They choose KK [42] among other force-directed techniques such as VLSI [55] and Eades [63], because they find KK as a more mechanically realistic model. They use Kamada-Kawai (its

energy function) as the fitness function of the genetic algorithm. The KK algorithm's energy function is not the single effective factor on the GA's fitness function; several aesthetic criteria are also considered in multilevel modeling (MM) technique. MM is an effort to account for both individual and group levels effect simultaneously within a model [35]. By this way, they have developed a hybrid technique which still inspires researchers for recent studies. They use dynamic weighting for aesthetic criteria, with this technique users can set coefficients of each fitness criteria. By this way, different weightings generate distinct graphs for the same dataset. A sample illustration of their study is given in Figure 2.11. This figure shows us that, in the usage of GA for drawing graphs, with a higher number of iterations better graphs are obtained.

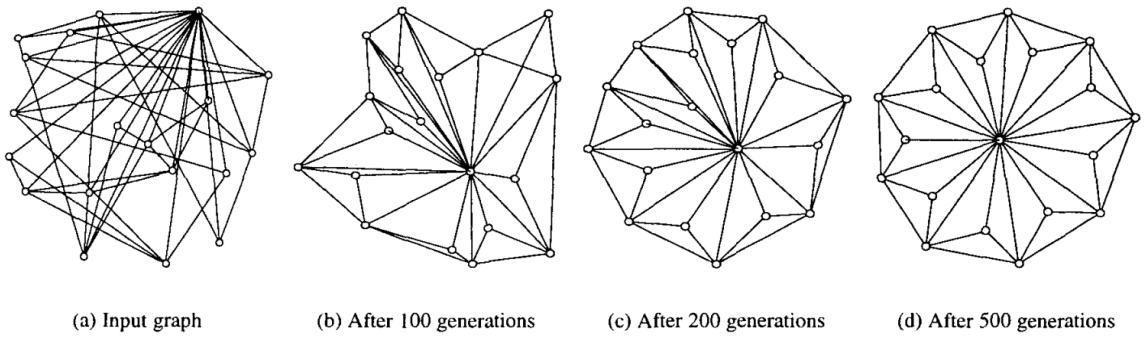


Figure 2.11: Layout of Barreto and Barbosa's algorithm in action [9].

Timo and Eloranta [63] have conducted one of the most comprehensive study in this field. This is the first study, where an undirected graph is drawn by using the genetic algorithm, without help of any force-directed algorithm. By considering some aesthetic criteria such as edge crossings and edge length deviation; their study produces nice-looking graphs. They have also implemented some new crossover techniques on graphs. In Figure 2.12, one of the crossover techniques used in this study is illustrated. First, the square, to be used in the selection of the vertices, is placed randomly on parent individuals. All of the vertices of each parent individual are transferred to its child without any modification. Then the same size rectangle with different coordinates is placed on the child individuals. The

child individuals exchange the positions of the vertices, which are selected in parent individuals (e.g., for child one the vertices selected on parent two are moved), inside the chosen rectangles and the rest of the vertices are kept unchanged. These new crossover techniques are based on the exchange of the vertices between two selected individuals. The crossover techniques in this study inspired us to develop our crossover mechanisms. In this study, some mutation techniques are also used; these techniques are introduced in prior studies by Groves et al [45]. A small number of vertices are randomly displaced within a certain range in used mutation techniques. A common point with previous studies is that they work on a small number of vertices too.

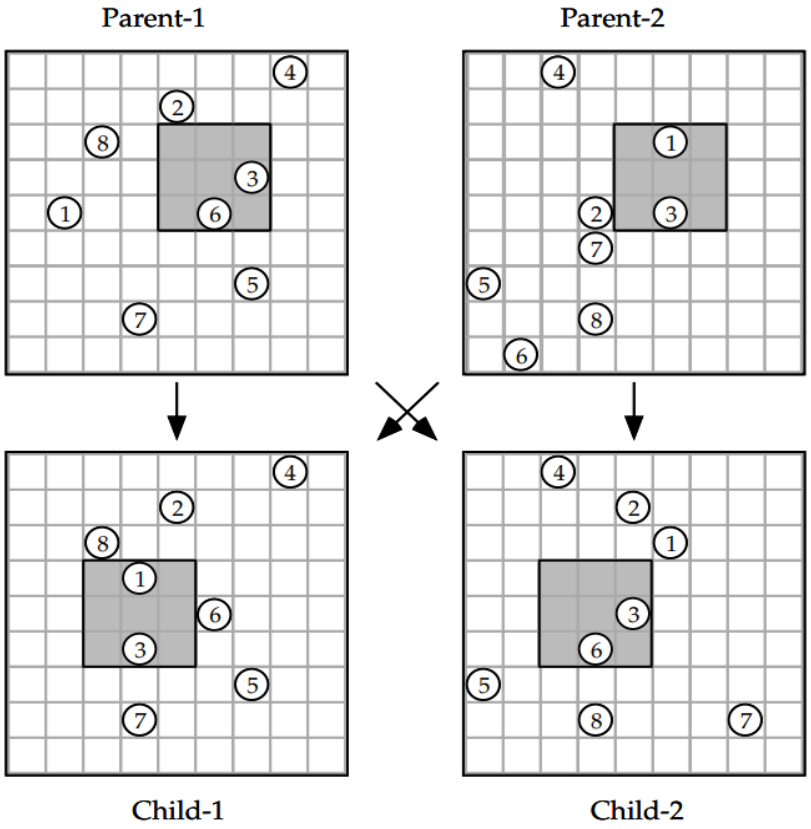


Figure 2.12: Rectangular crossover from TimGA [63].

Zhang [54] proposes an improved genetic algorithm for drawing undirected graphs. One of the significant improvement of this study is the use of inversion. The statement of the author about inversion is 'Inversion works by randomly selecting two

inversion points within a chromosome and inverting the order of genes between the inversion points, but remembering the gene's meaning or functionality.' . By using inversion, the randomness in the mutation is supported. Two types of mutation are also used, 'Single vertex mutation' and 'Exchange vertex mutation' these have an inspiration to use similar techniques in our study. With respect to some aesthetic criteria [61], authors claim that they produce better graphs than the ones produced by simple genetic algorithms [63], which is the original force-directed algorithm [18]. In this study, the termination mechanism of GA needs to be improved. In the final version of the study, as the author indicates 'the termination condition is just a check whether the algorithm has run for a fixed number of generations' [54]. By using certain adaptive techniques, this fixed approach can be avoided. In our study, by controlling the improvement between consecutive generations; our study can decide whether to terminate or not.

A study by Tossini et. al [24] uses an evolutionary algorithm with a force-directed algorithm in a different way, SA is proposed as a pre-processing step. They have thought that starting with a randomized initial graph decreases the performance of the force-directed algorithm [67, 42]. In their work, the number of edge-crossings is reduced by using a new approach to the initial graph. After the random initialization step, the graphs are pre-processed with SA, then these pre-processed graphs are used as initial graphs. They have been inspired by a prior work where instead of SA, GA is used by Toosi [23]. Fruchterman-Reingold's force-directed graph-drawing algorithm [27] is also used in this study. The results show us that with a small number of vertices, this technique generates successful graph layouts with decreased execution time of the force-directed algorithm. On the other hand, reducing the randomness of the initial graph may affect the resulting graph to be stuck into the local optima.

After the survey on above-given documents, we have reached a set of important points:

- The force-directed algorithms are already being used in combination with the heuristic approaches.

- There are various force-directed algorithms in use, and KK is among the most

preferred ones.

• A heuristic approach can be used in different roles such as a fine-tuner, a post-processor, or an initializer.

• For drawing undirected graphs, instead of SA, GA is more preferable because of its large solution set.

• Use of a force-directed algorithm as a fine-tuner has the biggest positive effect on the creation of nice-looking graphs.

• Most of the studies are not at the desired level in terms of execution time.

• In most of the studies, small graphs which have less than 100 vertices are used.

• The studies where the genetic algorithm or a similar heuristic approach is used, in contrast to the force-directed algorithms, the clustering is not employed before.

CHAPTER 3

ECLERIZE TYPE GA

3.1 Overview

Our work, named EClerize Type GA, is an improvement of EClerize [37]. We have integrated the previous study EClerize with a well-known heuristic approach, Genetic Algorithm (GA). The object of our study is to avoid the prior study's result from local-minima and to obtain global optimum solutions for EClerize.

In our study, the most effective improvement is the use of a heuristic to draw an undirected graph with the help of a force-directed algorithm as a fine-tuner. A fine-tuner makes small arrangements for something to achieve the best or the desired performance. In Tossini's study [24], this idea is conducted by the use of SA with the help of a spring-embedded technique by Eades [18]; we have implemented a similar study with some variations. Instead of SA, we have applied GA. Also, we haven't chosen the Eades algorithm for the selected force-directed algorithm. Instead, by using EClerize, we use KK as well. KK is a force-directed algorithm, which is used in native EClerize. In addition to these changes, in detail, there are pretty much differences from crossover techniques to selection algorithms.

We have built a system which supplies a controllable platform to the users with the help of presented parameters. In addition to the native EClerize parameters, we introduce some new parameters which are effective on the resulting graph.

In general terms, our work generates a graph population by considering the population size; then it runs GA on this group of graphs with the use of native EClerize as a fine-tuner. In the end, it finds the best graph and shows it to the users. The

pseudocode of EClerize Type GA is presented in Algorithm 2.

Algorithm 2 EClerize Type GA

INPUT: A Graph G = (V,E)
constant $size$, GA population size
$counter$, GA iteration counter
$settings$, Program Settings

OUTPUT: The Best Layout of G

1: $population \leftarrow$ Initialization(G, $size$, $counter$)
2: $fitnesses \leftarrow$ Calculate_Fitnesses($population$, $settings.fitness\_settings$)
3: $terminate$ = false
4: **while** $terminate$ is not true **do**
5: $selected\_parents \leftarrow$ Select_Parents($population$)
6: $population \leftarrow$ Do_Crossover($population$, $selected\_parents$ $settings.crossover\_settings$)
7: $population \leftarrow$ Do_Mutation($population$, $settings.mutation\_settings$)
8: $population \leftarrow$ Run_EClerize($population$, $settings.eclerize\_settings$)
9: $fitnesses \leftarrow$ Calculate_Fitnesses($population$, $settings.fitness\_settings$)
10: $terminate \leftarrow$ Check_Termination($population$, $settings.termination\_settings$)
11: **end while**
12: $best\_individual \leftarrow$ Choose_Best_Graph($population$)

As it is seen in Algorithm 2, the Algorithm of EClerize Type GA has various input parameters. Constant $size$ defines the population size, parameter $counter$ is used as the iteration counter of GA to terminate algorithm, the parameter $settings$ stores the program settings. In the algorithm, the initialization method is called first. This method performs a few preprocessing steps from loading sub-modules to preparing a logging mechanism; the initialization method returns an object, which is a new population containing randomized graphs as individuals. After Step 1, the calculation method is called to measure each individual's fitness values, then the

program starts to the main cycle. In this cycle, each step of GA, from Step 5 to Step 10, are called and EClerize is used as a fine-tuner in Step 8. This cycle is terminated when one of the termination condition occurs. Following the termination of this cycle, the best graph is chosen and it is shown to the users. In onward sub-sections, details of the algorithm are described.

3.2 Genetic Algorithm on EClerize Type GA

3.2.1 Initial Condition

Initialization part of the main algorithm is responsible for preliminary preparations. In this phase, the main objective could be split into two parts. The first part is interested in reading native EClerize settings. There are several parameters required by the previous study. As they are presented in Algorithm 1, there are some parameters used by native EClerize; the list of vertices which have EC number, parameter S to define Strengths of Virtual Edges in KK algorithm and a constant c which adjusts the inter-cluster distance. These settings are essential for the native EClerize. The remaining part of the settings serves for other than native EClerize, this part is reserved for the settings of the GA; by using these settings, our improvements perform properly. There are different types of settings:

• Native GA Settings: These parameters are used in the main operations of GA. A parameter for the population size and a constant for maximum iteration to terminate GA are these type of settings.

• Fitness Settings: This set keeps specific data for fitness calculation. These are configurable by end-users for flexible fitness measurement with different aesthetic criteria. For each fitness measurement, there is a weight to determine the level of impact of the selected measurement on overall fitness value. By setting a criterion's weight to zero, the criterion which uses this weight is made ineffective. By arranging coefficients of different fitness measurements, different graph layouts can be drawn.

• Mutation Settings: This set holds specific settings for the mutation mecha-

nism. A setting is reserved for keeping the mutation ratio, in GA with the usage of this parameter, the status of whether or not the mutation to be performed is determined. Other members of this setting section are used to determine the type of mutation. In the case of the mutation occur, there are different types of mutation techniques to be operated. For each mutation technique, there is a setting to keep its probability. After reading the mutation parameters, the parameters are transformed into a usable form where the probability of occurrence of each mutation operation is represented as percentage weight.

