Journal of Modern Technology and Engineering
Vol.8, No.3, 2023, pp.196-206

CIRCUIT COVERING THE EUCLIDEAN COMPLETE GRAPH

Fidan Nuriyeva[1,2]

[1]Dokuz Eylul University, Faculty of Science, Department of Computer Science, Izmir, Türkiye
[2]Institute of Control Systems, The Ministry of Science and Education of the Republic of Azerbaijan, Baku, Azerbaijan

Abstract. In this study, the concept of the Circuit Covering the Euclidean Complete Graph has been defined, and an algorithm has been proposed to find this circuit. The Circuit Covering the Graph can be utilized as a stage in solving various graph problems, such as the Routing Problems and Geometric Design Problems.

Keywords: Graph, Circuit Covering the Graph, Convex Hull, Traveling Salesman Problem.
AMS Subject Classification: 05C10, 05C38, 05C85.
Corresponding author: Fidan, Nuriyeva, Dokuz Eylul University, Faculty of Science, Department of Computer Science, Izmir, Türkiye, Institute of Control Systems, The Ministry of Science and Education of the Republic of Azerbaijan, Baku, Azerbaijan, e-mail: *nuriyevafidan@gmail.com*
Received: 10 July 2023; Revised: 29 October 2023; Accepted: 25 November 2023;
Published: 30 December 2023.

1 Introduction

Graph Theory are used for solving a multitude of complex and comprehensive problems in modern life. These applications encompass areas such as economics, management science, sales and marketing, information transmission, and transportation planning. Graph theory is also beneficial in defining problems and determining structural relationships (Agnarsson & Greenlaw (2007)).

Graph problems are widespread in computer science. Hundreds of interesting computational problems are expressed in graphs. Examples of these problems include the Hamiltonian circuit, Eulerian circuit, Traveling Salesman Problem, and others. After addressing these problems below, a new concept: 'Circuit Covering the Graph' is defined, and an algorithm is proposed to find this circuit.

The article first recalls the necessary concepts related to graphs, touches upon the Traveling Salesman Problem, provides concepts related to the Convex Hull, and showcases the two best-known algorithms. Subsequently, concepts related to the Circuit Covering the Graph are defined, and an algorithm is designed to find this circuit. Finally, a comparison between the Circuit Covering the Graph and the Convex Hull of the Vertex Set of the Graph is presented, and their applications are discussed.

2 Computational Geometry

Although geometry has been studied since ancient times, the development of algorithms for geometric problems is considered relatively recent. *Computational Geometry* is a branch of Computer Science that focuses on algorithms solving geometric problems (Preparata & Shamos (1985)).

In modern engineering and mathematics, Computational Geometry finds applications in various fields such as Computer Graphics, Robotics, VLSI Design, Computer-Aided Design, Molecular Modeling, Metallurgy, Manufacturing, Textile Design, Forestry, and Statistics.

The input to a problem in Computational Geometry is typically given as a set of geometric objects, such as a set of points, a set of line segments, or an ordered set of vertices of a polygon in counterclockwise order. The output often answers questions related to geometric objects, such as whether lines intersect or whether there is a new geometric object, like the convex hull (the smallest convex polygon envelope), associated with a set of points.

In this article, we will address a Computational Geometry problem in the plane, specifically in two-dimensional space. We will represent each object's input as $p_i = (x_i, y_i)$, where $x_i, y_i \in R$, with a set of points $p_1, p_2, p_3, \dots p_n$. For example, an n-sided polygon P will be represented by the array $p_1, p_2, p_3, \dots, p_n$, indicating the order in which its vertices appear along the boundary of P.

3 Graphs

A ***Graph*** has a finite set V called the ***Vertex set***. Let there be a relation $(a, b) \in R$ on V that is symmetric and non-reflective. Let E be the collection of 2-element subsets of V. In other words, for elements a and b in V, $(a, b) \in E$. Here, the set E is called the ***Edge set***, and each element is referred to as an edge of the graph.

A graph is represented as $G(V, E)$ or simply denoted as G.

If $(a, b) \in E$, it is said that the two elements a and b of the V set are *connected*.

A finite graph is depicted with shapes. The ***Vertices*** are represented by *points*, and the ***edges*** connecting the vertices are shown with *lines*.

Let v be any vertex in the graph G. The degree of the vertex v is the number of edges adjacent to v.

If every edge is labeled, it is called a *labeled/weighted* graph.

A *walk* in which each vertex is used once is called a path.

A path from the vertex v_0 to v_n in a graph is expressed as an array $(v_0, e_1, v_1, \dots, v_{n-1}, e_n, v_n)$, consisting of $(n + 1)$ vertices and n edges. In other words, a graph with end vertices of degree 1 and inner vertices of degree 2 is called a *path graph*. The path graph with n vertices is denoted by P_n.

