Gesture-Based Input for Drawing Schematics on a Mobile Device

Daniel Chivers, Peter Rodgers
Department of Computer Science, University of Kent, Canterbury
dc355@kent.ac.uk, P.J.Rodgers@kent.ac.uk

Abstract

We present a system for drawing metro map style schematics using a gesture-based interface. This work brings together techniques in gesture recognition on touch-sensitive devices with research in schematic layout of networks. The software allows users to create and edit schematic networks, and provides an automated layout method for improving the appearance of the schematic. A case study using the metro map metaphor to visualize social networks and web site structure is described.

Keywords--- **Gesture-Based Input, Sketching Input, Metro Maps, Schematics, Mobile Device**.

1. Introduction

Visualizing complex, interconnected information using a metro map is a common metaphor. Data from many application areas has the potential to be visualized in this way, for example metro map diagrams for astronomical data and web trends, are shown in [15]. Other types of data drawn as a metro map, such as thesis structure and a business plans can be found at [8]. Typically, these examples have been drawn by hand using vector graphics applications, requiring a great deal of time and effort. An alternative approach is to use an existing metro map and change the labels to make the new data fit the existing structure, as can be seen in [14], which is based on the London Underground. However, this method is restricted to data sets that can fit into an existing layout. These examples are evidence that users want to visualize data using a metro map metaphor but the difficulty in creating this style of diagram by hand means that it has not been explored to its full potential.

Mobile touch based devices have a great potential for creating schematics as they allow users to conveniently and effectively capture complex ideas in a clear, easy to read schematic at any time. With this in mind our aim was to develop a piece of software to meet these needs.

The application we have developed, SchemaSketch (see Figure 1), facilitates the fast drawing of metro map style schematics and allows the user to create them in such a way that the schematic contains information about the underlying connections. This makes it much easier for the user to reposition stations, as all lines remain connected when nodes move. The software can be downloaded from:
http://cs.kent.ac.uk/projects/schemasketch/iv2011.

As the diagram contains structural data, it is also possible to perform automatic layout techniques to attempt to improve the schematic. An example of this automatic layout has been implemented into SchemaSketch. Inspired by the methods developed in [7], the application attempts to position nodes to satisfy a series of criteria based upon aesthetic quality of the diagram.

We have developed SchemaSketch to run on portable Android devices, which allows the user to draw their ideas whilst away from a computer. For example, whilst on public transport or for workers out in the field where a larger computer may be impractical. In addition, touch screen devices, such as mobile phones, are commonly used which makes the system widely accessible.

There are few current applications that support schematic drawing of metro maps. Example applications which claim to, such as [2] and [5], are general purpose vector based graphic applications and do not allow easy modification of drawn schematics, as they do not preserve connectivity information.

We have examined previous work in gesture based input [1], sketch recognition and beautification of hand

Figure 1: SchemaSketch running on an Archos 7 HT device using Android v1.5

drawn sketches [6][9][12] in order to decide on an efficient and intuitive input mechanism for drawing schematics. Although full sketch recognition can provide more advanced functionality by supporting a variety of symbols, it comes with a performance overhead in recognition, as well as a mechanism to determine when the user has finished one symbol and moves onto the next. Mechanisms to circumvent this include a waiting time between pen strokes [12], but this hinders the input flow of the user. Simpler gesture recognition, where each gesture corresponds to a symbol, can provide the functionality we require and will ensure the user is not disturbed by workflow pauses.

Previous work on automatic metro map includes work that applies a force directed approach [13]. Other research uses a series of criteria to measure aesthetic elements of the schematic, such as line straightness, octilinearity and line length between nodes [7][10]. User tests, such as those carried out in [4], have shown that diagrams that conform strongly to combinations of these criteria have increased readability. Although these papers use different methods for optimisation, we have chosen to implement a method inspired by that in [7] due to its flexibility, as aesthetic criteria can be modified relatively easily.

In the remainder of this paper, Section 2 describes in detail the user interface of SchemaSketch, as well as implementation details of gestures, connections, and labels. Section 3 describes the layout and optimisation techniques including implementation of layout criteria used by SchemaSketch to optimise drawn schematics. Section 4 describes our results and provides some examples of the software in use, as well as discussing current problems. Section 5 outlines potential future work. Finally, Section 6 gives our conclusions.