• Crossover Settings: This set keeps the specific settings for crossover mechanism. A setting is reserved for keeping crossover ratio, in our study with the help of this parameter, crossover status is determined. In the case of crossover does not occur, the selected ancestors pass without any change to the next generation. Other members of this set are used to determine the type of crossover. In the case of crossover occurs, different types of crossovers service in our study. For each crossover, there is a setting to keep its probability of occurrence. After reading the parameterized crossover settings, our study transforms these settings to a usable form where the probability of occurrence of each technique is represented as percentage weight.

In our study, first, graph data such as vertices and edge lists are read. Then, the algorithm checks the vertices for existency of EC attributes. This operation is required by the native EClerize for clustering vertices. The algorithm adds virtual edges to the vertices which have the same EC class numbers. After the creation of the base graph, now which has virtual edges associated with the EC tree; the algorithm generates a population which consists of graphs as individuals. For the population generation, in addition to the base graph, the parameter reserved for population size is also used. A new graph is produced by changing the positions of vertices of the base graph randomly in the drawn area. Each generated graph has own randomized vertices, which are adjusted on the subsequent steps.

In this part, in addition to parameter preparation and population generation, one extra task is executed for loading worker modules such as the crossover and the mutation modules. The pseudocode of Initial Condition is presented in Algorithm 3.

Algorithm 3 Initial Condition

INPUT: $G = (V, E)$, A Graph
$size$, GA population size
$settings$, Program Settings

OUTPUT: New Graph Population $population$

1: $settings \leftarrow$ Read_And_Manipulate_All_Settings(G, $settings$)
2: $population \leftarrow$ Generate_New_Population($size$, $settings$)
3: Load_Modules()

3.2.2 Fitness Calculator

In our study, we supply a fitness mechanism to the users that allows them to choose which fitness measurement is active and how much it affects the overall fitness value. In this study, various aesthetic criteria of undirected graphs are served as fitness criteria in fitness calculation of GA.

Each fitness measurement serves to rate the quality of a graph from a different perspective. In general, researchers associate aesthetic more than nice-looking; they identify aesthetic with readability and understandable [12]. Fitness measures in this study are given below:

• Minimum Vertex Distance Sum: The sum of the distances of each vertex to the closest vertex on the same graph.

• Minimum and Maximum Vertex Distance: The maximum and the minimum values in the graph, which are the distances between two vertices.

• Edge Length Deviation: The length of each edge is measured and compared to the overall edge length. The average edge length is calculated by dividing the sum of all edge lengths on the graph by the number of edges.

• Number of Edge Crossings: Two different edges cross in a graph drawing if their geometric representations intersect; in our study, the number of the all intersections in the graph is used for fitness calculation.

• Drawing Area: Size of the area for drawing a graph, calculated by using the extreme vertices in the graph.

• Inter-cluster Distance: The distance between clusters. In our study the clusters are type of Enzyme Commission (EC) clusters, which consist of the vertices which have clustering information dedicated to EC number.

The pseudocode of Fitness Calculator is presented in Algorithm 4. First, the algorithm checks the fitness measures to be performed, then for each graph, it runs the selected measurements and obtains each individual's fitness values in parallel. Then, a comparison between individuals is conducted and the best individual is accepted as a hundred percent, then the rest of the individuals are calculated in proportion with the best one. After the calculation of the proportional percentage for each measurement, the final fitness values are computed. The fitness set, previously described in Section 3.2.1 part of 'Fitness Settings', determines how much a selected measurement affects overall fitness value. By using each measurement's fitness setting with its computed overall fitness value, the final fitness values are produced.

Algorithm 4 Fitness Calculator

INPUT: $population$, Graph Population
$size$, GA population size
$settings$, Program Settings

OUTPUT: Overall fitnesses $fitnesses$

1: $applied\_fitnesses \leftarrow$ Get_Applied_Fitnesses($settings.fitness\_settings$)
2: **for all** $fitness \in applied\_fitnesses$ **do**
3: $raw\_fitness \leftarrow$ Calculate_Fitness($fitness$, $population$)
4: **end for**
5: $proportional\_fitnesses \leftarrow$ Calculate_Proportional_Fitness(raw_fitness)
6: $overall\_fitness \leftarrow$ Calculate_Overall_Fitness(proportional_fitness)

3.2.3 Selection

Selection is practiced by using the overall fitness values of individuals which are obtained from the fitness calculation, as described in Section 3.2.2. In our study, the elitism is used for the selection.

For all iterations, the size of the population is the same. For providing the elitism, at the beginning of the selection phase, the best individual in the current generation is copied to the next generation without any modification. After that, for every two new individuals, two ancestors are selected randomly with our selection technique. In this technique, the chances of selection of individuals are directly proportional to their overall fitness values. Creation of two new individuals from the selected parents is described in Section 3.2.4. The pseudocode of selection is presented in Algorithm 5.

Algorithm 5 Selection

INPUT: $population$, Current Graph Population
$fitnesses$, Population Fitnesses
$size$, GA population size
$settings$, Program Settings

OUTPUT: $chosen\_parents$, Chosen parents

1: $new\_population \leftarrow$ Add_Best_Individual($population$, $fitnesses$)
2: **for** i = 1 to ($size$ / 2) **do**
3: $first\_parent \leftarrow$ Get_Individual($population$, $fitnesses$)
4: $second\_parent \leftarrow$ Get_Individual($population$, $fitnesses$)
5: $chosen\_parents \leftarrow$ Insert_Parents_To_Chosen_List($first\_parent$, $second\_parent$)
6: **end for**

3.2.4 Crossover

In GA, the crossover has a critical role in reaching the global optima. It's the source of diversity with mutation; in GA, diversity has the most critical role in avoiding local optima.

We apply crossover specifically for undirected graphs. After the step of fitness calculation, the algorithm continues with selection. To generate every two new individuals, two-parent individuals from the current generation are selected with a probabilistic technique which considers individuals' fitness values. Selection is described in detail in Section 3.2.3. After the selection of ancestors, the part of crossing these two individuals begins.

First, with the help of the crossover ratio settings, the decision of whether crossover to be applied or not is decided. If there is no crossover, two-parent graphs are passed without any modification to the next generation. If the crossover is to be applied, it is necessary to determine which crossover technique is to be used. A set of particular parameters, for keeping the selection ratio of each crossover technique, are used in this phase.

We have operated the crossover techniques which do not cross two selected individuals equally. Due to the integrated nature of the graphs within themselves, it is not easy to combine the genetic heritage from both parents equally into a new child. In our crossover techniques, the majority of the new individual is taken from only one of the parent individuals, while the remaining small amount is taken from the other individual.

There are two types of crossover techniques used in our work:

• Rectangular Crossover: One of the used crossover technique is inspired by Timo and Eloranta's study [63]. A sample illustration is given in Figure 3.1. In the rectangular crossover, first, two rectangles are placed randomly on parent individuals with different locations. Figure 3.1a shows the first parent with a placed rectangle and the vertices in this square colored with blue and Figure 3.1b shows the second parent with a placed rectangle and the vertices in this square colored

with red. All of the vertices of each parent are then transferred to their child except the vertices existing in the rectangle of other parent. In the given example, in the first child, all of the vertices except the vertex number 8 are transferred from the first parent. Then the remaining vertices existing over other parent's rectangle are transferred to the children. In the given example, in the first child, the vertex numbered with 8 is transferred from the second parent. The result state of the generated two new individuals are given in Figure 3.1c and Figure 3.1d.

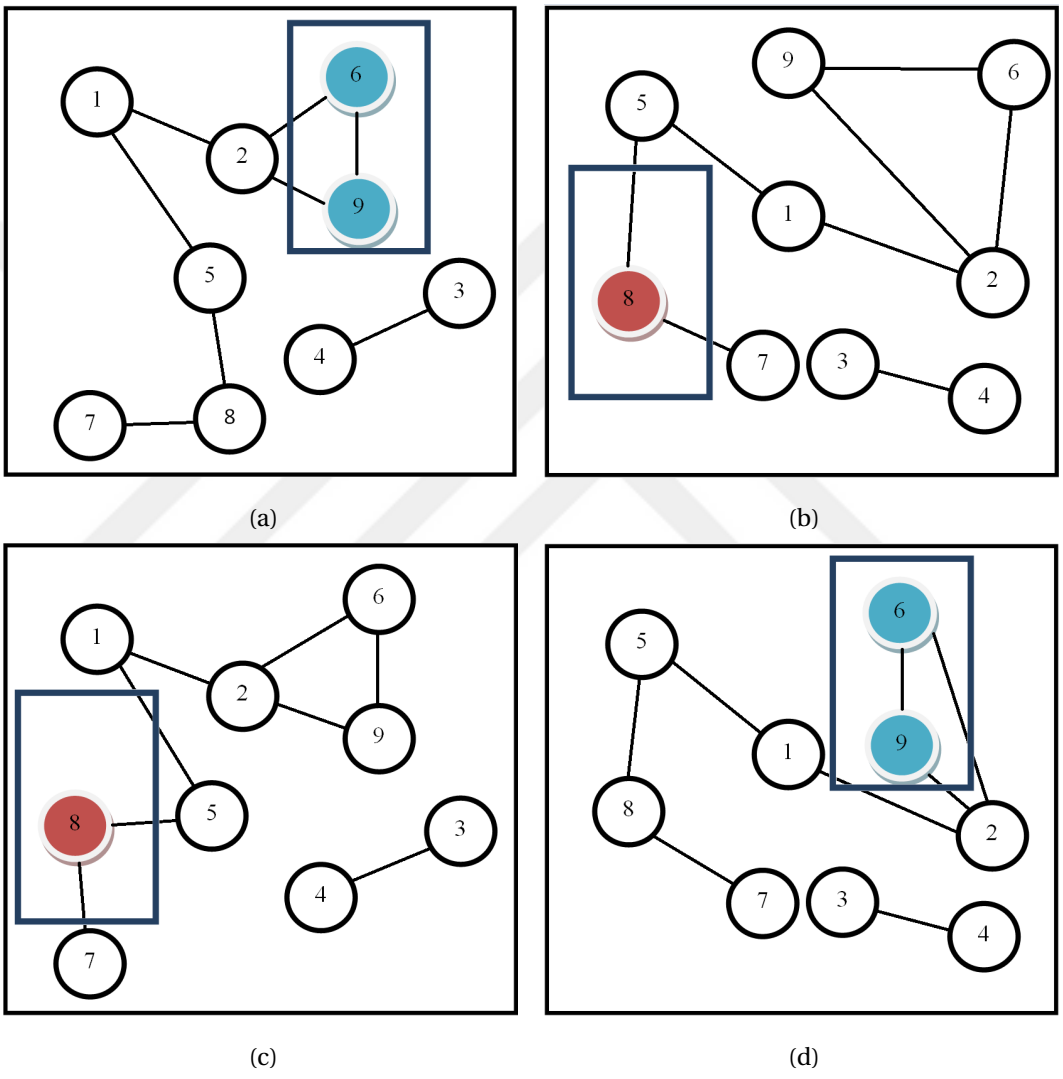


Figure 3.1: Illustration of rectangular crossover technique. In this figure, the edges are not shown to avoid complexity. a) First parent graph. b) Second parent graph. c) First child graph. d) Second child graph.

• Three Vertices Crossover: A sample illustration is represented in Figure 3.2. In this crossover technique, first, for each parent graph, the three connected vertices for which two of them have only one edge are selected randomly. Figure 3.2a shows the first parent with selected three vertices colored with blue and Figure 3.2b shows the second parent with the detected three vertices colored with red. All of the vertices of each parent individual are transferred to their child, then the child individuals swap the three-node vertices, and the rest of the vertices are kept unchanged. Generated two new individuals are given in Figure 3.1c and Figure 3.1d.

