Project report

Formation: Advanced Mechatronics (M1)

Subject: PROJ803 – Collective Project

Solving geometric constraint using Graph Method

Written by: Ajami Mohammad

Maaz Wael

Project owner: Camillo Hernandez

May,2022

I. Abstract

Geometric constraints are imposed conditions, rules or limiting factors. By applying constraints, you can control the location, size, distance, angle, and other parameters of objects in a drawing, as well as describing the relationship between geometric elements. The constraint solvers are mainly divided into four categories: numerical, rule oriented, symbolic and finally graph constructive solvers. This project states explicitly the constructive and general solvers in the graph method, as well as showing some examples for some methods.

Table of Contents

Table of Figures

1 Introduction

Geometric constraints find and manage the relationships of objects with respect to each other, the distance, length, angle, and radial values of objects. Geometric constraint solving integrated in various fields, as like as molecular modelling, tolerance analysis, geometric theorems and computer Aided Design (CAD). There are four main methods for solving geometric constraints: numerical, symbolic, rule oriented and graph method. The general methods are presented in section 2. Section 3 shows graph method with its constructive and general solvers. Conclusion is stated in section 4. Project management, the methods and tools for gathering and finding information are presented in section 5. Personal outcomes and self-assessment are illustrated in section 6. And finally, the references are introduced in section 7.

2 Background information

This section states brief explanation of the four geometric constraint methods used.

2.1 Numerical method

Numerical method uses equations of 3rd order or more to solve system constraints. Most of the numerical methods have difficulties in dealing with over and under constraint problems. Some systems use a preprocess treatment to decompose the system of equations before launching the numerical solver.[1] Cayley–Menger determinants can also be used to produce simpler algebraic systems . Other methods improve the resolution process by reducing the number of possible options.

2.2 Symbolic method

The restrictions are transformed into a system of equations using symbolic approaches. To find symbolic formulations for the answers, methods such as Gröbner bases or elimination with resultants are used. These procedures take long time to be solved [3].

2.3 Rule based method

Rule-based solvers rely on the predicates formulation[1]. . They give a qualitative analysis of geometric limitations, but the huge quantity of computations required (exhaustive searching and matching) makes them unsuitable for real-world applications [4].

2.4 Graph method

Graph theory gives rise to graph-constructive solvers. They are based on a study of the constraint graph's structure. The graph constructive technique allows for the creation of reliable and efficient algorithms. For the structural analysis of geometric constraint problems, graph theory provides a powerful idea [5].

3 Graph method

This section presents deeply how to construct graph of a structure, as well as showing the guidelines with examples.

3.1 Introduction

Geometric constraint solvers based on graphs have become the most popular. The geometric constraint system is graphed, with vertices representing geometric elements and edges representing stated constraints between them.

We focus on 2D graph-based solutions developed in the area of Computer Aided Design in this paper. Many of these solvers convert the GCS to a graph.[2]

They isolate under, over, and well-constrained sections of the constraint graph using decomposition techniques. A decomposition technique is used to locate small, solvable subgraphs in the well-constrained portion.

The final solution is created by combining the solved subgraphs according to a resolution order determined during the decomposition step. Decomposition/ Decomposition plan (DR-Plan) is the term for this. The major goal of this decomposition is to speed up the resolution process by confining direct algebraic resolution to the smallest possible subsystems (typically ruler and compass solvable problems).

