

Computer Science A Level Project.

THE TRANSPORTATION PROBLEM

JAMES PEDLEY

Analysis

Introduction:

Transporting goods from one place to another has been essential for most of human existence and in our modern world it has never been more important. Whether it is getting groceries delivered to your front door or sending a parcel to someone in another country, we rely on many services to carry out what have become these routine tasks. However, each individual delivery costs money, albeit not much, but on the scale of a large company the pennies add up and can potentially influence decisions. Companies need to find ways of reducing the cost of carrying goods from one place to another.

For individual drivers ferrying goods from one place to another can be very time consuming, especially if a longer route than needed is traversed. Since Journey time affects the final cost of delivery through petrol costs, man hours, and productivity. There is a need to reduce the travelling time. Distance travelled is a major factor that dictates the end cost and hence needs reducing in order to save money.

Problem Context.

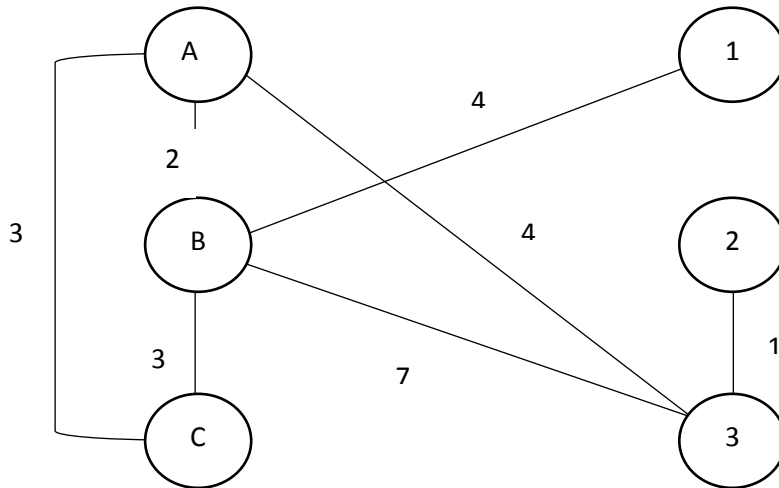
The system will be used by teachers in both the maths and computer science department at More House school in Farnham. The school teaches GCSE and A-level qualifications but the system created for this project will primarily be used by the students and staff doing A-level.

Identification of the problem.

1. Further Mathematics is a subject that is not very popular at More House, with between 1 and 5 students taking it each year. Due to this low number the teacher does not have as many teaching resources used in lessons. In particular the further maths decision maths modules don't have many resources and the textbook is heavily relied upon for questions and examples. From my experience I found that the transportation algorithms (from D2) and Dijkstra's algorithm (from D1) were two of the more difficult topics taking extra time to understand. It would have been more interesting and useful to see the application of such algorithms in the real world, for example in a business context. I think that it would be more fun if current students taking the further maths course could see how powerful the two methods are first hand when being run on a computer, rather than being told by a textbook. Furthermore, the students could work through the algorithms for the problems they come up with themselves and see if they get the same method as the computer program. Since the system generates a report as it performs the computation students will be able to see clearly where they have gone wrong and be able to understand how to continue with the method to obtain the correct answer. In addition, visually displaying the network that the user has inputted emphasises how adjacency matrices are used to represent graphs. Finally, the further maths students can learn how the transportation algorithm handle different supply and demand values, because previous network structures that have been entered into the system (and saved) can be loaded for different supply and demand values to be entered. This will be interesting as students can quickly see how the different values effect the overall result without having to perform the algorithms by hand which is very time consuming. My further maths teacher, Miss Hudson will be the end user or client for this project. [Lastly Miss Hudson would like the system to be as business oriented as possible to show one of the many applications of these algorithms.](#)

Example Problem (shortest route between nodes)

The following is an example demonstrating the shortest route between nodes. If goods need to be transported from warehouses A, B and C to demand points 1, 2 and 3, the shortest path between each warehouse needs to be computed. The network drawn below shows possible routes between each warehouse and demand point, where a warehouse and demand point is a labelled node.



Let's say the cost per unit length on the network is £0.50. Finding the shortest path between each supply and demand point will reduce the transportation costs.

By inspection we can come up with an adjacency matrix:

	1	2	3
A	6	5	4
B	4	7	6
C	7	8	7

By multiplying each distance by the cost per unit length we obtain our values for transporting one unit of our item from one node to another. Costs are shown in the following matrix: -

	1	2	3
A	£3.00	£2.50	£2.00
B	£2.00	£3.50	£3.00
C	£3.50	£4.00	£3.50

For simplicity, no additional costs such as surcharges for weight or size, were considered in this example.

It is easy to see how the cost of transportation can quickly grow as distance travelled and the complexity of the network increase.

An additional factor that needs addressing to reduce overall cost is balancing the number of items that need transporting from each supply point, to each demand point, for example warehouses to shops. This cost will be influenced by the price per unit \times distance that delivery people have to travel (This assumes all the items are of the same type). For example, three warehouses which house 1, 2 and 4 units of stock respectively and are 5, 3 and 2 miles away from a supply point respectively which requires 5 units of stock to be delivered. It will be most cost effective to transport 4 units from the third warehouse and 1 unit from the second.

It is important therefore to develop a system that reduces the overall cost of transporting goods from one place to another by considering the network on which the supply and demand points sit.

A History of Shortest Path Problems:

Edsger Wybe Dijkstra (11th May 1930 - 6th August 2002) was a Dutch computer scientist known for his essays on programming. However, he is best known for his famous algorithm for finding the shortest path between two nodes on a network. This algorithm was created in 1970 and forms the basis of many applications that routes calls through phone networks and emails over the internet, not to mention its massive implication in global positioning systems (GPS).

Dijkstra's algorithm is a greedy algorithm which means it is a child of an algorithmic paradigm that looks for simple solutions to complex, multi-step problems by deciding on the locally optimal choice to make, in the hope it will lead to a global optimum. Such algorithms are called greedy as the algorithm does not consider the problem as a whole and focuses on providing an immediate output. In short, once a decision has been made, it is never reconsidered.

An advantage of this paradigm is that the solutions of small sections of a problem can be easy to understand and candid. On the flip side it is possible that the most optimal short-term strategy may lead to a bad long term outcome. Although this might be the case it is important to consider the worst case scenario. In business this risk is an important consideration and should be taken as an upper bound so a company knows the maximum they might have to spend.

Dijkstra's algorithm is subject to some limitation which means it cannot be performed on all types of graphs. These constraints are as follows:

- The edges on the graphs must have nonnegative weights and each network must be connected.
- The algorithm can work on both digraphs and undirected graphs.

One of the reasons why Dijkstra's algorithm is so popular is because it is simple to understand. I have written pseudocode for this algorithm to show this.

Vertices = the set of all vertices

s = null set

#S is the set of vertices that have been visited

distance[s] = 0

Repeat for all i is an element of Vertices – [s]

where [s] is the source vertex

distance[i] = infinity

Q = Vertices

#Q is the queue of the vertices to be considered

While Q does not equal null set

u = minimum distance from current node to next node

S = the union of S and u

Repeat for all i is an element of neighbouring[u]

If distance[i] > distance[u] + w(u,v)

Then:

distance[v] = distance[u] + w(u,v)

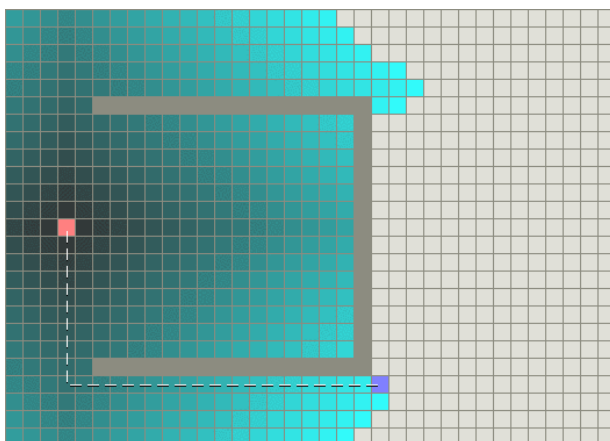
Return distance

The above algorithm is the old variation that only outputs the shortest path, not the path itself.

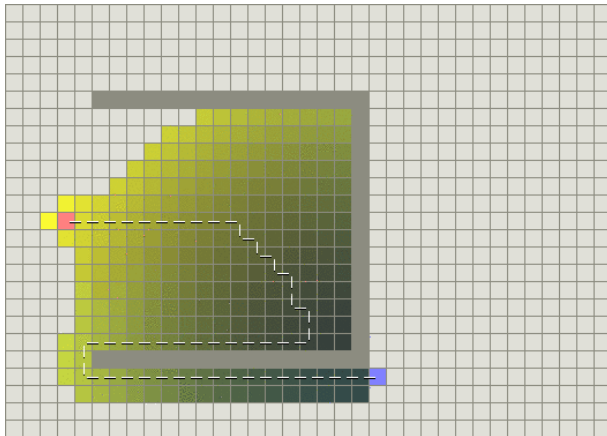
A history wouldn't be complete without a discussion of why this algorithm actually works. Intuitively it creates optimal substructures meaning the subpath of any shortest path is itself the shortest. One can argue with confidence that this works in reverse and hence so that an optimal path can be composed from all of the optimal subpaths between the points in between two vertices. If not convinced one can look at triangle inequalities. For example, if $\delta(u, v)$ is the shortest length between u and v, where u and v are two vertices on a graph or network. Let x be the third vertex that forms the triangle. $\delta(u, v) \leq \delta(u, x) + \delta(v, x)$.

Alternate Algorithms:

The A* algorithm is a popular algorithm when it comes to pathfinding that unlike Dijkstra's algorithm, is not greedy. It does not always optimise the next choice but looks at the whole problem and chooses a route accordingly. This can come in useful when you want to navigate round obstacles to a goal. The comparison can be seen in the two images below taken from <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>:



A* algorithm



Dijkstra's algorithm

The A* algorithm is obviously the better choice when there are obstacles in the network being used. However, Dijkstra's algorithm uses much less computation and on relatively simple network with little or no obstacles to navigate around, the A* is almost redundant because it uses far more resources and computation time.

Transportation Algorithm

This project heavily relies on the collection of algorithms introduced in my Further Maths decision maths course to produce the optimal number of items to transport from the stock nodes to the demand nodes. After undertaking research I could not find any code implementing such algorithms. I have therefore designed how to implement them myself.

The general method as described in the D2 textbook is as follows:

1. First find an initial solution that uses all the stock and meets all the demands
2. Calculate the total cost of this solution and see if it can be reduced by transporting some goods along a route not currently in the solution. (If this is not possible then the solution is optimal.)
3. If the cost can be reduced by using a new route, as many units as possible are allocated to this new route to create a new solution.
4. The new solution is checked in the same way as the initial solution to see if it is optimal. If not, any new routes found are included.
5. When no further savings are possible, an optimal solution has been found

Finding an initial solution is relatively simple. All one has to do is to create a table with one row and column for every destination. This represents the number of items to be transported from a source or supply node to a demand node. The supply and demand values of each node (given to the program by the user) line the rows and columns (supply values line the rows and demand values, the columns).

Once this has been done the north-west corner method is performed.

This method starts by considering the top leftmost cell and looks at the demand value (below) and the supply value (right). The smallest of these values is entered into the cell and subtracted from both – leaving one with a value of 0. Then consider the cell to the right if the current cell's corresponding supply value is not 0. If it is 0 now, consider the cell below. Continue this method until a path is achieved from the top left of the table to the bottom right. This is the initial solution to the problem.

Example:

Given 3 supply points with values (3, 5, and 4) and 3 demand points with values (4, 6, and 2) find the initial solution of a transportation problem:

Create our table.

	Demand node 1	Demand node 2	Demand node 3	Supply values
Supply node 1				3
Supply node 2				5
Supply node 3				4
Demand values	4	6	2	

Perform the north-west corner method

	Demand node 1	Demand node 2	Demand node 3	Supply values
Supply node 1	3			0
Supply node 2				5
Supply node 3				4
Demand values	2	6	2	

	Demand node 1	Demand node 2	Demand node 3	Supply values
Supply node 1	3			0
Supply node 2	1			4
Supply node 3				4
Demand values	0	6	2	

	Demand node 1	Demand node 2	Demand node 3	Supply values
Supply node 1	3			0
Supply node 2	1	4		0
Supply node 3				4
Demand values	0	2	2	

	Demand node 1	Demand node 2	Demand node 3	Supply values
Supply node 1	3			0
Supply node 2	1	4		0
Supply node 3		2		2
Demand values	0	0	2	

	Demand node 1	Demand node 2	Demand node 3	Supply values
Supply node 1	3			0
Supply node 2	1	4		0
Supply node 3		2	2	0
Demand values	0	0	0	

The last table is the initial solution.

Problems:

Unbalanced Problem.

A transportation problem is unbalanced if the total supply > total demand. The other inequality, when total supply < total demand is not considered for obvious reasons because the software should not be used if the supply is saturated in the first place. This creates a problem because the north-

west corner method as it will not leave all the supply and demand values with a 0 value, as in the example above, meaning there are still items to be transported which will cause problems in the next couple of algorithms.

In order to avoid this problem a dummy column needs to be added. This dummy column will have zero cost in the least cost matrix but have a demand value that equals the surplus stock value. Only when this change has occurred should the north-west corner method be performed.

Degenerate Solution.

This arises if the number of cells used is less than $n + m - 1$ where n = column number and m = row number. When a cell satisfies its corresponding demand and supply value. As they are both the same size. The following algorithms require that $n + m - 1$ cells are always in use. In order to get around this, a zero needs to be placed in this cell and the next cell to be considered can be either right or below the current one.

The next step is to see if the current solution is optimal. In order to do this, shadow costs, need to be generated. This involves finding the shadow costs which will be used in generating improvement indexes.

Shadow Costs

1. To do this all the cells in the current solution need to be replaced with their corresponding value in the table of cost matrix.
2. Then set the cost linked to the first source node equal to zero.
3. Then move along the row to other non-empty cells and find any other destination costs in the same way.
4. When all possible destination costs are in the current row, go to the start of the next row.
5. Move along this row to any other non-empty cells and using the destination costs found earlier find the source cost of the row. Then find all other possible destination costs.
6. Repeat steps four and five until all shadow costs have been found.

Improvement Indexes.

To calculate the improvement index in sending an item or unit from source node P to demand node Q is found $I_{PQ} = C(PQ) - S(P) - D(Q)$ where $C(PQ)$ is the cost value. The improvement indexes are produced for all cells that are currently not in the solution.

The solution is optimal if all improvement indexes are ≥ 0 .

If the solution is not optimal the most negative improvement index and the cell that it was generated for is considered the entering cell for the next algorithm.

The Stepping Stone Method.