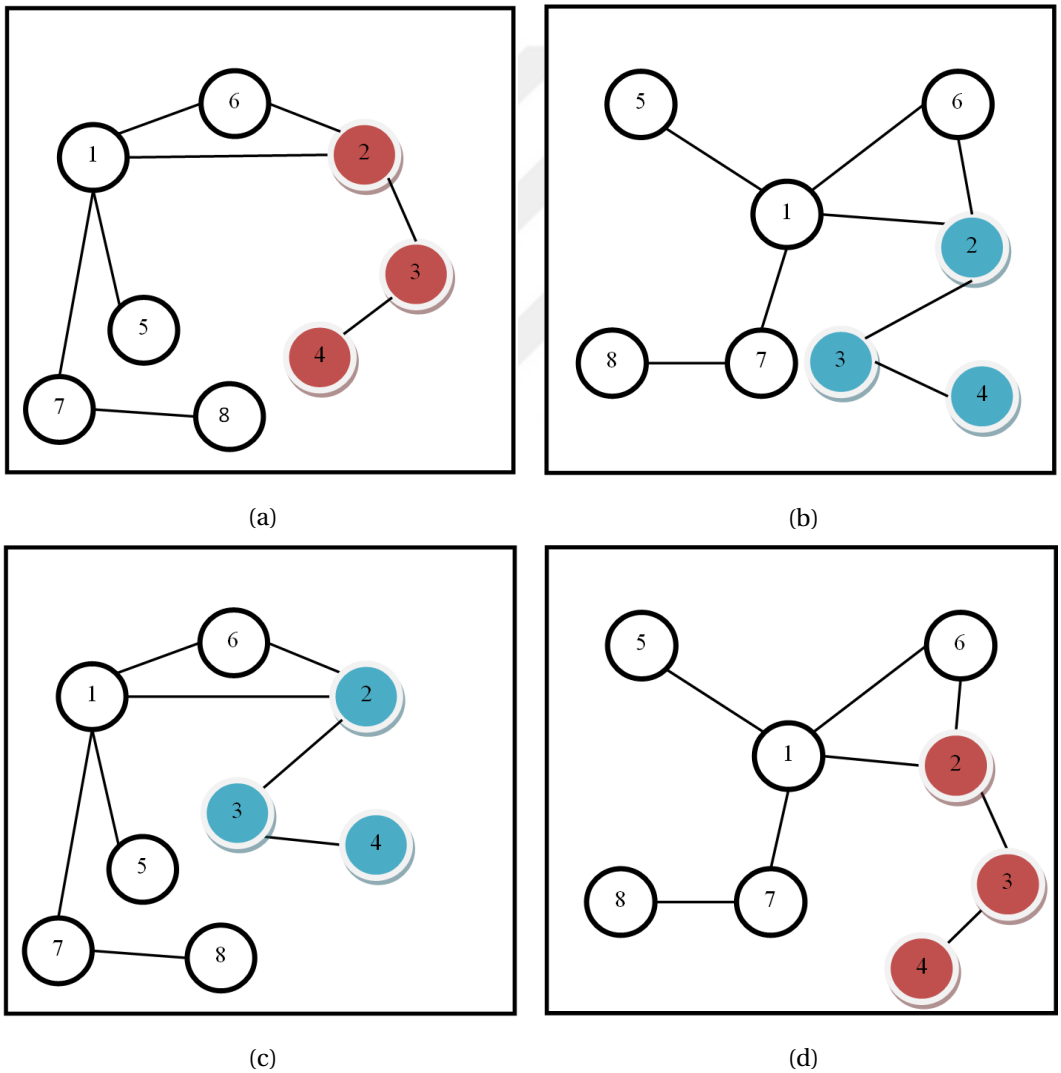


Figure 3.2: Illustration of three vertices crossover. a) First parent graph. b) Second parent graph. c)First child graph. d) Second child graph.

The pseudocode of Crossover is presented in Algorithm 6.

Algorithm 6 Crossover

INPUT: $chosen\_parents$, Chosen Parents
$population$, Population
$settings$, Program Settings

OUTPUT: $new\_population$, New population

1: **for** i = 1 to $chosen\_parents.size$ **do**
2: $first\_parent \leftarrow chosen\_parents$[i].$first\_parent$
3: $second\_parent \leftarrow chosen\_parents$[i].$second\_parent$
4: $do\_crossover \leftarrow$ Decide_Crossover($settings.crossover\_settings$)
5: **if** $do\_crossover$ is false **then**
6: $new\_first\_individual \leftarrow$ first_parent
7: $new\_second\_individual \leftarrow$ second_parent
8: **else**
9: $selected\_crossover \leftarrow$ Decide_Crossover_Technique($settings.crossover\_settings$)
10: $new\_first\_individual \leftarrow$ Do_Crossover($first\_parent$, $second\_parent$, $selected\_crossover$)
11: $new\_second\_individual \leftarrow$ Do_Crossover($second\_parent$, $first\_parent$, $selected\_crossover$)
12: **end if**
13: $new\_population \leftarrow$ Add_New_Children($first\_individual$, $second\_individual$)
14: **end for**

3.2.5 Mutation

We have practiced mutation specifically for undirected graphs. After the crossover step, mutation operations are conducted. Members of the new population, generated by crossover operations, need to be changed randomly not to get stuck into local minima. Otherwise, new individuals' fitness values could be as much as their parents.

First, with the help of mutation ratio settings, the decision of whether the mutation is to be applied or not is given. If there is no mutation, the selected individual is passed without any modification to the next generation. If the mutation is to be applied, it is necessary to determine which mutation technique is to be used. A particular parameter set for keeping the mutation ratio of each mutation technique is used in this selection procedure.

For graph drawing, the mutation has a more critical role than the crossover. In crossover techniques for graphs, it is not easy to combine the genetic heritage from both parents into a new child, due to the integrated nature of the graphs within themselves. For this reason in GA, diversity is mainly provided by mutation. Besides, the implementation of mutation techniques is easier than crossover techniques. The third reason is, the mutation is less costly in terms of execution time. For these reasons, more attention to the mutation is given in graph-related studies [63].

In our study, we have inspired from the studies of Timo and Eloranta [63]. There are five types of mutation techniques used in our work:

• Tiny Vertex Mutation: A vertex-based mutation technique. Selects a random vertex and moves it horizontally or vertically with a random distance on the graph. This mutation technique has less impact than others; for this reason, we have named this technique as tiny. An illustration of this technique is given in Figure 3.3. In this example the vertex numbered with 3 is selected for mutation, in Figure 3.3b this vertex is moved on the vertical axis and in Figure 3.3c this vertex is moved on the horizontal axis.

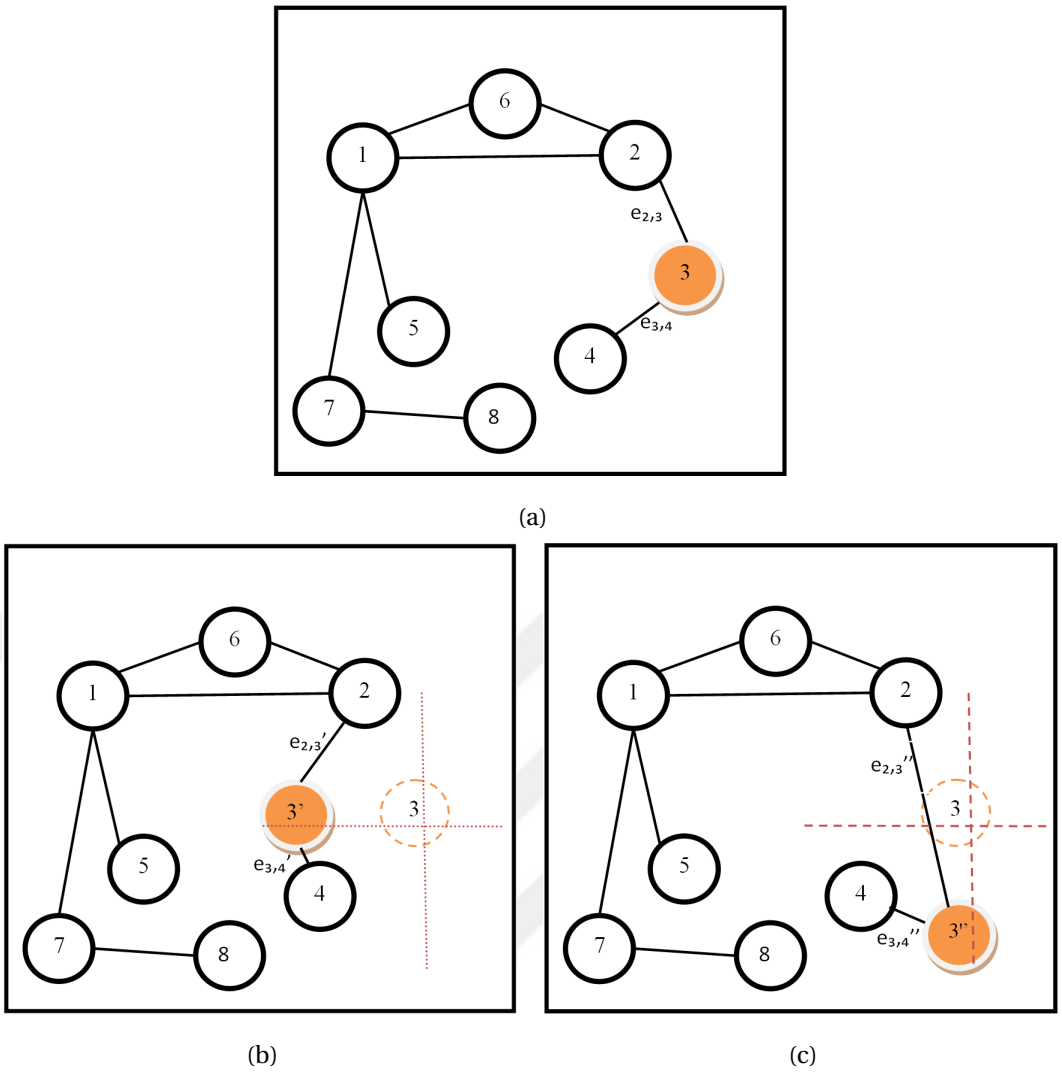
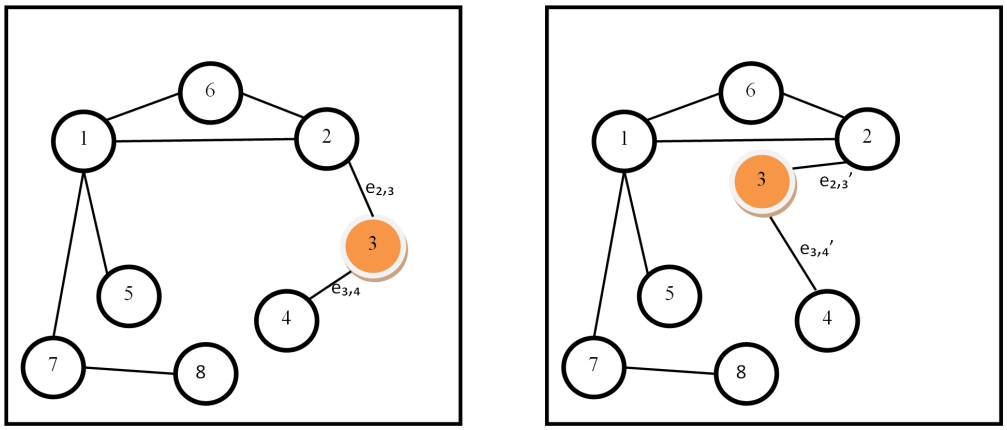


Figure 3.3: An illustration of the tiny vertex mutation. In this mutation technique, a random vertex is selected and moved horizontally or vertically with a random distance on the graph. a) Original graph. b) After the mutation on X-axis. c) After the mutation on Y-axis.

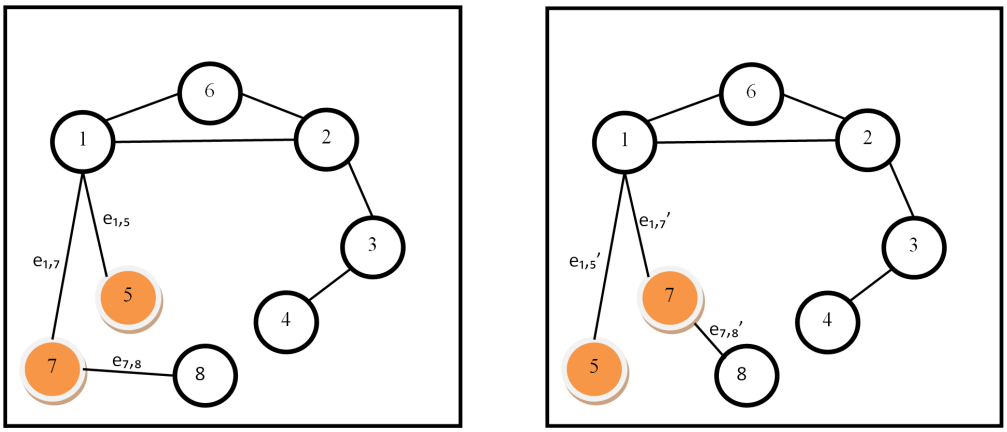
• Single Vertex Mutation: Selects a random vertex and moves it both horizontally and vertically in a limited random range inside the drawing area. An illustration of this technique is given in Figure 3.4. In this example the vertex numbered with 3 is selected for mutation, in Figure 3.4b this vertex is moved on both the X axis and the Y axis. The lengths and angles of the edges $e_{2,3}$ and $e_{3,4}$ are changed.