2. Interface and Implementation

We have developed a software tool that allows a user to hand draw schematics on a touch based mobile device. The application, SchemaSketch, has been written to run on mobile devices running Google's Android operating system. It has been developed on v1.5 but is compatible with newer releases and will accommodate a variety of screen sizes. It provides two operating modes for creation of schematics, draw (input) and move (modification), which can be toggled in the main menu. These two modes of operation are described in Sections 2.1 and 2.2 respectively.

2.1. Draw Mode

This mode allows schematics to be created by using sequences of gestures to input objects (see Section 2.3 for details on gesture recognition). The following list describes the objects that can be drawn in the schematic.

- *Station* – SchemaSketch provides two different types of station object, a circular station and a line station. These two stations are visually different and are input using different gestures, however they are treated the same from a connectivity perspective. Circular stations are intended to be used for representing junction stations, whereas line stations are intended for use in situations where the station has two or less incident edges. Stations are used to connect together multiple edges, of the same or different colour. A label can be added to either type of station object.
- *Edge* - Provides a connection between two stations. SchemaSketch provides support for different coloured edges to be drawn. Parallel edges (of a different colour) can be drawn between two stations. Edges allow the formation of metro lines. Metro map lines are considered to be several connecting edges of the same colour.

Whilst in this mode, the menu provides the following options:

- *Eraser* – Changes the pen to an eraser pen that will remove everything drawn over.
- *Undo* – Undo the last action.
- *Colour* – Change the colour of the Edge.
- *Clear All* – Clears all objects from the screen (requires confirmation).
- *Mode: Move* – Switch to the move mode (see Section 2.2).

The draw mode also allows the user to add labelling to the schematic, see Section 2.5 for details.

2.2. Move Mode

This mode allows the manual modification of a drawn schematic, by enabling drag and drop functionality for stations and labels.

Whilst in this mode, the menu provides the following options:

- *Eraser* – Changes the pen to an eraser pen that will remove everything drawn over.
- *Undo* – Undo the last action.
- *Optimise* – Uses a hill climbing multicriteria optimiser method to produce a more optimised schematic (see Section 3).
- *Load/Save* – allows the loading and saving of drawn schematics to a file.
- *Mode: Draw* – Switch to the draw mode (see Section 2.1).

The draw mode also allows the user to manually move labels on the schematic, see Section 2.5 for details.

2.3. Gestures

Multiple gestures are used to input the various objects defined in Section 2.1. Gestures are recorded as a sequence of time-stamped points. When the user makes a gesture, SchemaSketch will attempt to recognise the gesture based on a series of rules.

- **Minimum direct length to be classified as an edge.**
  Direct length refers to the distance between the start and end points of the gesture. For a gesture to be an Edge object, this distance must be greater than 45 pixels.

- **Minimum straightness to be classified as an edge.**
  The straightness of a gesture, *G*, is calculated using Equation 1.

$$straightness(G) = \left(\frac{dist(G_{Start}, G_{End})}{actualLength(G)} \right)$$

Equation 1

  actualLength(G) is calculated using Equation 2. The straightness calculation will produce a value between 0 and 1. A value of 1 is a perfectly straight line. For a gesture to be classed as an edge, *straightness(G)* must be greater than 0.9.

If a gesture passes the minimum direct length test and minimum straightness test, it can be classified as an edge, otherwise it is potentially a station. Differentiating between the two types of station is performed by the three following rules.

- **Minimum actual length to be classified as a station.**
  Actual length refers to the length of the gesture if it was straightened out, and is calculated using Equation 2, where n is the number of points in the gesture and p_i is the i^{th} point along the gesture.

$$actualLength(G) = \sum_{i=1}^{n-1} \left(dist(p_i, p_{i+1}) \right)$$

Equation 2

actualLength(G) must be greater than 10 pixels for the gesture to be a station. This means any gesture shorter than 10 pixels will not be recognised and nothing will be added to the diagram. This is useful for discarding unintentional screen touches.

- **Minimum straightness to be classified as a line station.**
  The straightness is once again checked using Equation 1 and if *straightness(G)* is greater than 0.5 then it will be classified as a line station. Although stations and edges are both straight lines, edges require a higher *straightness(G)* value because the longer a gesture, the easier it is to get a high *straightness(G)* value.