A closed walk with all vertices of degree 2 and the number of vertices $n \geq 3$ is called a *cycle graph*. The number of edges of a contour graph with n vertices is n, and this graph is denoted C_n.

Graphs with an edge between every pair of distinct vertices are called ***complete graphs***. A complete graph with n vertices is denoted as K_n.

An ***Euler Path*** in a graph G is a closed walk that includes all edges of G exactly once. A closed Euler path is called an *Euler circuit*. If a graph contains at least one Euler Path, it is an *Euler graph*.

If there is a path visiting each vertex only once, it is called a ***Hamiltonian path***. *A Hamiltonian circuit* in a graph is a circuit that passes through each vertex once. If a graph contains a Hamiltonian circuit, it is a *Hamiltonian graph* (Agnarsson & Greenlaw (2007)).

Applications of Hamilton circuits include interesting problems such as the Traveling Salesman Problem.

4 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem studied in the fields of Operations Research and Computer Science. The TSP aims to find the shortest or least costly tour that visits each of the n known points (cities, locations, or nodes) exactly once, where the distances between them are known. In graph theory, the TSP can be defined as finding the least costly Hamiltonian Circuit in a weighted graph (where nodes represent cities, edges represent the roads between cities, and weights represent the cost or length of the roads) (Nuriyeva (2023)).

This problem has a wide range of applications, including determining routes for vehicles distributing products to customers based on demands from one or more depots, airport routing for airplanes, and determining the locations of base stations for GSM operators in the fields of transportation, logistics, and communication. Additionally, given that the Traveling Salesman Problem forms the foundation for many other problems, developing efficient solution methods for this problem is crucial.

5 Convex Cover of the Vertices of the Given Graph

The *convex combination* of two distinct points, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, is any point $p_3(x_3, y_3)$ for some $0 \leq a \leq 1$, where $x_3 = ax_1 + (1-a)x_2$ and $y_3 = ay_1 + (1-a)y_2$. Additionally, it can be expressed as $p_3 = ap_1 + (1-a)p_2$. Intuitively, we can say that p_3 is any point on the line passing through p_1 and p_2, and p_1 and p_2 are either on or between p_3 on that line (Artigas et al. (2010)).

The ***Convex Hull*** of a set of points Q is denoted as ***CH(Q)*** and is the smallest convex polygon such that every point in $\boldsymbol{Q}$ lies either on the boundary or inside $\boldsymbol{P}$ (Buzatu & Cataranciuc (2015)). Indirectly, we can think of every point in $\boldsymbol{Q}$ as a nail protruding from the surface. Therefore, the Convex Hull is like a tight rubber band that envelops all the points. Figure 1 illustrates the set of V vertex points of a given graph and its convex hull.

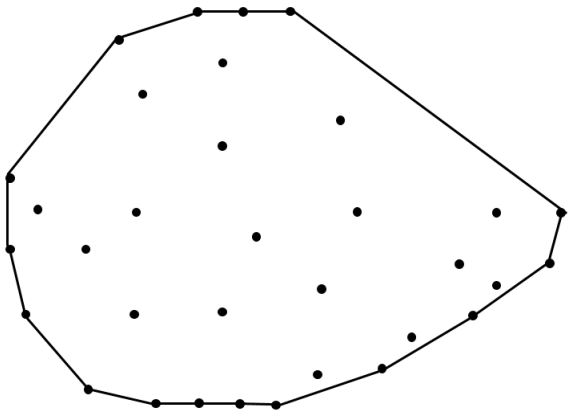

Figure 1: Convex Cover of the V Vertices of the Given Graph

We can demonstrate two algorithms that calculate the convex hull of a set of n points (Kirkpatrick & Seidel R. (1986)). Both algorithms employ a technique known as "rational sweep" which means they output the corners of the convex hull in the counterclockwise direction. The first of these algorithms is Graham's Scan, which runs in $O(nlgn)$ time (Graham (1972)). The second is Jarvis's March, which operates in $O(nh)$ time, where h represents the number of corners of the convex hull (Jarvis (1973)). As shown in Figure 1, each corner of ***CH(Q)*** is a point within $\boldsymbol{Q}$.

6 Graph Covering Circuit

Assuming a graph $G(V, E)$ where the set of V vertex points of the graph lies within a rectangle $ABCD$ with coordinates in the plane $[A(x_l, y_d), B(x_l, y_u), C(x_r, y_u), D(x_r, y_d)]$, and the coordinates of the vertex points $v_i = (x_i, y_i)$ are given as $x_i, y_i \in R$ (Figure 2).

