International Journal of Software Engineering and Knowledge Engineering

Improving Search-Based Schematic Layout by Parameter Manipulation

Daniel Chivers
School of Computing, University of Kent
Canterbury, Kent CT2 7NF, UK
dc355@kent.ac.uk

Peter Rodgers
School of Computing, University of Kent
Canterbury, Kent CT2 7NF, UK
P.J.Rodgers.ac.uk

This paper reports on a method to improve the automated layout of schematic diagrams by widening the search space examined by the system. In search-based layout methods there are typically a number of parameters that control the search algorithm which do not affect the fitness function, but nevertheless have an impact on the final layout. We explore how varying three parameters (grid spacing, the starting distance of allowed node movement and the number of iterations) affects the resultant diagram in a hill-climbing layout system. Using an iterative process, we produce diagram layouts that are significantly better than those produced by ad-hoc parameter settings.

1. Introduction

Search-based methods for graph drawing have been successfully used for a number of years [18]. Despite taking a relatively long time to produce layouts, they have a number of advantages, including that of targeting a fitness function that explicitly includes layout metrics, so allowing a direct measure of the quality of a graph.

Schematic layout, often called the metro map layout problem, is a variant on the graph drawing problem [21] where a number of aesthetics are present, such as a requirement that edges are restricted to a limited number of angles - typically octilinearity. Search is a common method for attempting to solve the schematic layout problem, perhaps because force directed techniques struggle to meet the aesthetic requirements (Section 2.1), and therefore this problem seems suitable for further investigation in improving the effectiveness of search in graph layout. Current search-based methods for schematic layout include hill-climbing (Section 2.2), mixed-integer programming (Section 2.3) and ant-colony systems (Section 2.4). Other work in automated layout of schematics include spider diagrams [14] and schematisation of road networks [1][6]. Studies of the effectiveness of schematics, their criteria and how they influence users have also been performed [11][17].

Parameter optimization in search has been widely studied [2] and relates to the

problem of meta-optimization [16] - the use of one optimization method to tune another; for example to find optimal parameter settings of a genetic algorithm. Search-based methods for the general graph drawing problem include simulated annealing [9] and genetic algorithms [4]; however, these methods make a wide search of the problem space, which requires many recalculations of fitness, so are not considered feasible with the computationally heavy fitness function required in schematic layout. It is one of the goals of the research described in this paper to see if a wider search can be conducted, whilst keeping runtime within computationally sensible bounds. This implies that both performance improvements to the fitness calculation are needed, and that a narrower search algorithm than the more general methods is required. Due to the large potential for performance improvement and ease of addition of new criteria, we have chosen to use hill-climbing for our automated schematic layout method.

Any search-based algorithm relies on the setting of a number of parameters. In the case of hill climbing, the parameters we considered in this paper were: grid spacing; the starting distance of node movement; and the number of iterations that the algorithm lasted for. As these parameters are typically set in an ad-hoc manner, we aimed to discover if current hill-climbing schematic layout had a tendency to reach local optima, indicating that current search-based methods are prone to sub-optimal results, and if we could avoid some of these local optima by varying the initial parameter settings, so finding a better layout for the diagram.

We examined four well known metro maps and our results indicate that the variation in final layout as the parameters change has underlying trends, but the result is unpredictable for specific parameter sets. In addition, the difference in final layout can be large, often between sets of parameters that have values that are close together. We interpret this as indicating that the system reaches a relatively poor local optima frequently and that there is no general characterization of parameter sets for any of the maps. As a result, in general, increasing the search space by attempting multiple layouts with different parameters results in a better schematic than can be reached by a search using a single parameter set.

This paper extends the work described in [8]. In this paper we add an overview of current layout techniques for schematic diagrams (Section 2). We describe the optimization method in Section 3, this differs from the previous method by allowing the last iteration of the search method to be run multiple times to help prevent cases of sub-optimal layout. Section 4 provides results and an evaluation of both phases of our testing, with a combined discussion in Section 5. Finally, Section 6 gives our conclusions and outlines possible future work.

2. Related Work in Schematic Layout Methods

This section provides an overview of current techniques used to automate the layout process of schematics. Schematic layout is a subset of the graph layout problem, and typically uses similar methods with additional criteria to force the layout process

Table 1: Maps used

Map	Junctions	Stations	Edges
Washington	9	77	53
Vienna	10	80	63
Mexico City	24	123	120
Sydney	24	151	103

Table 2: Parameters and values used

Parameters	Values
Grid Spacing	8, 10, 12, 14
Start Distance	13, 14, 15, 16, 17
Iterations	8, 12, 16, 20

to incorporate schematic characteristics. The list of methods examined here covers a number of alternate and promising approaches. These approaches use an initial geographic node embedding and each attempts to reposition nodes to fit a series of criteria.

Regarding terminology, this paper uses the term *junction* to mean a stop on a schematic network where lines meet. A *station* is stop where only one line is present. A *node* is either a junction or a station, and an *edge* connects two nodes. A *line* is a sequence of connected edges. In this paper, we show the edges of a line in the same color.

2.1. *Force-Based*

Hong et al. present an attempt at producing an automatically laid out metro map style schematic using a force-based layout algorithm [12]. They examine research carried out on existing hand-drawn metro maps [10], and devise the following criteria for good metro map layout: 1) Each line drawn as straight as possible 2) No edge crossings 3) No overlapping of labels 4) Lines mostly drawn horizontally or vertically, with some at 45° 5) Each line drawn with a unique color. Of which, criteria 1 and 2 have been shown to be effective for aiding graph readability [5]. The authors present five variations of the automation process, but each is based on GEM [3] - a modified force-directed layout algorithm which preserves edge crossings and includes a magnetic spring algorithm [20] to align edges to 45° angles. Along with this GEM algorithm, all but the initial method also use a preprocessing step which simplifies the graph. This preprocessing step removes all two-degree vertices in the graph thereby reducing the number of calculations required in the optimisation stage as well as significantly reducing the time taken to optimize. Hong et al. also

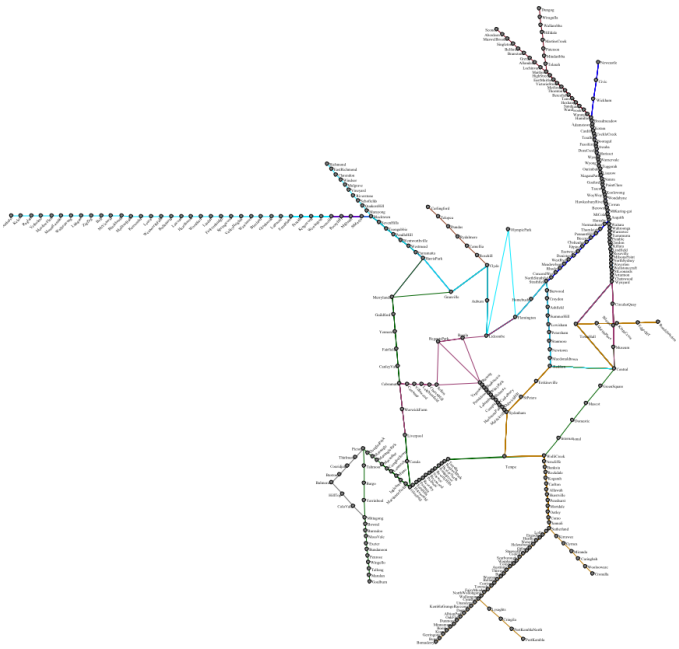


Fig. 1: Optimized Sydney metro produced using a force-based method by Hong et al.

implement an algorithm for the automated placement of labels; the label of each node has a possible eight positions that it can occupy, and a conflict map is created to identify node positions that cause occlusion. Each label is then positioned to reduce or eliminate the total number of label occlusions. Although their produced schematic shows labels that are angled at multiples of 45°, the paper does not go into more detail about possible label orientations.

