The use of GeoGebra in Discrete Mathematics

Ph.D. **Raúl M. Falcón**
School of Building Engineering. University of Seville, Spain.
rafalgan@us.es

Ph.D. **Ricardo Ríos**
I.E.S. Julio Verne, Seville, Spain.
profesofricardo@yahoo.es

ABSTRACT: In this paper we explain how to make use of the commands that are available in GeoGebra to deal with some realistic problems related to the field of Discrete Mathematics. We also expose how to define new tools that make possible the study of theoretical results in Graph theory.

KEYWORDS: Discrete Mathematics, Graph Theory, Shortest Path Problem, Bézier curves, Pattern recognition, Art Gallery Problem, Travelling Problem, Random graphs.

1 Introduction

Discrete Mathematics deals with structures formed by a set of objects that can be either finite or enumerated by the set of positive integers. From an educational point of view, this field constitutes an interesting subject to be included in the curriculum of Mathematics not only at the university level but also at the secondary level [Ouv14, RFR97]. It is due to the fact that Discrete Mathematics comprises distinct and varied topics that can be easily applied in the real world and immediately understood by any student without an extensive background in Mathematics. Examples of these topics are, for instance, Combinatorics, Computer Science, Cryptography, Decision theory, Discrete Probability, Game theory, Graph theory, Information theory or Operation Research. Particularly, Brousseau [Bro97] established Game

theory as a main tool to analyze the processes of teaching and learning Mathematics by conceptualizing the theory of didactical situations.

Incidence structures constitute one of the main structures studied in Discrete Mathematics. An *incidence structure* is a triple *(P, B, I)* formed by a finite set *P* of points, a finite set *B* of lines and an incidence relation $I \subseteq P$ x *B*, where, given a point *p* and a line *l* such that $(p,l) \in I$, it is said that *p lies on l* or that *l contains p*. This incidence structure constitutes a *graph* if every line passes exactly through two points. Equivalently, a *graph G = (V, E)* is a pair formed by a set *V* of points called *vertices* and a set *E* of lines connecting them called *edges* (see Figure 1). Since the problem of the Seven Bridges of Königsberg, solved by Euler [Eul36] in 1736, *Graph theory* has been established as a prolific source of applications to realistic problems like the art gallery problem, the map coloring problem, the traveling problem or the shortest path problem, amongst others.

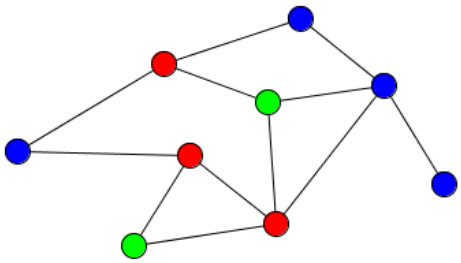

Figure 1: Graph of 9 vertices and 12 edges drawn in GeoGebra.

Nowadays, there exists a wide amount of graph drawing software systems that can be used to deal with distinct applications of Graph theory in realistic problems like analysis and visualization of networks, flows, concept maps or data mining, study of molecular interaction networks or generation of random graphs [JP04]. Nevertheless, because of their complexity, these systems do not constitute in general a good tool to teach basic concepts and results on Graph theory. To this end, more simple and environment-friendly systems are required like, for instance, the recent versions of *Grafos* [Rod10] or *GraphShop* [And11]. In this paper, we propose the dynamic geometry system *GeoGebra* as a very interesting alternative to deal with not only Graph theory but Discrete Mathematics in general. The series of commands devoted to Discrete Mathematics that GeoGebra has implemented by defect together with its dynamical, intuitive and friendly environment constitutes a good alternative to introduce basic concepts and results on the majority of the topics that we have previously mentioned.

The paper is organized as follows. In Section 2 we introduce the seven commands on Discrete Mathematics implemented by defect in GeoGebra 5.0 and we explain how to use them to solve distinct problems with application in the real world and which can easily be implemented in Maths classes at the secondary level. In Section 3 we explain how to develop new tools on Graph theory to be implemented in GeoGebra. Specifically, we expose how to construct random graphs whose edges are randomly generated according to a normal distribution. Even if the paper is self-contained, we refer to the monograph of Rosen [Ros99] for more details on basic concepts and problems on Discrete Mathematics.