3.2 2D Geometric graph and decomposition

A graph G(V, E), consisting of a vertex set V and an edge set E, can be used to represent any GCS. The geometric elements are represented by the vertices of G, and the constraints between them are represented by the edges. The size of G will be the cardinality of V. The required condition of generic solvability for any GCS is given by Laman's theorem [2]. Its constraint graph (also known as generically isostatic, minimally rigid, or Laman graph by rigidity theory and structural topology communities) must be structurally well-constrained in order to be solvable. Here are some important definitions [6]:

-1st definition: A geometric constraint graph G = (V , E) where |V | = n(n > 1) and |E| = m is structurally well-constrained if and only if m = 2n − 3 and m’ ≤ 2n’ − 3 for any induced subgraph G’ = (V’ , E’), where |V’ | = n’(n’ > 1) and |E’| = m’[2]

- 2nd definition: a constraint graph G = (V, E) contains a structurally overconstrained part if there is an induced subgraph G’ = (V’, E’) having more than 2n’ − 3 edges[2]

-3rd definition: A geometric constraint graph G = (V, E) is structurally under-constrained if it is not over-constrained and the number of edges is less than 2n − 3.

3.3 Requirements for graph-based solver

Graph-based solvers can be compared using many desirable properties as defined below. We note that those properties can be contradictory, the satisfaction of one of them can be to the detriment of another[2].

-Generality: this property characterizes the possible geometric configurations handled by the solver and determines the domain of GCS; a solver is general if it can decompose any possible configuration.

-Optimality: the size of the largest discovered subgraph must be near to minimal, and the number of identified subgraphs must be maximum

-Computational complexity: In practice, the solver must be quick to allow interactive use.

-Handling over and under-constrained parts: the solver must separate under- and over-constrained subgraphs for further correction.

3.4 Constructive solvers

There are many solvers, but they aren't complete; they only solve a subset of ruler and compass constructible geometric configurations, not every well constrained graph. The constraint graph must be biconnected, and the smallest solvable subgraphs must be triangles [2].

By assigning an explicit recognition method to each new type of shape, those algorithms can theoretically be extended to handle shapes other than triangles. Constructive solutions only deal with quadratic equations during the construction phase. As a result, calculating coordinates does not necessitate sophisticated mathematical calculations. Following that, we'll go over three well-known constructive solvers.

3.4.1 Decomposition analysis method

This is accomplished by repeatedly separating the graph into articulation pairs. An articulation pair is made up of two nodes with the same articulation.

The graph is divided into two new graphs as a result of removal. The method is repeated until the graph is simple enough to solve directly, such as using triangles or simple edges. In order to keep the graph stiff, As needed, virtual edges are added. Following that, virtual edges are removed to allow for further decomposition. This algorithm has an O (n^2) complexity[2].

Input: G: the geometric constraint graph;

Output: A plan of resolution.

1. Split the Geometric constraint graph into bi-connected components.
2. Split each bi-connected component into split components, inserting virtual edges as necessary.
3. At any articulation pair with no single edge and exactly one more complex subgraph, remove the virtual edge from the subgraph. Apply the algorithm recursively from stage 2.
4. The algorithm terminates when the graph cannot be split further. ‘

Figure 1 shows Owens' algorithm for solving geometric constraint [2]

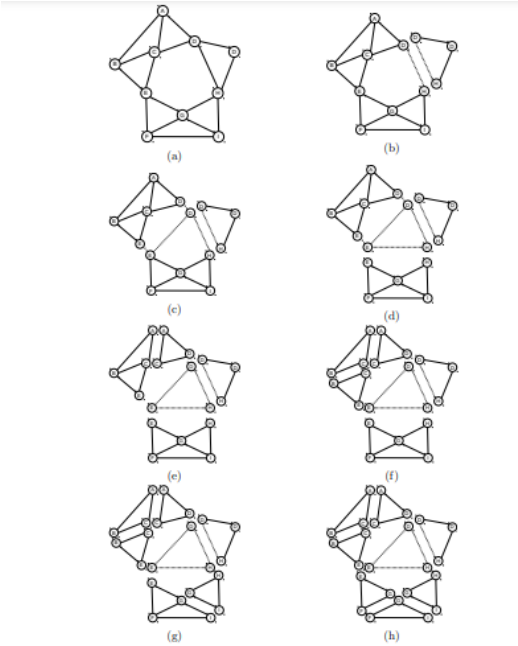

Figure 2 A geometric constraint graph decomposition by Owen's algorithm [2]

3.4.2 Reduction analysis method

Fudos, and Hoffmann proposed an O (n^2) approach for solving Owen's algorithm solves the same class of geometric constraint issues. There are two phases to the algorithm. The first is known as analysis. phase, where Fudos reduction procedure is as follows [5]:

-let the geometric constraint graph to be examined is G(V, E). The initial step is the formation of the initial collection of clusters is the subject of the algorithm.