The optimisation technique described has impressive optimisation times of a matter of seconds for fairly complex maps. However, this fast performance produces less aesthetically pleasing results than other schematic optimisation methods. Figure 1 shows an optimisation of the Sydney metro, and highlights some of the drawbacks of this method including: 1) Although the authors define a criteria to constrain lines to octilinear angles, this is not fully enforced and there are many lines that do not conform to this criterion 2) The distance between stations is not kept consistent across the diagram, with both extremely long and short edges 3) The figure appears unbalanced, with both very sparse areas and areas where stations appear to be overlapping.

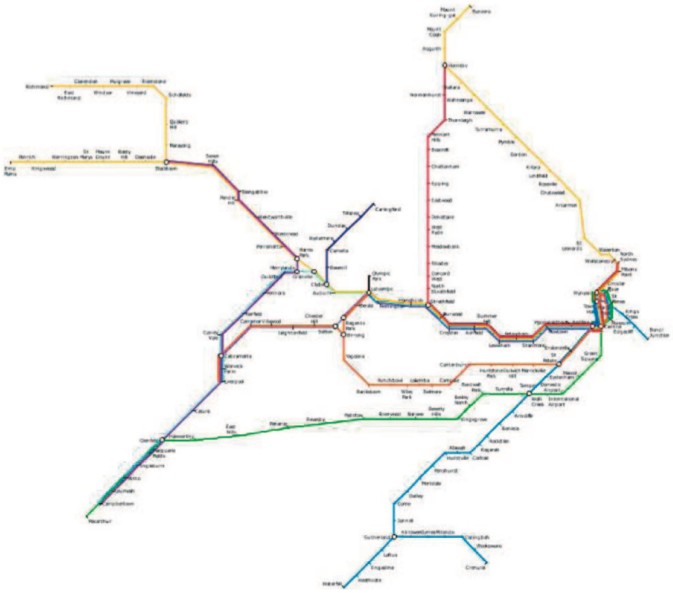


Fig. 2: Optimized Sydney metro produced using a multicriteria hill-climbing method by Stott et al.

2.2. *Multicriteria Hill-climbing*

Stott et al. present a multicriteria optimisation approach to schematic layout [19]. As in [12], a number of criteria have been devised to measure the aesthetic qualities of the layout. The authors have developed metrics for numerically quantifying these criteria that can be weighted and summed to find a fitness value for a given map layout. A hill-climbing optimizer is then used to reduce this fitness value and find improved layouts. This optimizer is explained in more detail in Section 3, as our chosen method is based upon this algorithm. In an attempt to help avoid local optima, clustering techniques are used to move groups of stations as well as individual stations. Along with station positioning, label positioning is also automated in a similar method to that by Hong et al. There are additional criteria for label positioning such as position priority and position consistency with nearby labels, although label rotation is not permitted.

This layout method uses more criteria and rules than the previously described force-based approach, and the result (Figure 2) is an output that looks far more like a hand-drawn metro map at the expense of optimisation time (although no optimisation times are given, the authors do mention that the optimizer is slow). However, the final quality of the produced map is the most important aspect for this situation rather than optimisation speed. Although the resulting layout is a vast improvement on the previous force-based method, there are still many sections that

are undesirable. For example, 1) Many lines have unnecessary bends, such as the multi-lined section near the right of the schematic 2) Many lines do not apply the octilinear criteria without any obvious reason, such as the single green horizontal line near the bottom of the schematic 3) The layout has struggled with the Sydney city centre section of the map (far right), which is very difficult to interpret. These issues are indicative of situations of local optima, where the optimizer is unable to move past a worse layout in oder to achieve a superior one. A great advantage of this optimisation is that it is very easily extensible, new criteria can be added without the need for changes in the optimisation process and this allows for relatively simple testing of new criteria. Criteria can also easily be toggled on or off to allow fast evaluation of their effectiveness.

2.3. *Mixed-Integer Linear Programming*

Nöllenburg and Wolff present another variant of optimisation technique for the optimization of metro maps using mixed-integer linear programming [15]. This method is different to the other optimisation methods described here because it defines its constraints as one of two types, *hard* and *soft*. Hard constraints must be met, whilst soft constraints are maximized. The main advantage of this method is that its optimisation technique guarantees that all hard constraints are met, whilst avoiding the problem of local minima which can be seen in the alternate methods explained here. The MIP optimisation technique used is a derivative of linear programming which is a method of determining the best outcome in a mathematical model based on a set of linear constraints (the hard constraints in this case) and a linear objective function. The linear constraints are plotted onto a graph to produce a feasible area, in which the linear objective function can be optimized.

Figure 3 shows the Sydney metro map layout produced by this method. The result is of a very high quality, and fully adheres to hard constraints such as octilinearity. The only obvious bad section of this schematic is the top yellow line, which unnecessarily performs a 90° bend. Although this method arguably produces the most aesthetically pleasing schematics, the optimisation quality is produced at the expense of run-time. Run time for the Sydney schematic shown here is quoted as 22 minutes. The implementation of this technique is also more complicated than a hill-climbing multicriteria approach, resulting in less potential performance improvement and less extensibility for additional criteria. A further disadvantage of this method is that if hard constraints are unable to be met, the optimisation will fail.

An alternate, faster, implementation of this technique is by Wang et al. [22]. However this is achieved with a smaller criteria set and so does not produce the same quality output.

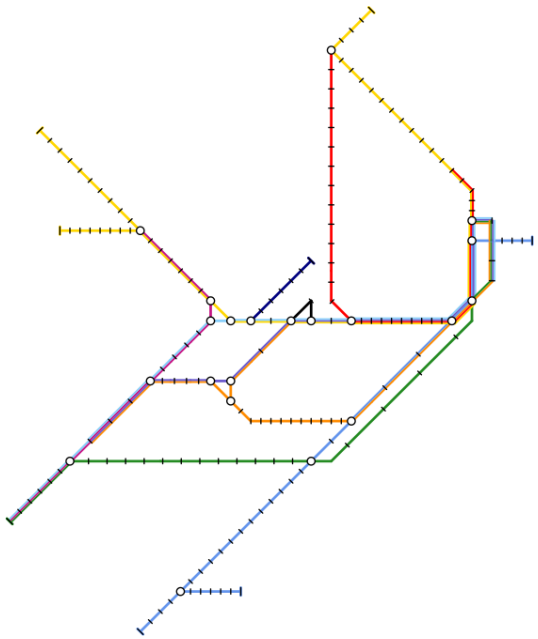


Fig. 3: Optimized Sydney metro produced using a mixed-integer linear programming method by Nöllenburg and Wolff

2.4. *Ant Colony System*

Ware and Richards present an ant colony system algorithm for automatically schematising network data sets [23]. Their ant colony system takes a similar approach to a multicriteria hill-climber, using a series of criteria in order to quantify map quality, combined with qualities from simulated annealing methods. Non-determinism is introduced into the node-positioning stage by moving the nodes in a random order; as opposed to in the same sequential order each iteration. Each non-deterministic iteration is run multiple times, each known as an ant, and at the end of each iteration the best resulting layout applies pheromones to its node positions in a global grid so as to affect all ants in future iterations. The pheromone increases the likelihood of ants moving nodes near to it, and it gets stronger over time in node positions that produce a good layout (as multiple ants will lay pheromones in the same position, creating a cumulative high pheromone value). As the pheromone gets stronger in node positions that produce a good layout, the algorithm eventually converges to a single solution.