2 Commands devoted to Discrete Mathematics in GeoGebra

There exist seven commands on Discrete Mathematics that are implemented by defect in GeoGebra 5.0: *ConvexHull*, *DelaunayTriangulation*, *Hull*, *MinimumSpanningTree*, *ShortestDistance*, *TravellingSalesman* and *Voronoi*. In this section we introduce them and we expose some possible applications in real-world problems.

2.1 ConvexHull

A polygon P is *convex* if whenever two points P_1 and P_2 lie inside of P, then the whole segment P_1P_2 is also inside of P. The *convex hull* of a set $S = \{P_1, \ldots, P_n\}$ of n points in the plane is the smallest convex polygon that contains all the points in S. In GeoGebra, this polygon can be obtained by using the command *ConvexHull[<List of Points>]*. This command can be used to solve *shortest path problems* related to *robot motion planning* as the next one:

> *"Let P_1 and P_2 be two points on the outside of the convex hull of a star-shaped polygon P such that the segment line P_1P_2 passes through P. Determine the shortest path between the points P_1 and P_2 such that none point of the path is inside of P."*

Recall that a polygon is said to be *star-shaped* if it contains a point from which the entire polygon boundary is visible. To solve this problem, we draw in GeoGebra both points P_1 and P_2 and the polygon P. After that, we use the command *ConvexHull* to obtain the convex hull related to P_1, P_2 and all the vertices of P. The boundary of this hull determines two paths from P_1 to P_2,

of which the shortest one is the solution of our problem. It can be determined by using the tool *Distance or Length* of GeoGebra (see Figure 2).

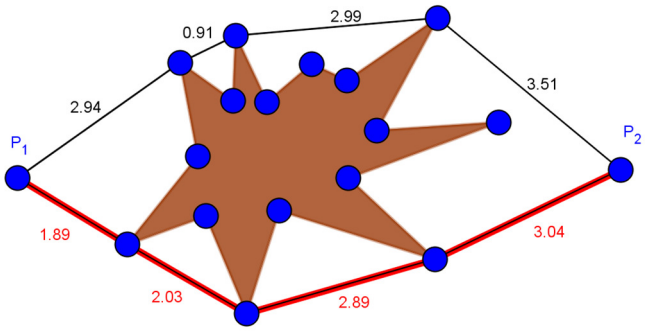

Figure 2: Shortest path related to a convex hull.

An interesting property that can be easily tested in GeoGebra is the fact that the ***Bézier curve*** of every set of control points is contained in their convex hull (see Figure 3). Bézier curves are piecewise polynomial functions defined from a set of control points, with the property that any affine transformation of the curve coincides with the Bézier curve of the transformed control points. Specifically, given a set $S = \{P_1, \ldots, P_n\}$ of n points in the plane, its Bézier curve is defined parametrically as

$$B(t) = \sum_{i=0}^{n} P_i \cdot \binom{n}{i} t^i (1-t)^{n-i},$$

where $0 \leq t \leq 1$. In GeoGebra, this curve can be defined by entering the next commands in the input bar:

n = Length[S]
T=Sequence[BinomialCoefficient[n-1,i]x^i (1 - x)^(n - 1 - i),i, 0, n - 1]
Px = Sequence[x(Element[S, i]) Element[T, i], i, 1, n]
Py = Sequence[y(Element[S, i]) Element[T, i], i, 1, n]
Sx(x) = Sum[Px]
Sy(x) = Sum[Py]
Curve[Sx(t), Sy(t), t, 0, 1]

For more details about the visualization of Bézier curves in GeoGebra we refer to the paper of Viera [Vie14].

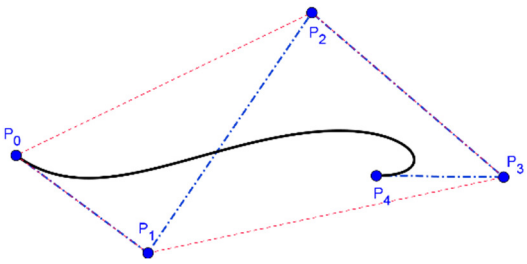

Figure 3: Bézier curve and convex hull of five points in GeoGebra.