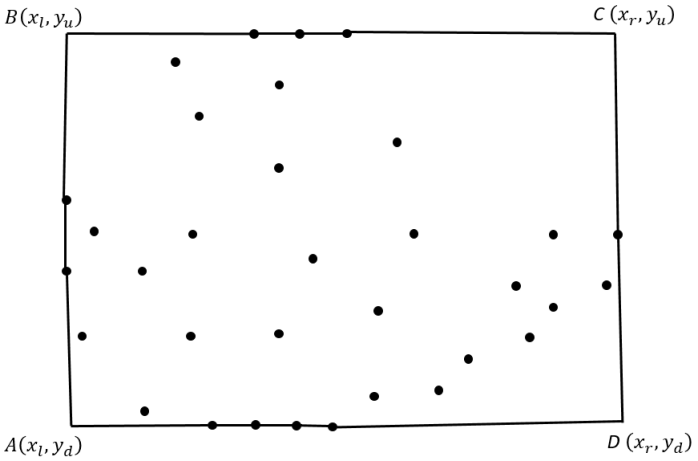

Figure 2: Rectangle Containing the Vertices of the Given Graph

We will refer to graphs of this type, given in the plane, as ***Euclidean Graphs***, similar to the Traveling Salesman Problem (TSP).

In practice, the graphs encountered in TSP generally take the form of such graphs. As it is known, the graphs of TSP are Complete Graphs (Figure 3).

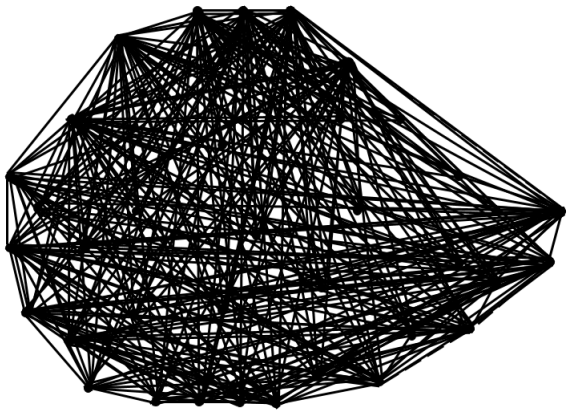

Figure 3: Complete Graph of Given Vertices

We will refer to the leftmost point (the point with the smallest first x-coordinate) in the set V as the *Left End Point*, and denote this set of points as ***L(V)*** (Figure 4).

We will call the second y-coordinate smallest among the Left End Point the *Lower Left End Point* and denote it as P_{dle}.

Similarly, among the Left End Points, the one with the largest second y-coordinate will be called the *Upper Left End Point*, and denoted as P_{ule}.

We will refer to the rightmost point (the point with the largest first x-coordinate) in the set V as the *Right End Point*, and denote this set of points as ***R(V)*** (Figure 4).

The second y-coordinate smallest among the Right End Points will be referred to as the *Lower Right End Point*, denoted as P_{dre}.

Similarly, among the Right End Points, the one with the largest second y-coordinate will be called the *Upper Right End Point*, and denoted as P_{ure}.

We will refer to the bottom most point (the point with the smallest second y-coordinate) in the set V as the *Down End Point*, and denote this set of points as ***D(V)*** (Figure 4).

The first x-coordinate smallest among the Down End Points will be referred to as the *Lower Left Down End Point*, denoted as P_{lde}.

Similarly, among the Down End Points, the one with the largest first x-coordinate will be called the *Lower Right Down End Point*, denoted as P_{rde}.

We will refer to the topmost point (the point with the largest second y-coordinate) in the set V as the *Upper End Point*, and denote this set of points as ***U(V)*** (Figure 4).

The first x-coordinate smallest among the Upper End Points will be referred to as the *Upper Left Upper End Point*, denoted as P_{lue}.

Similarly, among the Upper End Points, the one with the largest first x-coordinate will be called the *Upper Right Upper End Point*, denoted as P_{rue}.

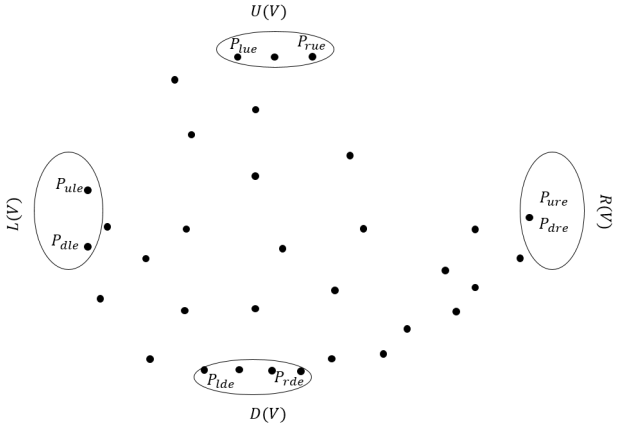

Figure 4: Sets of Graph Endpoints

As seen in Figure 4, $R(V)$ is a single point, so P_{ure} and P_{dre} are the same.