(a) (b)

Figure 3.4: An illustration of the single vertex mutation. In this mutation technique, a random vertex is selected and moved both horizontally and vertically in a limited random range inside the drawing area. a) Original graph. b) The graph after the mutation of the selected vertices.

• Swap of Vertices Mutation: A vertex-based mutation technique in our study. Swaps two randomly selected vertices' positions. This mutation technique is illustrated in Figure 3.5. In this figure, the vertices numbered with 5 and 7 are swapped. After the mutation the result graph is given in Figure 3.5b, in this figure the lengths and the angles of the vertices $e_{1,7}$, $e_{1,5}$ and $e_{7,8}$ are different from the original graph.



(a) (b)

Figure 3.5: An illustration of the swap of vertices mutation. In this mutation technique, two randomly selected vertices' positions are swapped. a) Original graph. b) The graph after the mutation of the selected vertices.

• Edge Based Mutation: Selects a random edge and moves it both horizontally and vertically by keeping its length and angle unchanged. This mutation technique is illustrated in Figure 3.6. In Figure 3.6a, the edge $e_{2,3}$ is selected for mutation. In Figure 3.6b, this edge is mutated by keeping its length and angle unchanged; but after the mutation, the edges $e_{1,2}$, $e_{6,2}$ and $e_{4,3}$ have new lengths and changed angles.

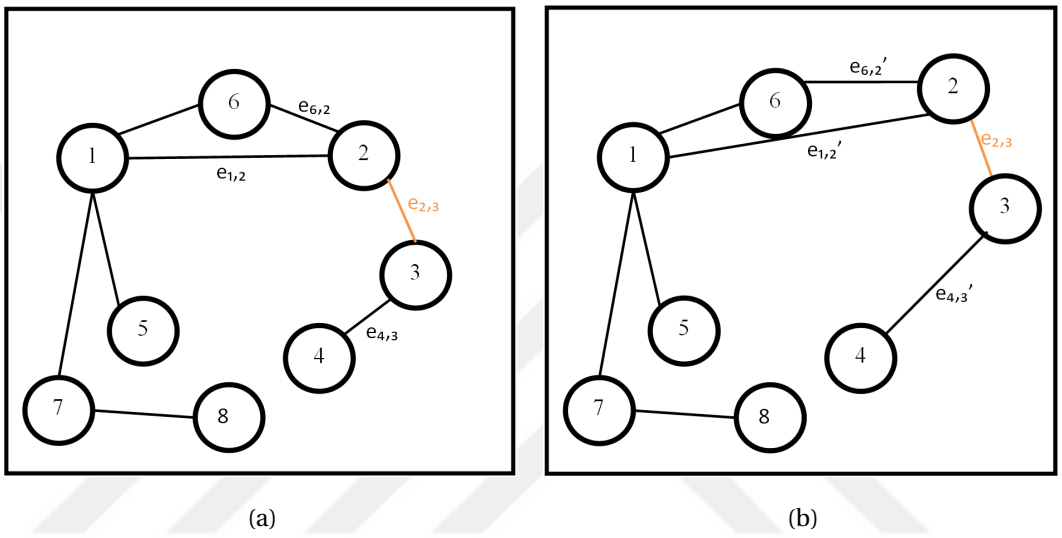


Figure 3.6: An illustration of the edge-based mutation. In this mutation technique, a random edge is selected and moved both horizontally and vertically by keeping its length and angle unchanged. a) Original graph. b) The graph after the mutation of the selected edge with its connected vertices.

• Two Edge-Based Mutation: An edge-oriented mutation technique. It is similar to the 'Edge Based Mutation' technique, the only difference from this technique is that instead of one edge, the two connected edges are mutating together. In this technique, a random vertex who has at least 2 edges is selected, then these two edges are moved to the new locations by keeping edge lengths and angles the same. Among all of the used mutation techniques, this technique has the biggest impact on a graph. The technique is illustrated in Figure 3.7.

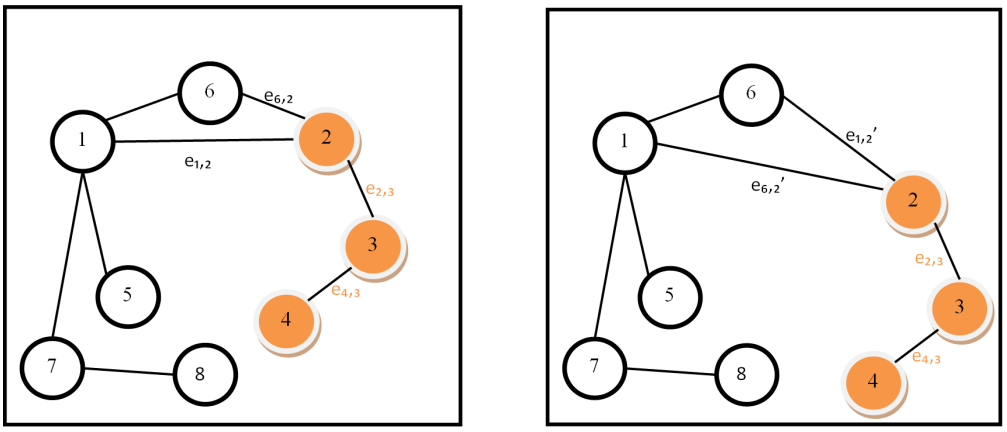


Figure 3.7: An illustration of the two edge-based mutation. In this technique, a random vertex who has at least 2 edges is selected, then these two edges are moved to the new locations by keeping edge lengths and angles the same. a) Original graph. b) The graph after the two edge-based mutation.

Algorithm 7 Mutation

INPUT: *population*, Graph Population
settings, Program Settings

OUTPUT: *mutated_population*, Mutated Population

1: **for** $i = 1$ to *population.size* **do**
2: *individual* ← *population*[i]
3: *mutate* ← Decide_Mutation(*settings.mutation_settings*)
4: **if** *mutate* is not true **then**
5: *mutated_individual* ← *individual*
6: **else**
7: *selected_mutation* ← Decide_Mutation_Technique(*settings.mutation_settings*)
8: *mutated_individual* ← Do_Mutation(*individual*, *selected_mutation*)
9: **end if**
10: *mutated_population* ← Add_Individual(*mutated_individual*)
11: **end for**

3.2.6 Termination

Termination condition has a significant role in the execution time of GA. Because of GA's non-deterministic nature, it's hard to estimate the duration for generating successful results. To overcome this problem, we have applied the following techniques.

One of the used technique is counting the iteration of GA; at the end of any iteration, if this value exceeds a predefined limit, GA is terminated. Stopping the GA too early results in premature graphs; on the other hand, GA does not provide better graphs when a certain number of iterations are exceeded, this case only causes longer execution times. We have executed our algorithm several times and we have obtained some results for defining a default iteration limit; the results of this study are given in the Section 4.2.1.

Another technique is tracking the obtained results and interpreting them to decide for terminating the GA or continue to the next iteration. At the end of each iteration, the best individual is chosen among the current population members; this mechanism is called elitism which is previously mentioned in Section 2.7. In our experiments, we have seen that in the case of the best individual is kept the same on consecutive iterations for a long time, terminating the program is essential. In our study, there is a mechanism that checks whether the current generation's best individual is same with previous generation's best individual; if this case occurs for consecutive three generations, the GA is terminated.

The pseudocode of the termination is presented in Algorithm 8.

Algorithm 8 Termination

INPUT: $population$, Graph Population
$settings$, Program Settings
$prev\_best\_individual$, Previous Generation Best Individual
$iteration\_counter$, GA Iteration Counter
$best\_individual\_counter$, Counter for Same Best Individual

OUTPUT: $terminate\_GA$, Terminate GA

1: $terminate\_GA \leftarrow 0$
2: **if** $iteration\_counter >= settings.iteration\_limit$ **then**
3: $terminate\_GA \leftarrow 1$
4: **else**
5: **if** $population.best\_individual = prev\_best\_individual$ **then**
6: $best\_individual\_counter \leftarrow best\_individual\_counter + 1$
7: **else**
8: $best\_individual\_counter \leftarrow 0$
9: **end if**
10: **if** $best\_individual\_counter >= 3$ **then**
11: $terminate\_GA \leftarrow 1$
12: **end if**
13: **end if**

CHAPTER 4

EXPERIMENTAL RESULTS

The implementation of our study is based on a well-known biological visualization framework Cytoscape [50]. Cytoscape is an open-source platform which provides visualization of a lot of different type of biological data such as molecular interaction networks and metabolic pathways contain vertices. In this section, the given results are obtained from our experiments which are performed on a computer which has the 64-bit Windows 7 operating system and on an Intel Core i5-2310 2.9GHz processor with 6 GB RAM.

4.1 Dataset

In our study, experiments are run on the same datasets with the base study ECleriize. In this manner, we have the chance of comparing our results with the base study's results.

The data sets used in this study were taken from the REACTOME database. REACTOME is an open-source database, founded in 2003. The website allows researchers to use biological information for data visualization, integration, and analysis by servicing their database on different platforms. One of the option provided by REACTOME is BioPAX, which is a well-known standard language that supplies visualization and analysis of biological pathway data. We have used the datasets in "BioPAX 3" format in our studies.

As in previous work, we have used three datasets, these datasets store biological graphs contain vertices that represent enzyme structure. In these graphs, there are

some vertices which have clustering information dedicated to Enzyme Commission (EC) number. These three datasets are Signaling by EGFR [4], Signaling by ERBB2 [5], and Visual phototransduction [6]. The information about used datasets are given in Table4.1 and Table 4.2.

Table 4.1: Primary information about used datasets.

Dataset	Stable Identifier	# of Vertices	# of Edges
Signaling by EGFR	R-HSA-177929	779	1209
Signaling by ERBB2	R-HSA-1227986	731	1109
Visual phototransduction	R-HSA-2187338	542	767

Table 4.2: EC class information about used datasets.

Dataset	Signaling by EGFR	Signaling by ERBB2	Visual Phototransduction
EC 1 - Oxidoreductases	0	0	9
EC 2 - Transferases	30	31	6
EC 3 - Hydrolases	3	2	11
EC 4 - Lyases	0	0	1
EC 5 - Isomerases	0	0	1
EC 6 - Ligases	6	5	0
Total	39	38	28

4.2 Parameter Setup

Kamada-Kawai uses parameters such as spring constant and maximum iteration number per vertices, while EClerize also employs its parameters such as spring constant for vertices that have EC numbers and a constant for distance vector.

Our study has lots of parameters which directly affect the resulting graphs. These parameters with optimal values are described.

4.2.1 Population Size ($size$) and Iteration Counter ($counter$)

Population size ($size$) and iteration counter ($counter$) are correlated and they directly affect the execution time of the program and the quality of the obtained graphs. In our study, a population consists of graphs. Parameter $size$ keeps the population size. For large graphs, which have a lot of vertices and edges, the parameter $size$ directly affects memory consumption and execution time. Due to the non-deterministic nature of the GA, populations with a large number of members, increase the chances of reaching the global optima.

In our experiments, first, we have determined the parameter $size$. By fixing the parameter $counter$ as equal to 1 and with the variable value of the parameter $size$, we have run our experiments. In spite of there are lots of different measures to understand how a graph is nicely drawn; in our parameter estimation works, we have used the number of edge crossings as the selected aesthetic measure because it has the biggest effect on the resulting graph. To find the optimal value for the population size, the number of edge crossings is our most important criteria.

We have run our study, with different population sizes from 1 to 10 on three datasets. We have observed that after $size > 10$ edge crossing number does not decrease any more, for this reason, in our experiments the value of the parameter $size$ is limited with 10. Figure 4.1, Figure 4.2 and Figure 4.3 present graphs that show the relation between parameter $size$ and the number of edge crossings when $counter = 1$. In these graphs, the line of min shows the obtained lowest edge crossing number for the selected value of parameter $size$, the line of median is the middle member of

ordered obtained edge crossing numbers of the selected value of the parameter $size$ and the line of max represents the observed highest edge crossing number for the selected value of the parameter $size$.

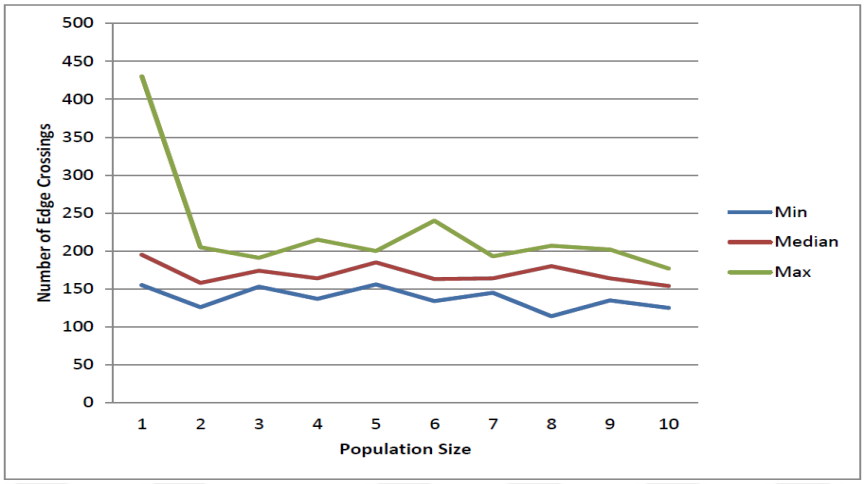


Figure 4.1: Relation between parameter $size$ and edge crossing number when $counter$ = 1 on Signaling by EGFR dataset.