2.2 DelaunayTriangulation

The *Delaunay triangulation* of a set $S = \{P_1, \ldots, P_n\}$ of n points in the plane is the subdivision into triangles of the convex hull of S such that no point in S is inside the circumcircle of any triangle of the triangulation. This triangulation is unique and constitutes the most regular possible triangulation because it maximizes the minimum angle of all the triangles. Particularly, any pair of closer points are always connected by an edge of the triangulation. Due to all these facts, the Delaunay triangulation has a wide range of applications in pattern recognition, terrain modeling, computer vision or image and video compression, amongst others. In GeoGebra, this triangulation can be determined by using the command *DelaunayTriangulation[<List of Points>]*. Figure 4 shows the Delaunay triangulation constructed by GeoGebra from a cloud of points that characterizes the colour and contrast changes in a picture of a rose.

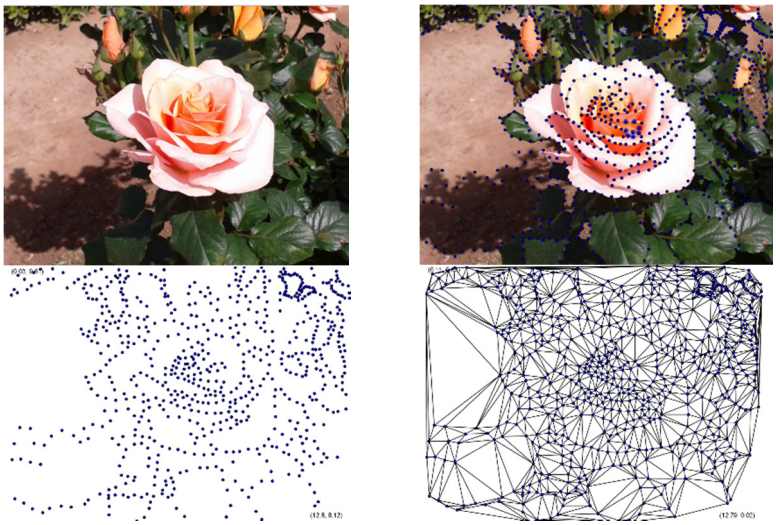

Figure 4: Pattern recognition in GeoGebra.

The command *DelaunayTriangulation* can also be used in GeoGebra to deal with other problems in Discrete Mathematics that require the use of a triangulation. This is the case, for instance, of the *art gallery problem*, which consists originally of determining the minimum number of guards that are required to look after an art gallery. This gallery is represented by a polygon on whose vertices must be placed the guards. To solve this problem, the polygon is triangulated and 3-coloured, that is, the vertices of the triangulation are coloured with three distinct colours so that every triangle contains all the three colours. After that, the smallest set of vertices with the same colour determines the position of the guards. This set has at most $\lfloor n/3 \rfloor$ vertices, where n is the number of vertices of the polygon. Figure 5 shows an example in this regard that has been solved by using GeoGebra. In the figure, the orange gallery has been considered as a polygon, its 24 vertices have been drawn and their Delaunay triangulation has been constructed. The vertices of those triangles contained in the gallery has then been 3-coloured. Remark that any of the resulting three sets of vertices having a same colour determines a possible distribution of guards because all of them have cardinality eight.

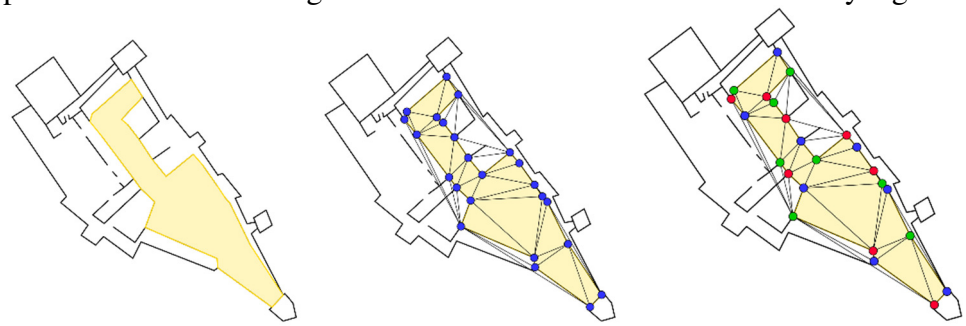

Figure 5: Art gallery problem in GeoGebra.