Input: a geometric constraint graph G;

Output: a plan of resolution.

procedure REDUCE(G)

1. Construct initial set of clusters S_G, each element of S_G is a set of two adjacent vertices of G.
2. Construct the cluster bipartite graph H.
3. Find all triangles in G.
4. Successively rewrite H by replacing a 6 cycle in H by a four node structure as explained in figure 4.3. Record a cluster merging operation for each such rewriting step
5. If H can be rewritten into a final graph that is a star with center a cluster and periphery the vertices of G, then G is solvable; otherwise, it is not solvable.

end procedure

Figure 3 shows reduction analysis algorithm [2]

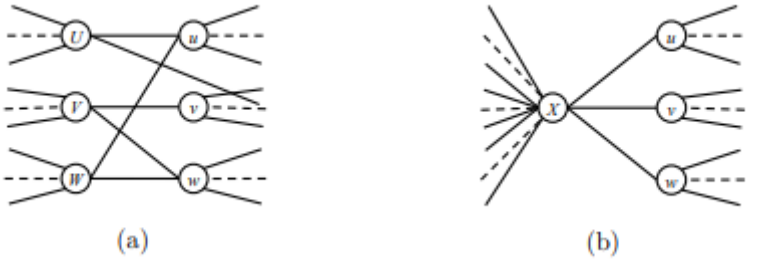

Figure 4 shows reduction of a 6-cycle bipartite graph H that corresponds to cluster merging[2]

3.4.3 Tree decomposition method

This proposed an O (n^2) new constructive solution that falls into the constraint shape recognition category, i.e., it solves the same domain of geometric constraints systems as Fudos' and Owen's solvers[2].

By looking for hinges in fundamental circuits of a given planar embedding of the graph, the technique decomposes biconnected geometric constraint graphs.

The algorithm has two main steps:

-Transform the GCG into a simpler, planar graph

-Compute a planar embedding for the transformed graph where hinges are identified as a set of three vertices shared by two faces

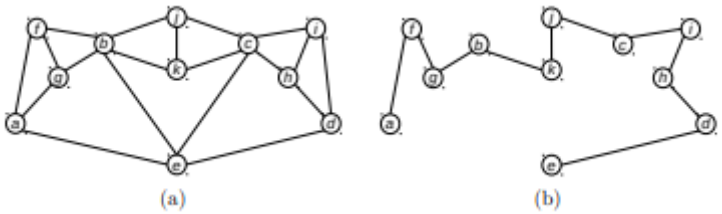

Figure 5 Geometric graph and its tree[2]

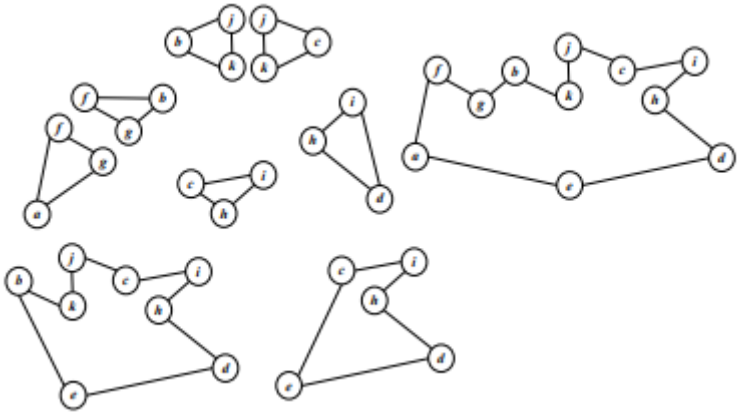

Figure 6 Fundamental circuits of the Graph above[2]