Layouts produced by this method (Figure 4 - Sydney) are cleaner than a standard hill-climbing approach, with the entire schematic fully enforcing octilinearity, and in most cases a consistent length between adjacent stations; this is partly due to the fact it can avoid some local optima by ants having a possibility to move to

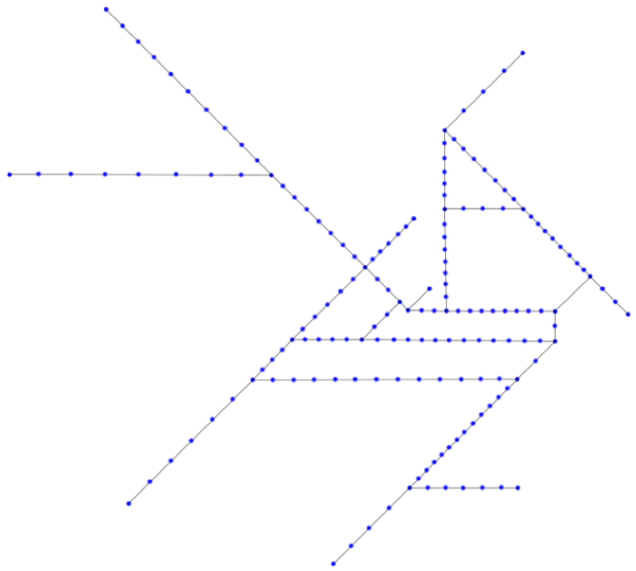


Fig. 4: Optimized Sydney metro produced using an ant colony system algorithm by Ware and Richards

a worse layout. There are still sections of this layout that show signs of remaining local optima, including the top right line which comes off at an angle not expected from the incoming two lines. It should be pointed out, however, that the Sydney schematic used in this paper has been simplified more than that found in the alternate methods; many sections where there was a "triangle" of nodes have been condensed into a single junction, and the complicated city centre section of the schematic has been reduced into a set of simpler connections. They also compare their map to an implemented version of a simulated annealing schematic layout technique [13], and conclude that *"in each case the ACS algorithm outperformed that of a previous SA algorithm in terms of cost (quality) and time"*. The running time is quoted as 150 seconds for the Sydney map shown, which is still in the realm of the search-based methods.

2.5. *Summary of Related Work*

This section has provided an overview of some notable methods in automated schematic layout. Many of these methods are search-based methods, and due to the infeasibility of performing a full search of the problem space, use a number of heuristics to speed up the search. This inevitably results in the arise of local optima situations, as can be seen in Figs. 1, 2 and 4. These situations of local optima indicate that a better layout can be found, and that alternate techniques must be tried to circumvent them. We are interested in how varying parameter sets of search-based schematic optimizers can affect the resulting output, with the aim of

producing an improved layout.

3. Overview of the Method

Inspired by the methods developed in [19] (Section 2.2), our automated layout algorithm uses a multicriteria hill-climbing search technique. This method operates by attempting to lower a set of measurable criteria by performing modifications to the schematic. There are two stages: firstly, the nodes are positioned; and secondly, the labels are positioned. In this paper we targeted performance improvements on the first, node positioning, as this is the major bottleneck.
There are three key parameters to the method. These are:

(1) **Grid Spacing**: A grid is placed over the canvas, and each schematic node must be positioned on a grid point. This parameter defines the grid resolution in pixels. When altered, this parameter affects the start layout as the nodes are initially snapped to the grid. It will also alter the number of potential sites that they can be positioned in when they move, and the distance by which they can move.
(2) **Start Distance**: This parameter defines the initial (and maximum) distance the nodes can be moved, in terms of grid positions. A method is applied to reduce this distance over the duration of the optimization, as explained below.
(3) **Iterations**: In each iteration, every node and cluster of nodes is examined to see if its location can be improved. Once an iteration has completed, the distance the nodes can move in the following iteration is re-evaluated. The initial iteration always uses the start distance, and the last iteration always uses a distance of one grid space. The distance is represented by an integer, and decreases along a floored linear interpolation between the initial and final iterations, so is not always reduced between iterations. The final iteration, with a distance of one, is repeated until no better solution can be found.

The method uses a series of criteria to calculate the aesthetic fitness. Each criterion has its own metric that calculates a value representing how well the current schematic adheres to it; and using this value we can calculate, weight, and sum it with all other values to produce a total fitness for the current schematic. The algorithm will attempt to reduce the total fitness value by moving around nodes, or clusters of nodes, and recalculating the fitness to determine if the schematic has been improved. The algorithm undergoes a number of iterations, and each node is moved once per iteration. Node movements are performed sequentially in a fixed order, to ensure determinism. A node can be either a junction (station with a degree of greater than 2) or a line bend point. A total fitness value of zero would indicate that all criteria have been perfectly met. The criteria include:

(1) **Octilinearity.** Lines should be at multiples of 45°.

Table 3: Overall time improvements across all tested parameter sets (minutes)

Diagram	Avg. Before	Avg. After	Speedup (times faster)
Washington	36.872	4.630	8.0
Vienna	51.929	10.581	4.9
Mexico	234.982	27.475	8.6
Sydney	518.813	61.340	8.5

Table 4: Layout time improvements by criteria (minutes)

Criteria	Avg. Before	Avg. After	Speedup (times faster)
Octilinearity	1.251	0.040	31.3
Edge Length	6.534	0.041	159.4
Line Straightness	13.289	3.062	4.3
Edge Crossings	14.671	1.870	7.9
Occlusions	110.223	5.257	21.0

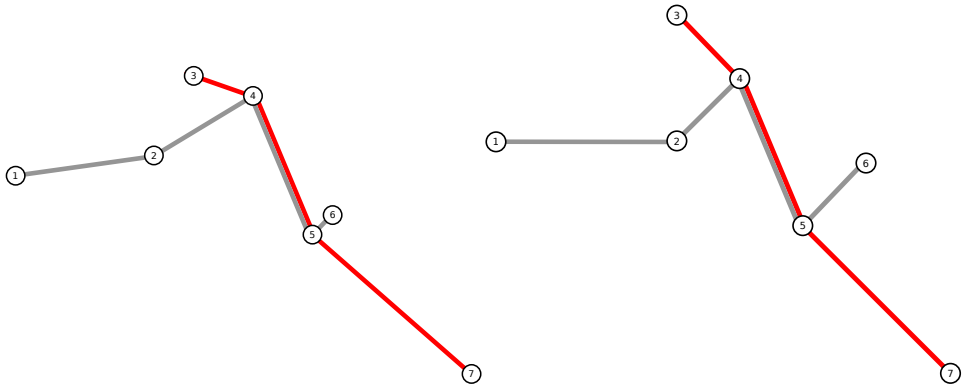
(2) **Edge Length.** Line sections between nodes should be a standard length.
(3) **Line Straightness:**
   (a) **Total.** The entirety of the line should be as straight as possible.
   (b) **Through Nodes.** Lines sections passing through junctions should be kept as straight as possible.
(4) **Edge Crossings.** Lines should not cross other lines.
(5) **Occlusion.** Nodes should not occlude parts of any edges.

Nodes can be moved in eight directions up to the current iterations distance value in grid spaces; *North*, *North-East*, *East*, *South-East*, *South*, *South-West* and *West*.

3.1. *Layout Example*

Figure 5 provides a minimum working example to illustrate the effects of the layout method. An unoptimised schematic (Figure 5a) is used as the starting point. As explained in the previous section, the algorithm moves each node in turn (numbers in the diagram indicate the order in which they are moved) attempting to lower the fitness value, calculated by summing the values of a set of metrics which measure the fitness of each criterion. Figure 6 shows an example of the node positions that are evaluated during an iteration. This example illustrates only the positions of node 2, during an iteration with a node movement distance of 4 units.