In Figure 4.1, there isn't a drastic change between edge crossing numbers on the median line for different values of population size. For initial values of parameter $size$, in view of these three lines, the consistency between obtained the number of edge crossings is worse than higher values of $size$. From the value of $size$ as 7 to the higher values, the consistency of the number of edge crossings is better than previous values of the parameter $size$.

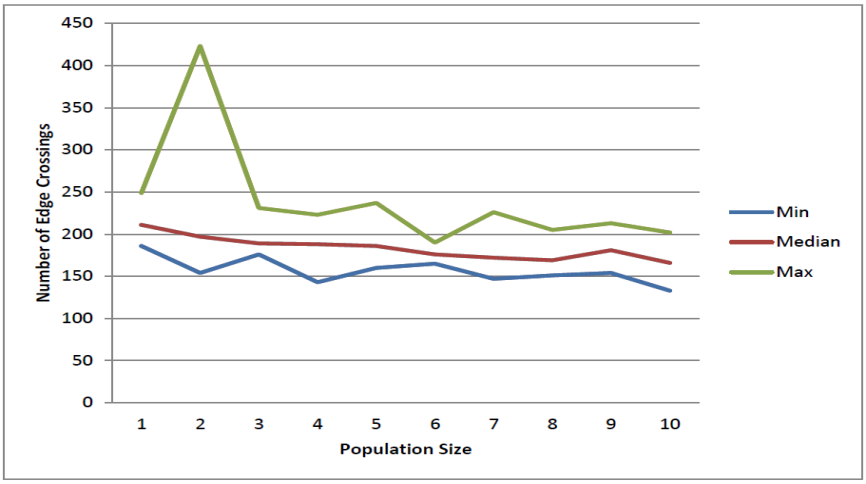


Figure 4.2: Relation between parameter $size$ and edge crossing number when $counter$ = 1 on Signaling by ERBB2 dataset.

In Figure 4.2, the graph shows a decreasing trend for the number of edge crossing for ascending values of population size. At the beginning of the graph, the first two initial values of the population size have an inconsistency between 3 lines. The value of $size$ as 6 can be accepted as the best value in terms of both the number of edge crossings and stability.

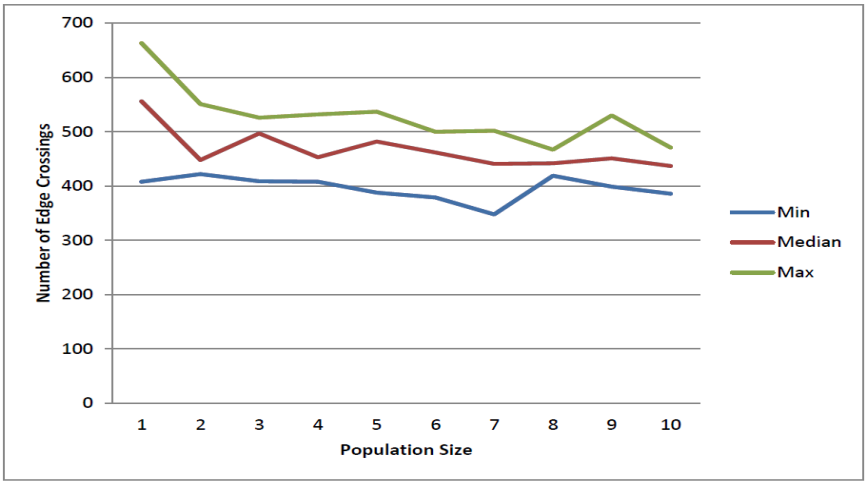


Figure 4.3: Relation between parameter $size$ and edge crossing number when $counter$ = 1 on Visual Phototransduction dataset.

In Figure 4.3, a similar state is seen with the previous graph. For initial values of the population size, especially for the first two values, stability between different lines is partially less than higher values. The value of $size$ as 8 is acceptable as the best value in terms of both the number of edge crossings and stability.

The number of edge crossings becomes stable when the value of S is bigger than 5 in all results. By looking at the above results, it is advised to use parameter $size$ as $6 \leq size \leq 8$.

The parameter $counter$ is used as the iteration counter in the GA. One of the termination condition of GA in our study is, exceeding a predefined limit on the creation of new generations. In our experiments, to determine an optimal value for parameter $counter$, we have fixed the value of $size$ as 6. We have run our study with different values of $counter$ from 1 to 10 on three datasets. We have observed that after $counter > 10$, edge crossing number does not decrease any more and also execution time becomes longer. Table 4.3, Table 4.4 and Table 4.5 show us the effect of parameter $counter$ on number of edge crossings and execution time.

Table 4.3: Iteration counter ($counter$), edge crossing numbers and execution time (second) when $size = 6$ on Signaling by EGFR dataset.

c	Number of Edge Crossings	Execution Time (second)
1	176	10
2	164	12
3	175	13
4	169	16
5	174	14
6	166	14
7	158	18
8	172	19
9	162	20
10	159	24

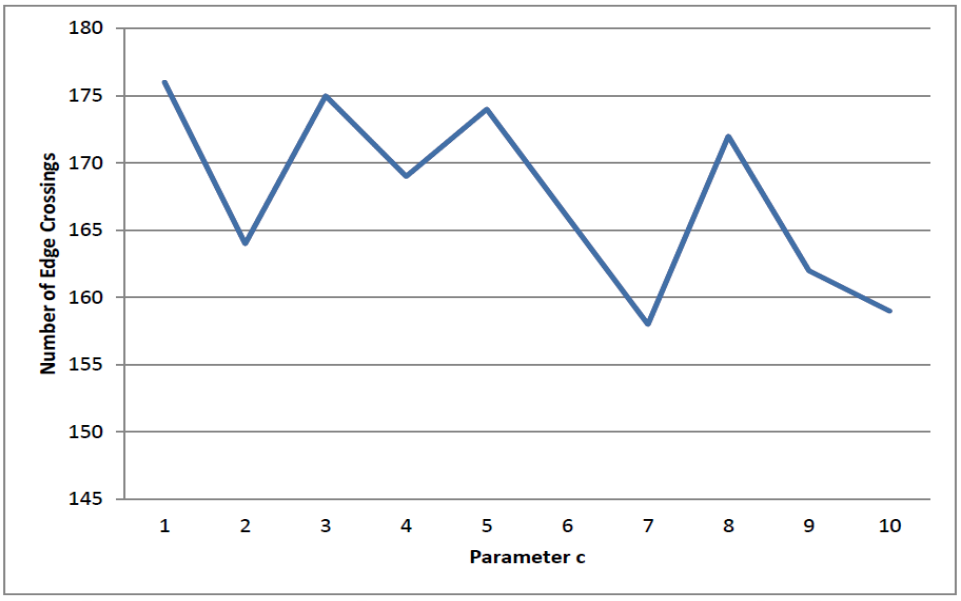


Figure 4.4: Number of edge crossings and execution time (second) when $size = 6$ on Signaling by EGFR dataset.

Table 4.4: Edge crossing numbers and execution time (second) when $size = 6$ on Signaling by ERBB2 dataset.

c	Number of Edge Crossings	Execution Time (second)
1	163	9
2	167	9
3	158	10
4	158	14
5	150	11
6	149	13
7	151	17
8	148	16
9	142	19
10	157	21

Figure 4.5: Edge crossing numbers and execution time (second) when $size = 6$ on Signaling by ERBB2 dataset.

Table 4.5: Edge crossing numbers and execution time (second) when $size = 6$ on Visual Phototransduction dataset.

c	Number of Edge Crossings	Execution Time (second)
1	462	34
2	427	33
3	404	39
4	415	45
5	410	50
6	424	59
7	405	66
8	391	76
9	387	108
10	403	80

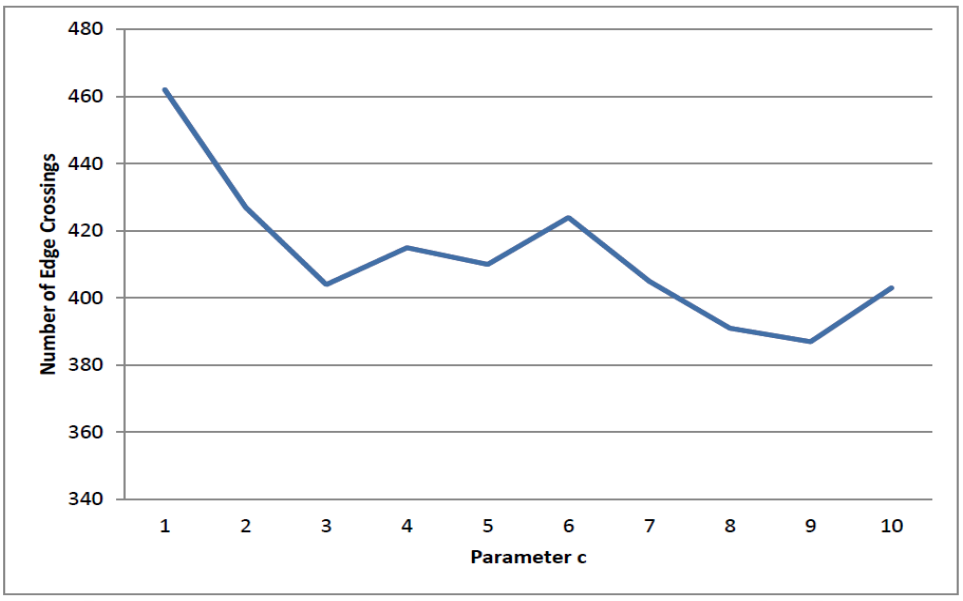


Figure 4.6: Edge crossing numbers and execution time (second) when $size = 6$ on Visual Phototransduction dataset.

From the tables above the following results are reached:

• An increasing number of iteration on populations generally results in increased execution times.

• In our study, there is a linear correlation between the number of edges and execution times.

• The experiments show us that there isn't a huge difference with regard to the number of edge crossings between different iteration counters.

• If we compare our results with the results from the base study EClerize, we see that there isn't too much increase in the execution time with our improvement. On the other side, the number of edge crossings has a significant decrease, by comparison, the case study.

The number of edge crossings becomes stable when $counter$ is bigger than 2 in all results. After values of $counter$ bigger than 8, there are longer execution times. So, it is advised to use parameter $counter$ as: $3 \leq counter \leq 8$.

We have conducted some experiments to observe the results when the population size or the iteration counter are set to the higher values. The results are given in Table 4.6. The results show that the higher values of the parameter 'iteration counter' is time-consuming and there is difficulty in producing immediate results; the parameter 'population size' in high values result in both increased memory consumption and long execution times. We have also observed that the usage in our recommended range produces quality results similar to the high-value configurations. Users who want to get results in a short time should stay within our recommended ranges. If the users have enough time to wait, they can work with the high values of these parameters; this helps them to achieve the global optimum of the resulting graph.

Table 4.6: Execution results from different values of the parameters $size$ and $counter$ on the datasets Signaling by EGFR, Signaling by ERBB2 and Visual Phototransduction.

Dataset	***size***	***counter***	**# of Edge Crossings**	**Execution Time (minutes)**
Signaling by EGFR	1	1	292	0.02
	6	4	138	0.11
	10	100	121	3.50
	10	500	119	14.00
	10	1000	139	29.00
	100	10	130	5.50
	1000	10	112	57.00
	100	100	97	35.00
Signaling by ERBB2	1	1	322	0.03
	6	4	164	0.13
	10	100	141	4.50
	10	500	143	21.25
	10	1000	174	43.50
	100	10	134	7.50
	1000	10	121	72.00
	100	100	130	41.00
Visual Phototransduction	1	1	686	0.08
	6	4	370	0.46
	10	100	336	17.00
	10	500	372	83.00
	10	500	354	171.00
	100	10	341	23.50
	1000	10	336	242.00
	100	100	355	175.00

4.2.2 Mutation Settings

In our study, there are two types of settings for mutation. The first type of mutation setting is used for keeping the mutation rate. In our experiments, we have observed that the mutation has a more prominent and positive effect than crossover on undirected graphs. For this reason, we use the mutation rate as 50% in our experiments; we have observed that higher rates than this value causes loss on the characteristics of an individual. Less than 40% also avoids the diversity in new generations.

