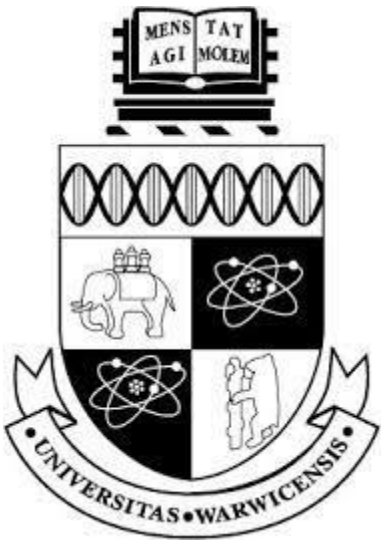
New Developments in Graph Editing Algorithms

Progress Report

Sam Moon [2001779]



Contents

1 Introduction 3
1.1 Abstract 3
1.2 Suitability of Project 3
2 Background Research 3
3 Development Progress 4
4 Challenges faced during the development 5
4.1 Updated Objectives 5
4.2 Methods 5
4.3 Timetable 6
5 Risk Assessment 8
6 Legal, Social, Professional, and Ethical Issues 8
7 References 9
8 Appendix – Project Specification 9

1 Introduction

1.1 Abstract

An investigation into New Developments in Graph Editing Algorithms, studying various algorithms relating to the vertex cover problem, including the König-Egerváry class of graphs, that is, graphs with the property that the maximum matching has equal size to the graph's minimum vertex cover, as well as parameterised vertex cover problems and the use of linear programming techniques in efficient algorithms.

1.2 Suitability of Project

This project aims to provide an understanding of the practicality and real-world performance of currently theoretical algorithms, which, while proven to be correct and to have polynomial runtime, have not been concretely tested. The project builds upon content learned previously at university, most notably Algorithmic Graph Theory, and so is a suitable difficulty for a third-year project.

2 Background Research

As was stated in the Project Specification, the objectives of the project were split into a total of 5 algorithms and the aim was to fully research, implement and analyse/test each before continuing to the next. Following this pattern, the research done so far is entirely in pursuit of the first object algorithm: Testing for a Unique Vertex Cover in König-Egerváry graphs.

The majority of research done into this was done in [1] as it was in this paper the algorithm was proposed and justified, however, there was also some exploration of other sources specifically involving the famous Edmonds Blossom Algorithm for Maximum Matchings [2] and Computing a Minimum Vertex Cover saturated by a maximum matching in König-Egerváry graphs [3]. These two algorithms were required to implement the main algorithm as we require a partition of the graph into a minimum vertex cover and an independent set such that the vertex cover is completely saturated by a maximum matching exclusively spanning across the cut.

A dataset [4] of example graphs was also found and is planned to be used in the testing phase for this algorithm, and potentially other algorithms later. This data set includes a range of both random general graphs and a range of bipartite graphs. These sets will be combined in a way such that a set of König-Egerváry graphs is produced using the characterization of this class: a König-Egerváry admits a partition of vertices (A, B) with A minimum vertex cover and max matching M as the subgraph induced by A could be any general graph, B the corresponding independent set and M spans across this cut. Using this we can take a general graph from the dataset and embed it into one side of a bipartite graph of the same number of vertices so that we obtain a König-Egerváry with clear partition and the edges of the bipartite graph as the max matching.

3 Development Progress

So far, the development of the first algorithm: Testing for a Unique Vertex Cover in König-Egerváry graphs [1], is almost complete. The majority of the required routines are fully implemented:

- Computation of maximum matching M of input graph G using Edmonds Blossom Algorithm [2].
- Naïve algorithm for the computation of (A, B) partition of vertices such that A is a minimum vertex cover of the input graph saturated precisely by M. *This will be replaced.*
- Construction of $G^{A->B}_{bip, M}$, the bipartite graph on the input graphs vertex set with matching edges oriented from A to B, other edges across the cut oriented from B to A, and all other edges deleted.
- Calculation of R(U), the set of vertices reachable from a vertex in U in $G^{A->B}_{bip, M}$, where U is the set of vertices not saturated by M in G. Computation of vertex set X, the intersection of R(U) and A.
- Computation of G/U graph, the subgraph of G induced by deleting all vertices in R(U).
- Computation of a minimum vertex cover containing a given subset, Z, of the vertices in G or returning that one does not exist. This uses M-alternating trees in G/U rooted at vertices in R-R(U), where R is the intersection of Z and B.
- Routine which makes use of all the above to decide whether an input König-Egerváry graph has a unique vertex cover by checking if we can swap any of the matched vertices from A with its partner from B and conserve the vertex cover property.
- Simple file reading routine to import test example graphs from the dataset [4] and instantiate a corresponding Graph object that the routines can use.

Remaining subroutines to implement for this entire section:

- More efficient computation of (A, B) partition of vertices such that A is a minimum vertex cover of the input graph saturated precisely by M [3].
- Generation of the dataset of König-Egerváry graphs as was described in the previous section using the general and bipartite graphs from the dataset [4].

4 Challenges faced during the development

At the time of writing the Project Specification, there were many unknowns, including many specifics about the algorithm, how long implementation of each of the subroutines would take, and external factors affecting the rate at which the project would progress. Therefore, there will have to be many changes made to the original objectives and timetable which were originally proposed.

The largest change is that 5 objective algorithms to fully research, implement and analyse was found to be too much by myself and my supervisor and so the more realistic goal of completing 3 of the 5 will be set, however, the remaining two will be kept as a stretch goal as it is likely more time will be dedicated to this project next term.

4.1 Updated Objectives

- (4.2.1a) An algorithm for testing König-Egerváry graphs for Unique Minimum Vertex Covers [1]
- (4.2.1b) An FPT algorithm for Local Search Vertex Cover problem on planar graphs [5]
- (4.2.1c) An algorithm for König-Egerváry vertex deletion problem [6]

Stretch Objectives:

- (4.2.1d) The "Simple Algorithm" for Vertex Cover Above LP [6]
- (4.2.1e) The "Improved Algorithm" for Vertex Cover Above LP [6]

In addition, for each algorithm:

- The program should be efficient enough to run smoothly, assuming the theorised algorithm in question is efficient
- The project will produce a summary of the various algorithms considered
- And a brief review of each, evaluating its strengths and weaknesses

4.2 Methods

After research was completed into the Unique Minimum Vertex Cover algorithm [1] the decision was made to implement bespoke data structures, classes and methods for the graphs and matchings so, while the original plan was to use graphing libraries BGL (Boost Graph Library) [7] and SNAP (Stanford Network Analysis Platform) [8], they were not used at all. This decision was due to the realisation that very little work was required to set up some simple `Graph`, `Edge` and `EdgeSet` classes, and the vertex cover could simply be represented by a standard C++ `list<int>` data structure.

> **`Graph`** – Maintains an array of `list<int>` corresponding to the list of neighbours of each vertex in the graph. This is the most common data structure used for graphs as it is easy to implement and is sufficiently efficient for both exploring and editing the graph. Some basic methods include the insertion and removal of edges, the removal of all edges attached to a given vertex and a copy constructor. Limitations of the class include a fixed vertex set after instantiation

due to the use of an array for the adjacency lists, in some contexts, this may be insufficient however for this algorithm it was never required, and just removing all the edges incident on some vertices worked fine.

`Edge` – Very simple class which stores an array `int[2]` corresponding to the two endpoints of the edge in a graph.

`EdgeSet` – Maintains a `list<Edge>` data structure with some methods to add and remove edges from the set, to augment the set by another set (that is, take the symmetric difference of two sets of edges. This is used in the Blossom Algorithm for Maximum Matching), and a method to return the set of vertices covered by the EdgeSet.

In addition to these classes basic set operations were implemented for general use across all subroutines: union, intersection, and set difference.

4.3 Timetable

Below, in Figure 1, is the original timetable that was proposed in the Project Specification.