Figure 5d shows the resulting output, and it is clear to see how the algorithm has adjusted node positions to adhere to each criterion: 1) Each edge is now at a



(a) Unoptimized Input (Fitness=7.63)

(b) First iteration of node movements (Fitness=4.22)

(c) Second iteration of node movements (Fitness=2.65)

(d) Optimized output after cluster movements (Fitness=0.74)

Fig. 5: Minimum working example of layout algorithm

multiple of 45°, satisfying octilinearity 2) Edge lengths have been equalised 3) Lines have been straightened as much as possible (without introducing occlusion) both in total, and as they pass through junctions 4) Line crossings have been avoided 5) Occlusions have been avoided. These changes result in the fitness value dropping from 7.63 to 0.74 as almost all criteria have now been fully applied. The resulting fitness value is due to it not being possible to fully straighten both grey and red lines without occlusion (which is weighted higher to enforce more strongly), requiring two 45° line bends.

Figures 5b and 5c have been included to show exactly how the schematic changes during each iteration of the algorithm. During the first iteration (5a to 5b), many nodes are moved into positions that satisfy octilinearity. This is due to octilinearity being weighted strongly as it is regarded as one of the most important criterion to schematic layout. It can be seen that there is one edge (from node 4 to 5) that does not manage to achieve this, and this is because moving either of the two end

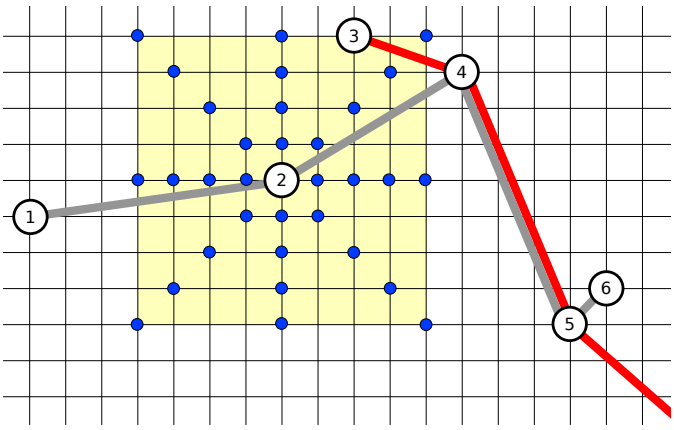


Fig. 6: Part of Figure 5a to show possible node movement positions (blue dots) for node 2 during an iteration with a distance of 4.

nodes to make the section octilinear would result in a negative effect on the other neighbouring nodes. Node 6 shows how the algorithm also moves nodes to attempt to standardise the edge lengths. This first iteration results in a drop in the fitness value from 7.63 to 4.22, as many octilinear issues are addressed. During the second iteration (5b to 5c), given that most nodes now already lie in octilinear directions from their neighbours, some over length edges are shortened (from node 1 to 2, and from node 5 to 7). Node 6 has also been moved down, improving the total line straightness of the grey line. These two changes further lower the fitness from 4.22 to 2.65. The transition between the last two schematics (5c to 5d) is a result of the clustering techniques used during the previous iteration (these are performed after single node movements, see [19] for more details on clustering methods used). Nodes 1 and 2 are clustered by angle, and both moved upwards; nodes 5, 6 and 7 are clustered by length, and are also moved upwards to make the edge between nodes 4 and 5 octilinear. These final clustering techniques further reduce the fitness from 2.65 to 0.74.

3.2. *Optimiser Performance*

The initial implementation of the search method in [7] was not optimized for performance, making the type of experimentation here infeasible. As a consequence, we spent some effort improving the performance of the method in order to make it run faster. The main performance increase was gained by caching all individual criteria fitness values for nodes and edges and re-using these as much as possible. We detect graph items that have moved and only recalculate the fitness values that are affected; these are then summed with the unaffected, previously calculated, values to obtain the new total fitness value.

In some cases, in particular edge crossings and occlusion, detecting items that

Table 5: Vienna results - Phase 1

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	15	20	10	0.972
2	17	20	10	0.981
3	16	16	10	0.986
4	16	20	10	0.986
5	17	16	10	0.986
6	15	16	10	1.037
7	14	20	10	1.135
8	17	12	10	1.180
9	16	12	10	1.195
10	13	16	10	1.317
...				
40	16	12	8	2.174
...				
80	17	8	14	4.233

have been affected by a node movement is not trivial as a change in the position of a single edge can also affect the edges or nodes along its length. In order to avoid having to re-check the moved edge with every other edge and junction, we place a second grid over the entire schematic. At the start of layout, each edge is examined and the edges that pass through grid cells are identified. This edge location grid is updated each time an edge is moved. Using such a grid to monitor the location of edges speeds up the testing for edge crossings and occlusions when checking for each, as the method can identify a subset of all nodes and edges as potential occlusion or crossing candidates by the grid squares in which changes have been made.

Table 1 lists the maps on which the testing has been performed, along with the number of junctions, stations and edges. The maps were chosen as being representative of reasonably sized schematics that demonstrated different characteristics and for which we could easily access the data. The table is listed by ascending order of the total number of junctions and stations. Table 3 shows the overall time performance increases gained by the algorithm improvements on each map. As seen in the table, there is a large performance increase from the implemented changes, averaging a 7.5 times speed improvement across the four schematics. This improvement in run time made the testing feasible by allowing us to carry out the required experiments in a reasonable time frame. Table 4 shows a breakdown of the time improvements on a per-criterion basis. There was a substantial performance increase per criterion, averaging 44.8 times faster. This is higher than the performance increase on the overall running time due to the algorithm performing other tasks that have not been optimized, such as label placement. The timings were performed on an ASUS Eee Pad Transformer TF101 running the Android operating system, version 3.2.1.

The device uses a 1GHz NVIDIA Tegra 2 with 1GB of RAM.

4. Evaluation

To test how changing the parameters impacts on the final diagram, we developed a test rig that allows us to explore a variety of settings for the chosen parameters. The testing rig outputs images of each schematic, and a file containing the fitness value of each. From this data, we investigated how modifying the parameters relates to the final fitness value. Of course, we can also pick the best layout (the one with lowest fitness) from those generated to be our output schematic.

4.1. *Phase 1*

In this section we present the results from the testing of each of the four maps. We have chosen Vienna as an example in which to go into more detail because it provides good variation in layout between the different runs. Diagrams generated from the other examples can been seen in the Appendix. Table 2 lists the values used for each parameter. Running a map from Table 1 with all possible parameter configurations from Table 2 results in a total of 80 layouts.

Table 5 shows an abridged table of the Vienna schematic results from the testing rig along with the parameter settings for each. It shows the top ten best schematics by fitness value, the median schematic and the single worst. An immediate trend can be seen from this table with a grid spacing value of ten appearing in all the best fitness values. For other maps, data given in the Appendix, only Sydney shows a similar trend; the other two maps do show some grouping by grid spacing, but in these cases it is less conclusive as the best grid also appears much lower down the ranking. For the four maps, only two share the same grid spacing for the best map, and there is no correlation between map size and grid spacing. As a result, it appears that grid spacing is perhaps the most important parameter to explore.

From Table 5 we can also see that there is a continuing improvement in fitness at the top of the list. This pattern, where the best fitness is found for only one set of parameters is also shown in two other maps (see the Appendix). We believe this is an indication that the system is not converging on the optimum solution, and so an even wider examination of the search space may be required to achieve the best final layout.