As it is mentioned in Section 3.2.5, in our study, there are five types of mutation techniques for providing diversity on individuals. The second type of mutation setting is used for detecting the mutation type to be applied. These group of settings are used to decide which mutation to be applied when the mutation occurs. In the mutation techniques we use, we recommend that similar mutation techniques have a similar probability of occurrence. According to Eloranta [63], the mutation operations applied to the edges usually have better performance than those used to vertices; we have also observed this circumstance in our experiments. For this reason, we recommend that the ratio of edge-based mutation techniques to be higher and corner-based techniques to be lower rates. Recommended rates of mutation techniques are given in Table 4.7.

Table 4.7: Recommended rates of mutation techniques used in EClerize Type GA.

Mutation Type	Probability of Occurrence
Two Edge Mutation	35%
Edge Mutation	35%
Single Mutate	10%
Tiny Mutate	10%
Vertex Swap Mutation	10%
Total	100%

4.2.3 Crossover Settings

Two types of settings for crossover are used in our study. The first setting is used for crossover occurrence on selected two individuals. In our experiments, we have observed that it's not easy to transfer the information from two ancestor graphs to the two child graphs by using crossover operations. For this reason, for undirected graphs, the crossover has less effect than the mutation [63]. Due to this reason, in our study, the crossover rate is smaller than the mutation rate. In our experiments, the crossover rate is used as 15%.

The second type of crossover setting is used for the applied crossover technique. The used techniques, in this study, are given in Section 3.2.4. There are two types of crossovers in our study. In our research, we could not find any recommendation on this settings. We recommend to use both techniques in equal proportions; since according to our observations, both of these methods affect the graphs at similar levels. Recommended rates of crossover techniques are given in Table 4.8.

Table 4.8: Recommended crossover rates of EClerize Type GA.

Crossover Type	Probability of Occurrence
Rectangular Crossover	50%
Three Vertices Crossover	50%
Total	100%

4.2.4 Fitness Settings

In our study, a configurable settings environment is provided to the users. For different considerations, different types of aesthetic criteria can be significant. Using our configurable fitness settings, the resulting graph can be figured in different forms.

Fitness settings affect various steps of our study. It has a critical role in the creation of a new generation. By identifying the individuals which have better fitness values at the current generation, the fitness calculation phase increases the likelihood of random selection of these individuals for being parent of the new generation. Furthermore, the individual which has the highest fitness value is selected in this phase, which is used for elitism. Another phase where fitness settings have a critical role, following to the termination of GA, the best individual in the last generation is selected as the resulting graph to be shown the users; by looking at the current fitness settings, our study chooses the best individual then it is showed to the user.

Recommended fitness settings are given in Table 4.9. The fitness coefficients given below have been determined by considering the aesthetic criteria that we care in our experiments. Since the number of edge crossings is the most important fitness criterion in our study, it has the highest rate. The remaining percentage is divided among other criteria according to their importance.

Table 4.9: Recommended fitness coefficients of EClerize Type GA.

Fitness Type	Impression Coefficient
Edge Crossing	50%
Inter Cluster Distance	15%
Minimum Vertex Distance Sum	10%
Edge Length Deviation	10%
Drawing Area	10%
Minimum and Maximum Vertex Distance	5%
Total	100%

In our studies, we have observed the effect of fitness values determined by the users on the obtained graphs. By activating one fitness criterion and inactivating others, EClerize Type GA has been run on Signaling by EGFR dataset for several fitness criteria. In the obtained results, it has been observed that each active fitness criterion has a significant effect on the result graph. It has been also observed that, other inactivated fitness criteria have shown a deterioration in these result graphs compared to the initial graphs. The obtained results are given in Table 4.10. The obtained graphs of the studies where a single fitness criterion is active are shown in Figure 4.7.

Table 4.10: Comparison of different active fitness criteria on Signaling by EGFR dataset.

Active Criterion	Drawing Area	# of Edge Crossings	Edge Length Dev.	Inter Cluster Dist.
Edge Crossing	1.15	166.00	162.66	3594.83
Edge Length Dev	1.20	195.00	150.48	3537.88
Min. Node Dist. Sum	1.32	232.00	157.98	3554.29
Inter-Cluster Dist.	1.12	226.00	154.77	3679.87

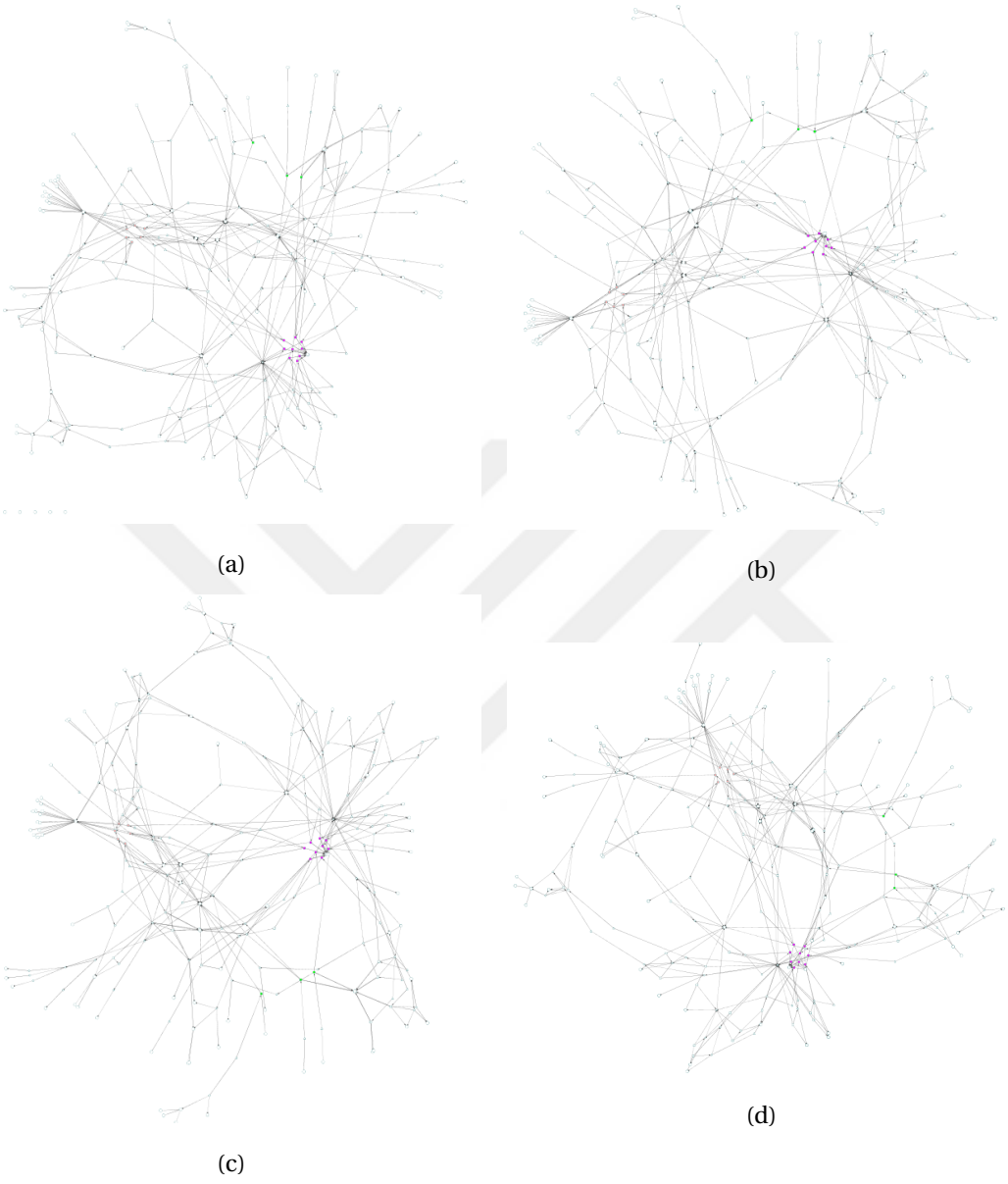


Figure 4.7: An illustration of different active fitness criteria on Signaling by EGFR dataset. a) Edge crossing is the only active criterion. b) Edge length deviation is the only active criterion. c) Inter-cluster distance is the only active criterion d) Minimum node distance sum is the only active criterion.

4.3 Results

With a small amount of additional execution time and with the help of various configurable settings, our study can produce better results than the base study.

EClerize Type GA works the same with base study when the parameters $size$ = 1 and $counter$ = 1. We have run our study and base study with above-given default parameters on three datasets for comparison. With an acceptable longer execution time, our study produced better results in view of selected aesthetic criteria. As it is mentioned before, the fitness type 'Edge Crossings' has a bigger impact on overall fitness than other settings. For this reason, the biggest progress in our results exists in this fitness setting. Table 4.11, table 4.12 and table 4.13 show obtained results for each fitness type for 3 datasets. Inter-cluster distance, minimum node value and, minimum node distance sum are the types of fitnesses with positive effect, the other fitness types have negative effect on overall fitness. For better comparison, Figure 4.8, Figure 4.9 and Figure 4.10 are also given.

Table 4.11: A comparison between result of our study and base study on Signaling by EGFR dataset.

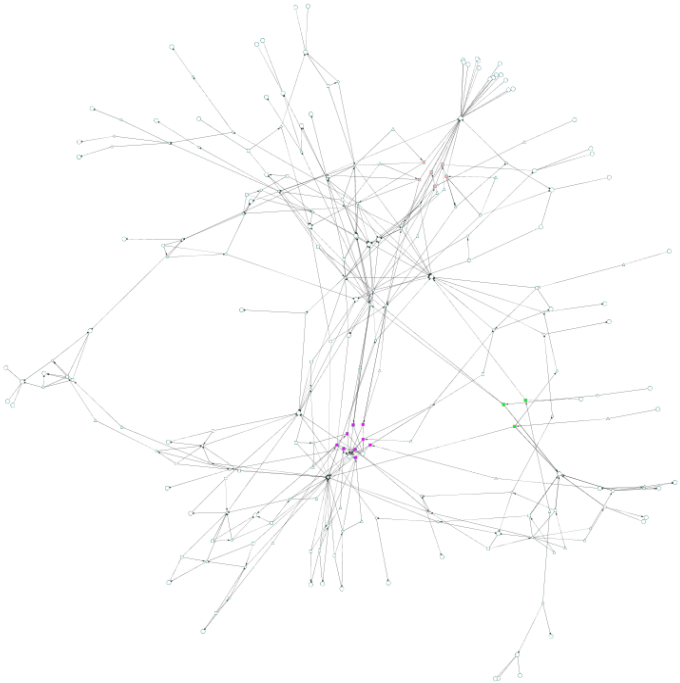
Fitness Type	Native EClerize	EClerize Type GA	change
Drawing Area	1.36	1.40	-3%
Edge Crossing	292.00	138.00	53%
Minimum Vertex Dist.	29.20	35.14	-20%
Edge Length Deviation	161.93	158.66	2%
Maximum Vertex Dist.	904.79	915.49	1%
Inter Cluster Dist.	3795.35	3268.54	17%
Execution Time (ms)	1100	7400	-

Table 4.12: A comparison between result of our study and base study on Signaling by ERBB2 dataset.

Fitness Type	Native EClerize	EClerize Type GA	Change
Drawing Area	1.50	1.51	0%
Edge Crossing	322.00	164.00	49%
Minimum Vertex Dist.	33.34	25.64	23%
Edge Length Deviation	161.65	153.08	7%
Maximum Vertex Dist.	922.19	1004.01	9%
Inter Cluster Dist.	3927.34	5228.09	33%
Execution Time (ms)	1300	8000	-

Table 4.13: A Comparison Between Result of Our Study and Base Study on Visual Phototransduction dataset.