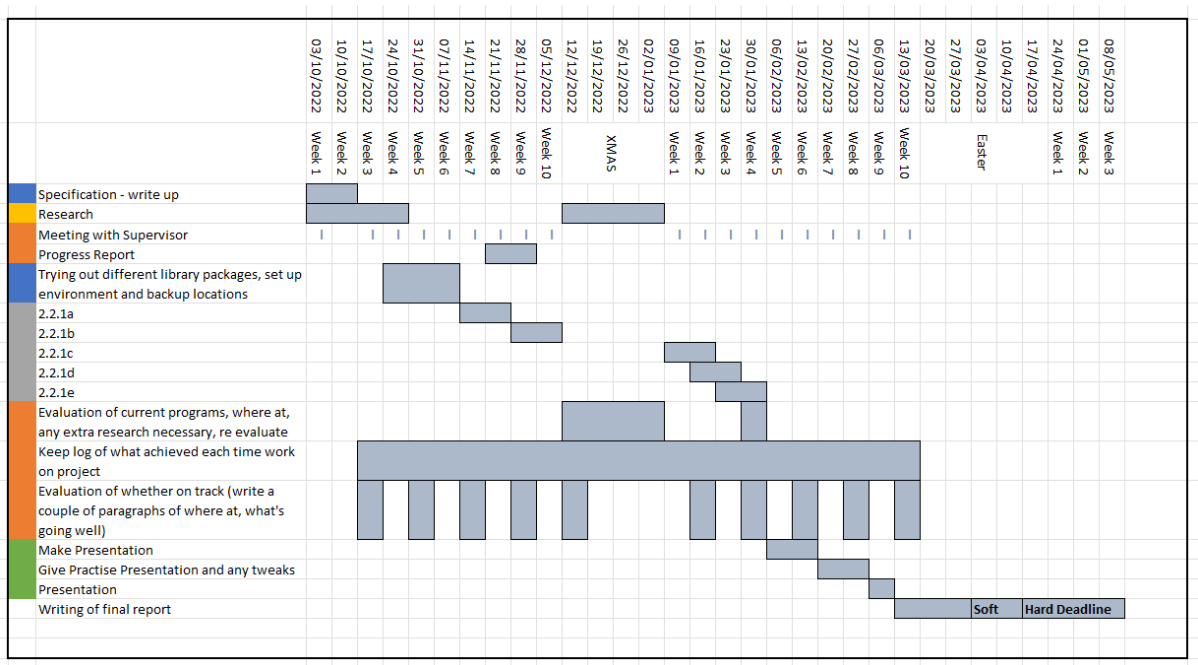


Figure 1: Gantt Chart of the project's original time management plan

As mentioned previously the development of the Unique Minimum Vertex Cover algorithm [1] has taken longer than this plan allowed for and so an updated plan is proposed in Figure 2.