Figure 9 shows the original geographic layout of Vienna, used as the starting point by the algorithm. Figures 10, 11 and 12 show the best, median and worst final schematics for Vienna respectively. We have included the median, as this is the expected layout when parameters are arbitrarily chosen from the ranges used in this paper. In this case, the fitness of the best diagram is 44.71% of the median. Various cases of local optima are visible in Figure 11, the median Vienna diagram. For example, it has generally worse line straightness than the best diagram, Figure 10, which can be seen in nearly all lines. Many more cases of local optima can be seen

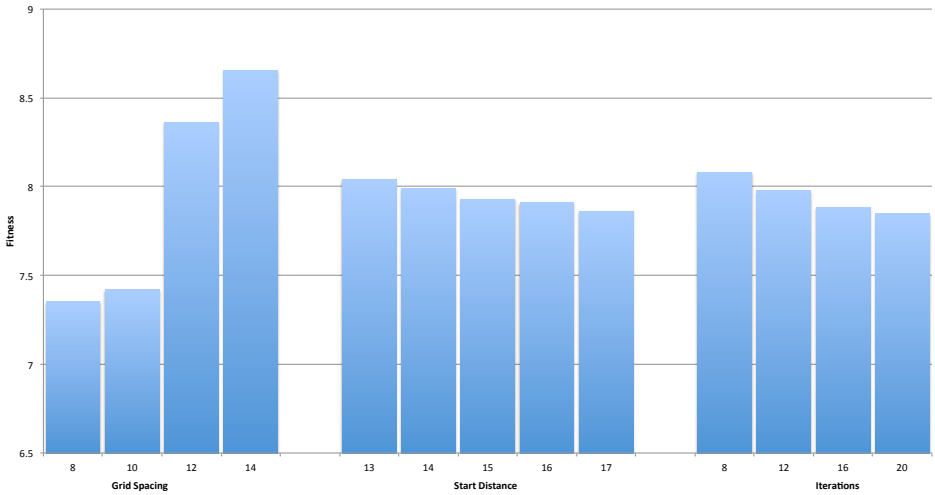


Fig. 7: Cumulative mean fitness value for all maps. Note: The y-axis starts from 6.5 - Phase 1

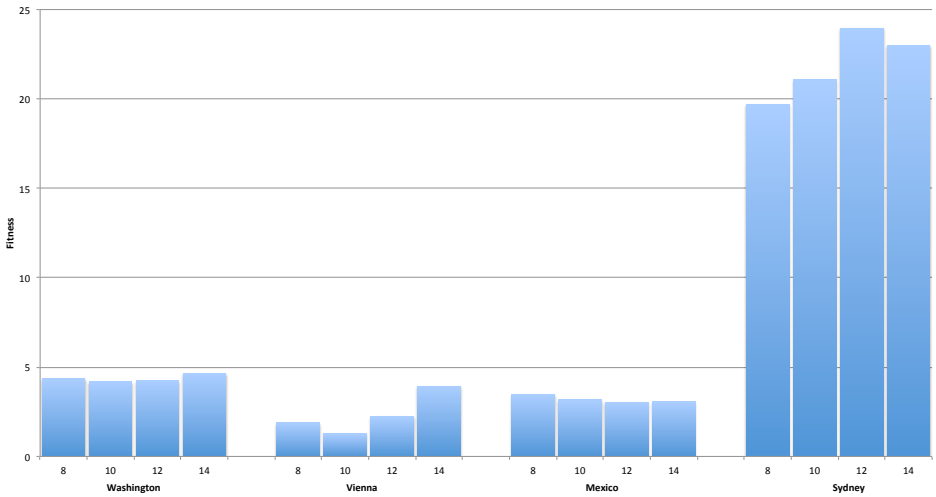


Fig. 8: Mean fitness value against grid spacing - Phase 1

in the worst diagram, Figure 12, which along with much poorer line straightness, has multiple edges breaking the octilinearity criterion.

Besides Vienna, we have also included the geographic and best layout of Sydney from the tests, shown in Figures 13 and 14 respectively. Figure 14, the best Sydney map, shows how this method has been able to improve problems with the layout

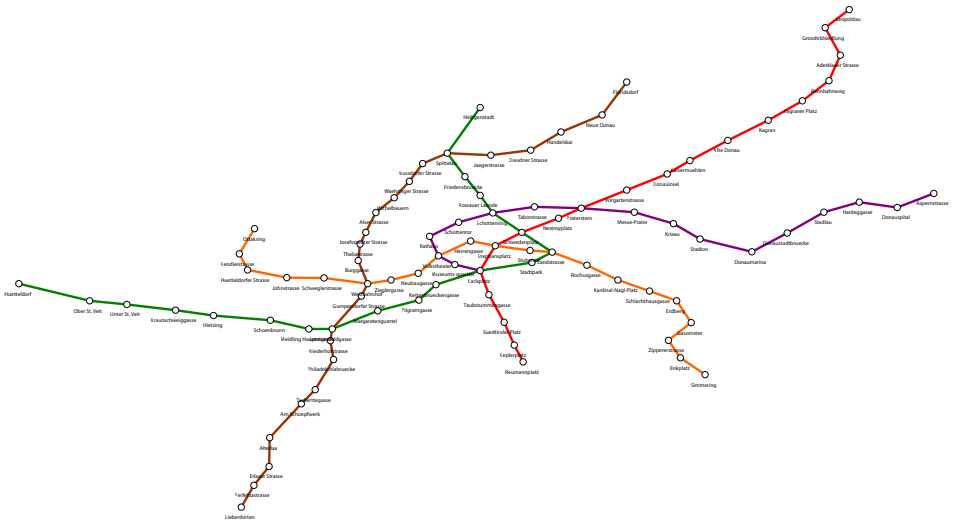


Fig. 9: Vienna - Geographic Map

Fig. 10: Vienna - Phase 1 - Rank 1 (Fitness = 0.972)

method by Hong et al. (Figure 1). Octilinearity has been improved, edge lengths are more similar in length across the schematic, and stations are more evenly distributed. An abridged table of results for Sydney can be found in the Appendix.

Fig. 11: Vienna - Phase 1 - Rank 40 (Fitness = 2.174)

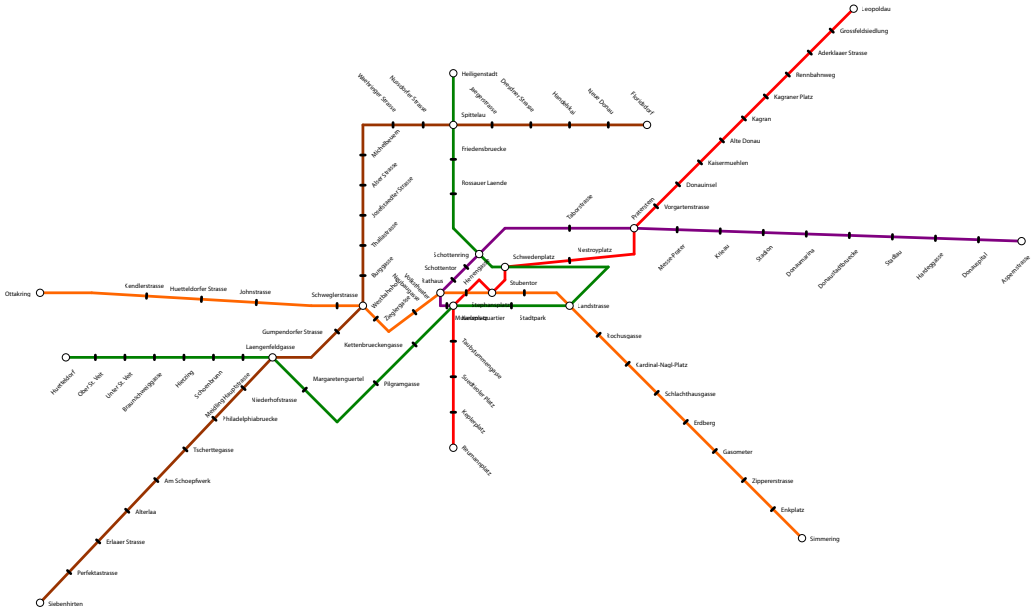


Fig. 12: Vienna - Phase 1 - Rank 80 (Fitness = 4.233)