- **Minimum average radius to be classified as a circular station.**
  If *straightness(G)* is less than (or equal to) 0.5 this last check is performed to identify a circular station gesture. We calculate the average radius of the shape (we know the shape is curved, as *straightness(G)* is low). First we calculate the centre point of the gesture, by averaging x and y co-ordinates of all points. Using this, we can calculate the average radius using Equation 3, where n is the number of points in the gesture and p_i is the i^{th} point along the gesture.

$$radius(G) = \frac{\sum_{i=1}^{n} \left(dist(G_{centre}, p_i) \right)}{n}$$

Equation 3

  If *radius(G)* is greater than 10 pixels, this gesture can now be classified as a circular station, otherwise the gesture will not be recognised and nothing will be added to the diagram.

These rules result in stations being drawn either by a short, straight gesture or a circular shape with start and end points close together. Edges are drawn by a long, straight gesture.

2.4. Connections

SchemaSketch connects edges to stations based upon location of the gesture. Starting an edge in the vicinity of a station will connect that edge to the station; conversely, drawing a circular station around an unconnected end of an edge (or multiple edge ends) will connect the edges to the newly drawn station. Line stations will also connect to multiple edges provided they are close enough. When the user is drawing, any object that an edge can connect to will display a highlighted "hotspot" which is the object's connection radius.

Drawing a line station that intersects an edge will insert it at that point along the edge, or if the station is close enough to a free end, it will attach to that.

2.5. Labelling

Labels can be added to stations whilst in draw mode. Touching on a station will open a text input dialog allowing the user to enter a label name.

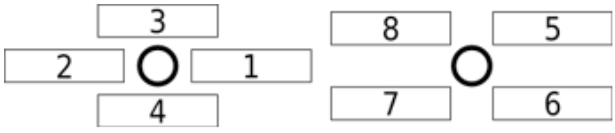

Figure 2: Possible positions for labels relative to their parent node. The values indicate the priority of each position

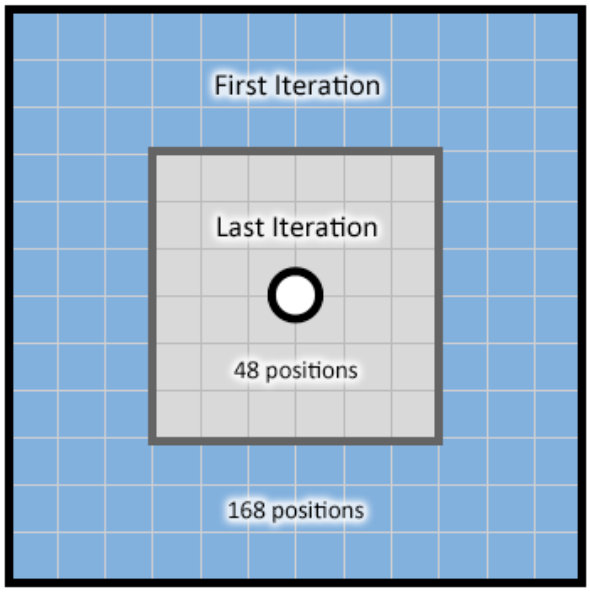

Figure 3: Distance nodes are moved during first and last optimisation iterations

Whilst in move mode, labels can be moved manually by dragging them around their parent. There are eight positions in which a label can be placed, these positions relative to the parent node are *North, North-East, East, South-East, South, South-West, West* and *North-West,* as shown in Figure 2. A label will initially be placed in position 4 (*South*).

3. Layout and Optimisation

SchemaSketch includes a multicriteria optimiser to produce more easily read schematics. This optimiser is inspired by work performed in [7]. SchemaSketch's specifics are outlined in Section 3.1. We have chosen to use a subset of the criteria used in this prior work for optimising node and label positioning, as many of these criteria are not appropriate for our case.

3.1. Optimiser

SchemaSketch's optimisation process uses a number of iterations of station movements, currently set to 10. During each iteration, each station is examined in turn and placed in available grid positions around it. As well as moving stations, clustering methods (based on those that group lines in [7]) are applied to move groups of stations.

The optimiser uses a cooling method to attempt to reduce the number of position checks the algorithm performs. Figure 3 shows the distance nodes are moved; during the first iteration, each node is tested in all positions up to six squares away (168 positions), this distance decreases linearly down to three squares (48 positions) during the last iteration. Fractional values are rounded up to the nearest integer. Stations are not permitted to go beyond the limits of the screen.