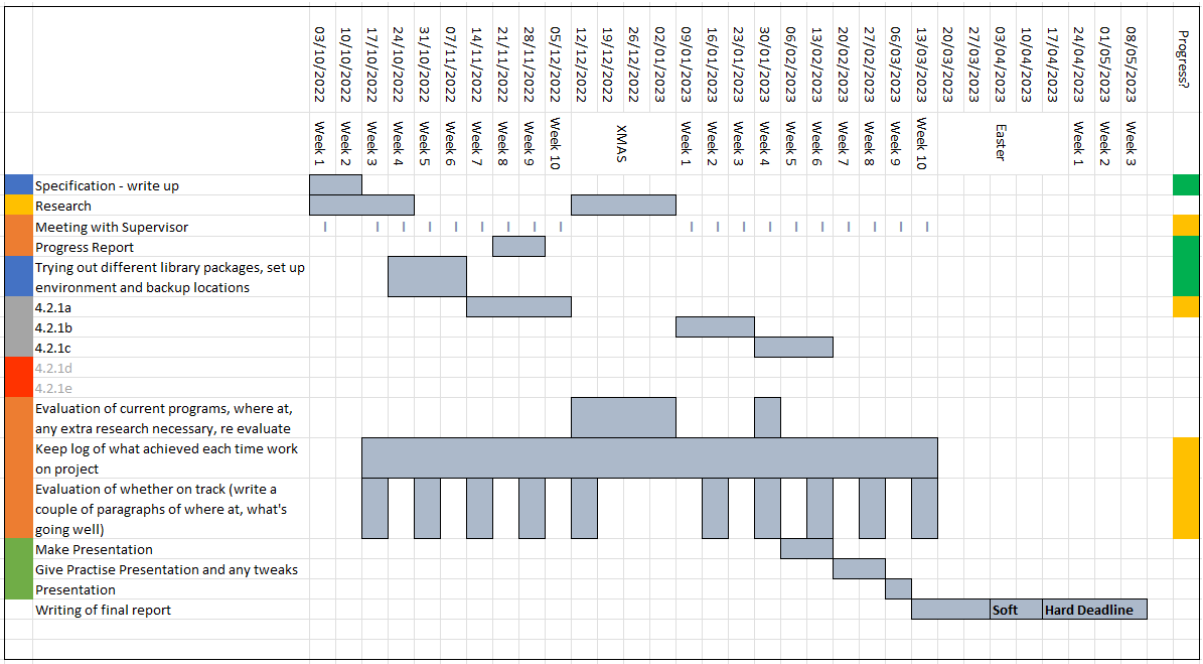


Figure 2: Gantt Chart of the project's updated time management plan

It is expected that next term more time will be available to be dedicated to the project and that some parts such as the bespoke classes and some relevant methods can be easily reused. Therefore, it is expected that the next two parts of the project will be completed faster than the first, taking just 3 weeks instead of 4.

Next term the weekly time management will also be different to the Term 1 timetable from the specification. Below is the updated timetable where the hours dedicated to the project are marked for a total of 11 hours a week.

Term 2	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm
Monday								
Tuesday		x	x			x		
Wednesday		x	x			x		
Thursday						x		
Friday								
Saturday			x			x		
Sunday			x			x		

5 Risk Assessment

As was mentioned in the specification, the main risks associated with this project are that the project could be lost in some way, most likely laptop malfunction. To avoid a catastrophe in the case that something like this happened, the solution files have been regularly backed up using Git and onto OneDrive to provide plenty of redundancy. These practises will continue for the rest of the project.

To avoid a problem occurring in case of illness, work will be completed with plenty of spare time before the deadline.

6 Legal, Social, Professional, and Ethical Issues

This project does not entail any legal, social, professional, or ethical issues as the product will not involve other people or be released into the public. The project will also use only free licence and/or open-source software to avoid legal issues.

7 References

[1] V. Raman et al., A characterization of König-Egerváry graphs with extendable vertex covers, Inf Process. Lett. (2020)

[2] Edmonds, J. (1965). Paths, Trees, and Flowers. Canadian Journal of Mathematics, 17, 449-467.

[3] Fănică Gavril, Testing for equality between maximum matching and minimum node covering, Information Processing Letters, Volume 6, Issue 6, 1977, Pages 199-202, ISSN 0020-0190,

[4] Kartelj, Aleksandar (2019): General graph datasets. 4TU.ResearchData. Dataset.

[5] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. A. Rosamond, S. Saurabh, and Y. Villanger. Local search: Is brute-force avoidable? Journal of Computer and System Sciences, (2012)

[6] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. ACM Transactions on Algorithms, (2014)

[7] Boost Graph Library, Accessed 10th October 2022

[8] SNAP Graph Library, Jure Leskovec, Accessed 11th October 2022

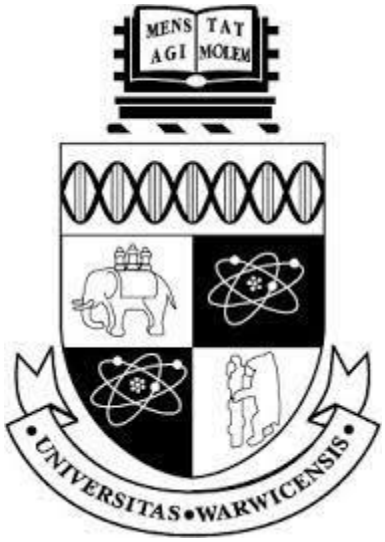
8 Appendix – Project Specification

The remainder of the document is the Project Specification.