3.5 General Solvers

The domain of the 2D geometric constraint configurations is unrestricted in the class of general solvers, and the smallest subgraph can be any irreducible well-constrained graph. Those approaches are complete, in the sense that they can decompose, solve, and recompose any well-constrained graph. Following that, we give Hoffmann algorithm's for detecting all solvable subgraphs (called also dense subgraphs) ... [2]

3.5.1 Geometric Constraint Bipartite Network

Every geometric element in two dimensions has two degrees of freedom, and all constraints between them are binary, eliminating one degree of freedom. To find irreducible solvable subgraphs, general solvers transform the constraint graph G(V, E, w) of a GCS to a bipartite directed network G∗ = (M, N, s, t, E∗ , w) where[2]:

-s is the source, and t is the sink.

-The vertices in N are the vertices in V

-The vertices in M are the edges in E

-The edges of G∗ are E ∗ = {(e, u),(e, v)|e = (u, v), e ∈ E

-Every directed edge (s, e) from the source node s to the set M is weighted by w€

-Each edge (v, t) from N to the source node t is weighted by w(v)

-Each edge (v, t) from N to the source node t is weighted by w(v)

-The edges from N to M are weighted by ∞

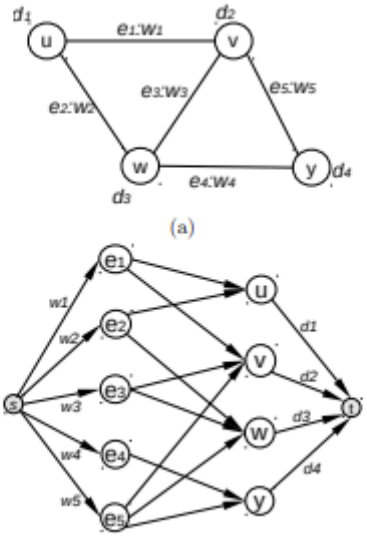

Figure 7 shows constraint graph and its associated network[2]

3.5.2 Detection of solvable subgraphs

Hoffmann and Vermeer proposed an O(n(m + n)) technique for separating solvable subgraphs (also known as dense subgraphs) from a constraint graph G(V, E) with |V | = n and |E | = m[2]. The suggested technique is general in that it can handle scenarios where the DOFs of geometric elements are greater than two and a constraint removes multiple DOFs.

3.5.3 Maximum matching approach

This approach can also distinguish between subgraphs that are under- and over-constrained. It is based on the greatest matching of the bipartite graph G(M, N, E) linked with the GCS being calculated[2]. Its algorithm is attached below.

Input: $G(M, N, E)$: a geometric constraint bipartite graph

Output: R: a decomposition plan

procedure DECOMPOSE(G)

1. Find a perfect matching M of G.
2. Build the directed graph G' from G by replacing each edge (x, y) in M by two arcs xy and yx, and by orienting all other edges from Y to X.
3. Compute the strongly connected components of G'. Each of these strongly connected components is irreducible.
4. To compute the dependency between these irreducible subgraphs, build the reduced graph R from G' by contracting each strongly connected component in a vertex. Each arc of R, say from s_1 to s_2, means: solve subsystem s_2 before $s1$. A compatible total order between subsystems can be obtained by any topological sorting of R.

return R

end procedure

Figure 8 shows the maximum matching algorithm for GCS decomposition

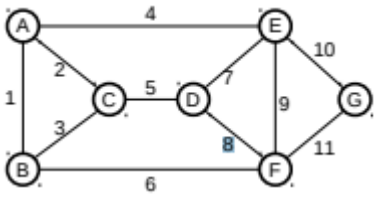

Figure 9 Geometric constraint graph[2]