NOTE 1: For some graphs, some of the sets mentioned above may be the same set, or one of these sets may be a subset of the other. When the number of elements in any of these sets is 1, the upper-lower or right-left elements will be the same (Figure 5).

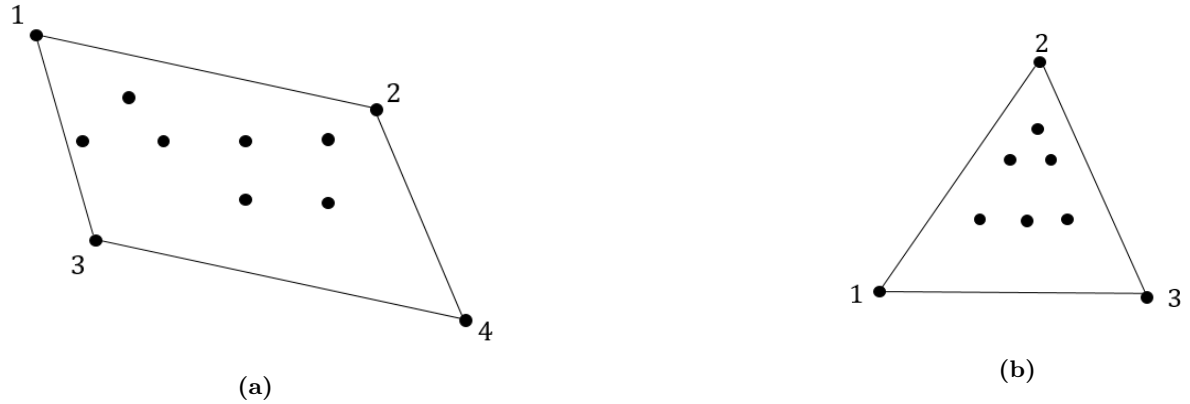

Figure 5: Examples showing when endpoints are set the same

In Figure 5(a), point 1 represents both the Left End Point and the Upper End Point; therefore, in this case, $L(V) = U(V) = \{1\}$. Also, point 4 represents both the Right End Point and the Down End Point; hence, $R(V) = D(V) = \{4\}$.

In Figure 5(b), point 1 represents both the Left End Point and one of the Down End Points; thus, in this case, $L(V) \cap D(V) = \{1\}$. Similarly, point 3 represents both the Right End Point

and one of the Down End Points; therefore, $R(V) \cap U(V) = 3$.

The *Set of End Points* ***SE(V)*** for the set of Vertex Points V will be as follows:

$$SE(V) = L(V) \cup R(V) \cup D(V) \cup U(V)$$

If $P(x_{dl}, y_{dl}) \in (V - SE(G))$ satisfies the following condition, we will refer to it as the *Down Left Limit Vertex*, and denote this set of vertices as ***DL(V)*** (Figure 6):

$$V \cap [(x_l, y_d), (x_l, y_{dl}), (x_{dl}, y_{dl}), (x_{dl}, y_d)] = P(x_{dl}, y_{dl})$$

Note 2: Here and below, the notation $[(x_l, y_d), (x_l, y_{dl}), (x_{dl}, y_{dl}), (x_{dl}, y_d)]$ represents the coordinates of the End Points, showing a rectangle with coordinates $(x_l, y_d), (x_l, y_{dl}), (x_{dl}, y_{dl}), (x_{dl}, y_d)$.

If $P(x_{ul}, y_{ul})$ $(V - SE(G))$ satisfies the following condition, we will refer to it as the *Upper Left Limit Vertex*, and denote this set of vertices as ***UL(V)*** (Figure 6):

$$V \cap [(x_l, y_{ul}), (x_l, y_u), (x_{ul}, y_u), (x_{ul}, y_{ul})] = P(x_{ul}, y_{ul})$$

If $P(x_{ur}, y_{ur}) \in (V - SE(G))$ satisfies the following condition, we will refer to it as the *Upper Right Limit Vertex*, and denote this set of vertices as ***UR(V)*** (Figure 6):

$$V \cap [(x_{ur}, y_{ur}), (x_{ur}, y_u), (x_r, y_u), (x_r, y_{ur})] = P(x_{ur}, y_{ur})$$

If $P(x_{dr}, y_{dr}) \in (V - SE(G))$ satisfies the following condition, we will refer to it as the *Down Right Limit Vertex*, and denote this set of vertices as ***DR(V)*** (Figure 6):

$$V \cap [(x_{dr}, y_d), (x_{dr}, y_{dr}), (x_r, y_{dr}), (x_r, y_d)] = P(x_{dr}, y_{dr})$$