At each station or cluster movement, a series of criteria are calculated and summed to produce a value representing a measurement of the aesthetic quality of the schematic; the lower this value the better - a value of zero indicates all criteria have been satisfied. The criteria used for this value are explained in Section 3.2. This value is recorded for each position the station or cluster is moved to, and once all positions have been tested it is moved to the position that yielded the lowest criteria value (indicating the best aesthetic quality).

After optimisation of the stations, the labels are examined in turn to determine their best position. They are tested in a single step. Figure 2 shows the order in which label positions are considered (from 1 to 8). At each position, the label criteria are calculated and labels are moved to the position with the lowest summed criteria value. The criteria used in the label positioning stage are explained in Section 3.3. Testing the labels in order of position preference ensures that if multiple positions have the same summed criteria value, it will be placed at the first found.

3.2. Station Criteria

This section explains the station positioning criteria used to determine the quality of the layout.

The criteria values often have a squared component, this is to ensure that the worst criteria are penalized more strongly. For criteria such as line straightness, this also provides the desirable behaviour that fewer sharper bends are penalised more than multiple smaller bends.

The calculations produce values which vary greatly by criterion (up to many orders of magnitude different), this is because the criteria are measured naturally on different scales. Using these unweighted values would put more emphasis on the criteria that were naturally larger, it is therefore necessary to weight the values so that they can be comparable.

Basic weighting involves multiplying the unweighted value by 1 over the maximum possible value; this constrains the value to between 0 and 1. However, it would be incorrect/not possible to scale all criteria in this way as they may never reach the maximum in practice, or alternatively they may not have a maximum. Therefore, to calculate weightings, we created a series of example graphs and recorded the unweighted criteria values. We averaged the values and used the inverse of the result as the weighting.

The first stage of optimisation is to snap all stations onto a grid. This is accomplished by examining all stations and moving them to the nearest grid position. If multiple stations contest a grid position, the original position of contested stations will be checked, and the closest one moved. This grid has multiple advantages to simplify the optimisation process. 1) By using a grid we can minimise the number of possible station positions that we are required to check, greatly speeding up the process 2) By moving stations to fit to a grid, we get the benefit of helping the octilinearity of edges between stations 3) Station/station occlusion checks are not

necessary providing the grid spacing is greater than the station's bounding box diameter.

The five station criteria that we use are:

1. **Octilinear Layout.** This criterion is to keep the graph as octilinear as possible; this means keeping all angles at multiples of 45°. The octilinear layout criterion sums the measure for each edge. The measurement for an edge is a square of the difference in angle from the nearest multiple of 45°.
2. **Minimise Edge Crossings.** Edge crossings should be kept to a minimum. The edge crossings criterion is measured by checking all pairs of edges for an intersection, and then summing the number of intersections and squaring it.
3. **Line Straightness.** Lines, a group of connected edges that share the same colour, should be as straight as possible and when bends are required they should be as small an angle as is attainable. The line straightness criterion sums a calculation for each line bend. The line bend calculation is the square of the angle the bend makes, penalising a line more if it contains sharper bends.
4. **Equal Edge Lengths.** Edges between stations should be of equal length, and they should also try to achieve a desirable target length, *t*. This length has been defined as three grid squares. The criterion sums a calculation for each edge. The edge calculation squares the difference between the edge length and the desirable length. As we are using an octilinear layout for the graph, we must account for diagonal edges. Because of this, we adjust the value of *t* to be three times the diagonal distance across a grid square when necessary.
5. **Occlusions.** Stations and edges should be positioned in a way that they do not obscure any other part of the schematic. Possible occlusions include station/station, station/edge, and edge/edge. Because the optimiser is based on a grid positioning system, it cannot place one station on top of another and therefore station/station occlusions cannot happen. Also, the *Minimise Edge Crossings* criterion includes edge/edge occlusions and so this need not be dealt with here. This means that this criterion only needs to check for station/edge occlusions. The number of indirectly connected station/edge occlusions is counted by checking each possible pairing for an intersection, and this result is squared to create the occlusion criterion.