2.3 Hull

Given a set $S = \{P_1, \ldots, P_n\}$ of n points in the plane, the command *Hull[<List of Points>, Percentage]* determines in GeoGebra a polygon that constitutes the *characteristic shapes* or χ*-shapes* of S according to the algorithm defined by Duckham et al. [DK08]. Depending on a factor of percentage that fit the adjustment of the polygon to the cloud of points, this polygon can always be embedded in the Delaunay triangulation of S. The maximum percentage corresponds to the convex hull of S and the minimum corresponds to the maximum adjustment. Due to its relation with the Delaunay triangulation, χ-shapes are applied on shapes analysis and recognition. Figure 6 shows an

example in this regard, where we have made use of a slider in order to determine explicitly the distinct adjustments constructed by GeoGebra for the given set of points.

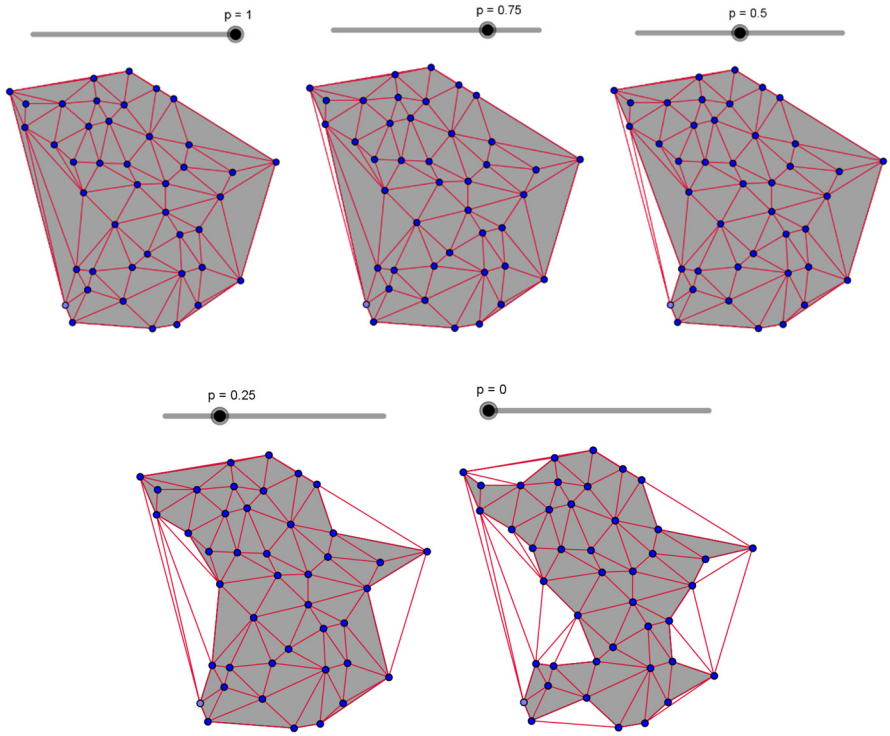

Figure 6: χ-shapes and Delaunay triangulation in GeoGebra.

2.4 MinimumSpanningTree

Given a graph $G = (V, E)$, the *degree* of a vertex $v \in V$ is the number of edges containing v. A *tree* is a graph with all its vertices having degree at most two. The graph G is said to be *connected* if, given two vertices v and v' in V, there exists a series of vertices $v_0 = v, v_1, \ldots, v_n = v'$ such that $v_i v_{i+1}$ determines an edge in E, for all $i \in \{0, \ldots, n - 1\}$. The series of edges $e_1 = v_0 v_1, \ldots, e_n = v_{n-1} v_n$ determines a *path* between v and v'. The graph G is said to be *weighted* if all its edges have associated a label or weight. A possible labelling is given by the length of each edge or, equivalently, by the Euclidean distance among its corresponding vertices. Finally, the graph G is said to be *complete* if every pair of vertices determines an edge (see Figure 7).