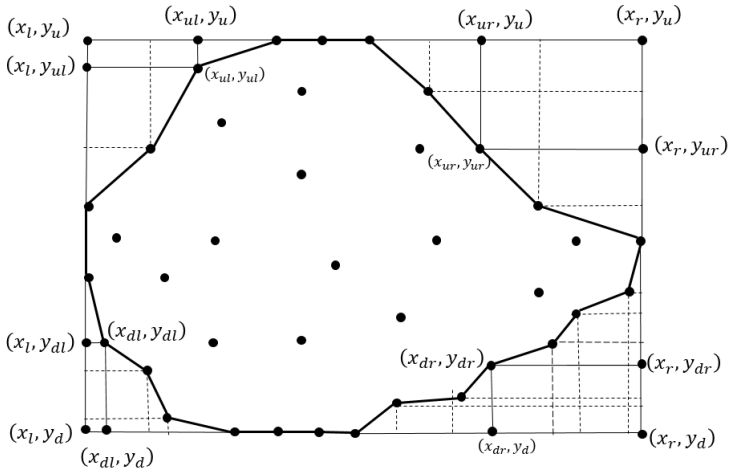

Figure 6: Displaying the Boundary Vertices of the Graph

Thus, *the Set of Edge Points* of the Vertex Points V consists of the union of the Set of End Points ∂V and the Set of Limit Points (Figure 7).

$$\partial V = SE(V) \cup DL(V) \cup UL(V) \cup UR(V) \cup DR(V)$$

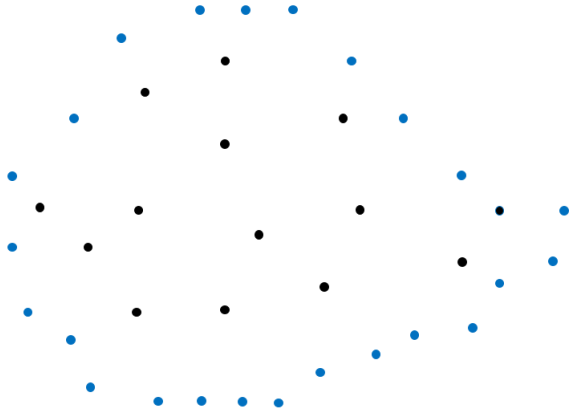

Figure 7: Set of Edge Points of Graph

In Figure 7, the Edge Points are highlighted in blue.

$V - \partial V$ forms the set of *Internal Points* of the Graph.

The ∂G Circuit passing through the Edge Points of the set of Vertex Points V composes the *Circuit Covering the Graph* (Figure 8).

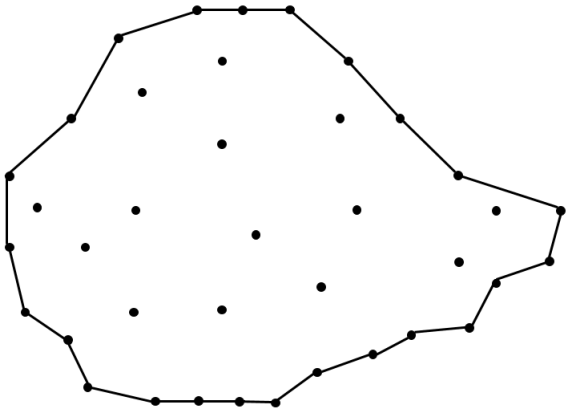

Figure 8: Circuit Covering the Graph

7 Circuit Covering the Graph Algorithm

1. **All vertex points of the graph are first sorted in ascending order based on their first coordinates, and then, if there are vertex points with the same first coordinates, they are sorted based on their second coordinates:**

$$\begin{aligned} A_1(x_1, y_1) \prec A_2(x_2, y_2) \prec \ldots \prec A_n(x_n, y_n) \\ x_i < x_j \Rightarrow A_i(x_i, y_i) \prec A_j(x_j, y_j) \\ y_i < y_j \Rightarrow A_i(x, y_i) \prec A_j(x, y_j) \end{aligned} \tag{1}$$

   In other words, the vertex points of the graph are initially sorted in ascending order based on their first x-coordinate. During this sorting, those with the same first x-coordinate will be placed consecutively within a range in this array. Then, within each of these ranges, the elements are sorted based on their second y-coordinate.

2. **The End Points of the graph are identified, and from these End Points, the bottom, top, right, and left points are determined (Figure 9)**.

2.1. To select the Left End Points ($L(V)$), the points with the smallest first x-coordinate in array (1) are chosen. These points may be one or more. If there is only one point, it becomes both the Left Bottom End and Left Upper End. If these points are two or more, they are consecutively connected to form the Left Side of the Circuit Covering the Graph.

2.2. To select the Right End Points ($R(V)$), the points with the largest first x-coordinate in array (1) are chosen. These points may be one or more. If there is only one point, it becomes both the Right Bottom End and Right Upper End. If these points are two or more, they are consecutively connected to form the Right Side of the Circuit Covering the Graph.