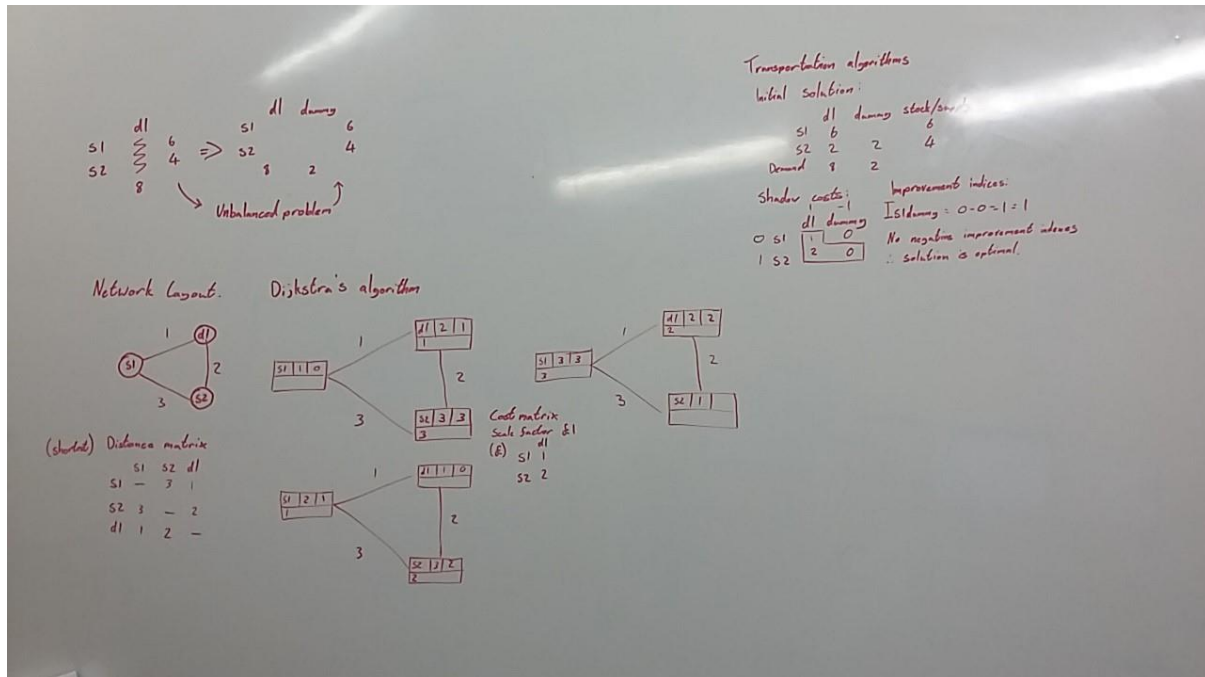
Taking the current solution again, enter the value θ into the empty 'entering' cell found from the improvement indexes. Create a cycle of changes using two basic rules.

1. In any row and column there can only be one increasing cell and decreasing cell
2. All adjustments (apart from the entering cell) must be made to non-empty cells.

Once the cycle of changes is determined, find the value of θ by finding the smallest number in the decreasing adjustments. After this, adjust the solution to include θ and remove the value of the cell from which you chose the value of θ from.

Possible Solutions:

It is possible to run all of the algorithms required to get a solution by hand but this will be very time consuming and will require lots of resources, for example paper. To demonstrate this the algorithms have been applied to a random network with random supply and demand values for the nodes by hand (image below). The specifics of the image are not important at this stage; however, it is easy to see that with more complex networks it would be inefficient to find a solution by hand, even if you had many people working on the same problem.



Alternatively, a computer could be used to run all of the algorithms required to find an optimal solution. This method can be done in a matter of seconds by a standard computer rather than the lengthy time required by a human computer. This will not only demonstrate the importance of computers using the algorithms talked about in my further maths course but will also demonstrate significant time savings if used by a business.

Given the above choices, programming bespoke software or finding existing software that performs the functionality seems the obvious approach in order to save time, money, and resources. In addition, in the context of a classroom it is much easier to demonstrate how powerful the transportation and Dijkstra's algorithm is if it is run on a computer, if nothing other than to show how quickly it obtains an answer. When students contrast this to the lengthy time it takes to run the algorithms by hand they will truly understand the value of running algorithms on computers.

Description of Current System(s):

Existing school system:

There is no existing computerised teaching tool for the decision modules taught in Further Maths.

Existing Business systems:

'ShortestPathFinder', by safe software is a pathfinding program that works out the shortest path between two nodes in a network, based on the length of the input or the cost (specified in an attribute) of each of the edges. This piece of software is similar to the original idea except it only works out the shortest path between a start node and a destination node. The concept proposed aims to calculate the optimal number of items to transport from each individual supply node to a set

of demand nodes. Although there is already software that exists online to find the shortest route on a network it does not perform the same task as this project.

<https://www.safe.com/transformers/shortest-path-finder/>

Caliper's TransCAD software does a similar job to what I have targeted my project to do. It aims to do the following:

“

- A powerful GIS engine with special extensions for transportation
- Mapping, visualization, and analysis tools designed for transportation applications
- Application modules for routing, travel demand forecasting, public transit, logistics, site location, and territory management

”

Quoted from their website: <http://www.caliper.com/tcovu.htm>

However, TransCAD is filled with lots of additional features which might overwhelm the user if they only want to perform the simple task of minimising the cost of transporting a number of items from one warehouse to another. By creating a single purpose system, a user can quickly and efficiently carry out the computations needed to find an answer on their network. The system that needs to be designed should be lightweight, because it should not take up much memory space and can easily be transferred from one computer to another. Furthermore, it is not designed for classroom use which is the main aim for my project.

Identification of the End User

The end user could be any business or organisation that needs to know how many items of stock to transport from a series of warehouses (supply points) to a set of shops (demand points) with the specific requirement to find the number of items that need to be transported from a single warehouse to each shop to minimise the overall cost. They should be interested in obtaining the optimal solution of the program to minimise cost.

The project will also be used by the maths and computer science departments at my school to demonstrate the powerful algorithms in their classes and to provide a template on how to layout A-level computer science projects in the future. I will be working with the mathematics department in particular to obtain feedback on the development of the overall system and on individual prototypes.

Interview with End User.

What functionality would you like to see in the system in addition to the results of the algorithms?

Ideally the system should be able to generate a visual representation. So a graph depicting the routes and networks obtained would be helpful as it would make the results much easier to interpret.

Do you think it will be useful for the results of each algorithm to be documented in a report?

Yes, as you're producing this as a teaching resource it would be helpful if that sort of data can be made available so that students can see the different results that are feeding into the final outcome.

Would it be beneficial to save the results of a computation to view at a later date?

Yes, for the same reason.

Do you think it would be useful to load previous network structures to make changes to?

Yes, either that or have some template networks available. Anything to make data entry faster is always helpful and as this is being used to demonstrate how decision maths may be used in the real world it's helpful to be able to easily input fairly similar networks so that students can look at how and why small changes to the input network alter the end result.

Identification of user needs.

1. The system should find the shortest paths between each individual supply node and each demand node in the network using a matrix inputted by the user.
2. The system should find the optimal number of items to transport from every supply node to every demand node to reduce the minimum cost. This should be done using the supply and demand values and distance weighting inputted by the user, and the results obtained from the above calculation.
3. The network inputted by the user should be displayed along with a report displaying the mathematical method the computer went through to generate the optimal solution along with said solution.
 - a. Report should be displayed in a text box so that the user can copy it into another document if they are writing project documentation. Or equally likely a teacher or student wants to make notes from the method.
 - b. Two graphs should be displayed. One based around a circle and another in a form similar to a bipartite graph (except nodes within a set can be joined together).
4. User should be able to save the report and the corresponding images of their network to a database.
5. Saved reports and the corresponding network graphs should be able to be loaded and displayed from a menu.
6. Users should be able to load a particular network structure that has already been inputted and saved into the system. This will make it quicker for users to make edits to current problems (say, if the scenario changed a bit such as the number of items requested by a particular demand node).
7. Users should be able to interact with the system without there being any discontinuities or errors.

Acceptable Limitations

The only limiting factor that will initially be worked with is the maximum size of the networks being entered. This is only due to the amount of computation time it will take and not because of the algorithms cannot handle the input. In the case that a large network was to be inputted into the system the user would be asked to simplify the network. Such a limit will be decided upon in the design stage or during the actual implementation of the system.

This limitation has been agreed with the end user.

Observations on research

A number of observations were made when undertaking this project.

School perspective:

My research has shown that no similar system to this project is being used in schools. If there is such a system, none is used within my school. Therefore, in order to enhance the teaching for the further maths decision modules there is a prompt for a new system to be developed.

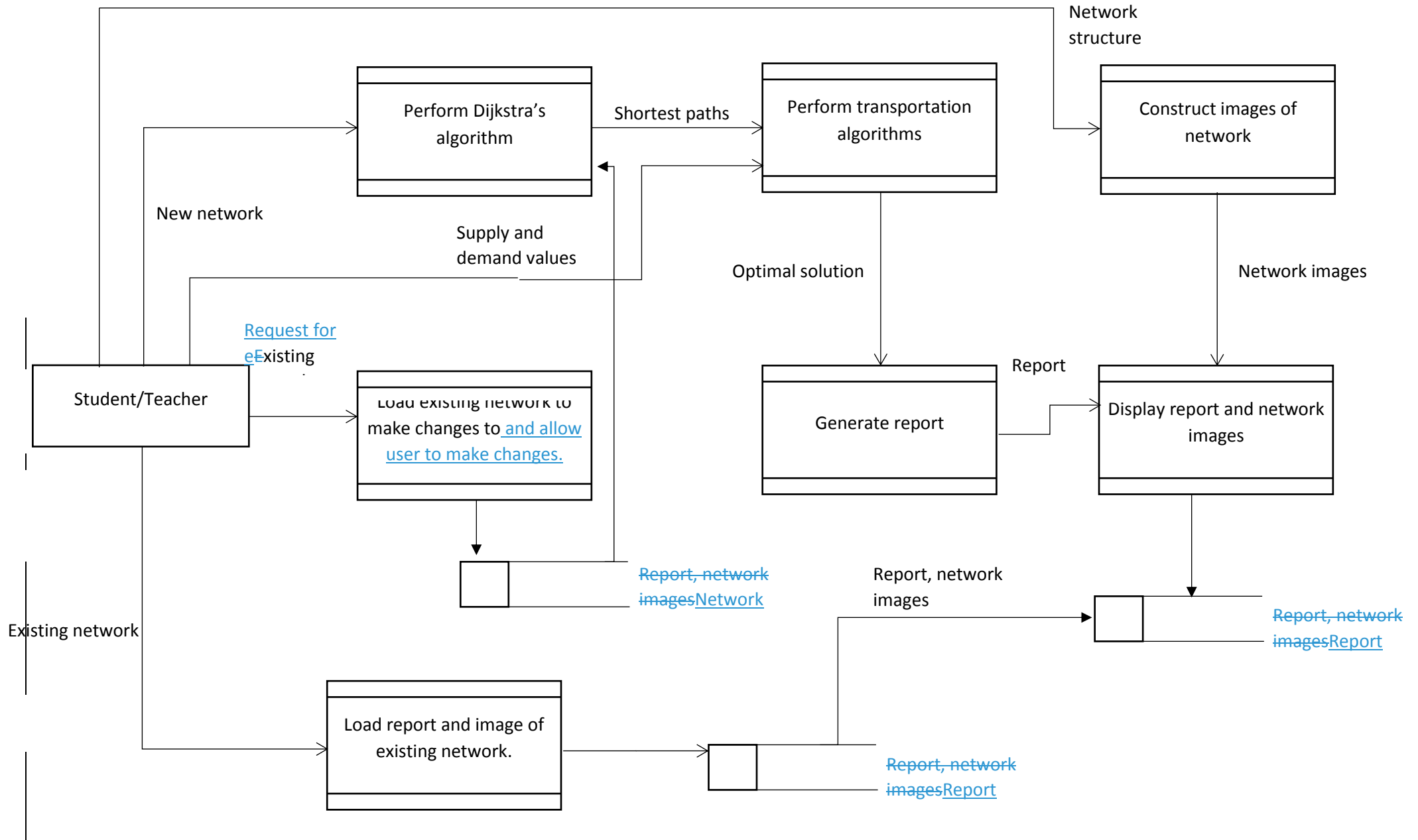
Business perspective:

Despite the idea being created before there appears to be no similar software that is purpose built for this type of problem. None of the researched applications are 'lightweight' and the fully functional pieces of software have a high cost. The proposed system to be developed should be cheap to produce, purpose built and designed for a single purpose.

Identifying Requirements:

Based on observations from existing systems the software that will be developed should be graphical. This will allow users to understand the process that they're going through and make the system more intuitive to use.

Data flow diagram for proposed system.



List of Requirements.

1. Based on 1. And 2. From identification of user needs:
 - a. The user should be able to input their desired network into the program as an adjacency matrix and all details about the nodes on the network through a series of graphical forms.
 - i. Adjacency / distance matrix should be scrollable to allow user to input all parts of the network.
 - ii. Title of the network should be displayed above the matrix (note: title of network should not scroll with the matrix).
 - iii. This process should be simple as to the user does not need to know they are inputting data into an adjacency matrix.
 - b. System should compute the shortest path from each supply node to every demand node.
 - c. System should use the transportation algorithms to find the optimal number of items to transport from each supply node to demand node.
 - i. System should perform the four steps of the transportation algorithm:
 1. North-West corner method to obtain initial solution
 2. Generate shadow costs
 3. Obtain improvement indices
 4. Perform the stepping stone method to obtain an improved solution.
 - ii. Algorithm should stop if an optimal solution cannot be found.
2. Based on 3 from identification of user needs:
 - a. System should generate a report that explains the computations that have been carried out.
 - i. Report should contain the conclusions obtained from the computation performed by the system.
 - b. System should graphically display the network the user has inputted alongside the report generated in two forms:
 - i. Bipartite graph form
 1. Bipartite graph should be scrollable when there are a large number of nodes.
 - ii. Circular graph form
 1. Circular graph should not display if the number of nodes in the network is greater than or equal to twenty as it will get too crowded.
3. Based on 4 and 5 from identification of user needs:
 - a. Allow user to save a report and network structure graphs.
 - b. Users should be able to load saved reports along with their network images.
4. Based on 6 from identification of user needs:
 - a. Allow users to load an existing network structure that has been saved and use it for a new problem
 - i. Changes can be made to the supply and demand values of nodes in an existing network.
 - ii. Changes can be made to the distances between nodes in an existing network.
5. Based on 7 from identification of user needs:

- a. System works together as a whole.
 - i. Individual modules and parts of the system work coherently with no errors or discontinuities.
 - ii. End user will try the system and give feedback.
 - iii. Outside user with no experience of the system should trial the piece of software to see if it is intuitive and easy to use. They will then present feedback in some form.
- b. Provide a graphical interface that is intuitive to use and flows smoothly between windows. It should be possible to access every part of the system through the GUI without having to restart the program.
 - i. All buttons should perform the action that it is intended to do and labelled appropriately (Eg. No buttons that are passive and don't do anything.)
 - ii. Drop down menus should contain clickable items
 - iii. Treeview structures (filing system) should be clickable and interactive using buttons.
 - iv. Allow users to use scrollbars.
- c. System should not terminate, or in other words not crash.
 - i. All points of system contact with user should be validated so that the algorithms that perform computation on the user data will not falter.
 - ii. All algorithms should not cause errors in the system.