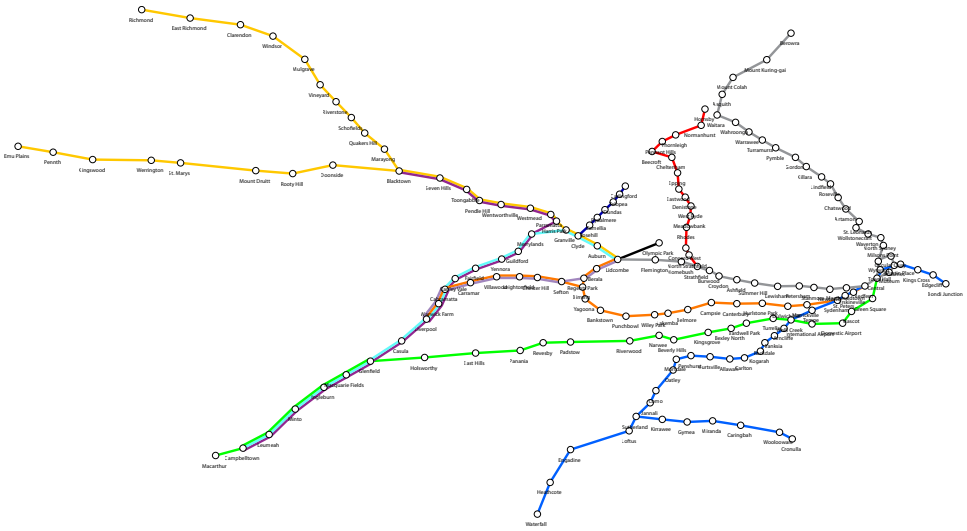


Fig. 13: Sydney - Geographic Map

Geographic and best images of Mexico and Washington, along with results, can also be found in the Appendix.

When the parameters are set in an ad-hoc fashion, and the wider search done here is not performed (as in the case in most current layout methods), the expected output layout is the median fitness value, and we can investigate how much our best diagram improves on this. In the case of Washington the best is 95.68% of the median, for Mexico the best is 86.79% of the median, and for Sydney the best is 91.28% of the median. As the percentage for Vienna was 44.71%, this implies that in some cases, something close to the best diagram can be found with a wide range of values, so allowing a more constrained (and faster) search to be applied. In other cases, a more restricted search will produce a much worse diagram.

It has been mentioned that a small change in the algorithm parameters can make a very large difference to the resultant layout. An example of this is the top ranked Mexico layout which has a fitness of 2.667, with parameters: start distance 13, iterations 20, grid spacing 14. A change from 20 to 16 iterations (one step in our testing) results in a drop to rank 50 and a fitness of 3.217.

Figure 7 shows the cumulative mean fitness for each parameter value. The graph indicates that for number of iterations and starting distance, a larger parameter will lead to a layout with a lower fitness; these trends also hold true on the map-specific graphs for start distance and iterations, shown in the Appendix. Increasing both of these parameters increases the search space, so indicating that an even wider search will tend to produce a better diagram. Grid spacing, which, as discussed previously, has the greatest effect upon the resulting layout's fitness, tends to produce better results when a smaller value is used.

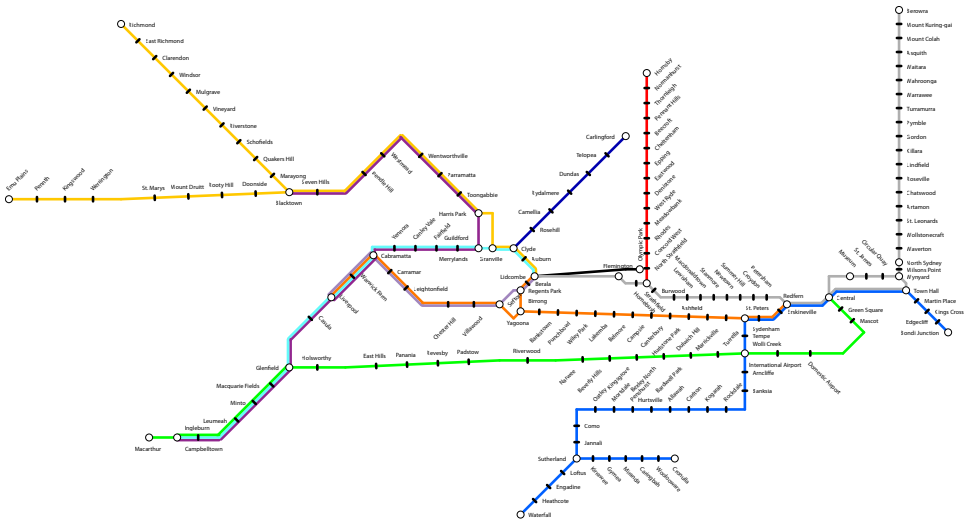


Fig. 14: Sydney - Phase 1 - Rank 1 (Fitness = 19.387)

However, Figure 8, which shows the mean fitness value of each layout produced by varying the grid spacing, does not show this trend on a by-map basis. This graph has been included because, unlike search distance and iterations, a clear trend cannot be seen in the cumulative data. Although grid spacing has a large effect on the fitness value of the produced layouts, there is no single optimum grid spacing value for all layouts, and each schematic has a specific value at which the best layouts are produced. Due to using a step size of two units for the tested grid spacing values, we are unable to determine the exact best value for each map. The best found value for Mexico and Sydney is also on the edge of the testing set, so we are unable to determine their best parameter value. To further investigate this we performed another series of tests, using a step of one unit and extending the range of both Mexico and Sydney values to attempt to find their best grid spacing values.

4.2. *Phase 2*

The initial phase of testing indicates that grid spacing is an important parameter with regards to the resulting fitness due to there not being a clear trend between it and better final fitness. Each map appears to have a particular grid spacing value at which the best fitness values are commonly found, and we therefore decided to run a new set of experiments using additional grid spacing values. The currently tested values have a step of two units, so it is possible that a better optimisation can be found between the tested values. Table 6 shows the new grid spacing values tested during phase 2, the values for each map have been chosen to test grid spacing values around the current best found schematic. For Mexico, where the best schematic used a grid spacing value that was on the edge of the data set, we have tested an

Table 6: Phase 2 grid spacing values

Map	Grid Spacing
Washington	9, 11
Vienna	9, 11
Mexico City	13, 15, 16
Sydney	3, 4, 5, 6, 7, 9

Table 7: Washington results - Phase 2

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	15	16	11	3.844
2	15	20	11	3.844
3	17	20	11	4.035
4	17	8	10	4.074
5	15	16	10	4.116
6	15	20	10	4.116
7	16	12	10	4.116
8	15	16	8	4.122
9	15	20	8	4.122
10	16	12	8	4.122
...				
60	13	8	9	4.249
...				
120	16	8	8	4.770

additional level. We have tested Sydney with six additional levels, this is because the first three additional values we tested (6, 7, 9) continued to show a decreasing fitness trend (ranks 2, 3, 4 with grid spacing 6). A grid spacing of 6 was again an extreme of the tested values, so we performed another series of tests using yet lower values of grid spacing: 3, 4 and 5.