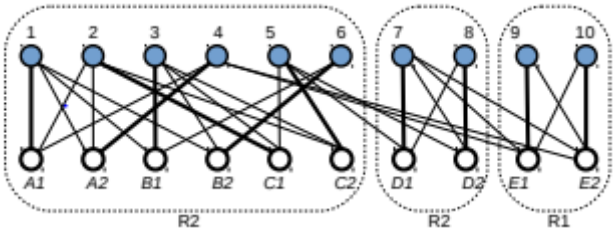

Figure 10 Bipartite graph of Fig9[2]

Figure 11 subgraph resolution order [2]

3.6 Frontier algorithms

The Frontier algorithm can be thought of as an improved version of the Condensed algorithm. The Condensed approach can be used to find minimum subgraphs and their sequential extensions. The simplification stage is the only variation between the two algorithms [9].

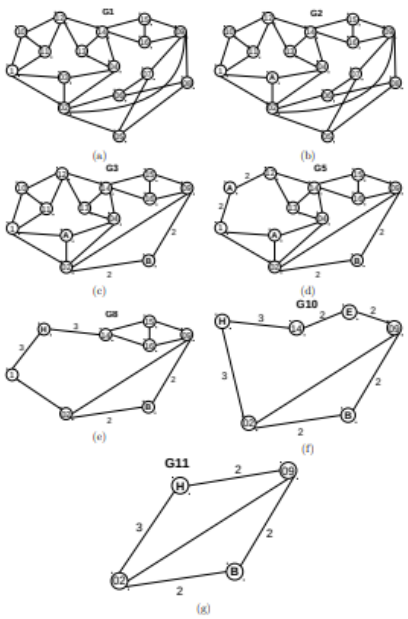

Figure 12 Example of Frontier algorithm[2]

3.7 Recursive Skeletonization Algorithm

This is used based on the following algorithm [10]

Input: G: a geometric constraints graph; $S = \{C_1, C_2, ..., C_n\}$: the set of detected cluster ;

Output: The skeleton graph $G_s(E_s, V_s)$.

procedure SKELETONIZE(G, S)

$V_s \leftarrow \emptyset$

for all $C_i \in S, C_j \in S, i \neq j$ **do**

$V_s \leftarrow V_s \cup (C_i \cap C_j)$;

end for

for each $C_i \in S$ **do**

$X \leftarrow C_i \cap V_s$;

let $X = \{n_1, n_2, ..., n_j\}$, triangulate X by doing:

$E_s \leftarrow E_s \cup \{(n_1, n_2)\}$

for $k \leftarrow 3, j$ **do**

$E_s \leftarrow E_s \cup \{(n_k, n_{k-1}), (n_k, nk - 2)\}$

end for

end for

return G_s

end procedure

Figure 13 recursive algorithm[2]

3.8 Constraints and Graph examples

Here are some constraint systems with their constraint graph

Figure 14a shows a dimensioning approach for a constraint problem. There are seven points and four lines in this puzzle. Nine point–point distances, two line–line angles, and eight point–line incidence limitations are the constraints. Figure 14b depicts the corresponding constraint graph. Distance or angle arguments are used to label the edges. The point–line incidence limitations are represented by the unlabeled edges.