New Developments in Graph Editing Algorithms

Project Specification

Sam Moon [2001779]



Contents

1 Abstract 12
2 Project Problem 12
2.1 Background 12
2.2 Objective 12
3 Methods 13
3.1 Research 13
3.2 Development 13
3.3 Testing 13
4 Timetable 14
5 Risk Assessment 15
6 Resources to be used 15
7 Legal, Social, Professional, and Ethical Issues 16
8 References 16
9 Appendix A 17

1 Abstract

An investigation into New Developments in Graph Editing Algorithms, studying various algorithms relating to the vertex cover problem, including the König-Egerváry class of graphs, parameterised vertex cover problems and the use of linear programming techniques in efficient algorithms.

2 Project Problem

2.1 Background

In recent years there have been many developments in our understanding of the relationship between Matchings and Vertex Covers; The well-known König-Egerváry Theorem (or simply König's Theorem) [1] asserts that the size of a maximum matching equals the size of a minimum vertex cover in bipartite graphs, and we denote the class of all graphs with this property as König-Egerváry graphs (of which bipartite graphs are a proper subset). More recently there have been further developments including a complete characterisation of such graphs which have a unique minimum vertex cover [2], this characterisation can be efficiently algorithmically checked. It is also known that the Vertex Cover problem is NP-Complete in general graphs however we can more efficiently solve parameterised versions of the problem (known as Fixed-Parameter Tractable or FPT problems) [3]. There have also been advances in using Linear Programming techniques to produce faster parameterised algorithms [4].

2.2 Objective

This project will attempt to produce an implementation of some previously theoretical algorithms in order to showcase them and assess their performance in a practical scenario. More specifically, the following algorithm ideas are proposed in their respective papers:

- (2.2.1a) A complete characterisation of König-Egerváry graphs with unique vertex covers [2]
- (2.2.1b) An FPT algorithm for Local Search Vertex Cover problem on planar graphs [5]
- (2.2.1c) The "Simple Algorithm" for Vertex Cover Above LP [4]
- (2.2.1d) The "Improved Algorithm" for Vertex Cover Above LP [4]
- (2.2.1e) An algorithm for König-Egerváry vertex deletion problem [4]

In addition:

- The program should be efficient enough to run smoothly, assuming the theorised algorithm in question is efficient
- The project will produce a summary of the various algorithms considered
- And a brief review of each, evaluating its strengths and weaknesses

3 Methods

3.1 Research

Before developing anything for the project, it will be necessary to complete significant research, so that a thorough understanding of the topics within the project is gained. Several weeks of research will be conducted, through Google Scholar, books within the department and University library, various academic texts, and online sources. This will ensure that the project is thoroughly considered and minimise the amount of reading to be done later in the project when the focus will be on the project and portfolio itself.

All research notes will be stored online, where they can be accessed and added easily, complete with references – this means if further clarification is required on an area, the original source can be returned to.

3.2 Development

In the development stage of the project, C++ will be used, along with various libraries (SNAP, BGL) to implement different algorithm ideas proposed by the papers above. Each algorithm will be fully implemented and tested before proceeding to the next, this will ensure that if the project is unable to fulfill all objectives set out above then there still exists an implementation and full review of some of the objectives, rather than potentially the implementations completed with no report written about any of them.

3.3 Testing

The various algorithms will be tested rigorously with various graphs, to ensure that it works and is functional. This will also show where some algorithms have weaknesses and could be improved, as they may be optimal for some specific graphs, but be inefficient in other cases. As more research is done into the algorithms, a more concrete test table will be produced for each covering a range of cases including typical and extreme inputs, such as very large graphs or very dense graphs.

In order to test the performance of the implemented algorithms, GNU Profiler, an open-source tool for performance profiling and optimisation, will be used to give accurate and accessible data and to ensure that all tests are carried out in a fair environment so that they are comparable.