Out of the four maps tested with additional grid spacing values, both Washington and Sydney managed to achieve a layout with a new best fitness level. The best ranking optimisation using new values for the other two maps is as follows: Vienna - 22 out of 120, Mexico - 79 out of 140. As a result we consider that Vienna and Mexico reached their best layout (by this method) in phase 1, and no further improvement was possible by manipulating the parameters considered in this paper. Hence, in this section we concentrate on Washington and Sydney.

Table 7 shows an updated table of results for Washington, where a grid spacing of 11 has produced the most optimum schematics; this is an additional 5.68%

Fig. 15: Washington - Phase 2 - Rank 1 (Fitness = 3.844)

improvement of the best schematic over the median. Figure 15 shows the new best Washington schematic produced. When compared to the previous best, Figure 22, a few enhancements can be seen that are resulting in a lower fitness value. For example, the centre section of the schematic has been spaced out further to allow extra room for stations; this change has allowed the removal of two line bend points, and has helped with the even distribution of nodes across the schematic. There are still a few situations of local optima present in this schematic, for example the blue periphery line section near the bottom is longer than necessary - this is either an indication of remaining local optima or a criteria conflict that cannot be resolved. Although running these tests with additional grid spacing values has produced a better layout, Figure 16 shows that on average a grid spacing of 10 is still best for the Washington schematic. It can also be seen that as the grid spacing gets further away from the optimum, the resulting fitness value increases.

Table 8 shows an updated table of results for Sydney, where a grid spacing of 4 has produced the most optimum schematic; this is an additional 2.28% improvement of the best schematic over the median Figure 17 shows the best ranked Sydney schematic produced using the new grid spacing values. As with the new best Washington schematic it shows a number of improvements over the previous best (Figure 14), for example more octilinear lines and more evenly distributed station spacing, but still has some apparent issues of local optima. Figure 18 shows the average fit-

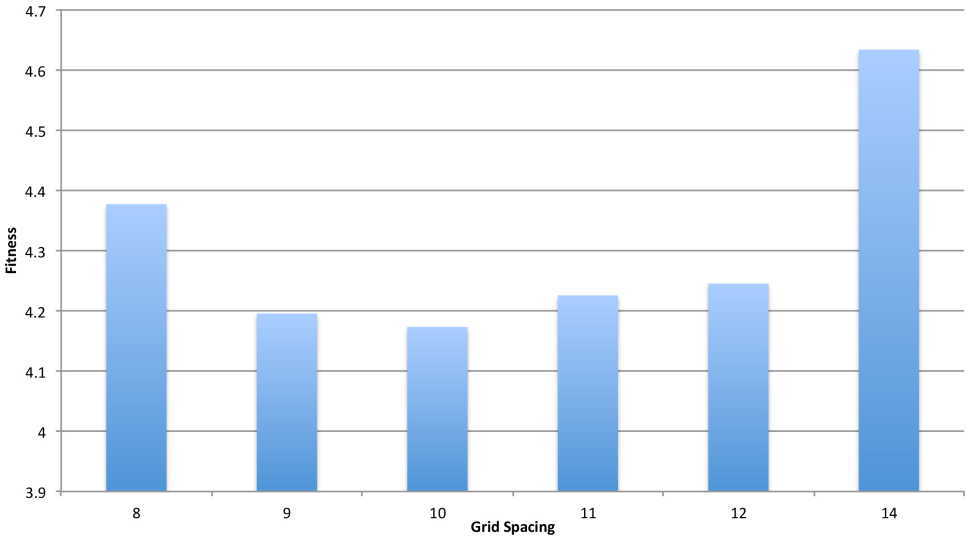


Fig. 16: Washington mean fitness value against grid spacing - Phase 2. Note: The y-axis starts at 3.9

Table 8: Sydney results - Phase 2

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	13	16	4	19.378
2	17	20	8	19.387
3	13	20	4	19.407
4	13	16	6	19.426
5	14	12	6	19.447
6	13	20	6	19.491
7	13	12	4	19.499
8	14	12	4	19.523
9	14	16	4	19.523
10	14	20	4	19.659
...				
100	13	8	7	21.742
...				
200	17	8	14	23.232

ness level for all tested values of grid spacing. It can be seen that for Sydney, there is a trend of grid spacing values between 4 to 8 producing a better result; this is in contrast to Vienna and Washington which prefer a spacing of 10, and Mexico at 14.

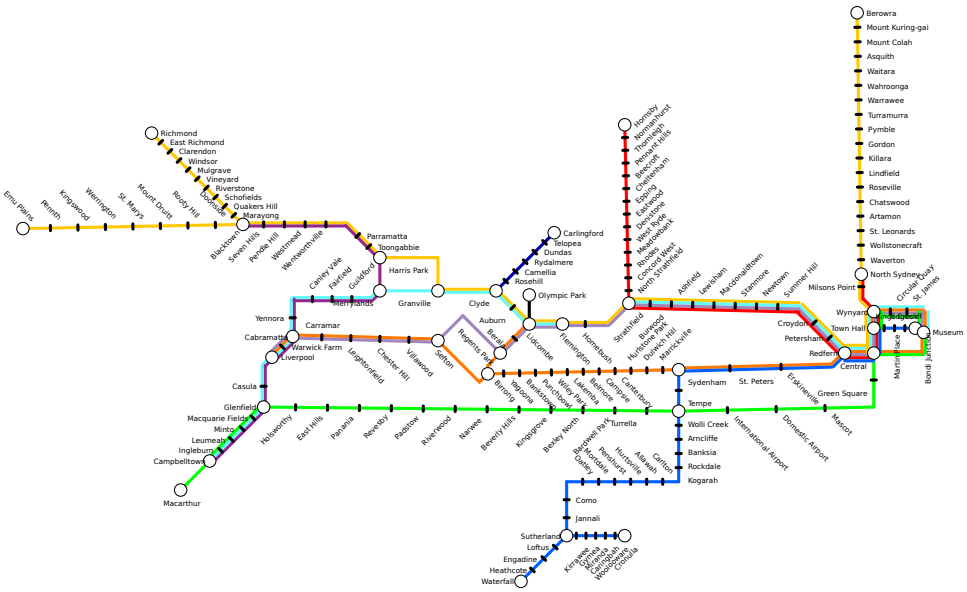


Fig. 17: Sydney - Phase 2 - Rank 1 (Fitness = 19.378)

5. Discussion

The previous section provided our test results and a separate discussion for each phase of testing. This section discusses an overview of the results obtained from both phases.

Our tests have indicated that for all four maps there is a slight trend towards better layouts being produced by a greater start distance and more iterations. This behaviour is expected, as increasing these parameters allows the system to evaluate more station positions, at the potential cost of increased run time. Unlike start distance and iterations, grid spacing did not display any obvious trend for improving fitness across all schematics; results indicate that each schematic has an optimum grid spacing value at which a lower fitness is more commonly produced, as can be seen in Figures 8, 16 and 18. A connection between optimal grid spacing and map density could be proposed, as the order in which maps increase in density inversely correlates to their best grid spacing value (ordered by perceived increasing density, best spacing in brackets): Mexico (14) < Vienna/Washington (10) < Sydney (4/8). This is intuitive, as a denser schematic would benefit from finer control of node positioning. However, this relationship needs to be verified with a larger number of schematics and a metric for quantifying density.