3.3. Label Criteria

This section explains the criteria used for label positioning. As label criteria are not included in the main layout of the schematic, they do not require weighting. In terms of priority, labels will only be placed in a consistent position when it is possible to do so without introducing occlusions. The two label criteria are defined as following:

1. **Occlusions.** Labels should not overlap edges or other labels. This criterion is measured by calculating the label bounding box and checking against each edge and other label for an intersection. The number of intersections is counted to create the value.
2. **Position Consistency.** It is desirable for adjacent labels to be similarly positioned. This is achieved by penalising labels that are not in the same position as their neighbours. All labels with exactly one or two neighbouring stations are checked and given a scoring based upon their position consistency (one point per difference in label positioning to both other stations). The value is the sum of the consistency values for all such stations.

The work described in [7] uses additional criteria, but these have been omitted because they are application area specific or ineffective in our model. In particular, some criteria are designed for use with data that has a spatial component such as metro maps that contain geographic relationships between the stations. These criteria include those that prevent large distortions and changes in topology, so retaining some geographical accuracy. Another criterion used by this previous work, angular resolution, which maximizes the angle between incident edges at a station has not been used. However, its effect is also performed by the octilinear criterion, so there is no need for a separate calculation.

4. Results and Examples

The following sections provide examples of data sets that can be displayed in a metro map style using the application. The examples shown also illustrate the optimisation method.

As illustrated in [15], there is clearly a use for software that allows the drawing and optimisation of metro map style schematics using abstract data, as the examples that can be found there have been time-consumingly drawn by hand.

SchemaSketch is still at the proof-of-concept stage, hence it cannot currently replace a vector graphics program for complete creation of metro map style schematics because of the aesthetic appearance of the final diagrams. However, we believe the software illustrates the potential for saving users considerable time by allowing fast and easy drawing of schematics. In addition, the built in optimiser can further aid users by helping them optimise their graphs according to a set of criteria. In Section 5 we discuss the export of drawn schematics to multiple file formats, which would allow the user to switch to a vector graphics software package for further editing.

4.1. Social Networks as a Metro Map

Here we demonstrate how the metro map metaphor can be used for the visualization of social networks. Stations are used to represent individuals, and different coloured lines correspond to the type of relationship between them. Figures 4 and 5 show an example of a

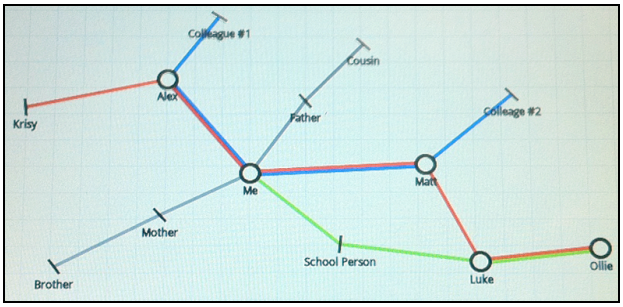

Figure 4: Social network before optimisation

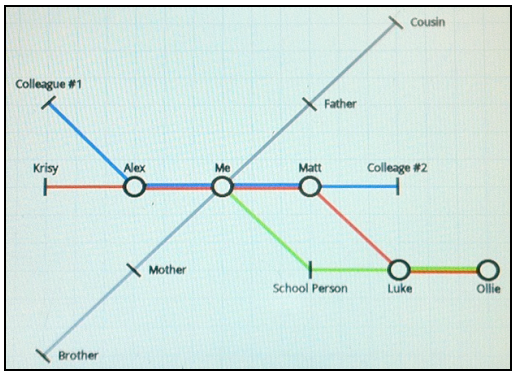

Figure 5: Social network after optimisation

social network drawn as a metro map, before and after optimisation. Here we show the family and friendship relationships centred on one individual.

Another example of where this type of visualization may be desired is for the display of personnel structure within a company or department, for example academic staff in a research institute can belong to multiple research groups. These can be represented as the coloured lines as shown in Figure 6. This schematic was conceived using SchemaSketch to draw the initial structure and plan an effective layout of the stations (as can be seen in Figure 9); it was then re-created using a