Chosen Solution:

Dijkstra's algorithm and the transportation algorithms could easily be implemented as a console program as they don't specifically require a graphical user interface (GUI). However, objective 5b which was agreed with the end user would not be achieved if this method was carried out. A GUI would be required in order to meet this objective. There are countless of languages with appropriate libraries that could be used to create a well-designed GUI. The two languages I am the most familiar with are Python and Visual Basic. Visual Basic would definitely be my first choice when it comes to make a graphical system due to the simple drag and drop IDE of visual studio. Unfortunately the maths department uses both Windows and Mac operating systems so this quickly nullifies the latter option as it is not as portable as the former.

Python has a few graphical libraries such as EasyGUI, PyQt, and WxPython but I have chosen to pursue further development of the project using the tkinter library. This decision was made based on the fact that every installation of python comes with the tkinter library built in. This will make the system very portable. Furthermore, Python tkinter tutorials are abundant on the internet so any problems were encountered then there was likely be a tutorial that will help with fixing the problem I came across.

The only disadvantage of creating a GUI in python is that you would have to manually create all of the GUI as there is no development tools as such to help like there is with Visual Basic or Java. Although Java is an attractive option because it is more portable than python and has integrated IDEs for developing GUIs the time that would be consumed learning how to program in the language would be too much and it would leave little time for actual development. Taking this into consideration with the time it would take to program the GUI manually it is almost intuitive that the system should be used to tackle the problem.

Python, whilst mainly being a procedural language supports the use of objects. I think the problem I am trying to solve is well suited to and object oriented style of programming. This is because each window of the GUI can be created using an array of classes and objects and from my intensive

research whilst I was getting used to programming with tkinter I found OOP is the standard for implementing GUIs using this library. In addition, abstract data types as easy to implement using this approach as well as It making the code organised and easy to read. Lastly OOP allows for faster development which will allow my end user to receive a finished product quicker as objects can be reused throughout the program.

To summarise: Python will be used to develop the system as it supports OOP programming which will be the programming paradigm used as it leads for solutions to be produced faster. Python as a language is very portable unlike the alternative programming language I know, Visual Basic.

Table of analysis performed:

Date	Action performed	Notes
October	Initial discussion with Miss Hudson	Talked about ideas for a solution and outlined how the computer science project worked.
October	Meeting with Miss Hudson	Showed her the specification and outlined requirements for the system
November	Talk with Miss Hudson	Continued speaking about requirements and finalised them.
November	Progress update with Miss Hudson	Presented my research and gave a progress update.

Documented Design

What will the program actually do?

The chief aim of this project is to create a piece of software that, given inputs of a network and a list specifying supply and demand nodes, is able to calculate the optimal number of items or stock to transfer from each supply point to each demand point.

This software will utilise a database which will store networks previously inputted into the system and display graphically each one so that the user can select the structure of the network before inputting details on top. This will allow the user to avoid the task of inputting the network into the system as an adjacency matrix. However, if the network does not exist inside the program, the user will be able to input the network manually.

The system will be fully integrated with a graphical user interface, making use of the in-built tkinter module in python. Dijkstra's algorithm will be used to find the shortest path between each node on the network. Each shortest distance will be entered into a table of least distance. Each value in this table will be multiplied by a constant (set by the user) representing price per unit distance. Using a set of transportation algorithms that are described below the program will calculate the optimal number of items to be transported from the supply points (warehouses etc.) to demand points (shops etc.) to reduce the overall cost.

Modular Decomposition of the Project.

To make the project easier to manage and control I have broken it down into different modules. Each module is listed below:

- Planning/design
- Algorithms
- Graphical user interface
- Testing



Modules should be completed in this order

Planning/Design:

In this stage of development every aspect of the project will be considered and discussed as to what is the best way to implement a certain feature or if it is really necessary.

Table Showing Main Development steps

Step number	Description
1	Create initial designs for the system's graphical user interface.
2	Design plans for all the screens the user will interact with.
3	Create a prototype to test out network drawing in the python tkinter module.
4	Crte a prototype to generate an adjacency matrix.
6	Create a data flow diagram to show how data will flow through the system.
7	Write pseudocode for the final project
8	Write program code
9	List differences from original plan (if any exist)
10	Defend code that you have written.
11	Test program against original requirements

Graphical User Interface (GUI) – What is the need?

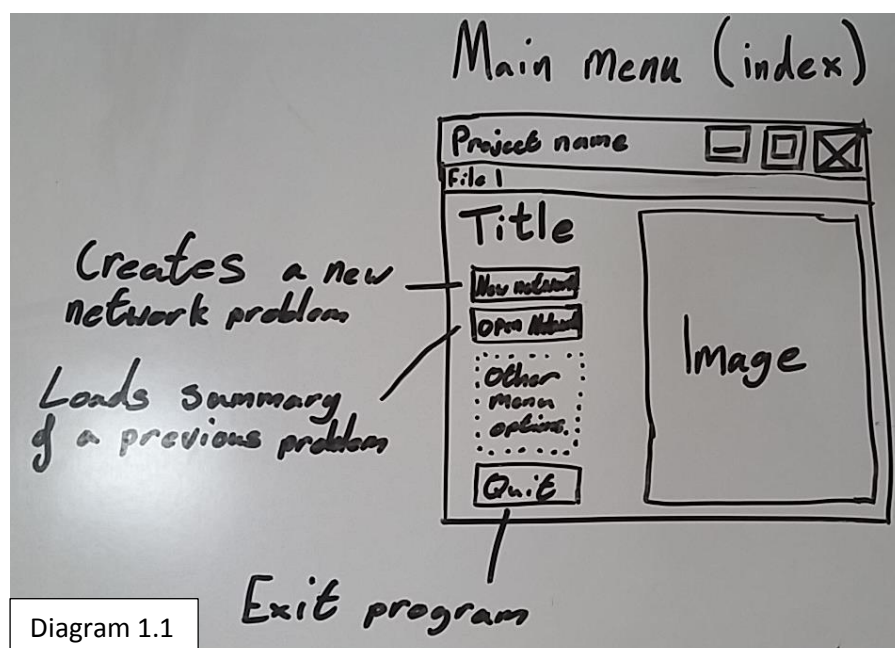
The alternative to a graphical user interface is a text based interface. This type of interface will negatively impact the user experience of the system and make it more difficult and time consuming to input a network (which evidently could lead to a loss in money for companies and organisations) to input a network. This problem is exaggerated for larger networks. On the other hand, a well-designed GUI that is intuitive to use should reduce the time it takes for new people to learn how to use it. Network input should be far more straightforward than possibly learning text commands for a text based UI.

A feature of the program being developed is that the user should be able to see what their network looks like graphically which simply wouldn't be possible if the system was text based. Furthermore, the speed at which a graphical application can be operated is much faster than a text based application due to user interaction with devices such as the mouse. Ergo, producing a graphical application only seems a natural way to develop the system.

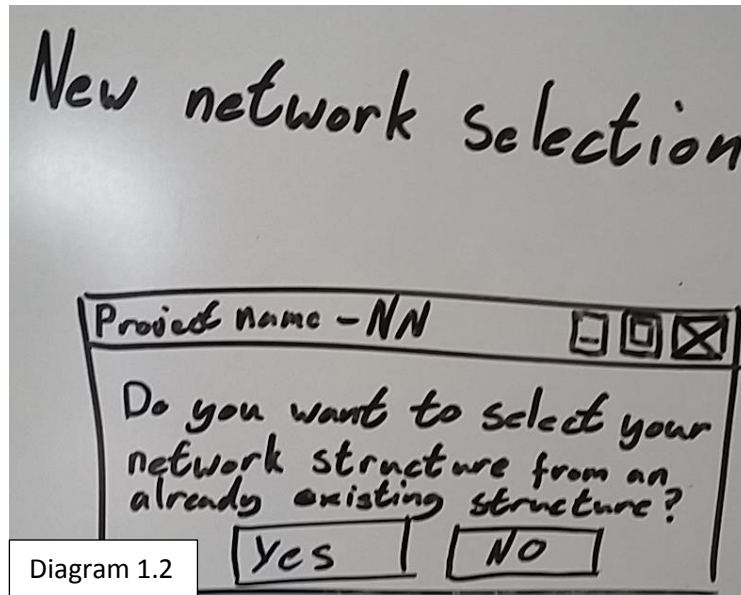
Graphical User Interface – A Walk Through the Program:

The main menu is one of the most important parts of a graphical user interface. I think it is imperative for it to be simple and intuitive to use and not too crowded. However, it mustn't be so simple that the user doesn't know what to do. From my research (evident in the analysis section) I have found that some of the most successful and intuitive main menus have an image corresponding to the functionality of the program. From the main menu designed the user has a choice of four buttons (one not included in the diagram below). These are the new network buttons, open network button, about button and the quit button.

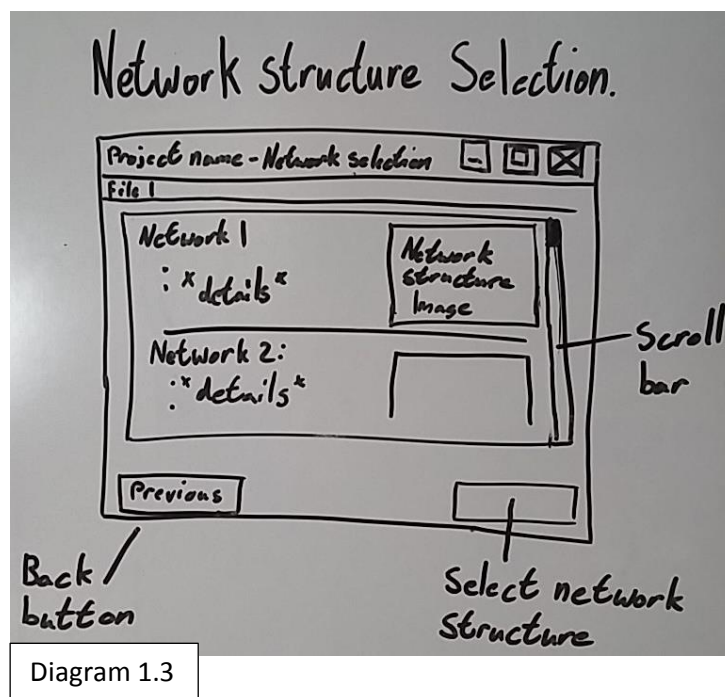
The new network button will open a separate window (Diagram 1.2) asking if the user wants to create their network based on a previously inputted network's structure or manually create a new network from scratch. Obviously if the network structure does not exist in the database they are forced to manually create a new network. An addition button that should be added to this window that was thought of after the initial design was created was a 'main menu button' necessary in case the user has pressed the 'new network' button by accident.



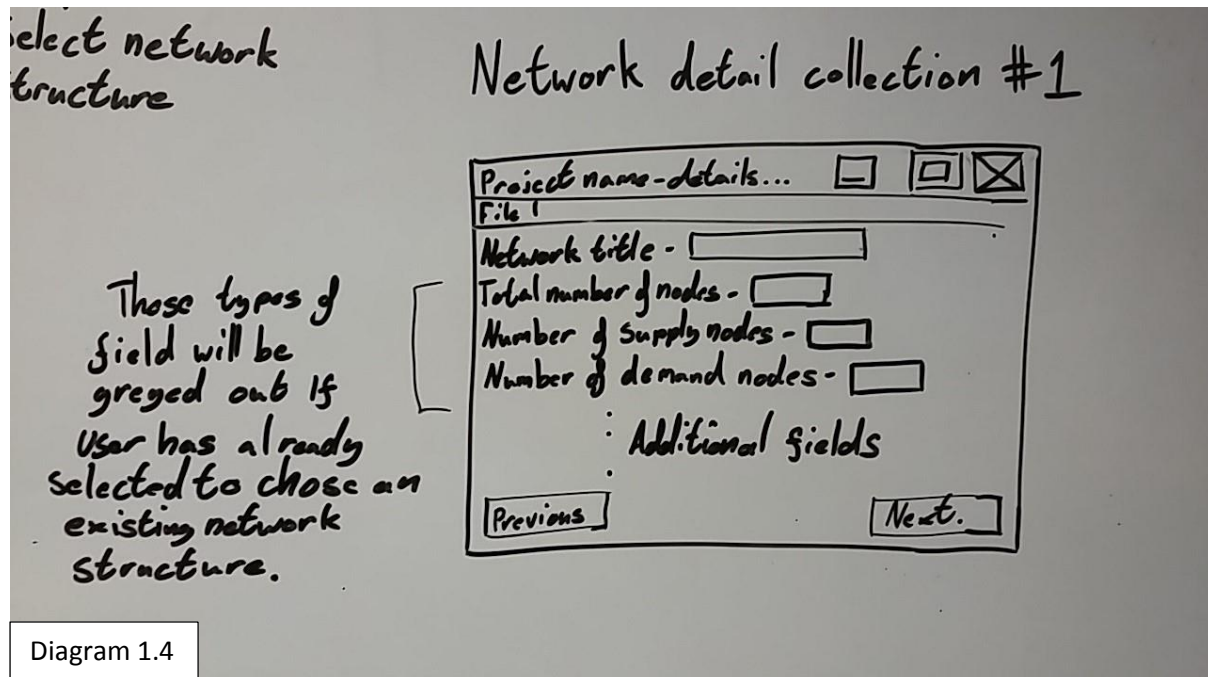
The open network button in the main menu window will cause another window to open where the user can select from results that have been calculated in the past. Once one is selected and confirmed by clicking the 'finished' button the corresponding report will appear in a new window. This is a useful feature and will save computation time as the report has already been generated and is stored in a database.



If the user selects the 'Yes' button then the following window will appear, closing the new network selection window. It is important that windows close when new ones open in order to give an 'uncluttered' feel to the application. For ease of use you wouldn't want to manually close all the open windows by yourself when you no longer feel the need for them.



This menu shows content taken directly from the attached database. The content is all of the information about previously entered networks along with an image of the structure of the network. This information will include the number of nodes in the network and the vacancy of each node (the number of arcs leaving/joining the node) in the network. This window will include a scroll bar to avoid the problem of trying to include all of the database information on one page. The 'previous' button will be used to return to the new network window. Each 'frame' in the window will have a property which will allow it to be selected. Once a user has clicked on the 'next' button (labelled as 'select network structure' in the diagram 1.3) and a frame is selected, the window will close and a network detail form will appear.



Some fields will be greyed out if the user has selected to choose an existing network structure because they would have been filled in already. Fields such as 'title' and 'number of supply nodes' will still need to be filled. If the user has chosen to set up the network manually they will need to enter data into all of the fields. Once 'next' has been selected and all relevant forms have been filled in the second detail collection form will open (the current window closes).