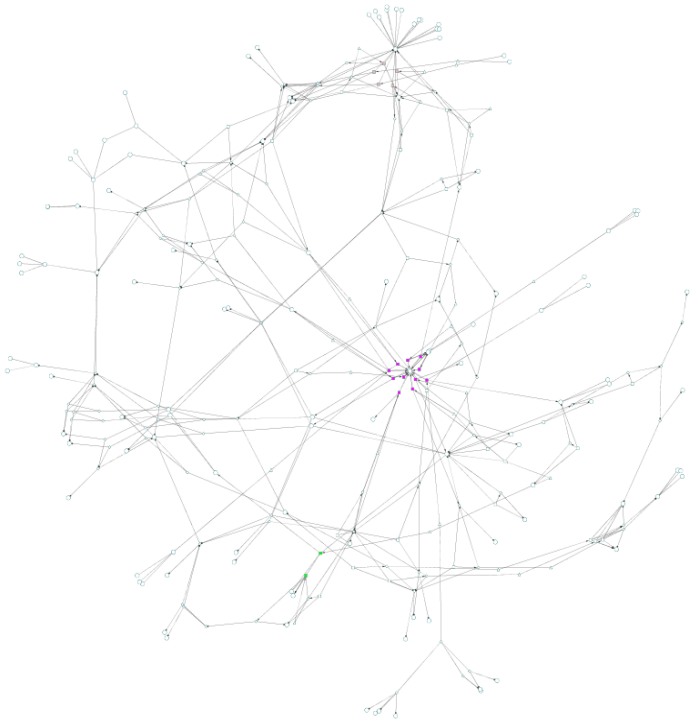
Fitness Type	Native EClerize	EClerize Type GA	Change
Drawing Area	3.99	4.08	0%
Edge Crossing	686.00	370.00	46%
Minimum Vertex Dist.	32.20	35.46	-10%
Edge Length Deviation	190.94	157.21	18%
Maximum Vertex Dist.	1806.72	1225.99	-32%
Inter Cluster Dist.	2308.56	3254.40	40%
Execution Time (ms)	5300	27500	-



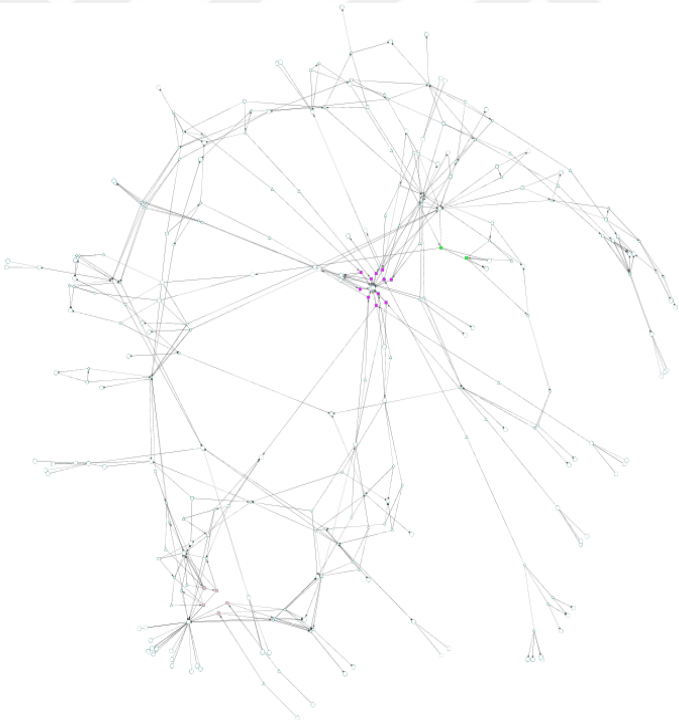
(a)

(b)

Figure 4.8: Different Layouts of Signaling by EGFR dataset. (A) Layout by EClerize Type GA with parameters c = 3 and S = 6. (B) Layout by Native EClerize with its default parameters.

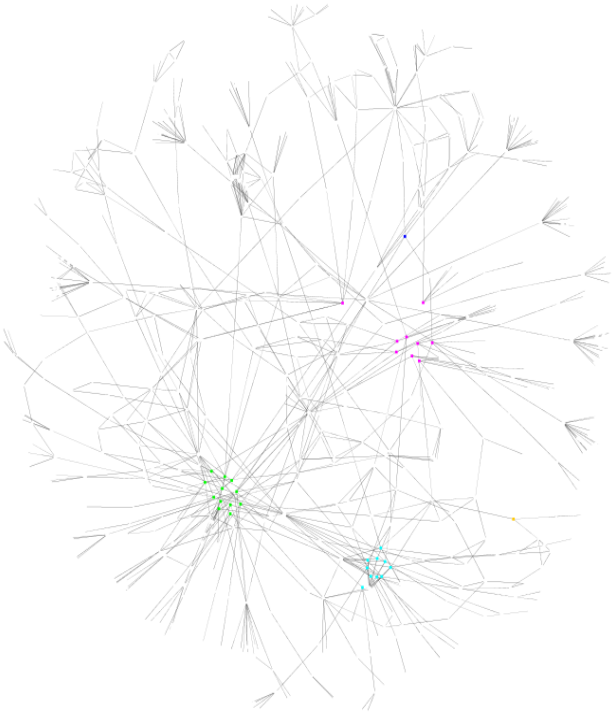


(a)

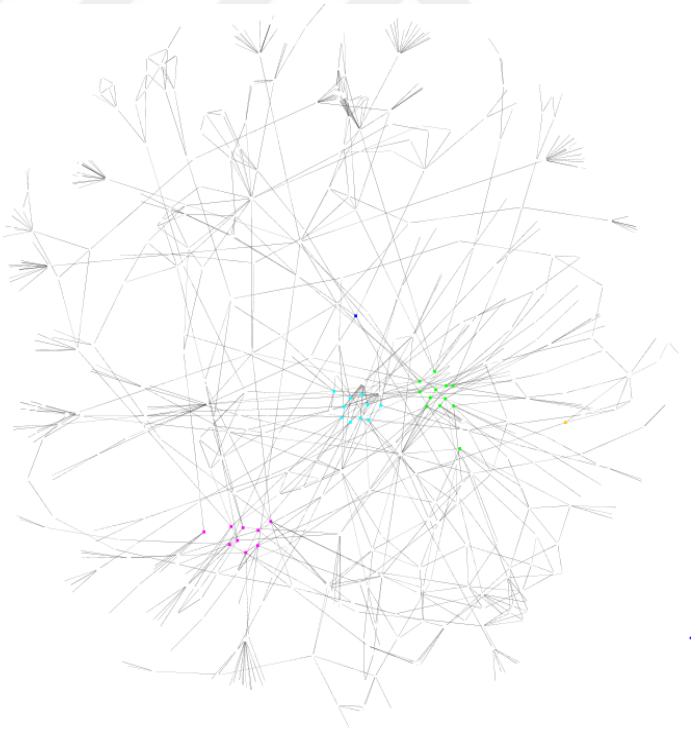


(b)

Figure 4.9: Different Layouts of Signaling by ERBB2 dataset. (A) Layout by EClerize Type GA with parameters c = 3 and S = 6. (B) Layout by Native EClerize with its default parameters.



(a)



(b)

Figure 4.10: Different Layouts of Signaling by Visual Phototransduction dataset. (A) Layout by EClerize Type GA with parameters c = 3 and S = 6. (B) Layout by Native EClerize with its default parameters.

It is useful to focus on the obtained layouts by zooming to better understand the progress. A closer comparison of the images obtained from 3 different datasets is given in Figure 4.11. In order to ensure that the observed area is the same in both compared graphs, we have focused on the same colored vertices which are the members of the same clusters. When the given examples are examined closely, the progress in the edge crossings and inter-cluster distances can be seen clearly. In the first two pairs of figures, it's easy to see the decrease in the number of edge crossings. In the last pair of figures, there is a significant decrease in the distance between two clusters.

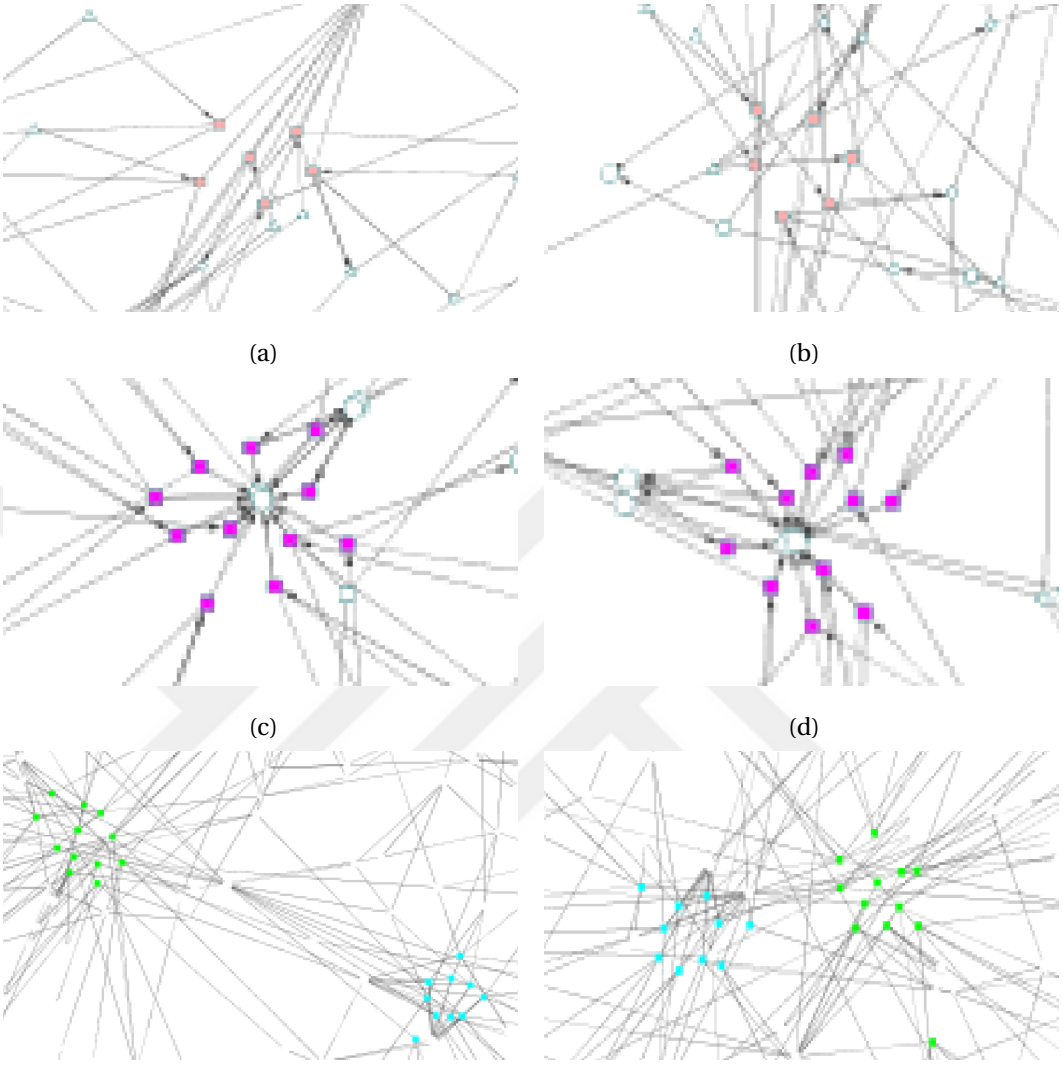


Figure 4.11: Zoomed visuals from the layouts of Signaling by EGFR, Signaling by ERBB2 and Visual Phototransduction datasets by EClerize Type GA and Native EClerize. a) Layout by EClerize Type GA on Signaling by EGFR dataset. b) Layout by Native EClerize on Signaling by EGFR dataset. c) Layout by EClerize Type GA on Signaling by ERBB2 dataset. d) Layout by Native EClerize on Signaling by ERBB2 dataset. e) Layout by EClerize Type GA on Visual Phototransduction dataset. f) Layout by Native EClerize on Visual Phototransduction dataset.

CHAPTER 5

CONCLUSION

5.1 Summary

Enzymes are biological molecules (typically proteins) that significantly speed up biochemical reactions in living organisms. The Enzyme Commission (EC) number is a number appointed to enzymes about their chemical functions [20]. Metabolic pathways are the biological graphs which may contain vertices that represent enzyme molecules. For biologists, understanding a metabolic pathway could be essential. A pathway can be visualized as a graph whose layout is drawn by a force-directed algorithm. Force-directed graph drawing algorithms or called as spring embedder algorithms define a system of forces acting on the vertices and edges. A force-directed algorithm can be used for any domain-specific work under the favor of its simplicity and flexibility. EClerize is a customized and improved Kamada Kawai (KK) force-directed algorithm in order to visualize pathways that contain nodes with attributes as EC numbers.

In our study, we have improved EClerize. Like as other force-directed algorithms, KK algorithm has a disadvantage of getting stuck into local optima. For this reason, result graphs generated by EClerize could be converged in local minima. In this study, our purpose is to avoid the local optima and obtain global optimum solutions for EClerize during graph drawing.

For some problems, there isn't enough time to find and verify that, whether the found solution is the optimal solution or not; on the other side, for some problems the annoyance is not the time, there is incomplete information about the problem. A heuristic is usually an optimization or a strategy that provides a good enough an-

swer. By use of a well-known heuristic technique, Genetic Algorithm (GA), we have improved the result graphs produced by native EClerize. Due to its random nature, GA never guarantees to find the best solution for any problem like other evolutionary algorithms, but it often finds a good enough solution which is relatively close to the best solution.

Our study, named EClerize Type GA, is a version of based work EClerize with the integration of GA. The components of the GA have been implemented in a graph-oriented manner. EClerize is used as a fine-tuner, a fine-tuner makes small arrangements for something to achieve the best or the desired performance; in our study, after the end of each iteration of GA, EClerize makes small arrangements on individuals (graphs). In our work, we have built a system which supplies a configurable settings platform to the users. By setting program parameters, users have a direct impact on the result graphs.