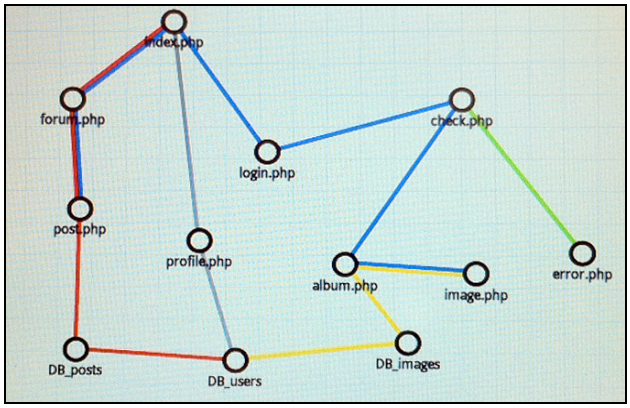

Figure 7: Website before optimisation

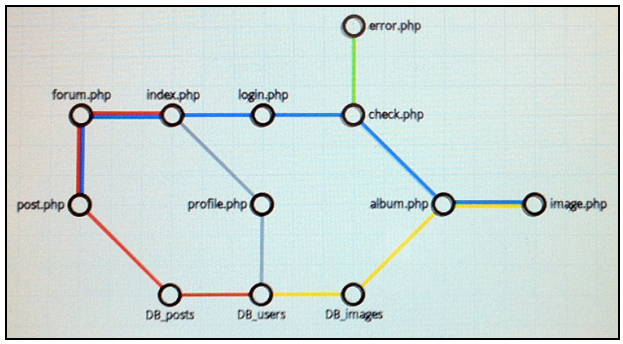

Figure 8: Website after optimisation

vector graphics drawing application.

4.2. Website planning as a Metro Map

As well as social networks, the structure of a website (from either an end user or a developers perspective) can be effectively visualized using the metro map metaphor. Figures 7 and 8 illustrate how a website can be visualized from a developers perspective by representing the individual pages and database tables as stations, and

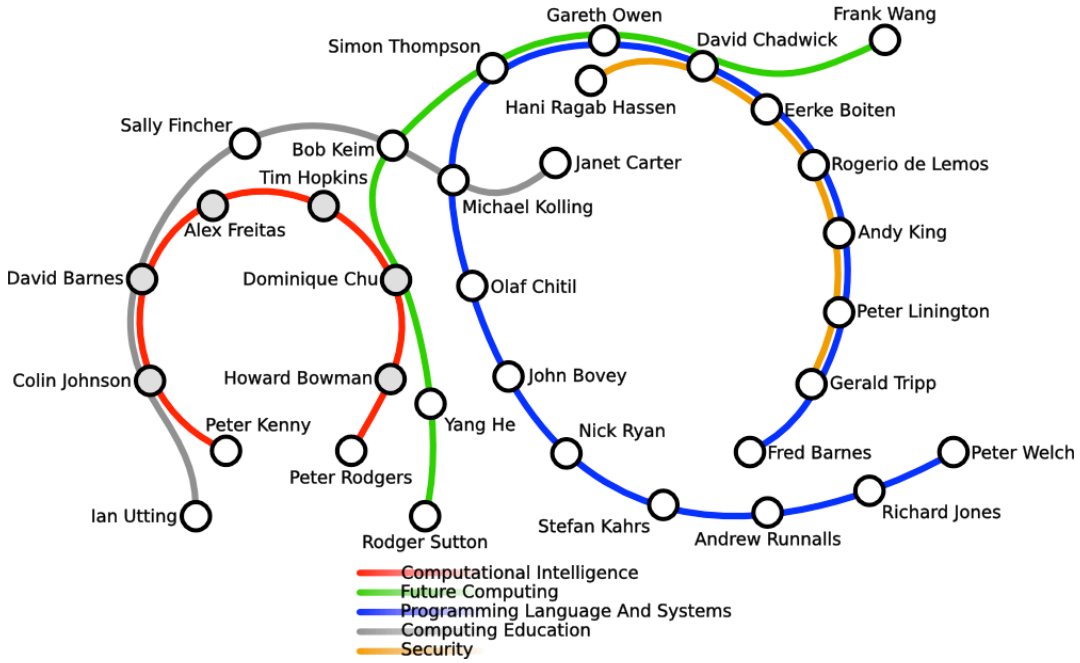

Figure 6: Metro map style schematic showing staff research areas in the University of Kent, Canterbury

the lines as aspects of the system, for example pages that require user authentication.

A designer may wish to plan the pages or services of a website for personal use or to show to a customer. The metro map at [3] illustrates an example of this use in practice. There, the designer of this diagram explains how he struggled to understand how aspects of the system were related when designing a course plan. Designing the system as a metro map allowed him to see the related aspects that could be combined into topics for the course.

In addition to this, a metro map based site diagram could be made into an interactive diagram to allow users to click on the stations to take them to that page, as well as providing a more interesting overview than the commonly used hierarchical text structure.