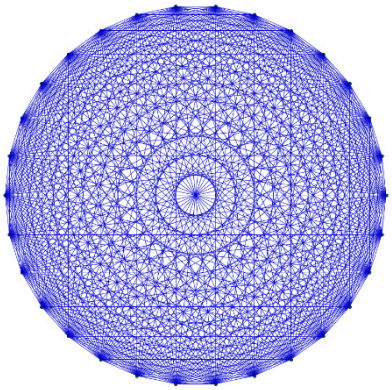

Figure 7: Complete graph of 30 vertices drawn in GeoGebra.

A *spanning tree* of a connected graph $G = (V, E)$ is a tree containing all the vertices of V and whose edges are in E. If the graph G is weighted, then the spanning tree is said to be *minimum* if there does not exist any other spanning tree with a lower total sum of weights in its edges.

In GeoGebra, given a set S of points, the command *MinimumSpanningTree[<List of Points>]* determines the minimum spanning tree of the weighted complete graph having S as set of vertices and the Euclidean distance between vertices as weight. This command can be used, for instance, to solve the next *shortest path problem*:

> "*Determine the road network of minimum cost joining the following central-European cities: Paris, Brussels, Luxembourg, Amsterdam, Berlin, Prague, Bern, Vaduz, Vienna, Bratislava and Budapest.*"

To solve this problem, we insert in GeoGebra a political map of Europe and mark the mentioned cities with eleven points $C_1, \ldots, C_{11}$. After that, we write in the input box the command *MinimumSpanningTree[{*$C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}$*}]*. The solution of the problem is shown in Figure 8.

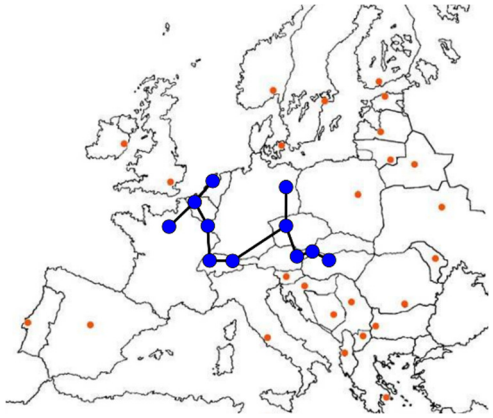

Figure 8: Road network of minimum cost.

2.5 ShortestDistance

Given a weighted graph *G = (V, E)*, the ***shortest distance problem*** consists of finding the path of minimum weight between two given vertices of the graph. In GeoGebra, this problem can be solved by using the command *ShortestDistance[<List of Segments>, <Start Point>, <End Point>, <Boolean Weighted>]*, where the Boolean value can be *true* or *false*, depending, respectively, on whether we use the Euclidean distance or the number of edges between points. Figure 9 shows an example in this regard. The red edges determine the shortest path between the points B and E according to the Euclidean distance, whereas the green ones determine the path with the minimum number of edges between these two points.

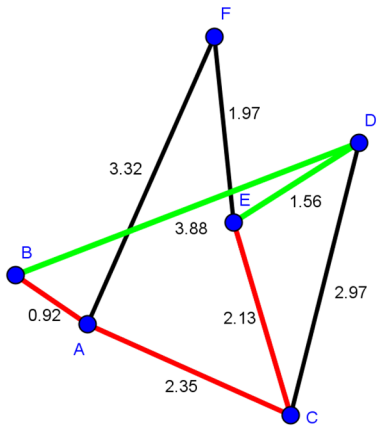

Figure 9: Shortest distances in GeoGebra.

Figure 10 shows a similar construction, based on a real situation.

"Which is the shortest path in New York to walk from the Empire State Building to the United Nations Headquarters?"