2.3. To select the Upper End Points ($U(V)$), the points with the largest second y-coordinate in array (1) are chosen. These points may be one or more. If there is only one point, it becomes both the Upper Left End and Upper Right End. If these points are two or more, they are consecutively connected to form the Upper part of the Circuit Covering the Graph.

2.4. To select the Bottom End Points ($D(V)$), the points with the smallest second y-coordinate in array (1) are chosen. These points may be one or more. If there is only one point, it becomes both the Bottom Left End and Bottom Right End. If these points are two or more, they are consecutively connected to form the Bottom part of the Circuit Covering the Graph.

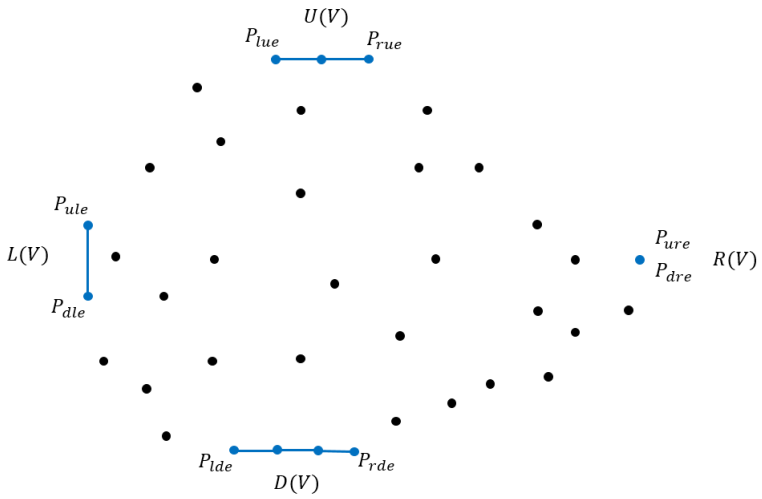

Figure 9: Finding the endpoints of the graph

3. **Starting from the Left End Points, first, the Left Down and Upper Limit Points of the Graph are identified and connected. Then, starting from the Bottom and Upper Right End Points, the Bottom Right and Upper Limit Points of the Graph are identified and connected.**

3.1. *To determine the Left Down Limit Points of the Graph* ($DL(V)$)*, the following steps are taken* (Figure 10):
Let x_{dle} be the first element in the array (1) first coordinate. Assume $x_{dl} = x_{dle}$, and $y_{dl} = y_{dle}$.
Let $x_i > x_{dl}$ be the smallest first coordinate after x_{dl} in this array. The following procedure is applied to the interval starting with x_i in the array (1):
Consider the element with the first coordinate x_2 and the smallest second coordinate $y_i f$ in the array (1). This element becomes the first element of the interval reserved

for x_i in the array (1). If this coordinate is less than y_{dl}, $A(x_i, y_i f)$ becomes one of the Left Down Limit Points of the Graph, and $A(x_d l, y_d l)$ is connected to $A(x_i, y_i f)$, and $x_{dl} = x_i$ and $y_{dl} = y_{if}$ are assumed. Otherwise, $A(x_i, y_{if})$ becomes an internal point of the Graph.
In the following steps, the smallest $x_(i + 1)$ after the first coordinate x_i, then the smallest $x_{(i+1)+1}$ after x_{i+1}, and others are selected from the array (1), and the above procedures are applied for appropriate intervals.
These procedures are completed when $x_{dl} = x_{lde}$ and $y_{dl} = y_{lde}$.