Network detail collection #2

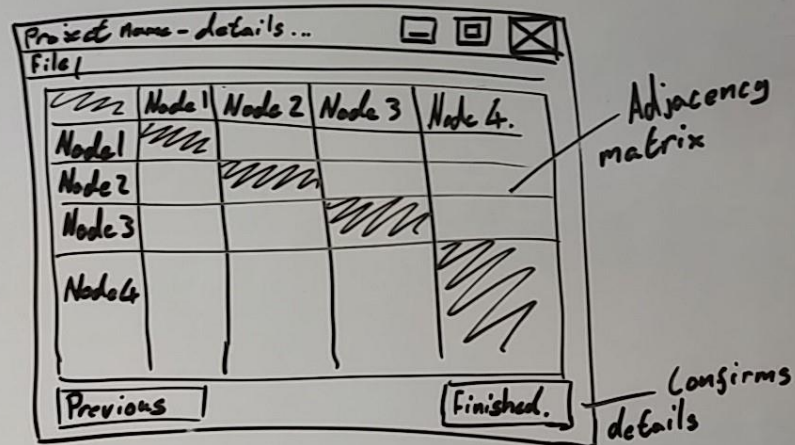
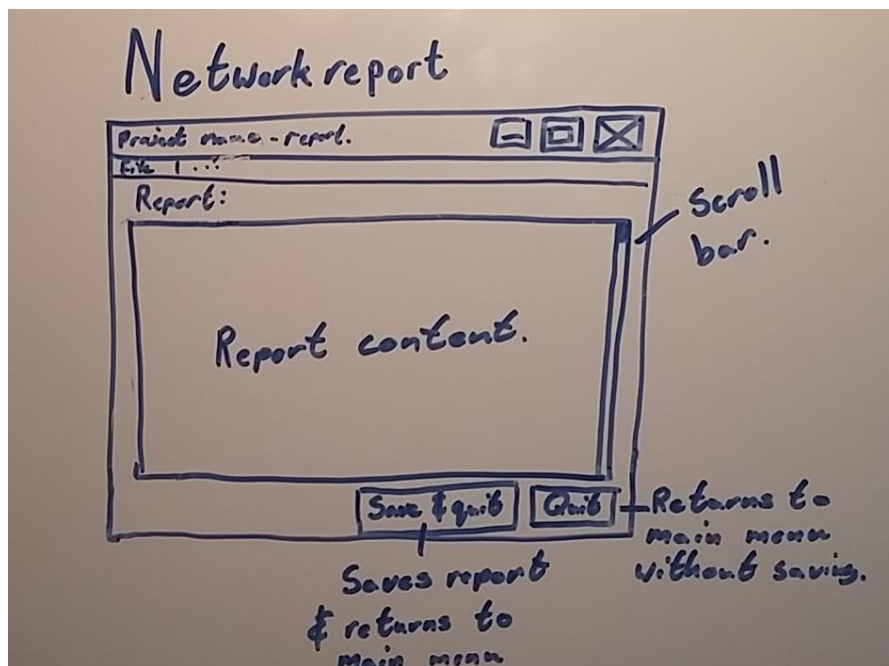


Diagram 1.5

This window contains an adjacency matrix where the user fills in the distances between each node. These will be implemented in Dijkstra's algorithm. The matrix is autogenerated by the program using the data collected in the previous forms. The data collection will be complete when the user clicks 'Finished'. Intuitively, the 'Previous' button will take you back to the last form opened.



This is the final GUI window in the system and will contain the report and the results of all calculations including the images of the network. It has been decided that two images of the network will be displayed, one of the network drawn round a circle and one of a bipartite graph. Although not displayed in the diagram a print button will be added.

General Explanation

The program will be designed and developed using the Python programming language as it supports object oriented programming. This will allow the code to be organised into classes and objects can be created as instances of the classes – similar to a real world situation. Python is mainly a string based language so all the in-built string handling functions will be readily available to utilise in addition to the useful SQLite and tkinter modules that come with the language. The system will be graphical and will be heavily based around the tkinter module. Furthermore, reports and networks created by the user will be stored to edit and view at a later date. This will require a database to store this data. Luckily – (and another reason why I chose to use python) is the fact that a module called SQLite is built into the language.

System

Index

Select Report

Select network structure

Get selected
report

Network detail collection

Distance
collection

Transportation
algorithms

Class diagrams are in other document (Object document.).

File Organisation, Record Structure and Processing.

The system will use a single database file whose structure and layout can be *paste database section here.

Identification of validation required

Form	Data Item	Check type	Check Details
Network detail collection	Network title	Length	Must be less than 25 characters long
Network detail collection	Total number of nodes	Format	Must be a decimal number in order to be handled by later code.
Network detail collection	Total number of nodes	Range	Total number of nodes must be less than 60 for the algorithm to run efficiently (and for the user to efficiently input the data).
Network detail collection	Number of supply points	Format	A number needs to be entered by a user here. Any other data type will cause an error with the algorithms it is used by.
Network detail collection	Number of supply points	Range	This number needs to be greater than zero and less than the total number of nodes
Network detail collection	Company name	Length	Must be less than 25 characters long
Network detail collection	Creator first name	Length	Must be less than 25 characters long
Network detail collection	Creator second name	Length	Must be less than 25 characters long
Distance collection	All matrix fields (or cells)	Format	Number must be an integer to be used by dijkstra's algorithm
Distance collection	All matrix fields (or cells)	Range	Number must be less than 999 as this is the 'large' number used by the algorithm to represent infinity. In addition the number must be greater than or equal to zero – a zero represents no connection between two nodes.
Weight collection	All Stock (supply) fields	Format	All stock must be an integer number since

			you can't have 0.5 of an item being transported. In addition, the value cannot be a string.
Weight collection	All Stock (supply) fields	Range	Realistically there should be no cap on the number entered but the size is limited to 9999.
Weight collection	All Demand fields	Format	All demand values must be an integer number since you can't have 0.5 of an item being transported. In addition, the value cannot be a string.
Weight collection	All Demand fields	Range	Sum of all demand values
Weight collection	Cost of travel per unit distance	Format	Must be a float (decimal)
Weight collection	Cost of travel per unit distance.	Range	Must be greater than zero.

Identification of Suitable Algorithms.

Dijkstra's algorithm

Dijkstra's algorithm is a greedy algorithm used to find the shortest path on a graph. It follows 5 main steps. Each node in the network is assigned a working value that changes throughout the algorithm. The working value is the cost of travelling from the source node (start node) to the current node.

1. Start by considering the source node which should be given a final value of 0.
2. For each node connected to the current arc being considered (eg. The source node if first iteration of algorithm) calculate the new working value for the connected node from:
 - a. Working value = final value of current node (which is 0 for the source node) + weight of connecting arc.
 - b. Replace the previous working value of the connected node if the newly calculated value is less.
 - c. Else, discard this new working value.
3. Look at the working values at all nodes that have not been finalised into final labels. Choose the smallest working value and finalise it into a final label. Consider this new arc in the next iteration of the algorithm.
4. Repeat steps two and three until the working value of the destination vertex is given a finalised label.
5. The shortest path to get from A to B is found by tracing back from the destination node to the source node. Given that B already lies on the route, include arc AB whenever final label of B-final label of A = weight of arc AB where A and B are arcs on the network.

I have decided to not to use point five as part of my algorithm because the shortest path will be not be needed in the final display of the results.

Below is the object oriented pseudocode I plan to base my technical solution algorithm off.

```
FUNCTION dijkstra(self, start node, graph):
    track = False
    route = []
    current_node = start_node
    WHILE algorithm not complete
        FOR number of nodes connected to current arc being
considered
            IF final label of current arc + weight of arc to
connecting node < working value of connecting arc than is not
finalised THEN
                Connected arc's working value ← final label
of current arc + weight of arc to connecting
node
                Connected arc's current route ← Current arc's
route + connecting arc
            END IF
            Compare weight = 999
        FOR all nodes in network
            IF all nodes weighting < compare weight AND all
nodes is NOT finalised:
                compare_weight = all nodes weighting
//weighting obtains the working value for node.
            END IF
            IF compare_weight is 999 THEN
                ESCAPE WHILE LOOP
            ELSE
                Current node ← node object which is found using the
Graph object
                if Current node != Start node:
                    Finalise current node
            END IF
        Store results in start node object.
```

Transportation Algorithms

Initial Solution

This algorithm generates an initial solution to the transportation problem. It is important because the rest of the methods used by the system to solve this problem are built on what is generated. The computer first needs to consider if the total supply is greater than the total demand. If this is the case the problem is said to be unbalanced and a column containing dummy values is added to the matrix.

```
IF total_supply > total_demand THEN
    Add_dummy_column() // abstract function/method that adds a dummy
    column
END IF
```

The dummy column in the cost matrix contains the value 0 in its cells and the demand value for this column is equal to the difference between the total demand and the total supply values. This change allows the algorithm to run as it can only work with a balance problem.

Once this check has been made the program should consider the cell in the top left hand corner of the cost matrix (cost matrix will be generated using the values obtained about the network and Dijkstra's algorithm. In addition its main purpose was described in the analysis). The supply value and demand value for the row and the column that contains the cell are looked at. The smaller value of these two is then subtracted from the larger and the number obtained from this calculation is placed in the cell. If the demand value is greater than the supply value then the next cell to be considered is right from the current one. Otherwise, the cell directly below the current one should be considered next. This comparison process repeats until the bottom right hand corner is reached.

```
WHILE the comparison process has not been made for the cell in the
bottom right hand corner
```

```
    IF current supply node > current demand node THEN
        Current cell value ← current supply value - current
        demand value
        Consider cell below next.
    ELSE
        Current cell value ← current demand value - current
        supply value
    END IF
```

A check should be made each time a cell is being considered to see if the supply value and the demand value for the cell equal each other. If this happens the solution is said to be degenerate and a cell from the one to the right and directly below the current cell should be chosen from randomly. The following code should be added to the above pseudocode selection statement.

```
ELSE IF supply value == demand value AND current cell IS NOT bottom
most right THEN
    Current cell value ← 0
    Try
```

```

        Consider cell below next

        // error handling statement as the cell below might not
        exist due to the size of the matrix

    CATCH

        Consider cell right next

ELSE IF current cell is bottom most right THEN

    Current cell value  $\leftarrow$  0

```

The new matrix obtained from this algorithm is called the initial solution. This solution is used to generate shadow costs which will be explained in more detail in the next section

Shadow Costs:

The next step in the transportation algorithm is to generate shadow costs. This is done by taking the values in the cost matrix whose cells are also filled by values in the initial solution. The following is an example of the calculation of shadow costs using a three by three cost matrix and shows the corresponding initial solution

Cost matrix:

	Demand node 1	Demand node 2
Supply node 1	5	3
Supply node 2	2	5

Initial solution:

	Demand node 1	Demand node 2
Supply node 1	3	2
Supply node 2		3

The values in the cost matrix are placed into the initial solution matrix where there are values.

	Demand node 1	Demand node 2
Supply node 1	5	3
Supply node 2	*Cell not used*	5

Each supply and demand node has its own shadow cost. The system carries out the following steps to generate the shadow costs.

1. Set the shadow cost for supply node 1 to zero.
2. Move to the right along the cells until reaching one containing a value (i.e. that is not empty).
 - a. Subtract the shadow cost of the supply node of the row or the demand node of the column, whichever one contains a value from the cost in the cell.
 - b. Enter the result of this calculation as the value for the shadow cost for the row or the demand node of the column, whichever one is empty.
 - c. If both the shadow costs already have values keep moving along the row.
3. If you have reached the end of the row move down and repeat process 2.
4. Algorithm terminates when the shadow costs for each node has been filled in.

Obviously some sort of indefinite iteration should be used here as we want the algorithm to repeat until a certain condition is met. In this case the algorithm should repeat until all shadow costs all

filled in. In most cases it is important to consider what will happen if the shadow costs are not filled in because the program might continue in an infinite loop. However, this process will always terminate if it is run on a balanced solution. As precautions have been made when generating the initial solution this algorithm will always terminate.

The standard way of performing indefinite iteration in python is to use a while loop. Hence, this the approach that will be taken in the technical solution.

Improvement Indices

Improvement indexes are generated to check whether the current solution obtained is optimal, i.e. the number of items to transport from each supply node to each demand node reduces the overall cost of transport to a minimum value. The algorithm to generate these is run every time a new solution is obtained by the algorithm. There are two actions that can be performed based on the results of the improvement indexes:

1. If at least one improvement index is a negative value then the new solution obtained is not optimal and can be improved upon. This is done by using the stepping stone algorithm which I will detail in the next section. The stepping stone algorithm requires an entering cell to be chosen (the meaning of this term will be explained later). The chosen cell will be the cell whose improvement index is the most negative.
2. If all improvement indices are positive numbers then the current solution is optimal and the whole transportation algorithm can terminate.

Improvement indices are generated for all cells that are not currently in the optimal solution. Using the cost of each cell not in the solution and the shadow costs for the supply node of the row and the demand node of the column, the improvement index can be calculated.

Cell improvement index \leftarrow Cost of cell in cost matrix - (shadow cost of supply node + shadow cost of demand node)

This is done for all cells not in the optimal solution and the following pseudocode shows how I am planning to implement this in the technical solution.

```
FOR number of cells not in new solution
```

```
    Cell improvement index of cell being considered  $\leftarrow$  Cost of cell  
    in cost matrix - (shadow cost of supply node + shadow cost of  
    demand node)
```

```
END FOR
```

Definite iteration is used in this case because the nested code only needs to be run a finite number of times which can be determined before the loop begins. The number of times as explained above is the number of cells that are not in the current solution.

The nested calculation generates an improvement index for each cell in the loop.

Stepping Stone Method

This method is a heuristic process which is relatively simple for a human to perform but it more complicated for a computer to perform because humans wouldn't use a set algorithm for using it.

Wikipedia states the definition of a heuristic algorithm as, "In computer science, artificial intelligence, and mathematical optimization, a heuristic is a technique designed for solving a

problem more quickly when classic methods are too slow or for finding an approximate solution when classic methods fail to find any exact solution”.

This method (example shown in analysis) enters a value theta into the ‘entering cell’ found by taking the cell with the most negative improvement index if the solution can be improved at all. Then, in order to keep the problem balanced theta must be subtracted and added in a sort of ring shape in the solution matrix to make sure that the total supply equals the total demand because theta is normally non-zero, if theta is equal to zero then the exiting cell is also zero. To find the value for theta, once theta has been added and subtracted to the surrounding cells, the smallest value in the matrix which theta is being subtracted from is taken to be its value. The cell which has its value taken becomes the exiting cell. If more than one cell has the same value (and theta is being subtracted from it) as theta then the exiting cell is chosen randomly from these values. The remaining cell is given the value of zero and is not removed.

This can be summarised into a few steps:

1. Choose entering cell from most negative improvement index
2. Enter theta into this cell in current solution
3. ‘Step’ round the solution adding and subtracting theta.
 - a. There must only be two values of theta in every row and column.
 - b. Add theta to cell if it is in the same column or row as a cell where theta is being subtracted from a value and complies with a.
 - c. Subtract theta from a cell if it is in the same column or row as a cell where theta is being entered or added to the value and complies with a.
4. Find the value of theta by seeing which is the smallest value that theta is being subtracted from. Theta now has this value.
 - a. Existing cell is also this cell
 - b. If two or more cells have the same value of theta and theta is being subtracted from them a random cell is chosen to be subtracted from.

I have identified that a recursive algorithm is the best type to use for the main part of the stepping stone algorithm – adding and subtracting values of theta in a ring to keep the problem balanced. This is because multiple attempts can be tried for adding and subtracting theta and only when they satisfy the conditions 3, a, b, and c is theta added to the solution. However, there is one main disadvantage to this approach. With the increase in size of the matrix used the number of possible solutions that will be explored increases at an alarming rate. For this reason there is a cap on the size of the network that the user can input into the system.