4.3. Issues with the System

Currently the canvas size is restricted to the size of the screen. This can be problematic as it limits the size of the schematic that can be drawn. A larger, scrollable canvas, and/or a zoom function would be beneficial to users by allowing them to draw schematics that are not limited in size. This would of course increase computational time for optimisation, but we believe this to be a reasonable trade-off. This size limitation problem sometimes manifests when the optimisation method is run. The optimiser will attempt to spread out schematics that are very dense, because it will attempt to normalise edge lengths, and if there is not enough room for the expansion, schematics will remain squashed into the available space. Figures 9 and 10 show an example of where a dense graph has been squashed onto a canvas that is too small for the optimiser to function correctly. The optimiser does not have enough canvas space to be able to move the stations to more desirable positions.

Besides canvas space, it is possible that the optimiser will remain in a state of local minima. A method in which the optimiser is allowed to make changes for the worse (such as Simulated Annealing) may be able to alleviate these local minima problems, but would increase the search space of the optimiser and increase optimisation time.

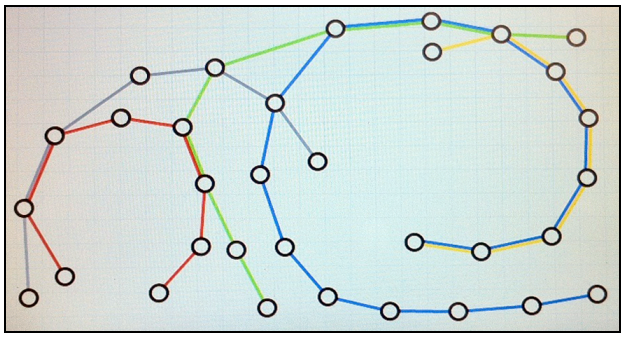

Figure 9: Dense graph before optimisation

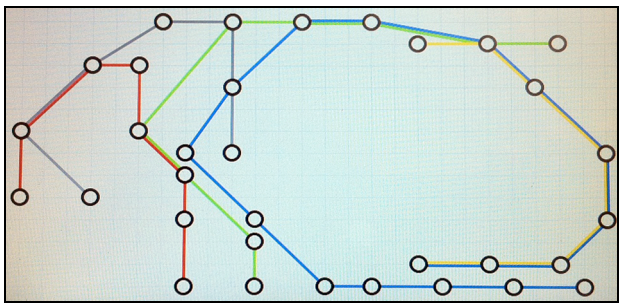

Figure 10: Dense graph after optimisation

5. Future Work

The multicriteria optimiser implemented in SchemaSketch allows the easy addition of new criteria. We plan to introduce new experimental criteria for use in user tests in an attempt to find out how much these affect the readability of metro style schematics. Examples of these additional criteria include symmetry, station balancing, parallel lines and station alignment.

The current version of SchemaSketch supports loading and saving of schematics, but this is limited to a bespoke file type. Future work would allow implementation of a mechanism to allow the user to export the schematic for use in other applications. It would be particularly beneficial to allow export to a vector format, for example *.svg*, so that users could import the diagram into other applications and make further edits.

SchemaSketch is not at the stage at which it could replace a vector graphics program for illustration of metro map style schematics in terms of aesthetic quality. However, with implementation effort, this may be an achievable goal and future projects could involve improving the visual quality of the schematics produced.

We have found that a metro map can be an interesting and practical visualization for any data set that contains multiple items which share relationships. There are many data sets like this around on the Internet, where many items are given “tags” (metadata describing a theme or concept) to relate them to the other items. For example, a metro map schematic could be produced from a set of photographs that have been tagged with metadata (for example, Flickr encourages this form of tagging). It would be possible to create a schematic by using line colours to represent tags and so produce a visualization of how all the items are linked together. Unlike simple tagging, it would be easy to see past the first order relations, which may reveal interesting results.

The application area for SchemaSketch is also currently limited due to it supporting only a small number of symbols. The gesture based input system can be modified to allow addition of new symbols which would allow SchemaSketch to be used for specialist applications. For example, electrical symbols could be introduced to allow the design of electrical schematics. This is a viable example of the particular benefits of a mobile device – an electrician may want to plan out electrical circuits whilst out on call, and it would be

much more practical to use a small mobile device than a laptop.