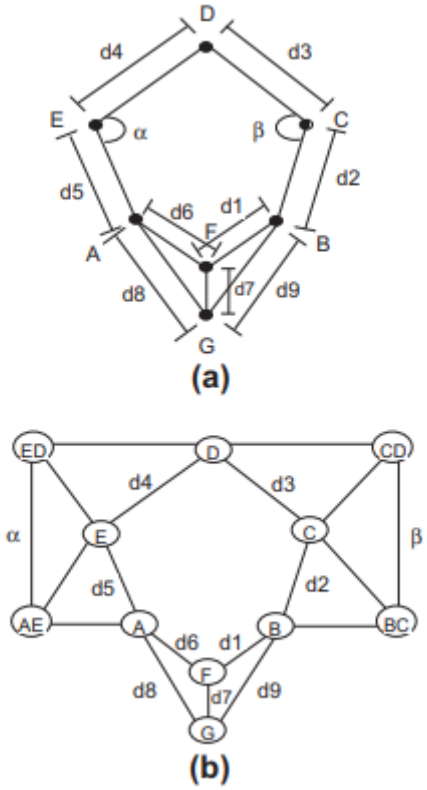

Figure 14 constraint problem with its graph[1]

The figure below shows the generated graph on the right for the problem (on the left), but for under constraint problem

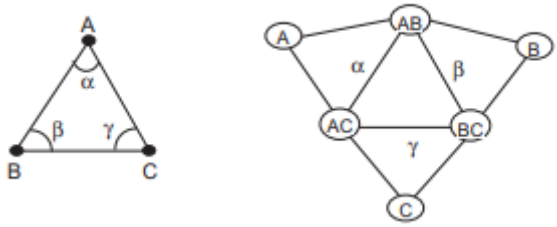

Figure 15 under constraint problem with its graph [1]

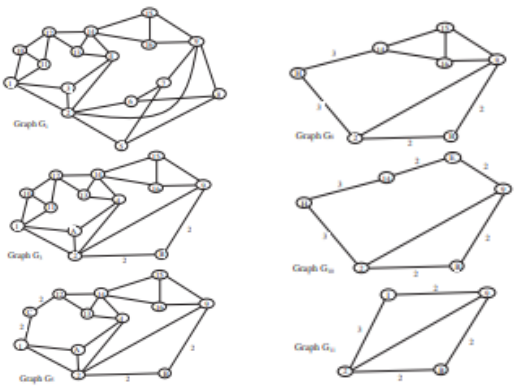

Figure 16 shows graphs generated using MFA[1]

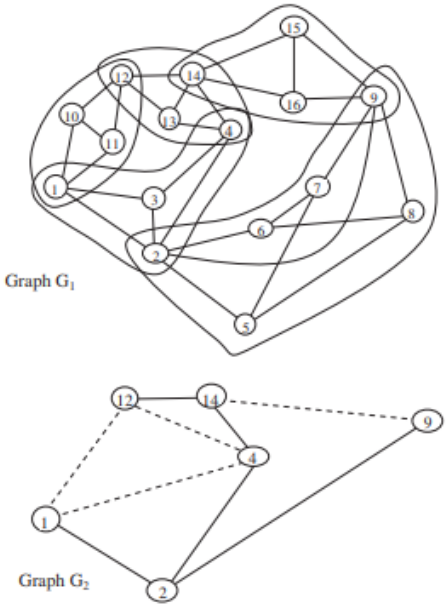

Figure 17 graph generated by SD-R

S-DR employs a graph-based reduction approach to solve systems of geometric constraints. This approach is solvability-preserving and can handle 2D constraint graphs with each vertex having two degrees of freedom and each edge reducing one degree of freedom of each item. Under-constrained settings are likewise handled by S-DR.

4 Conclusion

The goal of this project was to look at geometric constraint solving methods and the creation of a geometric constraint graph generator that can be used to analyse the performance of geometric constraint solvers.

The main objective of the project was achieved, which is solving geometric constraint using graph method. We concluded that graph method is a way of reduction for approaching, reducing time and makes the problem easy to be solved. Also based on our individual projects since they are kindly related, its necessary to state the DOF method [8], it's another method that used for solving geometric constraint It not only provides faster solution and numerical stability, but also the ability to handle redundant constraints rather than some other methods which are not efficient, and can have stability and robustness problems. Several experiments examples were reported in this project.