The pseudocode roughly shows the structure that will be used in the technical solution for placing theta within the current solution:

```
FUNCTION stepping_stone(path) :  
  
    IF not complete THEN //runs nested code if algorithm is not  
finished  
  
        # Checking if row or column contains another theta.  
  
        Possible rows ← []  
  
        Possible columns ← []
```



```

t_count_row ← 0
t_count_column ← 0
Add possible cells in current row that could contain
theta to Possible rows
IF number of theta occurrences in row THEN
    # Clearing possible_rows so that the algorithm
    doesn't add any θs to it.
    Possible_rows ← []
END IF

Add possible cells in current column that could contain
theta to Possible columns
IF number of theta occurrences in column > 1:
    // Clearing possible_columns so that the algorithm
    doesn't add any θs to it.
    possible_columns = []
END IF

if number of theta occurrences in current row == 1 and
number of theta occurrences in current column == 1:
    // Checks if stepping stone method is complete.
    IF obeys conditions in 3.a,b,c.
        // Returns True so that no more is added to the
matrix.
        complete_step ← True
        complete_path ← path //complete new solution
        return True
    ELSE
        // Returns False so that more can be added to
the matrix.
        return False
    END IF

ELSE IF there is no new possible cells for theta
    return False
ELSE

```

```
// Recursion attempts to add addition values of  $\theta$  to
rows and columns to try and get a cycle of  $\theta$ .
```

```
IF number of cells that could contain a theta in
current row is not zero THEN
```

```
FOR number of rows in current solution THEN
```

```
save ← instance of cell that is going to be
modified //so that if the algorithm reaches a dead end with it,
original value can be retrieved.
```

```
New cell being considered += New cell being
considered add or subtract theta according to rules described in 3.
```

```
IF stepping_stone(path, position of current
cell in solution) returns FALSE THEN
```

```
Restore original value of cell stored in
save.
```

```
ELSE
```

```
ESCAPE LOOP
```

```
END IF
```

```
END FOR
```

```
END IF
```

```
IF number of cells that could contain a theta in
current column is not zero THEN
```

```
FOR number of columns in current solution THEN
```

```
save ← instance of cell that is going to be
modified //so that if the algorithm reaches a dead end with it,
original value can be retrieved.
```

```
New cell being considered += New cell being
considered add or subtract theta according to rules described in 3.
```

```
IF stepping_stone(path, position of current
cell in solution) returns FALSE THEN
```

```
Restore original value of cell stored in
save.
```

```
ELSE
```

```
ESCAPE LOOP
```

```
END IF
```

```
END FOR
```

```

        END IF
    ELSE:
        //If algorithm not complete return False.
        return FALSE
    END IF
END FUNCTION

```

- Function (will be a method in the technical solution) has a parameter called 'path' which is the current solution that will be utilised and changed each layer of the recursion stack.
- In each layer of the recursion the following method is applied
 - Possible cells where theta could be added are identified in both the row and the column of the position in the solution that is currently being considered.
 - If there are already two cells in the current row that have a value of theta these possible cells are removed as there can only be two values of theta per row.
 - If there are already two cells in the current column that have a value of theta these possible cells are removed because there can only be two values of theta per column.
 - If there are no cells that theta can be added to in both the row or column of the current cell, the solution is to check to see if it matches the correct criterial for a balanced problem (seeing if the values of theta form a 'ring')
 - If this is true than the layer of recursion returns a TRUE Boolean
 - Else it returns a False Boolean.
- When a TRUE Boolean value is returned to an instance of the recursive function it then, in turn returns a TRUE Boolean value. This signals that this part of the algorithm is complete.

Transportation Algorithms.

The above sections describe what happens in each part of the whole algorithm and explains how I plan to tackle each part in the technical solution. Once the initial solution has been found, shadow costs will be generated for it. Using these shadow costs the algorithm will calculate improvement indices. If there are some negative improvement indices, the solution is not optimal and can be improved. Using the stepping stone method a new improved solution is found. The process of calculating shadow costs and improvement indexes are repeated again. If the new solution is not optimal then the stepping stone method is performed again. This process is repeated until an optimal solution is found.

FINAL EDIT

It was discovered that during the development of the technical solution there were some problems that didn't have an optimal solution. For these types of problems I decided that the initial solution would be returned to the user. I decided that the cut-off point to decide whether a solution had an optimal solution or not was after 10 iterations of the transportation algorithm. This is because all of the solution I had previously tested they produced an optimal solution before 5 iterations. Allowing for 10 to pass is a safety net just in case a problem that I haven't tested takes a long time to produce an optimal solution.

User Interface Design Rationale (Human Computer Interface)

The system will be graphical as already discussed above. This GUI will need to be consistent to avoid confusion. This means there all windows will have the same colour scheme (grey background with blue window borders). All text will be black so that it stands out.

All windows must follow the same design. As shown in the walkthrough of the program (below), each window will have a toolbar having a file option which will reveal a dropdown menu when it is clicked on. From this menu users will be able to access different functionalities of the program such as creating a new project or quit (i.e. return to main menu). The buttons contained in each window will be displayed at the bottom in all but the main menu which will have the button options listed at the left side. Lastly, any matrix or form that is too big to fit into the fixed size of the window (when the network the user is entering is particularly large) will be scrollable. This will avoid the problem of users having to resize the window if a matrix or form is too large to be contained onscreen.

The distance collection window will contain a n by n matrix where n is an integer specified indirectly by the user in the network detail collection form. As specified above this matrix will be scrollable if it is too large to fit inside the window. Adjacency matrices (often referred to as distance matrix in mathematical graph theory) are symmetrical about the diagonal spanning from the top left hand corner to the bottom right hand corner. In order to save users entering the same information twice (and possibly risking making a mistake which will cause a runtime error in a future algorithm) the opposite side of the matrix will update to contain the same value as the one just entered by the user in real time. This will cut the time the user spends entering distances into the matrix by half.

Libraries and Modules Used.

Modules and packages in Python are similar to Libraries in languages such as Java and C. Throughout this project I will use the Python's tkinter module and SQLite. Tkinter is a graphical user interface package which will provide the main backbone for the project because I will use it to structure each individual module of my own code.

SQLite is a database package which is lightweight and already preinstalled with the most recent installation of Python (version 3.6 at the time of writing). This package enables me to utilise all the main features of MySQL without having to install it separately (making the program very portable).

These packages are discussed separately, and in more detail in the above sections.

Prototyping: Network Drawing

One of the features discussed during the Graphical User Interface walkthrough was the images of the network structure listed along with its details. I decided that a circle would be the best way to model any given number of nodes around, because it would be easy to connect any given node to any other. To test this idea a prototype was developed.

This idea will be implemented in the python programming language with the additional in-built tkinter library. This module will let me develop a graphical user interface for which the network will be drawn on.

The GUI window will consist of an Entry field and a button. Once the user has entered the number of nodes they want to appear in a network and clicked the 'draw' button the program will generate the graph on a canvas element (used for drawings in tkinter).

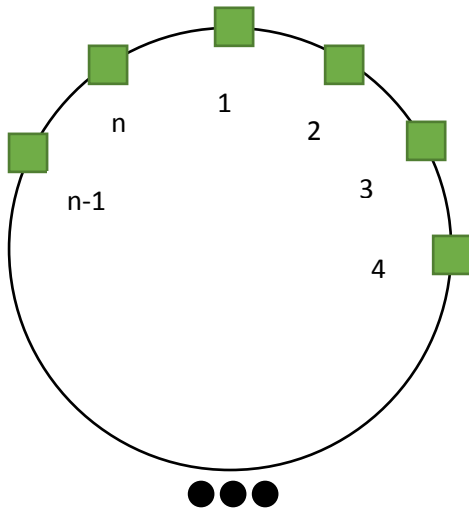
The maths required is fairly simple as we are using a circle to model the network.

The angle subtended from the arc between two adjacent nodes will be calculated as shown below.

$$\theta = \frac{2\pi}{\text{number of nodes}}$$

This formula measures the angle in radians as this is the format required for sine and cosine functions discussed later.

The nodes (represented by a box) will be equally spaced around the circumference of the circle



The coordinates of the centre of each node will be $(\sin(n\theta), \cos(n\theta))$ where n is the number of the node currently being considered.

Pseudo code:

Function draw (number of nodes)

Canvas.clear # clears canvas of current drawing

Radius = 100px

Angle = $2\pi/\text{number of nodes}$

n = 1

Repeat for each node

drawBox(coordinates($\sin(n\theta), \cos(n\theta)$))

n = n + 1

Repeat for each node

Connect(node) #connects current node to every other node in the network

The draw function is called whenever the user has inputted a new value into the Entry field in the GUI and will create a network on the canvas.

Program Code

```
#imports the tkinter module which is required to create a work GUI
from tkinter import *
#import the math module in order to use the sine and cosine functions
import math

#Sets up the main graphical window
root = Tk()

def circle(nodes):
    canvas.delete("all")
    radius = 100 #defining the radius of the circle
    radians = (2*math.pi)/nodes #Obtains the angle subtended by the circle arc between two adjacent nodes in radians
    #This loop draws the n number of nodes around a circle of radius 100px. Each node is a little green box (10x10px)
    for nodeLoop in range(nodes):
        box = canvas.create_rectangle(canvasSize-math.sin(nodeLoop*radians)*radius-5,
                                     canvasSize-math.cos(nodeLoop*radians)*radius-5,
                                     canvasSize-math.sin(nodeLoop*radians)*radius+5,
                                     canvasSize-math.cos(nodeLoop*radians)*radius+5,fill = "green")

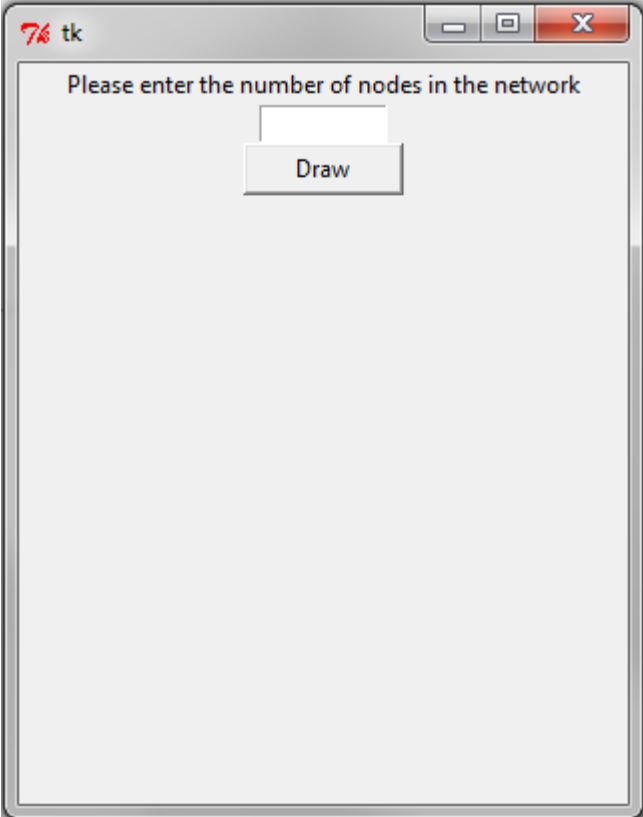
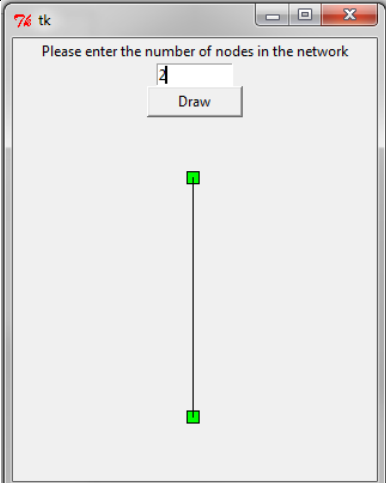
    #This set of nested loops goes through every node in the network and draws an arc from that node to every other node
    for nodeLoop in range(nodes):
        for lineLoop in range(nodes):
            #Drawing arcs
            line = canvas.create_line(canvasSize-math.sin(nodeLoop*radians)*radius,
                                     canvasSize-math.cos(nodeLoop*radians)*radius,
                                     canvasSize-math.sin(lineLoop*radians)*radius,
                                     canvasSize-math.cos(lineLoop*radians)*radius)

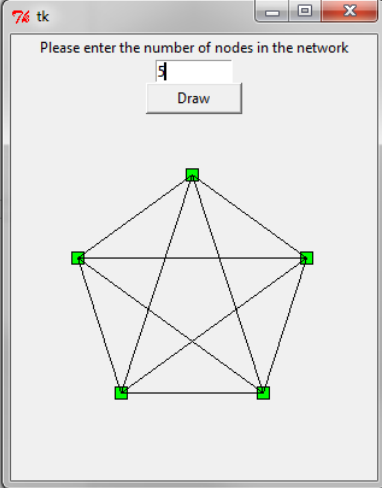
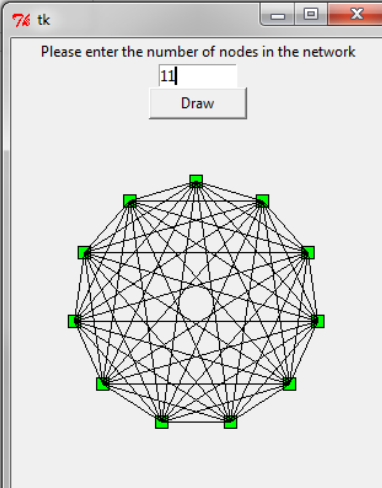
#Displays error message
def Error():
    messagebox.showinfo("Error", "That is not a valid input")
#This function

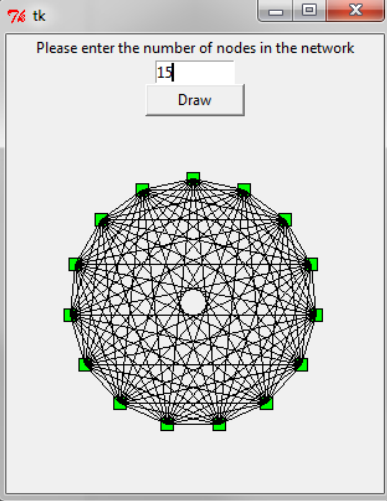
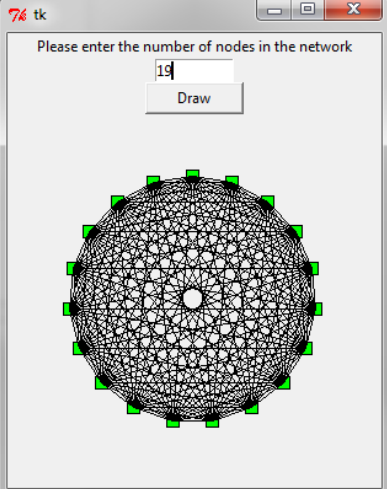
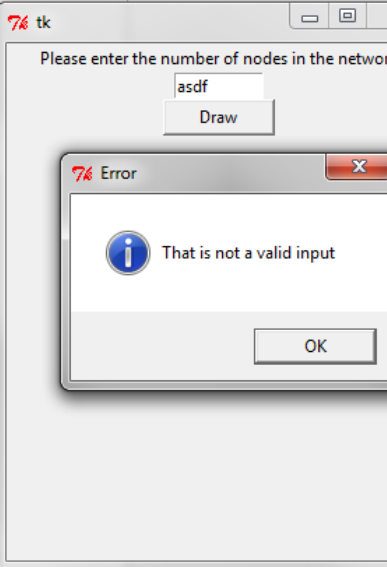
def select ():
    #Input validation
    #Except statement triggered if letters are inputted instead on a number
    try:
        nodes = int(mode.get())
        if nodes <=1:
            Error()
        elif nodes >= 20:
            Error()
        else:
            #passes the number of nodes inputted by the user into the circle function
            circle(nodes)
    except:
        Error()

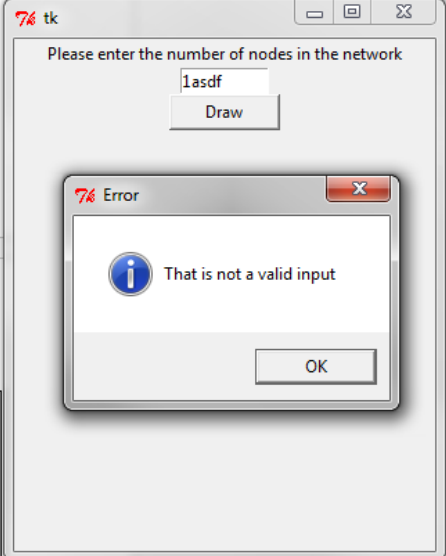
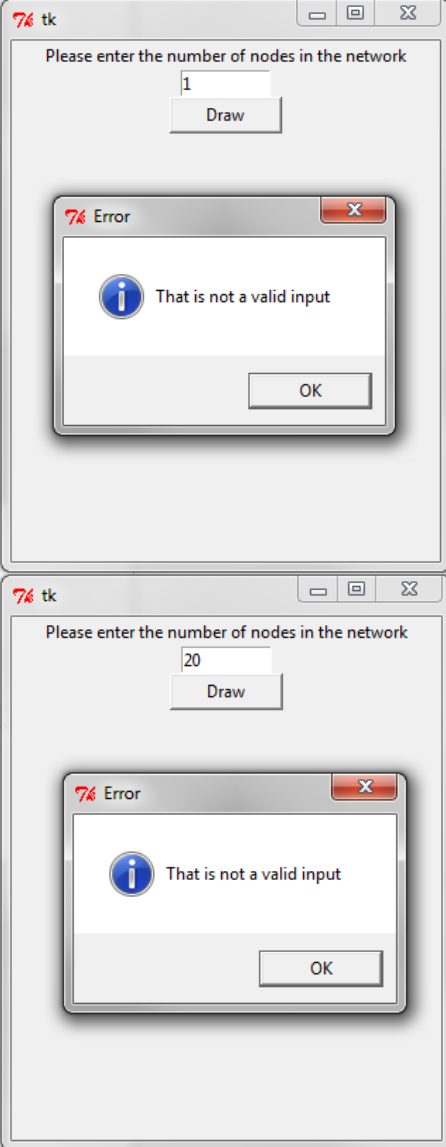
#The mode variable allows the program to collect the value in the Entry field when the button is pressed.
mode = StringVar()
#Writing an instruction for the user
instruction = Label(root,text = "Please enter the number of nodes in the network")
instruction.pack()
#Defining the entry field
e = Entry(root, width = 10, textvariable = mode)
e.pack()
#Defining an 'action' button that begins the drawing process.
drawButton = Button(root, text = "Draw",width=10,command=select)
drawButton.pack()
#Creating a canvas for the network to be drawn on.
canvasSize = 300
canvas = Canvas(root,width = canvasSize, height = canvasSize)
#The canvas size loop is altered as it is used later in finding the centre coordinates of the circle.
canvasSize = canvasSize/2
canvas.pack()
#This line keeps the window open (instead of closing straight away).
root.mainloop()
```