4 Timetable

The project timetable will be displayed as a Gantt Chart (Appendix A). The following table shows the hours per week which will be dedicated to Project Work. This approximates the amount of work per week necessary. The progress of the Project will be regularly monitored, both individually, and via weekly meetings with the project supervisor. In the weekly meetings, the progress and direction of the project will be discussed, so as to ensure that it is on track, and any necessary adjustments made, be that to the timescale or the project itself. Therefore, the provisional timetable is subject to change, with further understanding of the processes required, the limitations of the project, and any further research that may be necessary.

In Term 1, less time will be dedicated to the project, due to module selection. Consequently, progress during Term 1 will be less than what was initially hoped to achieve, so it will be necessary to set realistic and achievable goals. Considerable time will be spent during the Christmas break to work on the project, to ensure sufficient progress is made. As there is no specific software or material that is only on Campus or available during Term time requirements, this will be possible and achievable. Term 2 will also have considerably more time dedicated to the project.

Each task will be inserted into Trello.com, a web-based list-making application, which enables you to make lists, and sub-lists, and move items between these. This will be a necessary visual display of the tasks: what is done, what is left, and what is in progress. It will also aid in time management and prioritisation.

Term 1	10	11	12	1	2	3	4	5
Monday			x	x		x	x	
Tuesday	x							
Wednesday						x	x	x
Thursday				x	x	x		
Friday								
Saturday								
Sunday								

5 Risk Assessment

The main risks will be loss of data or inability to work due to a laptop malfunction. To mitigate this, backups of the data will be regularly taken to ensure that any possible issues are minimised – Git will be used for this. This should ensure that there is little work lost due to hardware issues. There are possibilities of theft, fire, and vandalism, so there will be a backup completed somewhere away from where the project is being completed (namely the personal residences of the student completing the project). This backup will be on a computer in the CS department.

There is also the risk of illness, which is incontrollable, so appropriate measures will be in place should this occur, such as giving earlier deadlines than necessary.

6 Resources to be used

The resources that are going to be used are all freely available to use, with prior experience in using some of the following.

- **Git** will be used for version control, to store a backup of the project, and allow it to be worked on from multiple computers, be that at home or on the DCS machines on campus.
- **C++** will be used to implement the algorithms mentioned in the Objectives.
- Two libraries for graph data structures and standard algorithms may be used, depending on the requirements for each algorithm:
    - The **BGL** (Boost Graph Library) may also be used for implementing the relevant algorithms. [8]
    - The **SNAP (**Stanford Network Analysis Platform) will also be used in the project for implementing the algorithms. [9]
- **gProfiler**, an open-source performance profiling, and optimisation tool, will be used in the testing phase for each algorithm. [10s]
- **Computers in the DCS** will be used during the project so that the project can be worked on during breaks between lectures, and spare time on campus. Backups will also be completed on a Computer in DCS to mitigate any setbacks due to any hardware issues on personal devices.
- **Trello.com** will be used to coordinate and work through the requirements of each task in a visual way, monitor progress made, and decompose each section into manageable chunks that are achievable, attainable, and measurable. This ensures that the progress is maintained, and the workload manageable.

7 Legal, Social, Professional, and Ethical Issues

This project does not entail any legal, social, professional, or ethical issues as the product will not involve other people or be released into the public. The project will also use only free licence and/or open-source software to avoid legal issues.

8 References

[1] D. König, "Graphs and matrices" Mat. Lapok, (1931)

[2] V. Raman et al., A characterization of König-Egerváry graphs with extendable vertex covers, Inf Process. Lett. (2020)

[3] Christian Komusiewicz, André Nichterlein, and Rolf Niedermeier, Parameterized Algorithmics for Graph Modification Problems: On Interactions with Heuristics (2016)

[4] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. ACM Transactions on Algorithms, (2014)

[5] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. A. Rosamond, S. Saurabh, and Y. Villanger. Local search: Is brute-force avoidable? Journal of Computer and System Sciences, (2012)

[6] A. Chandra Babu, P.V. Ramakrishnan. New Proofs of Konig-Egervary Theorem and Maximal Flow - Minimal Cut Capacity Theorem using O.R Techniques, (1991)