As for future work, the next step is to move from 2D graph generator to generate 3D graphs. It's also worth looking into the issue where geometric elements have more than two degrees of freedom and a geometric constraint can eliminate multiple degrees of freedom. Also, it's very interesting and useful to implement the studies done on a real life example, as like as solving geometric constraint of a robot.

5 Project management, method, and tools

The project information was collected at the start of the project, and the task was reviewed with the project manager. The first step was to learn fundamental knowledge. This foundational knowledge was acquired mainly from the individual projects.

For further information, Google Scholar was used to find various scientific resources. Also, we contacted many professors and expertise in solving constraint domain, as well as mathematic teachers for understanding deep mathematical equations. To organize the resources, the tool Zotero was used.

To share information and documents we used tools like WhatsApp, mails, and Dropbox. WhatsApp was used for quick sharing of documents because it was fast and could be used on the mobile phone and at the PC. Dropbox was used to have all documents together

Communication was held at face-to-face meetings at university or during leisure time. But also, via online platforms such as WhatsApp, and Zoom. WhatsApp is also good for retrospectively getting an overview of what was being discussed.

To have the task organized and keep an overview on the assignments the enterprise tool Trello was used. The advantages are that both team member had access to it, and it was given a clear overview. The only disadvantage was that for some features Trello-premium would have been better. To overcome this problem, the goals were also entered in a Gantt-chart on Microsoft-Excel. This Gantt-chart, as illustrated in Figure 18, visualised the timetable to be more organized and to estimate the time needed for each task.

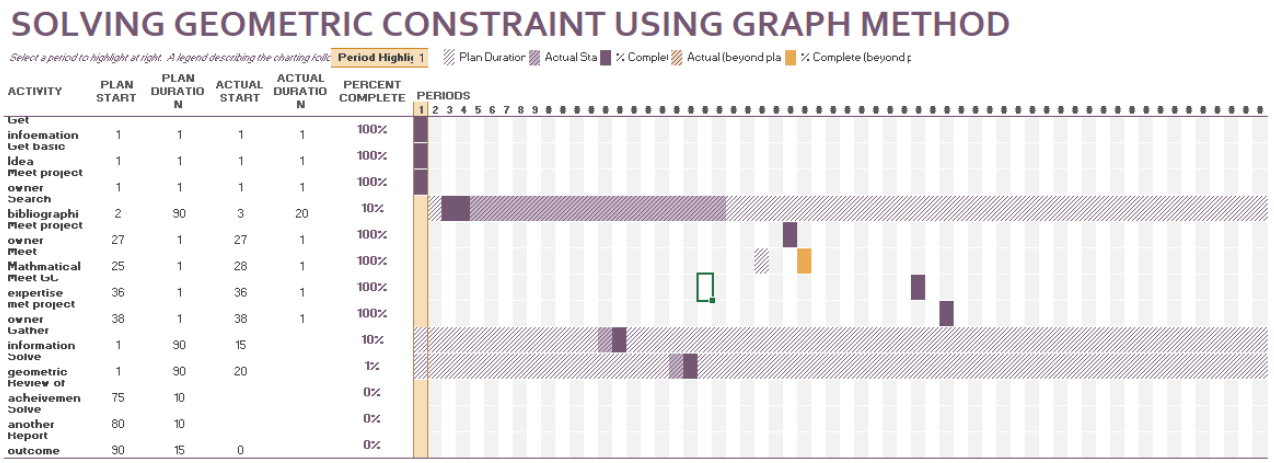

Figure 18: Gantt-chart of the project

To work together on the same assignments Google-Docs and Microsoft-Word were a good option for us. It was possible to work together in real-time and each of us could see the changes of the other.

For a good distribution of responsibilities and tasks we talked to each other what to do in the next period. If the workload is different, we try to manage the organization.

We had some milestones and we split up the workload in packages which were more suitable of one person.