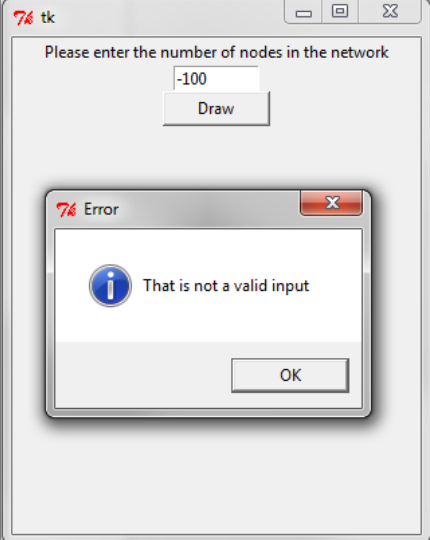
Testing

Test	Expected outcome	Actual outcome	Changes made
Program opens when code is run.	Main window opens.	 A screenshot of a Tk window titled "tk". The window has a standard Mac OS-style title bar with minimize, maximize, and close buttons. The main content area contains the text "Please enter the number of nodes in the network" in a blue font. Below the text is a white text input field. At the bottom center of the window is a button labeled "Draw".	None
Program draws network for small networks (that contains nodes 2- 10)	Program draws networks	 A screenshot of the same Tk window. The text input field now contains the number "2". The "Draw" button is still present. Below the input field, a network diagram is displayed, consisting of two small green square nodes connected by a vertical black line.	None

				
<p>Program draws network for larger networks (that contains nodes 11- 19)</p>	<p>Program draws networks</p>			<p>None</p>

		 	
<p>Program does not crash and shows an error message when erroneous data is inputted into the entry field.</p>	<p>An error message is displayed.</p>		<p>None</p>

			
<p>Program does not crash and shows an error message when numbers greater than or less than the accepted range (2-19) are inputted</p>	<p>An error message is displayed.</p>		<p>None</p>

			
Program closes when the red 'X' button is pressed	Program closes	Program closes	None

[Feedback for Prototype.](#)

[Miss Hudson was impressed with this prototype and commented on how quickly she received feedback on her suggestion. She agreed with my suggestion that there should be a limit on the number of nodes for the circle network to be displayed because it can get crowded round a circle with a fixed radius. In addition, Miss Hudson would like to see a graph with two columns, with one column containing supply nodes and the other containing demand nodes – much like a bipartite graph.](#)

Prototype 2: Generating an adjacency matrix from user input.

When the user needs to create a new network from scratch because the current network structure does not exist in the database connected to the program, it should be as simple and as easy as possible to do in order to inconvenience them as much as possible. In order to cater for the user's needs and to make the initial process of creating a new network manually a form has been carefully thought out (shown below).

Network title:

Number of nodes:

Number of supply nodes:

Number of demand nodes:

The naming of some fields could be changed in the future to cater for users who don't know network terminology. For example, the field 'Number of supply nodes' could be changed to 'Number of warehouses'.

Once the necessary details have been input an adjacency matrix will be generated using those details. An adjacency matrix is symmetrical about the diagonal from the top left corner and will be created from cells in which the user can input the distance between each node. If the user thinks that there is no connecting node between two nodes they will input a '-' to represent this. When the user updates one cell (or field) in the matrix the corresponding cell across the diagonal will automatically update.

A 4 x 4 adjacency matrix with 2 supply nodes and 2 demand nodes.

	Supply node 1	Supply node 2	Demand node	Demand node
Supply node 1				
Supply node 2				
Demand node 1				
Demand node 2				

(The diagonal is greyed out so data cannot be entered into these cells.)

Below is the pseudocode to generate the matrix based on the input already entered by the user in the form discussed above.

Code

```
#Importing the tkinter module to build GUI windows.
from tkinter import *

#This function creates the form in which the user enters
#details for the new network
def form1():

    #Set up of the GUI form.
    root1 = Tk()

    title = StringVar()
    titleLabel = Label(root1,text="Network title: ")
    titleLabel.grid(row = 0, column = 0)
    titleEntry = Entry(root1, textvariable = title)
    titleEntry.grid(row=0,column = 1)

    nodeNumber = StringVar()
    nodeLabel = Label(root1,text="Number of nodes: ")
    nodeLabel.grid(row = 2, column=0)
    nodeEntry = Entry(root1,textvariable = nodeNumber)
    nodeEntry.grid(row=2,column =1)

    supplyNode = StringVar()
    supplyLabel = Label(root1, text ="Number of supply nodes: ")
    supplyLabel.grid(row=3,column=0)
    SupplyEntry = Entry(root1,textvariable = supplyNode)
    SupplyEntry.grid(row=3,column=1)

    #the lambda command allows parameters to be passed into the function.
    finishedButton = Button(root1,text = "Finished",
                            command = lambda:gen(title.get(),
                                                  nodeNumber.get(),
                                                  supplyNode.get(),
                                                  root1))

    finishedButton.grid(row=5,column=0)
    #the mainloop function keeps the form open instead of closing right away.
    root1.mainloop()
```

```

def form2(title,nodeNumber,supplyNode):
    root2 = Tk()

    global entryStr
    global otherEntryStr
    #This array keeps track of all of the
    entryStr = []
    otherEntryStr = []

    track = 0

    title = Label(root2,text= title, font = "bold")
    title.grid(row=0,column=0)

    #Handles the display of the labels beside the matrix
    for i in range(1,nodeNumber+1):
        if i <= supplyNode :
            l = Label(root2,text="Supply Node "+str(i))
            l.grid(row=1,column=i)
        else:
            l = Label(root2,text="Demand Node "+str(i-supplyNode))
            l.grid(row=1,column=i)

    for i in range(2,nodeNumber+2):
        for x in range(nodeNumber+1):
            #Handles the display of the labels beside the matrix
            if x == 0:
                if i <= supplyNode+1:
                    l = Label(root2,text="Supply Node "+str(i - 1))
                    l.grid(row=i,column=0)
                else:
                    l = Label(root2,text="Demand Node "+str(i - supplyNode))
                    l.grid(row=i,column=0)
            #Places entry fields as 'cells' in the adjacency matrix
            elif i-1 < x:
                entryStr.append(StringVar())
                e = Entry(root2,textvariable = entryStr[len(entryStr)-1])
                e.grid(row=i,column=x)

            elif i-1 > x:

                pos = 0

                #Mapping the new cell to it's counterpart cell across the diagonal.

                #Complex code
                #for u in range(x-1):
                #    #pos += nodeNumber-1-u
                #    #pos += i-x-2

                #simpler code.
                pos = int( ((nodeNumber-1)**2+(nodeNumber-1))/2 ) - (((nodeNumber-x)**2+(nodeNumber-x))/2)
                pos += i-x-2

                #otherEntryStr.append(StringVar())
                e = Entry(root2,textvariable = entryStr[pos])

                e.grid(row=i,column=x)

    #mapp(otherEntryStr,entryStr,nodeNumber)

    root2.mainloop()
    #This function handles the generation of the new network window and input validation
def gen(title,nodeNumber,supplyNode,root1):
    try:
        nodeNumber = int(nodeNumber)
        supplyNode = int(supplyNode)
        if nodeNumber < supplyNode:
            messagebox.showinfo("Check input","The total number of nodes must be strictly greater than the number of supply nodes.")

    else:
        #closes the form 1 window.
        root1.destroy()
        #calls the form2 function and passes in parameters to deal with the generation of the matrix
        form2(title,nodeNumber,supplyNode)
    except:
        #Outputs and error message if input is not valid
        messagebox.showinfo("Error","Please check your input fields")

#This function is called at the start of the program.
form1()

```

Efficiency Problem.

In the program a block of code can be observed that has been commented out. This is because after it was written a more efficient way of finding the mapping was thought of.

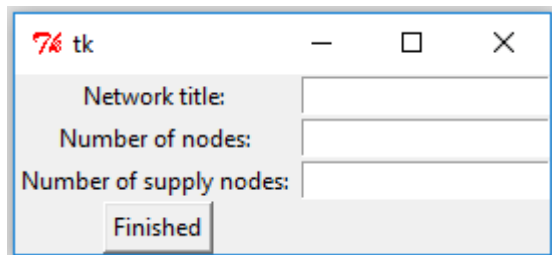
```
#Mapping the new cell to it's counterpart cell across the diagonal.  
  
#Complex code  
#for u in range(x-1):  
    #pos += nodeNumber-1-u  
#pos += i-x-2  
  
#simpler code.  
pos = int( ((nodeNumber-1)**2+(nodeNumber-1))/2 ) - (((nodeNumber-x)**2+(nodeNumber-x))/2)  
pos += i-x-2
```

The old code (the block in comments) had a time complexity of $O(x-1)$ whereas the new code has a complexity of $O(1)$, linear time. This means it will always take the same length of time to generate the position regardless of the input. Importantly this makes this part of the program more efficient.

This improvement was developed when it was noticed that the iteration was being used to calculate the number of cells in the columns before it matched the triangle numbers in decreasing order, the highest number being the number of rows in the far left column. The formula for the n th triangle number is $t_n = \frac{n}{2}(n + 1)$. This eliminates the need for iteration.

Improvements to be Implemented in Final Program.

Although there wasn't any technical improvements that would need to be implemented in the final program there are a few design changes that could be made. To start with, the form where the user enters details needed to generate the matrix is small and fiddly. This could made it difficult for the user to enter their details correctly. In the final program the form should be larger in order to make the software more user friendly.

A screenshot of a Tkinter window titled 'tk'. The window contains three input fields with labels: 'Network title:', 'Number of nodes:', and 'Number of supply nodes:'. Below the input fields is a button labeled 'Finished'. The window has standard window controls (minimize, maximize, close) in the top right corner.

Furthermore, the window title should be changed to something more appropriate such as 'Network details.'

Another observation is that every time the user does not enter the correct details into the form a messagebox is displayed. Closing this box could become tiresome after repeated misentries. This

could be improved by having the error message display inside of the form window. In addition, the error message itself could be tailored more towards the entry that went wrong instead of the general message displayed in the prototype.

Testing.

Similar tests were carried out on this prototype as the previous one. All tests were passed and no problems were encountered. This meant no adjustments needed to be made.

Feedback from Miss Hudson:

Miss Hudson liked the fact that she did not have to fill in both sides of the diagonal on the distance matrix. She commented that it cuts the time the user spends entering data in half. Her final comment on the prototype was that it would be helpful if the cells that had errors in on the matrix were highlighted in some way.

After further discussion we agreed that a "Fix" string should appear in the cell where there was an error.

Database Design.

In order to make the system user friendly it is important to include a database to efficiently store network structures that have been used in the past so networks don't have to be manually created again. In addition users may want to see summary reports for old problems. By using a database to store all data, everything is immediately accessible.

A database is normalised to reduce data redundancy and to try and maximise data integrity. These goals should be obtained by putting the database attached to the system in third normal form of normalisation.

Un-normalised Form:

Table1(networkTitle, numberOfNodes, numberOfSupplyNodes, creator, companyID, companyName, dateCreated, networkCircleImage, networkBipartiteImage, report)

Primary keys are underlined and foreign keys are followed by an asterisk. For example companyID is a primary key and creatorID* is a foreign key.

First Normal Form (1NF):

Company (companyID, companyName, creatorID*)

Creator(creatorID, creatorFirstName, creatorSecondName, companyID*)

Network(networkID, creatorID*, networkCircleImage, networkBipartiteImage, dateCreated, numberOfNodes, numberOfSupplyNodes, networkTitle, report)

To put a database into first normal form it has to be ensured that all tables do not contain repeating attributes (also known as repeating groups) and all data is atomic – as in all fields are broken down into their smallest components (for example, the field address can be broken down to address line 1, address line 2, etc.).