Mobile devices use much slower processors than desktop computers, and therefore the optimisation process can take a long time to run. Optimisation time also increases rapidly as more stations and edges are added to the schematic. To make it suitable for mobile devices, optimisation techniques can be introduced to minimise processing time, for example when a station is moved all criteria are currently recalculated. It would be possible to optimise this process so that only the required criteria are calculated on the stations that have changed. Stations with degree 2 could also be combined into a single edge with a weighting indicating the number of stations along them, this would greatly reduce the number of criteria calculations.

6. Conclusions

There are no applications which successfully support schematic drawing of metro maps in the style of Henry Beck's classic London Underground design. Nevertheless, data from a variety of areas is suitable for visualization in this manner.

In answer to this problem we have created SchemaSketch, an application that facilitates the drawing of metro map style schematics on Android devices using a gesture based touch interface. SchemaSketch contains information about the underlying graph structure and this allows easy use of automatic layout techniques to optimise the schematic. SchemaSketch includes a multicriteria optimiser which repositions nodes to satisfy a series of criteria based upon aesthetic quality.

Using SchemaSketch, we have investigated the use of the metro map metaphor for diagramming abstract data collections, such as social networks and websites. During these investigations we have found that metro maps can be an interesting and practical visualization for data sets that consist of multiple items which share relationships.

We have demonstrated that there are practical applications for software such as SchemaSketch, and that even at this early stage of the project's life it can greatly aid users who wish to visualize information in this style.

References

[1] Dean Rubine, *"Specifying Gestures by Example"*, in Proceedings of the 18th annual conference on Computer Graphics and interactive techniques, vol. 25, 1991, pp. 329-337.

[2] Edraw Soft, http://www.edrawsoft.com/, accessed 23/02/2011.

[3] The Moodle 2.0 Administration Map, http://www.synergy-learning.com/blog/moodle/the-moodle-2-0-administration-map/, accessed 24/02/2011.

[4] Helen C. Purchase, Robert F. Cohen and Murray James, *"Validating Graph Drawing Aesthetics"*, in proceedings Graph Drawing 1995. Lecture Notes in Computer Science, vol. 1027, 1996, pp. 435-446.

[5] iMapBuilder, http://www.imapbuilder.com/, accessed 23/02/2011.

[6] Isaac Freeman and Beryl Plimmer *"Connector semantics for sketched diagram recognition"*. AUIC '07 Proc. 8th Australasian conference on User interface 64. ACM, 2007.

[7] Jonathan Stott, Peter Rodgers, Juan Carlos Martínez-Ovando, and Stephen G. Walker. *"Automatic Metro Map Layout Using Multicriteria Optimization."* Transactions on Visualization and Computer Graphics, 16(1):101-114, January 2011.

[8] Keith V. Nesbitt, "*Getting to more Abstract Places using the Metro Map Metaphor"*, in Proceedings of the Information Visualisation, Eighth International Conference, 2004, pp. 488-493.

[9] Levent Burak Kara and Thomas F. Stahovich, "*Hierarchical Parsing and Recognition of Hand-Sketched Diagrams"*, in Proceedings of the 17th annual ACM symposium on user interface software and technology, 2004, pp. 13-22.

[10] Martin Nöllenburg and Alexander Wolff, *"A Mixed-Integer Program for Drawing High-Quality Metro Maps"*, in proceedings Graph Drawing 2006. Lecture Notes in Computer Science, vol. 3843, 2006, pp. 321-333.

[11] Maxwell J. Roberts, *"Underground Maps After Beck"*, Capital Transport Publishing, 2005.

[12] Milda Gusaite, E. Kazanavičius and T. Barkowsky, *"Dynamic Scene Analysis and Beautification for Hand-draw Sketches"*, Masters thesis, Kaunas University of Technology, 2006.

[13] Seok-He Hong, Damian Merrick and Hugo A.D. do Nascimento, *"The Metro Map Layout Problem"*, in APVis '04: Proceedings of the 2004 Australasian Symposium on Information Visualization, vol. 35, 2004, pp. 91-100.

[14] Simon Patterson, The Great Bear, http://www.olivercloke.com/simon-patterson-the-great-bear, accessed 23/02/2011.

[15] Ten examples of the Metro Map Metaphor, http://blog.visualmotive.com/2009/ten-examples-of-the-subway-map-metaphor/, accessed 23/02/2011.