The importance of delegating tasks and scheduling their completion cannot be overstated. Following that, we did:

- Created an organizational plan for the tasks
- Identified the various levels of responsibility
- Organized all the project's papers and procedures in one place
- Created a tracking system

In Figure 19 the distribution and organization of the tasks are shown. Also, the connection between the individual projects is shown.

Figure 19: mind mapping for gathering and planning information

6 Personal outcome and self-assessment

In this section the personal outcome of each team member is described.

6.1 Maaz Wael

This collective project not only made me share information even traditions with my team member which he is from electronic background, but also taught me the following:

- Complicated work should be broken down into sections and steps.
- Time management and planning
- Discussion and explanation are helpful to clarify comprehension
- Improve my communication abilities
- Attempt to solve more difficult issues
- Share a variety of viewpoints
- Hold each other responsible
- Encouragement to try new things
- Citing references
- Usage of project and time managing tools
- Be able to work with any student regarding his background

6.2 Ajami Mohammad

This project has allowed me to develop in many areas. Not only the understanding of how to solve geometric constraints, but also be able to think in a larger scale.

The project also contributed a lot to the development of management skills. Different tools were used that can help both in professional and private life.

Through the project, different scientific approaches that were learnt. Finding solutions to new challenges was promoted.

Because my team member had a different technical background, problems could be analysed from different angles and new solutions could be found.

7 Bibliography

[1] S. Ait-Aoudia and S. Foufou, 'A 2D geometric constraint solver using a graph reduction method', *Adv. Eng. Softw.*, vol. 41, no. 10, pp. 1187–1194, Oct. 2010, doi: 10.1016/j.advengsoft.2010.07.008.

[2] A. Moussaoui, 'Geometric Constraint Solver', PhD Thesis, Ecole nationale Supérieure d'Informatique (ex INI), Alger, 2016.

[3] R. Joan-Arinyo and A. Soto, 'A correct rule-based geometric constraint solver', *Comput. Graph.*, vol. 21, no. 5, pp. 599–609, Sep. 1997, doi: 10.1016/S0097-8493(97)00038-1.

[4] 'gao.pdf'. Accessed: May 17, 2022. [Online]. Available: http://www.mmrc.iss.ac.cn/pub/mm15.pdf/gao.pdf

[5] R. Joan-Arinyo and A. Soto, 'A correct rule-based geometric constraint solver', *Comput. Graph.*, vol. 21, no. 5, pp. 599–609, Sep. 1997, doi: 10.1016/S0097-8493(97)00038-1.

[6] I. Fudos and C. Hoffmann, 'A Graph-Constructive Approach to Solving Systems of Geometric Constraints', *ACM Trans. Graph.*, vol. 16, Mar. 2004, doi: 10.1145/248210.248223.

[7] C. Jermann, G. Trombettoni, B. Neveu, and P. Mathis, 'Decomposition of Geometric Constraint Systems: a Survey', *Int. J. Comput. Geom. Appl.*, vol. 16, no. 5–6, pp. 379–414, 2006, doi: 10.1142/S0218195906002105.

[8] G. A. Kramer, 'Using degrees of freedom analysis to solve geometric constraint systems', in *Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications - SMA '91*, Austin, Texas, United States, 1991, pp. 371–378. doi: 10.1145/112515.112566.

[9] J. Oung, M. Sitharam, B. Moro, and A. Arbree, 'FRONTIER: fully enabling geometric constraints for feature-based modeling and assembly', in *Proceedings of the sixth ACM symposium on Solid modeling and applications - SMA '01*, Ann Arbor, Michigan, United States, 2001, pp. 307–308. doi: 10.1145/376957.376995.

[10] V. Minden, K. L. Ho, A. Damle, and L. Ying, 'A Recursive Skeletonization Factorization Based on Strong Admissibility', *Multiscale Model. Simul.*, vol. 15, no. 2, pp. 768–796, Jan. 2017, doi: 10.1137/16M1095949.