Second Normal Form (2NF):

Company (companyID, companyName, creatorID*)

Creator(creatorID, creatorFirstName, creatorSecondName, companyID*)

Network(networkID, creatorID*, networkCircleImage, networkBipartiteImage, dateCreated, numberOfNodes, numberOfSupplyNodes, networkTitle, report)

Second normal form is achieved by making sure the database is in first normal form and then removing partial key attributes (attributes that depend on the primary key in the table but other foreign keys as well.)

By putting this database into first normal form it passed the criteria for second normal form.

Third Normal Form (3NF):

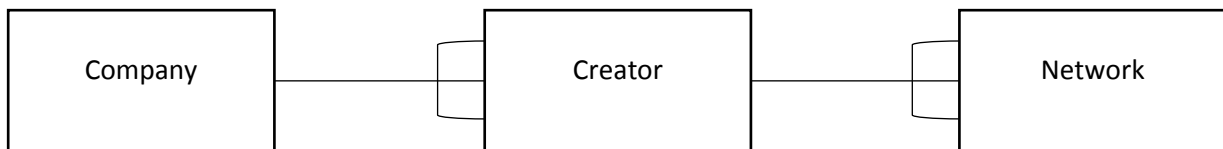
Company (companyID, companyName, creatorID*)

Creator(creatorID, creatorFirstName, creatorSecondName, companyID*)

Network(networkID, creatorID*, networkCirlcelImage, networkBipartitelImage, dateCreated, numberOfNodes, numberOfSupplyNodes, networkTitle, report)

A database is put into third normal form by making sure all non-key attributes that depend on other non-key attributes are removed and placed into other tables. This has already been achieved with this database by putting it into first normal form.

Below is the entity relationship diagram for the database:



This shows that a single company has many creators which in turn can create many networks. The reason why the company and creator tables exist is to make the reports generated by the program appear more official and allow them to be 'branded' by company and creator. Therefore it is logical to structure the database in this way after normalising it.

My SQL vs SQLite:

After normalising and considering the actual use of the system that is being created it was decided that SQLite should be used instead of MySQL. This decision was made for the many reasons, listed in the table below.

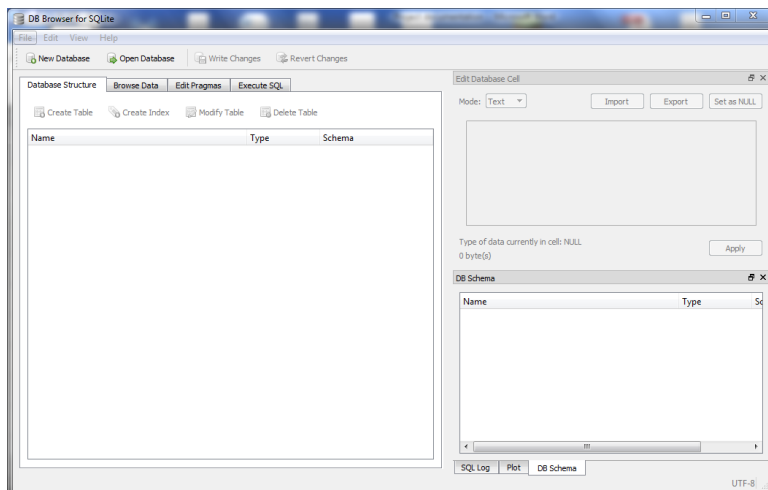
My SQL	SQLite
Designed for systems with multiple users. eg. Can be used on a network	Designed for systems with single users. Network capabilities can be added.
Designed for systems that require a degree with concurrency (eg. If multiple queries take place at the same time.)	SQLite has these capabilities but they are not nearly as extensive as My SQL.
Used when the database needs to scale with large amounts of data (eg. Terabytes)	Made to work efficiently with the amount of data stored on a typical home or work computer.

From this comparison My SQL is database with the most functionality. However, most of these features are not needed when considering the current project. This system is designed to work on a single computer and deal with the size of data stored on such a machine. Last is the fact that the database structure is not very complex and because this does not require a database that is as powerful as My SQL. Note that if the system needed to be improved for network implementation so that multiple users could work and contribute to the database the move to My SQL might be wise as it can handle concurrency issues much better than SQLite.

To summarise, SQLite is the best choice of database to use in this situation as it is suited to dealing with relatively small amounts of data and little or no concurrency issues.

Database Set-up.

A database a database browser was downloaded for SQLite to visually manage the database. A database browser is a universal table editor that allows users to access any database and modify it's tables and run SQL scripts. By using this browser, the initial set up of the database could be done without the need for writing code. (The screenshot below shows the initial interface of the SQLite database browser.)



The following SQL code was used to create the normalised database tables shown above.

Company table:

```

1 CREATE TABLE `Company` (
2     `companyID` INTEGER PRIMARY KEY AUTOINCREMENT,
3     `companyName` TEXT,
4     `creatorID` INTEGER
5 );

```

Creator table:

```

1 CREATE TABLE `Creator` (
2     `creatorID` INTEGER PRIMARY KEY AUTOINCREMENT,
3     `creatorFirstName` TEXT,
4     `creatorSecondName` TEXT,
5     `companyID` INTEGER
6 );

```

Network table:

```

1 CREATE TABLE `Network` (
2     `networkID` INTEGER PRIMARY KEY AUTOINCREMENT,
3     `creatorID` INTEGER,
4     `networkCircleImage` TEXT,
5     `networkBipartiteImage` INTEGER,
6     `dateCreated` TEXT,
7     `numberOfNodes` INTEGER,
8     `numberOfSupplyNodes` INTEGER,
9     `networkTitle` TEXT,
10    `report` TEXT
11 );

```

The following SQL commands will be executed every time the user saves data to the database. Although you cannot delete a database table from within the program, the code stops any possible errors occurring if a table is accidentally deleted manually.

```

CREATE TABLE IF NOT EXISTS `Company`(`companyID` INTEGER PRIMARY KEY AUTOINCREMENT,
`companyName` TEXT, `creatorID` INTEGER);

```

```
CREATE TABLE IF NOT EXISTS `Creator`(`creatorID` INTEGER PRIMARY KEY AUTOINCREMENT,  
`creatorFirstName` TEXT, `creatorSecondName` TEXT, `companyID` INTEGER);
```

```
CREATE TABLE IF NOT EXISTS `Network`(`networkID` INTEGER PRIMARY KEY AUTOINCREMENT,  
`creatorID` INTEGER, `allPaths` BLOB,`graph.rejected_nodes()` BLOB, `dateCreated` TEXT,  
`numberOfNodes` INTEGER,`numberOfSupplyNodes` INTEGER, `networkTitle` TEXT, `report` Text,  
`saveDistances` BLOB);
```

Test plan

Test Number	Window	Purpose	Test data	Expected outcome	Test data type	Result
1	Main menu	Check program closes when quit button is clicked	Quit button with be clicked	Program closes	Normal	Pass
2.a	Main menu	Check file drop down menu works when clicked	File option on option bar clicked	Dropdown menu is displayed	Normal	Pass
2.b	Main menu	Check network detail collection window opens when new project button is selected from the dropdown menu	Click new project button from dropdown menu	Main menu window closes and network detail collection window opens	Normal	Pass
2.c	Main menu	Check select report window displays when open report button is clicked from the dropdown menu	Click open report button from dropdown menu	Main menu window closes and select report window is displayed	Normal	Pass
2.d	Main menu	Check quit button closes the program when selected from dropdown menu	Click quit button from dropdown menu	Main menu window closes	Normal	Pass
3.a	Main menu	Check submenu displays when the new project button is clicked	Click the new project button from the submenu	Sub menu is displayed on top of the main window frame	Normal	Pass
3.b	Main menu	Check new project submenu disappears when the back button is clicked	Check the back button from the submenu	Submenu disappears when button is clicked	Normal	Pass
3.c	Main menu	Check network detail collection window opens when manual network button is clicked from submenu	Click the manual network button from the submenu	Main menu window closes and the network detail collection window opens	Normal	Pass
3.d	Main menu	Check network select network	Click the select	Main menu	Normal	Pass

		structure window opens when select report button is clicked from submenu	report button from the main menu	window closes and the select network structure window opens when select report.		
5	Main menu	Check select report window opens when open report button is clicked from the main menu.	Click open report button from the main menu	The select report window will be displayed and the main menu window will close	Normal	Pass
4	Main menu	Check program closes and minimises when square buttons are clicked at the top right hand corner of the window.	Click array of buttons in the top left hand	Program closes when the 'X' button is click, minimises when the '_' button is clicked, but does not maximise when the full screen button is clicked.	Normal	Pass
5.a	Network detail collection	Check file drop down menu works when clicked	File option on option bar clicked	Drop down menu displays.	Normal	Pass
5.b	Network detail collection	Check error message displays when new project is selected from dropdown menu as the user is already creating a new project.	Click new project button from dropdown menu	Error message is displayed notifying the user that they are already on a new project	Erroneous	Pass
5.c	Network detail collection	Check select report window displays when open report button is clicked from the dropdown menu	Click open report button from dropdown menu	Current window closes and select report window will open.	Normal	Pass
5.d	Network detail collection	Check quit button returns the program to the main menu.	Click quit button from dropdown menu	Current window closes and main menu opens.	Normal	Pass

6	Network detail collection	Check program doesn't display next window when all fields are not filled in.	Next button will be clicked when no fields are completed in the matrix	Error message will be displayed.	Erroneous	Pass
7	Network detail collection	Check a valid network name can be inputted into title field	"New network one" will be entered into the title field.	Value accepted.	Normal	Pass
8	Network detail collection	Check mixed (numbers and letters) can be inputted into title field	"123NewProject" will be entered into the title field	Title will be accepted	Boundary	Pass
9	Network detail collection	Check valid total number of nodes can be entered	The integer '4' will be entered into the total number of nodes field	Value will be accepted	Normal	Pass
10	Network detail collection	Check error message displays if a string is inputted instead of a number as the total number of nodes	The string "asdf" will be entered in the total number of nodes field	Error message is displayed (value rejected)	Erroneous	Pass
11	Network detail collection	Check error message displays if a negative number is inputted (invalid data) in inputted as the total number of nodes	The integer '-5' will be entered into the total number of nodes field	Error message is displayed (value rejected)	Boundary	Pass
12	Network detail collection	Check valid number of supply nodes can be inputted by the user (0< input < Number of supply nodes)	The integer '15' will be entered into the total number of supply nodes field	Value will be accepted	Normal	Pass
13	Network detail collection	Check error message displays if a string is inputted as the total number of supply nodes	The string "error" will be entered as the total number of supply nodes	Error message is displayed (value rejected)	Erroneous	Pass
14	Network detail collection	Check Error message displays if	The number '-4'	Error message is	Boundary	Pass

	collection	a negative number of inputted as the total number of supply nodes	will be entered as the total number of supply nodes	displayed (value rejected)		
15	Network detail collection	Check a valid company name can be inputted into company name field	The name "New company" will be entered	Name is accepted	Normal	Pass
16	Network detail collection	Check mixed (numbers and letters) can be inputted into company name field	The name "COmpany1" will be entered	Name is accepted	Boundary	Pass
17	Network detail collection	Check a valid network name can be inputted into creator first name field	The name "Alan" will be entered into the creator first name field	Name is accepted	Normal	Pass
18	Network detail collection	Check mixed (numbers and letters) cannot be inputted into creator first name field	The name "A1an" will be entered into the creator first name field	Error message is displayed (value rejected)	Boundary	Pass
19	Network detail collection	Check a valid network name can be inputted into creator second name field	The name "Turing" will be entered into the creator second name field	Name is accepted	Normal	Pass
20	Network detail collection	Check mixed (numbers and letters) can be accepted into creator second name field	The name "Tur1ng" will be entered into the creator second name field	Error message is displayed (value rejected)	Boundary	Pass
21	Distance collection	A small distance matrix is generated from the data inputted by the user in the Network detail collection form	The value '4' will be entered as the value for the total number of nodes.	A 4x4 distance matrix will be generated.	Normal	Pass
22	Distance collection	A large, scrollable distance matrix is generated from the data inputted by the user in the Network detail collection form	The value '20' will be entered as the value for the total number of nodes.	A 20x20 scrollable distance matrix will be generated.	Normal	Pass

23	Distance collection	Matrix fields update in real time to show symmetry when a value is entered in one side of the matrix.	The number '32' will be entered into a field in the matrix	The corresponding field will also update with the value 32.	Normal	Pass
24	Distance collection	Program identifies where the user needs to amend the mistake in the input.	The string "asdf" will be entered into a field in the matrix	"Fix" should be displayed next to the string.	Normal	Pass
25	Distance collection	Check that system does not allow user to continue if they haven't filled in the matrix.	Next button will be clicked when no field have been filled in.	An error message will be displayed.	Erroneous	Pass
26	Distance collection	Check that system allows valid input for all matrix fields.	The number '1' will be entered into all fields in the matrix	Input will be accepted.	Normal	Pass
27	Distance collection	Check that system does not allow strings to be entered into matrix fields.	The string "asdf" will be entered into a matrix field	Error message will be displayed and a 'Fix' will appear in the field with the string.	Erroneous	Pass
28	Distance collection	Check that system rejects floats when they are entered as distances between points on the matrix.	The float '4.3' is entered as one of the field in the matrix	Error message will be displayed and a 'Fix' will appear in the field with the string.	Normal	Pass
29	Distance collection	Check that system allows 0 to be entered to represent there is no arc between two nodes.	'0' will be entered in random fields in the matrix	Program will accept these values.	Normal	Pass
30	Distance collection	Check that only one 'Fix' is displayed in each field containing a user mistake when next button is clicked multiple times.	"No" will be entered in one field and then the next button will be clicked 4 times	Only one 'Fix' will be displayed in the field with the error.	Normal	Pass
31	Report display	Check to see Dijkstra's	Algorithm will be	I will get the same	Normal	Pass

	window	algorithm works with a connected graph and is displayed in report	run by hand and check with the answers given by the system in the end report	answer by running the algorithm by hand as the computer		
32	Report display window	Check to see Dijkstra's algorithm works with a graph with minimal connections between nodes and is displayed in report	Algorithm will be run by hand with a network subject to these conditions and compared with the computer running the algorithm on the same network	I will get the same answer by running the algorithm by hand as the computer	Normal	Pass
33	Report display window	Check to see Transportation algorithm works with a connected graph network when the problem entered is balanced.	Algorithm will be run by hand with a network subject to these conditions and compared with the computer running the algorithm on the same network	I will get the same answer by running the algorithm by hand as the computer	Normal	Pass
34	Report display window	Check to see Transportation algorithm works with a connected graph network when the problem entered is not balanced. I.e. the total supply value > total demand value	Algorithm will be run by hand with a network subject to these conditions and compared with the computer running the algorithm on the same network	I will get the same answer by running the algorithm by hand as the computer	Boundary	Pass
35	Report display	Check to see Transportation	Algorithm will be	I will get the same	Boundary	Pass