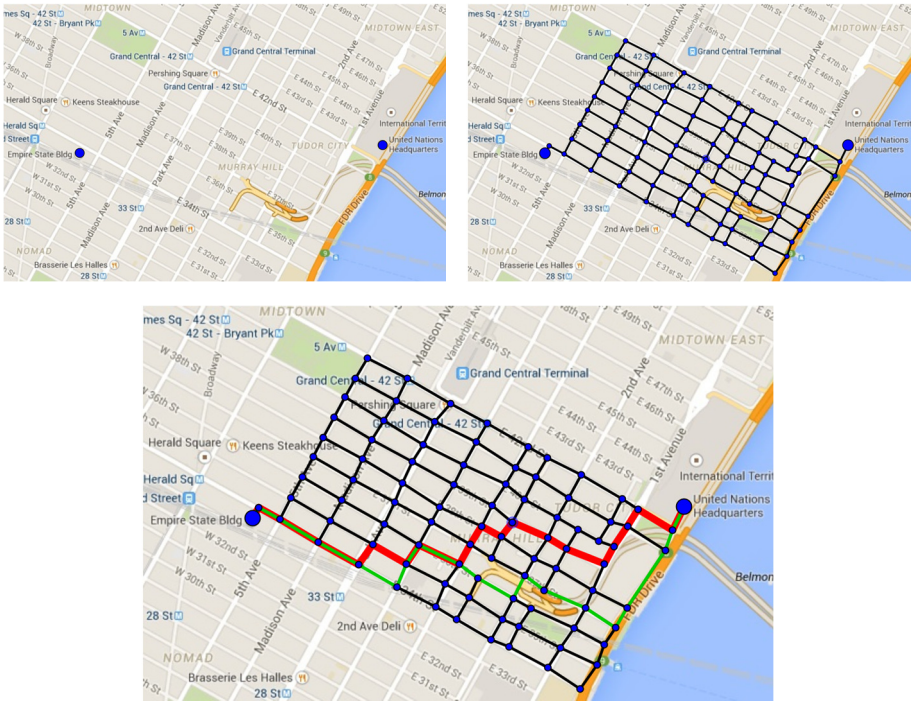

Figure 10: Shortest paths in New York by using GeoGebra.

2.6 TravellingSalesman

Given a set $S = \{P_1, \ldots, P_n\}$ of n points in the plane, the *travelling salesman problem* consists of finding the *tour* or cyclic path of minimum length that passes through all the points of S exactly once. This tour is determined in GeoGebra with the command *TravellingSalesman[<List of Points>]*. A possible problem in this regard is

"*Determine the cyclic road network of minimum length joining the following central-European cities: Paris, Brussels, Luxembourg, Amsterdam, Berlin, Prague, Bern, Vaduz, Vienna, Bratislava and Budapest.*"

To solve this problem in GeoGebra, it is enough to replace the command *MinimumSpanningTree* that we used in the shortest path problem exposed in Subsection 2.4 by the command *TravellingSalesman*. The solution obtained is shown in Figure 11.

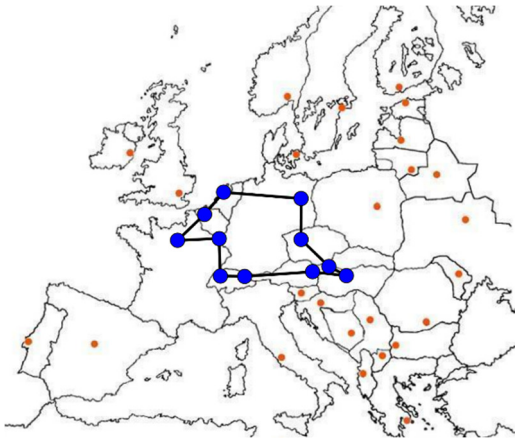

Figure 11: Tour of minimum length.

2.7 Voronoi

The *Voronoi diagram* of a set $S = \{P_1, \ldots, P_n\}$ of n points in the plane constitutes the dual of its Delaunay triangulation. It consists of the subdivision of the plane into n cells, one for each point of S, with the property that a point P lies in the cell corresponding to the point P_i if and only if $d(P,P_i)<d(P,P_j)$, for all $j \neq i$, where d represents the Euclidean distance in the plane. The corresponding Delaunay triangulation can then be obtained by joining those points whose related cells are neighbours. In GeoGebra, the Voronoi diagram can be obtained by using the command *Voronoi[<List of Points>]*.