In order to compare our results with the base study's results, we have run native EClerize and our study with recommended values on three datasets. With an acceptable longer execution time, our study produces better results in view of selected aesthetic criteria. For both of the three datasets, the biggest improvement among the fitness criteria is seen on edge crossing number; in all of our experiments, the number of edge crossing has been reduced by at least 40%. In the obtained results on Signaling by EGFR dataset, in spite of the negative changes on some criteria in the results by our study, the overall fitness value is still better than the achieved result from native EClerize; because these criteria, in which negative change has occurred, are less effective than other criteria in which positive change has occurred. By using a higher impression coefficient; users can be sure on that selected fitness criteria are always improved, just like the criterion 'edge crossing number' in our experiments. The obtained results on Signaling by ERBB2 dataset, all of the fitness criteria have positive changes. The obtained results on Visual Phototransduction dataset are similar to EGFR results, there are negative changes on two criteria but, our results are still better than the obtained results from native EClerize.

The achievements obtained by the studies are as follows:

• The obtained results are better than the results produced by the native ECLerize:

-The execution times of our study for the same dataset are in acceptable long.

-In view of several metric measurements, our study produces better results.

-We have improved the produced drawings to the best or nearly the best in the global range.

• We have integrated a force-directed drawing algorithm with the genetic algorithm. Although there are similar studies, we could not find any other study that integrates GA with KK.

• In the implementation of the components of the GA, we have revealed various innovative techniques:

-5 techniques in mutation phase and for crossover 2 techniques are employed. In this way, we provide significant support to the diversity in which the GA takes its power to reach the global optima.

-We provide a system, in which multiple criteria are utilized in the determining of individuals' fitnesses in the GA.

-By interpreting individuals from a broad perspective, we have prevented the overestimation of an individual who is good in a single point of view but weak in other aspects.

• We have created a platform where the resulting graph is shaped on the user's settings preferences. Our study allows users to select various criteria with the option of use with different rates.

• Our study differs from other studies in terms of both the number of vertices used in the GA, our experiments have been conducted on the datasets which have more than 700 vertices and more than 1200 edges.

• Although our study may seem peculiar to the biological graphs which rep-

resent pathways in which the vertices are associated with EC numbers, we have created the environment that provides nice-looking undirected graphs by reaching global optimum with the integration of GA, KK and several clustering techniques for different types of datasets.

5.2 Perspectives

In our study, the most time-consuming part is by EClerize. An improvement in EClerize directly affects the whole performance of our research. EClerize is based on a force-directed algorithm KK. This algorithm's execution time is $O(|V|^3)$ where V indicates the number of vertices in the graph. More effective algorithms, with different aesthetic concerns and better execution times, are present for a long time.

It is necessary to take into account that algorithms have specific characteristics besides their execution times. The case of the fastest algorithm produces the most desired result, is not always true; the algorithm can be considered by keeping all of the needed aesthetic criteria. Considering all this, Fruchterman-Reingold's algorithm [26] could be preferred instead of Kamada-Kawai.

Multi-level approaches can be used to draw larger graphs. There are some previous studies in which multi-level methods were used [68, 19, 33, 35].

Force-directed algorithms are quite old. At this point, it may be useful to implement the EClerize with more recent or today's methods. For example, GPU-based graph drawing algorithms or use of neural networks can be useful at this point. Some studies already exist about GPU usage on graph drawing [25, 40, 51].

In our study, there are numerous and different parameters used by the GA. While some of the parameters, such as the size of the population, directly affect the functioning of the GA; the others determine the sub-details, such as determining which mutation technique to be applied. In our studies, although we have determined some optimal ranges for the parameters population size and iteration counter on various datasets, the remaining parameters have been determined by intuitively. The future works may include determining the optimal values of all parameters.

REFERENCES

[1] Enzyme commission number. https://en.wikipedia.org/wiki/Enzyme_Commission_number. Accessed: 2019-03-15.

[2] A force-directed method for large crossing angle graph drawing - scientific figure on researchgate. https://www.researchgate.net/figure/The-spring-embedder-model-taken-from-Brandes-3_fig2_48172268. [Online; accessed March 24, 2019].

[3] Network science - scientific figure on researchgate. researchgate.net. [Online; accessed March 24, 2019].

[4] Signaling by egfr. https://reactome.org/content/detail/R-HSA-177929. [Online; accessed March 26, 2019].

[5] Signaling by erbb2. https://reactome.org/content/detail/R-HSA-1227986. [Online; accessed March 26, 2019].

[6] Visual phototransduction. https://reactome.org/content/detail/R-HSA-2187338. [Online; accessed March 26, 2019].

[7] V. K. Balakrishnan. *Graph Theory*. McGraw-Hill, 1997.

[8] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. (CMU-CS-95-141), May 1995.

[9] A. M. S. Barreto and H. J. C. Barbosa. Graph layout using a genetic algorithm. In *Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks*, pages 179–184, Nov 2000.

[10] D. Battista. Inclusion in a planar st-graph. page 172–179, 1998.

[11] M. Y. Becker and I. Rojas. A graph layout algorithm for drawing metabolic pathways. *Bioinformatics*, 17(5):461–467, 05 2001.

[12] C. Bennet, J. Ryall, L. Spalteholz, and A. Gooch. The aesthetics of graph visualization. pages 57–64, 01 2007.

[13] J. Branke, F. Bucher, and H. Schmeck. Using genetic algorithms for drawing undirected graphs. 03 1998.

[14] M. L. Brian S. Everitt, Sabine Landau and D. Stahl. *Cluster Analysis*. Wiley Publishers, 2011.

[15] G. Chartrand. *Introductory Graph Theory*. Courier Corporation, 1977.

[16] A. Cornelsen, Sabine; Karrenbauer. Accelerated bend minimization. *Journal of Graph Algorithms and Applications*, 16(3):635–650, 2012.

[17] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics (TOG)*, 15(4):301,331, 1996.

[18] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[19] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In S. North, editor, *Graph Drawing*, pages 101–112, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[20] W. EC. *Enzyme nomenclature 1992: recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes*. Academic Press, 1992.

[21] A. E. e. a. Eiben. Genetic algorithms with multi-parent recombination. *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, page 78–87, 1994.

[22] G. L. H. M. Emilio Di Giacomo, Walter Didimo. Area, curve complexity, and crossing resolution of non-planar graph drawings. In *Theory of Computing Systems*, volume 49, page 565–575, 2011.

[23] N. S. N. F. Ghassemi Toosi and M. Eaton. Evolving smart initial layouts for force-directed graph drawing. *In Proceedings of the Companion Publication*

of the 2015 Annual Conference on Genetic and Evolutionary Computation, pages 1397–1398, 2015.

[24] M. E. Farshad Ghassemi Toosi, Nikola S. Nikolov. Simulated annealing as pre-processing step for force-directed graph drawing. *ACM*, 2016.

[25] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1310–1319, 2007.

[26] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, 21(11):1129–1164, 1991.

[27] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. In *Software: Practice and Experience*, volume 21, pages 1129–1164, 1991.

[28] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Morgan Kaufmann, 1990.

[29] R. T. G. Di Battista, P. Eades and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.

[30] D. S. Garey, M. R.; Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(4):312–316, 1983.

[31] R. Garg, Ashim; Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

[32] Z. Gu, L. Gu, R. Eils, M. Schlesner, and B. Brors. Circlize implements and enhances circular visualization in r. *Bioinformatics*, 30(19):2811–2812, 06 2014.

[33] M. Hachul, Stefanand Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In J. Pach, editor, *Graph Drawing*, pages 285–295, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[34] F. Harary. *Graph Theory*. Addison-Wesleys, 1969.

[35] D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. In *Graph Drawing*, page 183–196, 2001.

[36] T. Harju. *Lecture Notes on GRAPH THEORY*. CreateSpace Independent Publishing Platform, 2014.

[37] V. A. H.F. Danaci, R. Cetin Atalay. Eclerize: A customized force-directed graph drawing algorithm for biological graphs with ec attributes. *Int. Journal of Bioinformatics and Computational Biology*, 16(04), 2018.

[38] J. Hopcroft and R. Tarjan. Efficient planarity testing. *Journal of the Association for Computing Machinery*, 21(4):549,568, 1974.

[39] K. Inoue, S. Shimozono, H. Yoshida, and H. Kurata. Application of approximate pattern matching in two dimensional spaces to grid layout for biochemical network maps. *PloS one*, 7:e37739, 06 2012.

[40] T. Jeowicz, M. Kudelka, J. Plato, and V. Snáel. Visualization of large graphs using gpu computing. In *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, pages 662–667, 2013.

[41] D. Jong. Special issue on genetic algorithms. In *Machine Learning*, volume 5, pages 285–319, 1990.

[42] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31:7,15, 1989.

[43] S. Kobourov. Spring embedders and force-directed graph drawing algorithms. 01 2012.

[44] K. "Kojima, M. Nagasaki, E. Jeong, M. Kato, and S. Miyano. An efficient grid layout algorithm for biological networks utilizing various biological attributes. *BMC Bioinformatics*, 8(1):76, Mar 2007.

[45] P. E. L. Groves, Z. Michalewicz and C. Janikow. Genetic algorithms for drawing directed graphs. *Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems*, pages 268–276, 1990.

[46] X. Lin. *Analysis of Algorithms for Drawing Graphs*. PhD thesis, University ıf Queensland, 1992.

[47] A. Lubiw and M. Petrick. Morphing planar graph drawings with bent edges. *Electronic Notes in Discrete Mathematics*, 31:45–48, 08 2008.

[48] M. Mitchell. A genetic algorthm tutorial. In *Statistics and Computing*, volume 2, pages 65,85, 1994.

[49] M. C. M. F.-R. A. J. B. S. O. Garcia, C. Saveanu and T. Aittokallio. Golorize: a cytoscape plug-in for network visualization with gene ontology-based layout and coloring. *Bioinformatics*, 23(3):394–396, 2007.

[50] O. O. N. S. B. J. T. W. D. R. N. A. B. S. P. Shannon, A. Markiel and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[51] A. Panagiotidis, G. Reina, M. Burch, T. Pfannkuch, and T. Ertl. Consistently gpu-accelerated graph visualization. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction*, pages 35–41. ACM, 2015.

[52] E. B. Peter K. Robinson. Spring embedders and force-directed graph drawing algorithms. *Essays in biotechnological applications*, 59:1–41, 11 2015.

[53] H. C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages and Computing*, pages 501–516, 2002.

[54] W. Z. Qing-Guo Zhang, Hua-Yong Liu, K. C. Ya-Jun Guo, L. Wang, and Y. Ong. Drawing undirected graphs with genetic algorithms. *ACM Transactions on Graphics*, page 28 – 36, 2005.

[55] N. Quinn and M. Breuer. A force directed component placement procedure for printed circuit boards. *IEEE Transactions on Circuits and Systems*, 26(6):377–388, 1979.

[56] P. P. R. Horst and N. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, 2000.

[57] Y. S. R.G. Strongin. Global optimization with non-convex constraints: Sequential and parallel algorithms. *Kluwer Academic Publishers*, 8, 200,2013,2014.

[58] M. M. Stephanie Forrest. What makes a problem hard for a genetic algorithm?

some anomalous results and their explanation. In *Machine Learning*, pages 285–319, 1993.

[59] M. G. Stjepam Picek. *Dealings with problem hardness in genetic algorithms*, volume 8. 05 2009.

[60] G. Strang. *Linear Algebra and Its Applications*. Thomson, Brooks/Cole, 2006.

[61] D. B. G. B. C. Tamassia, R. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):61–79, 1988.

[62] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):635–650, 1987.

[63] E. M. Timo Eloranta. Timga - a genetic algorithm for drawing undirected graphs. *Divulgaciones Matematicas*, 9:155–170, 01 2001.

[64] C.-K. Ting. On the mean convergence time of multi-parent genetic algorithms without selection. *Advances in Artificial Life*, page 403–412, 2005.

[65] R. Trudeau. *Introduction to Graph Theory*. Dover Books on Mathematics. Dover Publications, 2013.

[66] J. Tsay, B. Wu, and Y. Jeng. Hierarchically organized layout for visualization of biochemical pathway. In *2008 International Conference on BioMedical Engineering and Informatics*, volume 1, pages 153–157, May 2008.

[67] W. T. Tutte. How to draw a graph. *Proc. London Math. Society*, 13(52):743–768, 1963.

[68] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In J. Marks, editor, *Graph Drawing*, pages 171–182, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[69] K. Wegner and U. Kummer. A new dynamical layout algorithm for complex biochemical reaction networks. *BMC Bioinformatics*, 6:212–212, 2004.