	window	algorithm works with a connected graph network with the supply and demand values set up so that a degenerate solution is produced	run by hand with a network subject to these conditions and compared with the computer running the algorithm on the same network	answer by running the algorithm by hand as the computer		
36	Report display window	Check to see Transportation algorithm works with a minimal connected network when a balanced problem is entered	Algorithm will be run by hand with a network subject to these conditions and compared with the computer running the algorithm on the same network	I will get the same answer by running the algorithm by hand as the computer	Normal	Pass
37	Report display window	Check to see Transportation algorithm works with a minimal connected network when an unbalanced problem is entered.	Algorithm will be run by hand with a network subject to these conditions and compared with the computer running the algorithm on the same network	I will get the same answer by running the algorithm by hand as the computer	Boundary	Pass
38	Report display window	Check to see Transportation algorithm works with a minimal connected network with a problem that causes a degenerate solution	Algorithm will be run by hand with a network subject to these conditions and compared with the computer running the	I will get the same answer by running the algorithm by hand as the computer	Boundary	Pass

			algorithm on the same network			
39	Report display window	Check to see if you can quit and return to main menu from the report display window	Quit button is clicked	System should return to main menu.	Normal	Pass
40	Report display window	Check that the save and quit button performs it's function	The save and quit button will be clicked.	Program returns to main menu and the appropriate network details are stored in the multiple database tables.	Normal	Pass
41	Report display window	Check to see if circular network does not display when the total number of nodes in the network is greater than or equal to 20	A 20x20 matrix will be filled in.	Circular network is not displayed.	Normal	Pass
42	Report display window	Check to see if 'bipartite' network image is scrollable when the number of supply or demand nodes in the network is greater than 9	A 20x20 matrix will be filled in with 5 supply nodes and 15 demand nodes	Graph will contain the information on the network but will also be scrollable.	Normal	Pass
43	Select report window	Check that you can return to the main menu from window using the back button.	Back button will be clicked	System returns to the main menu.	Normal	Pass
44	Select report window	Check that report can be highlighted when clicked on in the treeview menu	A report into the treeview will be clicked	Report should be highlighted.	Normal	Pass
45	Select report window	Check that you can delete selected record in treeview but clicking the delete button	A report will be selected and the delete selected report button will be clicked on.	Report should be deleted.	Normal	Pass
46	Select report	Check that you can view	A report will be	The report, along	Normal	Pass

	window	selected report by clicking the display selected report button	selected and the display selected report button clicked.	with the graphs should be displayed to the user.		
47	Select report window	Check system alerts user with an error message if the user tries to view a report but has not selected anything in treeview.	The display selected report button is clicked when no report is selected	An error message is displayed to the user.	Erroneous	Pass
48	Select report window	Check an error message is displayed when user tries to delete a report but there are no reports left.	The delete selected report button is clicked when all reports have been deleted (leaving no reports left to be deleted)	An error message is displayed to the user.	Boundary	Pass
49	Select network structure window	Check that you can return to the main menu from window using the back button.	Click the back button	System returns to the main menu	Normal	Pass
50	Select network structure window	Check that records can be highlighted when clicked on in the treeview menu	Click on a network structure in treeview	Structure is highlighted	Normal	Pass
51	Select network structure window	Check that you can delete selected record in treeview by clicking the delete button	The delete selected structure button is clicked	Report is deleted	Normal	Pass
52	Select network structure window	Check system alerts user with an error message if the user tries to continue with a report but has not selected anything in treeview.	Click use selected structure button when there is no network structure selected	Error message is displayed	Erroneous	Pass
53	Select network structure window	Check an error message is displayed when user tries to delete a report but there are no	Delete selected structure button is click when there	Error message is displayed	Boundary	Pass

		network structures left.	are no network structures left			
54	Select network structure window	Check network structure is loaded into Network detail collection window	Existing network structure is selected and use selected structure button is clicked	Selected network structure is loaded into network detail collection window	Normal	Pass
55	Select network structure window	Check network structure is loaded in Distance collection window.	Same as in test 55 except structure of network is loaded into adjacency matrix	Saved data is loaded into adjacency matrix	Normal	Pass

Testing documentation: (see written test document).

Appraisal

Requirement Number:

1a. The user should be able to input their desired network into the program as an adjacency matrix and all details about the nodes on the network through a series of graphical forms.

i. Adjacency / distance matrix should be scrollable to allow the user to input all parts of the network.

This requirement has been met. The user can scroll any matrix (as long as it isn't too large) using the vertical and horizontal scrollbars.

ii. Title of the network should be displayed above the matrix (note: title should not scroll with the matrix.)

This requirement has been met. Any valid title that the user enters in the previous form is displayed above the matrix. In addition this title does not scroll with the matrix.

iii. This process should be simple as the user does not need to know they are inputting data into an adjacency matrix.

This objective has been met. Miss Hudson commented on the ease of inputting data into the matrix due to the symmetry of the table ensuring the user only has to fill in half. She also expressed how difficult it would be to simplify this process any further.

b. System should compute the shortest path from each supply node to every demand node.

Requirement has been met. The system calculates the shortest path using Dijkstra's shortest path algorithm. In addition the shortest paths are displayed in the final report. This process uses the data the user enters into the adjacency matrix.

c. System should generate a report that explains the computations that have been carried out.

i. System should perform the four steps of the transportation algorithm

All requirements have been met (1, 2, 3, and 4). The system uses the weight of the shortest paths calculated by Dijkstra's algorithm in conjunction with the details collected from the user about the number of supply nodes and number of demand nodes, in addition to the cost per unit distance collected in the four parts of the algorithm. Each step in the process is recorded in the report and displayed to the user after the computation has been performed. If the solution generated by the North-West corner method is optimal then the stepping stone method (point 4) will not be performed. This has been tested against checks made using manually calculated algorithms.

ii. Algorithm should stop if an optimal solution cannot be found.

Requirement has been met. If the transportation algorithm loops more than ten times then the algorithm will stop and the initial solution will be return to the user. This is done so that the program does not continue in an infinite loop.

2a. System should generate a report that explains the computation that has been carried out.

Requirement has been met. An extensive report is generated by the system that explains all parts of the computation carried out by the computer. The report steps through each algorithm explaining the important choices that have been made – much like the workings a human would do when performing the same algorithm.

i. Report should contain the conclusions obtained from the computation performed by the system.

Requirement has been met. The conclusions obtained from each algorithm are displayed in the report. This includes the number of items to transport from supply node to each demand node which is displayed right at the very top of the report (If the user does not want to read through all the workings) and at the end of the report.

b. System should graphically display the network the user has inputted alongside the report generated in two forms.

i. Bipartite graph form.

Requirement has been met. A graph containing two sets of nodes – one set containing all of the supply nodes and the other containing the demand nodes, side by side, is displayed by to the user. This graph is generated from the data the user inputs into the distance matrix.

1. Bipartite graph should be scrollable when there are a large number of nodes.

Requirement has been met. If the number of supply nodes or number of demand nodes (or both) is greater than nine then the scrollbar at the side of the graph becomes active and allows the user to scroll the canvas widget to view the entire graph.

ii. Circular graph form.

Requirement has been met. For any network with fewer than twenty nodes a circular network is displayed alongside the report showing the relationship between each node. (Graph is colour coded. Red for supply node, blue for demand node.)

1. Circular graph should not display if the number of nodes in the network is greater than twenty because the graphic will get too crowded

Requirement has been met.

3a. Allow user to save a report and network structure graphs.

Requirement has been met. Once all computations have been carried out on the data collected from the user and the report has been displayed, the user is allowed to save the report. By saving the report the network structure is also saved allowing it to be reused at a later date.

b. User should be able to load saved reports along with their network images.

Requirement has been met. User is allowed to select an existing network report and view its content. The graph(s) that were generated when the algorithms were initially run will also be displayed.

4a. Allow users to load an existing network structure that has been saved and use it for a new program.

i and iii.

Requirement has been met. When a network is saved by the user, it can be accessed at another time (even after the application has been closed and reopened) and all properties apart from the size of the network and the number of supply and demand nodes there are can be fully edited. This means changes to the supply and demand values can be adjusted as can the distances between each node on the network. The details for the distances remain filled in as to save the user time if they only want to change the supply and demand values for a large network.

5a. System works together as a whole

i. Individual modules and part of the system work coherently with no errors or discontinuities.

Requirement has been met. This will be discussed in more details below in the user feedback.

ii. End user will try the system and give feedback.

Requirement has been met. This will be discussed in more detail below in the user feedback.

iii. Outside user with no experience of the system should trial the piece of software to see if it is intuitive and easy to use. They will then present feedback.

Requirement has been met. This will be discussed in more detail below in the user feedback.

b. Provide a graphical interface that is intuitive to use and flows smoothly between windows. It should be possible to access every part of the system through the GUI without having to restart the program.

i. All buttons should perform the action that it is intended to do and labelled appropriately.

Requirement has been met. This has been rigorously tested in the previous section and the people who have tried the system have not commented about any faults concerning the functionality or naming of the buttons.

ii. Drop down menus should contain clickable items.

Requirement has been met. As mentions in (i) this has been tested extensively and no users have commented about it not working.

iii. Treeview structures should be clickable and interactive using buttons.

Requirement has been met. This feature has been tested in the testing section. Users can click on single items in the treeview and perform actions based on what has been selected.

iv. Allow users to use scrollbars.

Requirement has been met.

c. System should not terminate, or in other words not crash.

i. All points of system contact with the user should be validated so that the algorithms that perform computation on the user data will not falter.

Requirement has been met. System has never crashed due to invalid inputs either by me (during the testing process) or the users that have trialed the system.

ii. All algorithms should not cause errors in the system.

Requirement has been met.

User Feedback.

I have obtained feedback from the end user, Miss Hudson.

Miss Hudson was impressed to see that I had met all of the objectives and incorporated all of the points we made during our discussions. She commented specifically on how intuitive the system was to use and how data was very easy to enter into the system. She also liked how the matrix that the user entered the distances between nodes on the network was symmetrical. This allows the “student” to quickly find the results of their calculation.

Miss Hudson was pleased to see that previously created networks could be loaded back into the system and modified. For example, if the conditions of the problem have changed such as the number of items demanded by a shop has increased. She commented that this was an important feature as it allowed students to experiment with how the supply/demand values impacted the result of the algorithms. The fact that the reports generated in the past could be loaded and viewed again was noted and Miss Hudson liked the idea that students could copy the report into a word processing file and send it to the printer. She said it would make it easier to compare their work with the correct solution produced by the system.

The graphs that were identified by Miss Hudson in the initial stages of the project were commented on saying that it was useful that the user could see how the network could be laid out. She said it allowed students to show how distance matrices are a powerful way of representing a network which is a key learning point in the D1 module.

Furthermore, Miss Hudson was taken with the system validated user input especially with the adjacency matrix. She said it was very useful when making sure you have entered the correct data into every cell in the matrix. She said this feature would be very important when the matrix is very large.

Another prominent point made by Miss Hudson, was about the detail of the report in documenting every aspect of how the problem was handled by the computer, including points about degenerate solutions and unbalanced problems. She said it would be a really useful resource for students next year as the algorithms followed by the system are the same as the ones in the textbook. She liked how the system “shows how complex maths is used in the real world.”

Overall Miss Hudson commented “The system ran well, without any crashes perceived or any error messages generated. Popup windows for entering data worked well and data entry seemed intuitive.” She said further that the system was consistently hitting all of the objectives laid out in the earlier part of the report and that the project had successfully been completed.

Analysis of Feedback and Constructive Criticism

When Miss Hudson was asked to give some constructive criticism she said the following. “As the requirements of the system were quite strictly written vis-à-vis the display of networks the system has met the requirements, however the extent to which the network images produced would be useful to the end user is limited as the images generated may not match given images of the network and there did not appear to be any way for the user to identify which node was which or move the nodes so that the generated picture matched a given image.” This is true because the image of the network is isomorphic to that of what the user might have had in mind when entering the network into the program. In addition, the nodes were not labelled in order not to make the image too congested however, if the program should be developed further then the user should be able to explore the graphs in more detail.

Miss Hudson was obviously happy with the work I had done in developing the project from start to finish and was especially impressed with how easy it was to enter the details of the problem into the system. No negative comments were presented about the main functionality of the system and I was told that the report read well and provided enough information about each step of the algorithms. Miss Hudson did mention how the working values could have been written about in the report to help students learn more about the “nitty gritty” of the algorithms but she also understood that the project is a simulation of a business solution.

She was also happy with the interface of the whole system and said it was much better than a program solely based on a command line.

From the discussion I picked up some ideas that could be explored further and they are discussed in the next section.

Possible extensions

From a teaching perspective it would be helpful if the user could see the network graphically with all of its working values and indexes. It would also be useful to be able to progress through each step of the algorithm with a slider or arrow buttons to see in what order steps happen. This would enable students to quickly grasp the concept as they can repeat steps that they do not understand. This would equally be possible for the transportation algorithms.

Although the final report is structurally sound, would be more visually appealing with the addition of tables instead of the ones created using the ‘|’ symbol. The entry widget makes it very difficult to format objects such as text tables and sometimes they can become misleading or confusing to the reader. It would be good to have tables that could easily copy into other documents along with the text from the rest of the report.

A further function of the report window is the ability to print from the inside of the application. At the moment potential users have to copy the report into a word processing document and print it from there. This is additional hassle for the user and could be avoided with the addition of a print button. If this feature was to be implemented the graphs displayed in the report window would have to be converted into images as they are currently displayed in a canvas. This design choice was made solely because of the portability of the project. In order for the system to run on a wide range of platforms it must not contain as little external modules as possible. This is because most installations of Python will not have the ones required for image handling. These modules tend to be quite large because image handling can become complex.

I would like to introduce some sort of ‘collaboration mode’ using the TCP/IP as this would enable multiple users to send projects to each other and possibly work on the same problem together. I think this would increase the rate of productivity in a business setting (which is what this project is designed to emulate).

From the school perspective

Miss Hudson suggested that clearing identifications of saved networks when choosing them from the treeview widget should be used both when selecting a report to load and choosing a network structure to build the rest of your project upon. I agree with this because if you have multiple projects with similar titles all made by the same person it can get confusing if the company has not written down when their project was made. However, the date created that is displayed alongside each network, is a key way of determining between networks. If the date when the project is created is not identified by the company or individual who made it. This process could be made easier by

allowing the users to see the graph of the network alongside it's key details or by making the treeview cascading to display even more information about the network.

Furthermore, the graphs displayed alongside the report in the report display window could be labelled with the node numbers along with the cost for travelling between each arc. This should be done only on graphs that are small otherwise it would become too crowded.

Finally, a student who tested the application pointed out that there was no 'About' or information window that you find in most high end programs. Although this is a good observation and is definitely something that should be done as an extension for the project, it is not imperative, as a teaching tool. Only in a few cases would it be useful, for example, in the scenario when a new teacher starts using the program.

Conclusion.

Analysing the feedback from Miss Hudson and the two students in lower sixth it is apparent that this system will be a useful teaching resource because it demonstrates how the algorithm works and in what environment it can be implemented. This positive feedback is proof that I have achieved what I set out to achieve.

There are some small modifications that could be made to the program in the future to improve the human-computer interface. Although there are some developments that could be made to increase the functionality, they are not necessary and the minor improvements will not take long to implement.

Overall I think this project has been a success.

Technical solution (In project code document.)