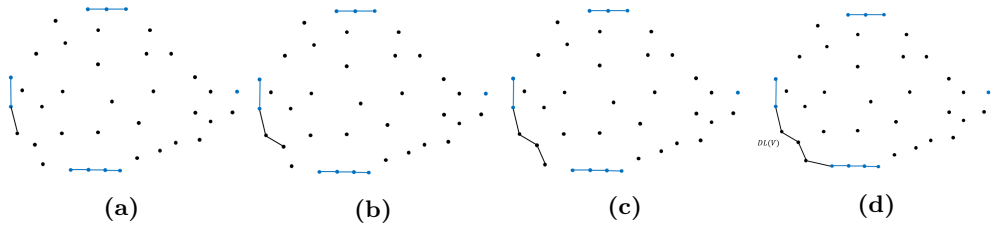

Figure 10: Examples showing when endpoints are set the same

3.2. *To determine the Left Upper Limit Points of the Graph ($UL(V)$), the following steps are taken (Figure 11):*
Let x_{ule} be the first element in the array (1) first coordinate. Assume $x_{ul} = x_{ule}$, and $y_{ul} = y_{ule}$.
Let $x_i > x_{ul}$ be the smallest first coordinate after x_{ul} in this array. The following procedure is applied to the interval starting with x_i in the array (1):
Consider the element with the first coordinate x_i and the largest second coordinate $y_i e$ in the array (1). This element becomes the last element of the interval reserved for x_i in the array (1). If this coordinate is greater than y_{ul}, $A(x_i, y_{ie})$ becomes one of the Left Upper Limit Points of the Graph, and $A(x_{ul}, y_{ul})$ is connected to $A(x_i, y_{ie})$, and $x_{ul} = x_i$ and $y_{ul} = y_{ie}$ are assumed. Otherwise, $A(x_i, y_{ie})$ becomes an internal point of the Graph. In the subsequent steps, the smallest x_3 after the first coordinate in the array (1), then the smallest x_4 after x_3, and so on, are selected, and the above operations are performed for appropriate intervals.
These procedures are completed when $x_{ul} = x_{ule}$ and $y_{ul} = y_{ule}$.

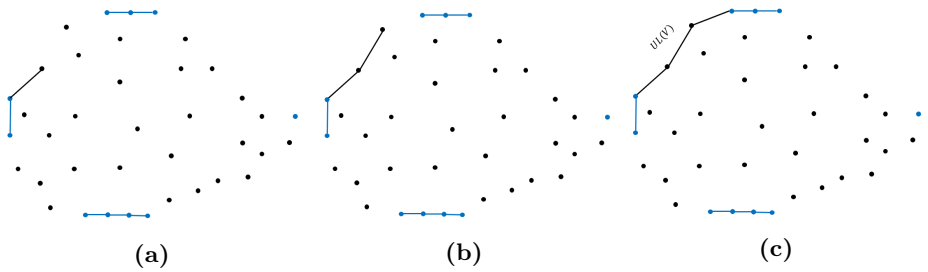

Figure 11: Combining the graph by determining the Upper Left Boundary points

3.3. *To determine the Upper Right Boundary points of the graph ($UR(V)$), the following steps are performed:*
Let's assume $x_{ur} = x_{ure}$, and $y_{ur} = y_{ure}$.
$x_i < x_{ur}$, and in this array, x_{ur} is the largest first coordinate after x_i. For the interval starting with x_i in the (1) array, the following process is carried out:
An element is selected in the array (1) with the first coordinate x_i and the largest second coordinate y_{ie}. This element becomes the last element of the interval allocated

for x_i in the array (1). If the coordinate is greater than yur, $A(x_i, y_{ie})$ becomes one of the Upper Right Boundary points of the graph, and $A(x_{ur}, y_{ur})$ is merged with $A(x_i, y_{ie})$, and $x_{ur} = x_i$, and $y_{ur} = y_{ie}$ are accepted. Otherwise, $A(x_i, y_{ie})$ becomes an internal point of the graph.
In the subsequent steps, the following steps are performed for suitable intervals by selecting the largest $x_{(i+1)}$ after first coordinate x_i, then the largest $x_{(i+1)+1}$ after $x_{(i+1)}$, and so on:
These procedures are completed when $x_{ur} = x_{rue}$ and $y_{ur} = y_{rue}$.

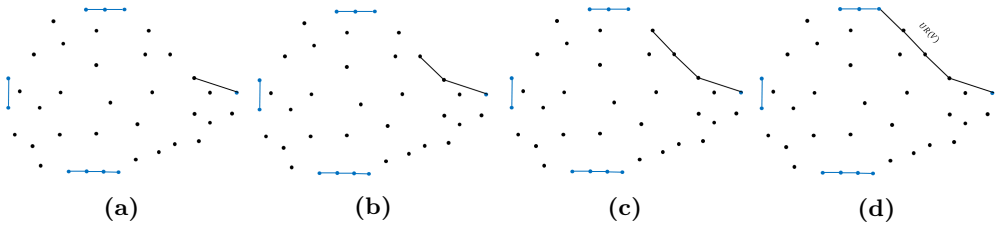

Figure 12: Combining the graph by determining the Upper Right Boundary points

3.4. To determine the Lower Right Boundary points of the graph ($DR(V)$), the following steps are performed:
Let's assume $x_{dr} = x_{dre}$, and $y_{dr} = y_{dre}$.
$x_i < x_{dr}$, and in this array, x_{dr} is the largest first coordinate after x_i. For the interval starting with x_i in the array (1), the following process is carried out:
An element is selected in the array (1) with the first coordinate x_i and the smallest second coordinate y_{ie}. This element becomes the first element of the interval allocated for x_i in the array (1). If the coordinate is less than yur, $A(x_i, y_{ie})$ becomes one of the Lower Right Boundary points of the graph, and $A(x_{dr}, y_{dl})$ is merged with $A(x_i, y_{ie})$, and $x_{dr} = x_i$, and $y_{dr} = y_{ie}$ are accepted. Otherwise, $A(x_i, y_{ie})$ becomes an internal point of the graph.
In the subsequent steps, the following steps are performed for suitable intervals by selecting the largest $x_{(i+1)}$ after the first coordinate x_i in the array (1), then the largest $x_{(i+1)+1}$ after $x_{(i+1)}$, and so on:
These procedures are completed when $x_{dr} = x_{rde}$ and $y_{dr} = y_{rde}$. This completes the Circuit Encompassing Algorithm for the graph.