[7] Vadim E. Levit, Eugen Mandrescu. A Characterisation of Konig-Egervary Graphs Using a Common Property of All Maximum Matchings, (2011)

[8] Boost Graph Library, Accessed 10th October 2022

[9] SNAP Graph Library, Jure Leskovec, Accessed 11th October 2022

[10] gProfiler, an open-source, continuous profiler for production across any environment, at any scale. Accessed 10th October 2022

9 Appendix A

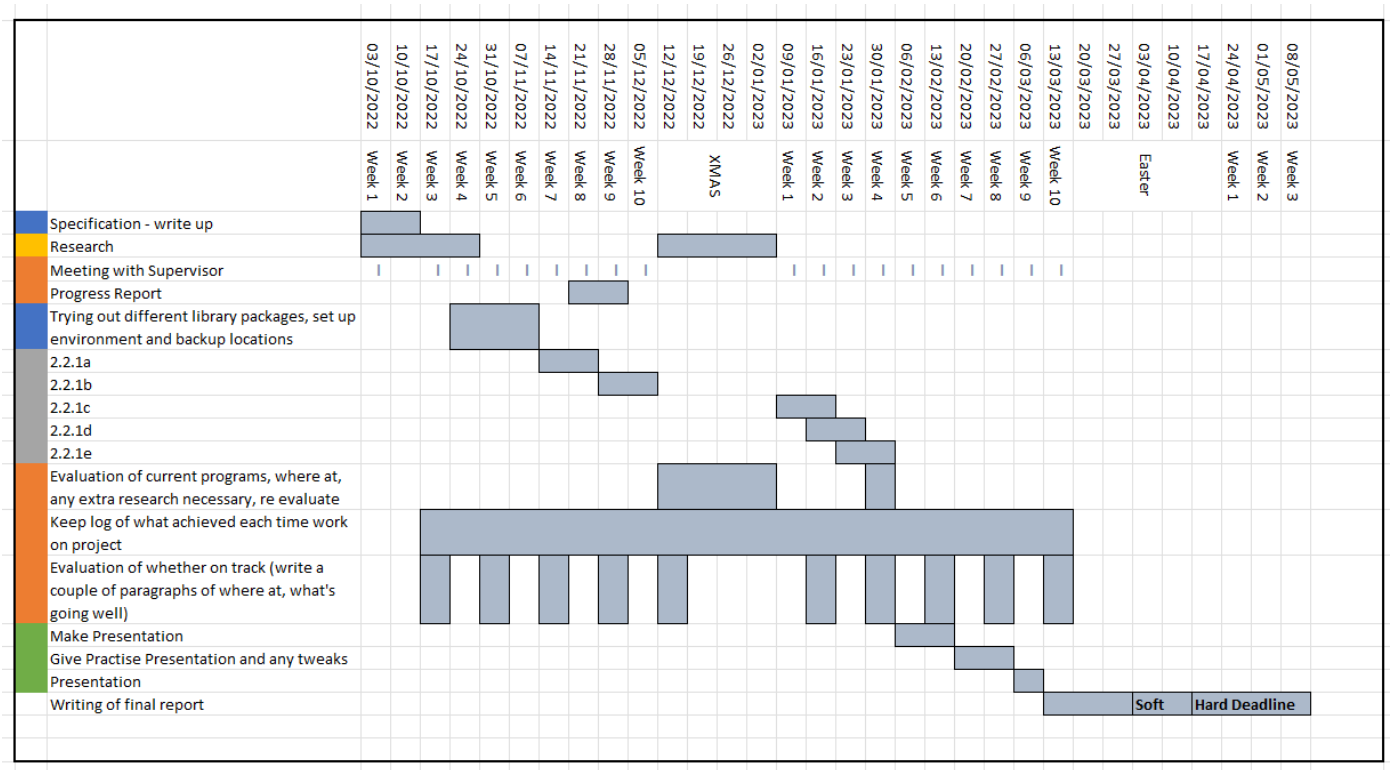


Figure 3: A Gantt Chart of the project's time management