Voronoi diagrams are applied in a wide range of distinct fields like optimization, location, networks, computer graphics, engineering, biology or chemistry, amongst others. A realistic problem that can be solved in GeoGebra is

> *"The council of Seville wants to construct a new public bicycle rent service point in the center of the city, but it has to be as far as possible of the already existing service points. Determine the best location to construct the new service point."*

Figure 12 shows the distinct steps to be followed in order to solve this problem with GeoGebra: Once the city map is inserted in the graphics view and the existing service points are marked, we determine their Voronoi diagram and their convex hull. The vertices of the Voronoi cells and the intersection points between the Voronoi diagram and the convex hull are the possible candidates to be the location of the new service point. The exact location is the point having maximum distance with the service points of its neighbourhood.

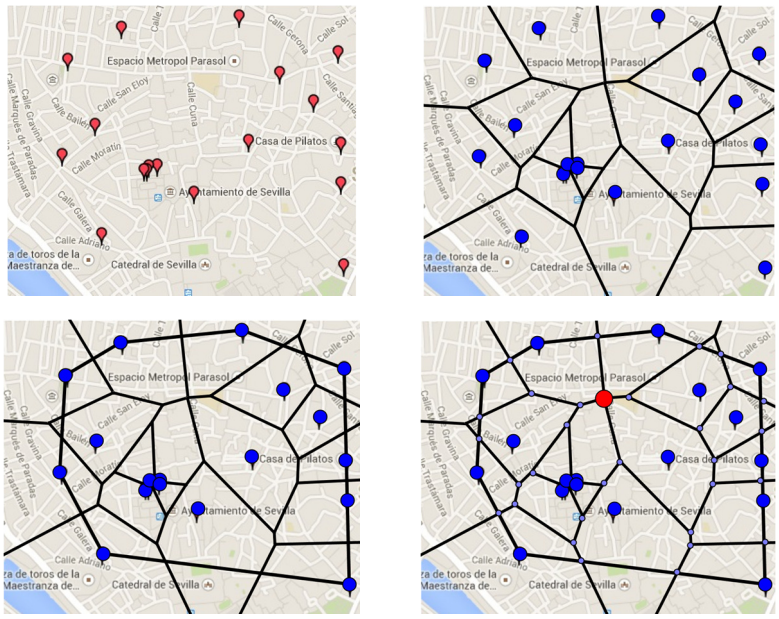

Figure 12: Location of new services by using GeoGebra.

3 Creating new tools in Graph theory

Even if there do not exist specific tools in GeoGebra to analyze results on Graph theory apart from the commands that have been introduced in the previous section, it is possible to create our own tools. In this section we expose a first example in this regard. Specifically, we focus on the explicit construction of random graphs and the definition of distinct commands that make possible the study of basic related concepts. To this end, in a new worksheet of GeoGebra, we create a slider n with integers values defined in the interval $[1, 30]$, which will determine the number of vertices of our graph (the maximum value 30 can be changed if it is required). After that, we define

explicitly these vertices as the n^{th} roots of unity, which are homogeneously distributed in the circumference of center *(0, 0)* and radius *1*. To this end we enter in the input bar the sequence

V=Sequence[cos(2k π / n) + í sin(2k π / n), k, 0, n-1]

The edges of our graph will be determined from its adjacency matrix, that is, from an *n x n* binary symmetric matrix $M = (m_{ij})$ such that $m_{ij} = 1$ if there exists an edge between the vertices *i* and *j*, and *0*, otherwise. The entries of this matrix will be randomly generated according to a normal distribution of a certain probability *p* that we introduce previously as a second slider defined in the interval *[0, 1]* with increment *0.01*. Particularly, the graph will be complete if $p = 1$ and will not have any vertex if $p = 0$. In order to generate our adjacency matrix and because of its symmetry, we create firstly a random list with the entries that are above its main diagonal

L=Sequence[Sequence[RandomBinomial[1,p],j,i+1,n],i,1,n-1]

The adjacency matrix and the edges of our graph are then respectively defined as

M=Sequence[Join[Join[Sequence[Element[Element[L,j],i-j],j,1,i-1],{0}], Sequence[Element[Element[L,i],j],j,1,n-i]],i,1,n]