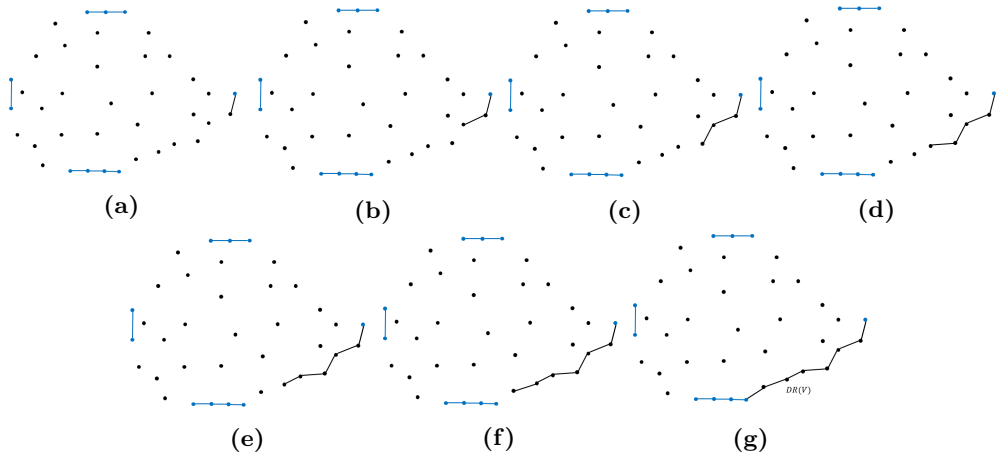

Figure 13: Combining the graph by determining the Lower Right Boundary points

8 Conclusion

The complexity of the proposed algorithm is $O(nlgn)$. This algorithm can be further developed to find the Convex Hull of the Graph. The Graph Covering Circuit always remains within the Convex Hull of the Graph and forms a tighter envelope. The common parts of the Convex Hull and the Graph Covering Circuit may exist, meaning they can intersect. The Vertex points of the Graph are present both on the Convex Hull and on the Covering Circuit (Figure 14).

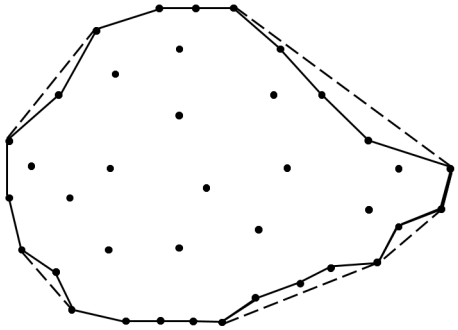

Figure 14: Convex Cover of Graph and Covering Circuit

In Figure 14, the Graph Covering Circuit is shown with solid lines, and the non-intersecting parts of the Convex Hull of the Graph with the Covering Circuit are indicated with dashed lines.

The Graph Covering and the Convex Hull of the Vertex Points Set in the Graph can be used as a stage in solving various Graph Problems, such as the Traveling Salesman Problem (Nuriyev et. al. (2018)), and in Geometric Design Problems for the future works.

References

Agnarsson, G., Greenlaw, R. (2015). *Graph Theory: Modeling, Applications, and Algorithms.* Prentice Hall, 464p.

Artigas, D., Dantas, S., Dourado, M.C., & Szwarcter, J.L. (2010). Convex covers of graphs. *Matematica Contemporanea, Sociedade Brasileira de Matematica*, *39*, 31-38.

Buzatu, R., Cataranciuc, S. (2015). Convex graph covers. *Computer Science Journal of Moldova*, *23*(3), 251-269.

Graham, R.L. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, *1*(4), 132-133.

Jarvis, R.A. (1973). On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, *2*(1), 18-21.

Kirkpatrick, D.G., Seidel, R. (1986). The ultimate planar convex hull algorithm. *SIAM Journal on Computing*, *15*(2), 287-299.

Nuriyeva, F. (2023). Heuristic Algorithm for Solving Travelling Salesman Problem: Adding the Endpoints of the Longest Edges. *Proceeding of the 5.th International Conference on "Problems of Cybernetics and Informatics (PCI-2023)*, 368-372.

Nuriyev, U., Ugurlu, O. & Nuriyeva, F. (2018). Self-Organizing Iterative Algorithm for Travelling Salesman Problem. *IFAC-Papers OnLine*, *51*(30), 268-270.

Preparata, F.P., Shamos, M.I. (1985). *Computational Geometry: An Introduction.* Springer, 398.