A=Sequence[Sequence[If[Element[M,i,j]≠0,Segment[Element[V,i], Element[V, j]]], j,i+1,n],i,1,n]

Our construction (see Figure 13) facilitates the definition of basic concepts related to a graph like its *order* (number of vertices), its *size* (number of edges) or the number of triangles formed by its edges:

Order=n
Size=Sum[Sequence[Sum[Element[L,i]],i,1,n-1]]
Triangles=Sum[Sequence[Element[M^3,i,i],i,1,n]] /6

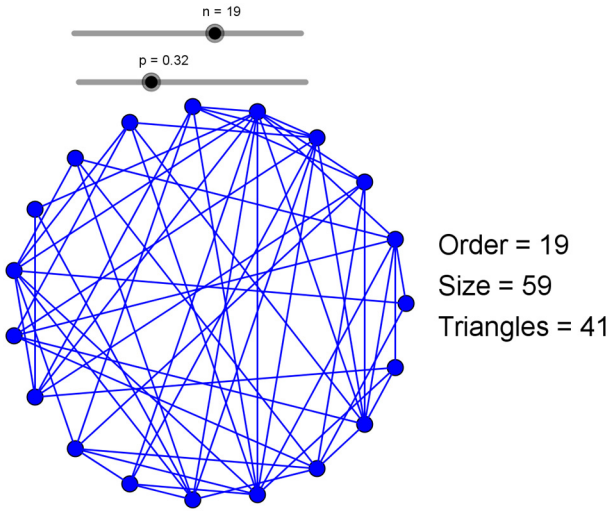

Figure 13: Random graph constructed in GeoGebra.

4 Conclusions

GeoGebra can be an excellent software to introduce distinct concepts and problems on Discrete Mathematics. In this paper we have dealt with the commands that are already implemented in GeoGebra to this end, we have exposed how to apply them to solve distinct realistic problems and we have shown how to generate random graphs and determine some basic properties. All the underlying concepts that are necessary to understand and solve the exposed problems are easy enough even for secondary students, who can be attracted by the simplicity of solving them by using a dynamic software like GeoGebra. In any case, it is only a first and very general approach to the wide range of possibilities that offer this software to deal with the field of Discrete Mathematics. A more comprehensive analysis of each exposed tool and their consequent effect in the teaching-learning process at the secondary and university level would be very interesting to be further developed.

References

[And11] **A. Andersen** – *GraphShop: An interactive software environment for graph theory research and applications*, Master Thesis, Utah State University, 2011.

[Bro97] **G. Brousseau** – *Theory of didactical situations in mathematics*, Mathematics Education Library, vol. 19, Springer Netherlands, 1997.

[DK08] **M. Duckham, L. Kulik, M. Worboys, A. Galton** – *Efficient generation of simple polygons for characterizing the shape of a set of points in the plane*, Pattern recognition, vol. 41 (10): 3224-3236, 2008.

[Eul36] **L. Euler** – *Solutio problematis ad geometriam situs pertinentis*, Commentarii academiae scientiarum Petropolitanae, vol. 8: 128-140, 1736.

[JM04] **M. Jünger, P. Mutzel** (eds.) – *Graph Drawing Software*, Springer-Verlag, New York, 2004.

[Ouv14] **C. Ouvrier-Buffet** – *Discrete Mathematics Teaching and Learning*, in Encyclopedia of Mathematics Education, pp.181-186, 2014.

[RFR97] **J. G. Rosenstein, D. S. Franzblau, F. S. Roberts** (eds.) – *Discrete mathematics in the schools*, DIMACS: Series in Discrete Mathematics & Theoretical Computer Science, vol. 36, American Mathematical Society & NCTM, Providence, 1997.

[Ros99] **K. H. Rosen** – *Handbook of Discrete and Combinatorial Mathematics*, CRC Press, 1999.

[Rod10] **A. Rodríguez Villalobos** – *Grafos: software para la construcción, edición y análisis de grafo*s, Bubok Publishing S.L., España, 2010.

[Vie14] **F. R. Viera Alves** – *Visualizing Bezier´s curves: some applications of Dynamic System Geogebra*, GeoGebra International Journal of Romania, vol, 3 (2): 57-68, 2014.