Table 9 shows the percentage improvement of the best schematic over the median for each map, indicating that if the correct parameters are chosen, the produced schematic will have on average a 22.25% lower fitness value than when using ad-hoc parameter values. This value is potentially larger - as only Mexico and Washington

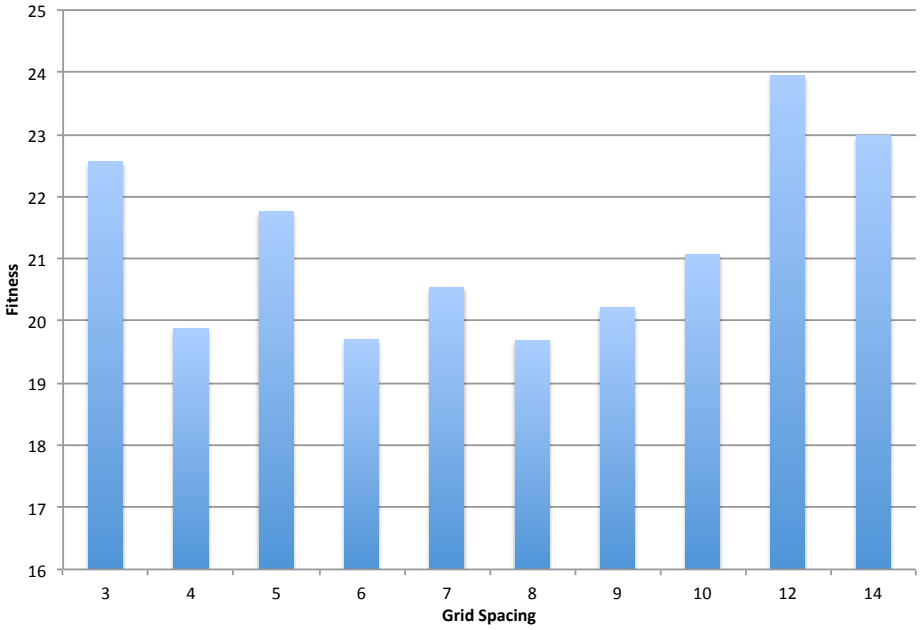


Fig. 18: Sydney mean fitness value against grid spacing - Phase 2. Note: The y-axis starts at 16

Table 9: Percentage improvement of best schematic over median

Map	Best	Median	% Improvement
Mexico	2.667	3.073	13.2%
Sydney	19.378	21.742	10.9%
Vienna	0.972	2.174	55.3%
Washington	3.844	4.249	9.6%
Mean			22.25%

managed to repeatedly produce the lowest fitness value (indicating a possible best), and that for at least Sydney and Vienna it is likely possible to generate a better fitness.

Interestingly, although trends have been identified in all parameters (grid spacing on a by-map basis), it can be seen in the results tables that the best layouts are not when all the parameters are at their best settings as indicated by the individual trends; there is still considerable variation.

6. Conclusion

When implementing any multicriteria search method, many parameters are used to configure the algorithm, for example those studied in this paper. We have performed a test to examine how the modification of three such parameters affects the final fitness result, and identified a number of trends. Increasing the start distance and number of iterations improved the mean result across all maps each time it was raised, and this is understandable as these directly increase the search space. A trend for grid spacing does not appear across all maps, but is visible on a per-map basis. This is an indication that grid spacing is related to the characteristics of a schematic.

Besides identifying trends, our results show that optimizing the three parameters leads to an overall mean improvement of 22.25% in the resulting fitness calculation over using ad-hoc values. When algorithms such as schematic layout are designed for the highest possible output quality without regard to time, any performance improvement is desired; and our test provides evidence that optimizing parameter values has a large effect on the performance of methods prone to local optima.

We have begun work on examining parameters for improving layout and characterizing maps. Firstly, other parameters may be investigated, for example, the number of bend points on diagrams and those that control clustering. Secondly, and perhaps most importantly, we plan to develop more sophisticated characterizations of schematics such as geographic density variance, and identifying radial and orbital maps. We are most interested to see if we can find a characterization of initial schematics that correlates to the identified patterns in grid spacing. If such a connection exists it would then be possible to automatically set optimizer parameters based on the schematic at the start of the layout, helping to produce a more optimal result than would otherwise be achieved using default values, without broadening the search space.

References

[1] Silviana Avelar and Matthias Muller. Generating topologically correct schematic maps. In *In Proceedings of the 9th International Symposium on Spatial Data Handling*, 2000.

[2] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, March 1993.

[3] François Bertault. A force-directed algorithm that preserves edge crossing properties. In *Lecture Notes in Computer Science*, volume 1731, pages 351–358, 1999.

[4] Jürgen Branke, Frank Bucher, and Hartmut Schmeck. Using genetic algorithms for drawing undirected graphs. In *The Third Nordic Workshop on Genetic Algorithms and their Applications*, pages 193–206, 1996.

[5] Helen C, Purchase, Robert F. Cohen, and Murray James. Validating graph drawing aesthetics. In *Lecture Nodes in Computer Science*, volume 1027, pages 435–446, 1996.

[6] Sergio Cabello, Mark de Berg, Steven van Dijk, Marc van Kreveld, and Tycho Strijk. Schematization of road networks. *Symposium on Computational Geometry*, pages 33–39, 2001.

[7] Daniel Chivers and Peter Rodgers. Gesture-based input for drawing schematics on a mobile device. In *Proceedings of Information Visualization*, pages 127–134, 2011.
[8] Daniel Chivers and Peter Rodgers. Exploring local optima in schematic layout. In *Visual Languages and Computing (VLC) in The 19th International Conference on Distributed Multimedia Systems (DMS 2013)*, 2013.
[9] Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331, October 1996.
[10] Ken Garland. *Mr Beck's Underground Map*. Capital Transport Publishing, England, 1994.
[11] Zhan Guo. Mind the map! the impact of transit maps on path choice in public transport. *Transportation Research Part A: Polocy and Practice*, 45(7):625–639, 2011.
[12] Seok-Hee Hong, Damian Merrick, and Hugo A. D. do Nascimento. The metro map layout problem. In *Proceedings of the 2004 Australasian symposium on Information Visualisation - Volume 35*, APVis '04, pages 91–100, Darlinghurst, Australia, 2004. Australian Computer Society, Inc.
[13] J. Mark Ware, George E. Taylor, Suchith Anand, and Nathan Thomas. Automated production of schematic maps for mobile applications. *Transactions in GIS*, 10(1):25–42, 2006.
[14] Joao Mourinho, Teresa Galvao, João Falcão e Cunha, Fernando Vieira, and Jose Pacheco. A software framework for the automated production of schematic maps. *Lecture Notes in Business Information Processing Volume*, 107:64–78, 2012.
[15] Martin Nöllenburg and Alexander Wolff. A mixed-integer program for drawing high-quality metro maps. In *Proceedings of the 13th international conference on Graph Drawing*, GD'05, pages 321–333, Berlin, Heidelberg, 2006. Springer-Verlag.
[16] Ibrahim Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623, October 1996.
[17] Maxwell J. Roberts, Elizabeth J. Newton, Fabio D. Lagattolla, Simon Hughes, and Megan C. Hasler. Objective versus subjective measures of paris metro map usability: Investigating traditional octolinear versus all-curves schematics. *International Journal of Human-Computer Studies*, 71(3):363–386, 2013.
[18] Jonathan Stott and Peter Rodgers. Metro map layout using multicriteria optimization. In *Proceedings 8th International Conference on information Visualization (IV04)*, pages 355–362, 2004.
[19] Jonathan Stott, Peter Rodgers, Juan Carlos Martínez-Ovando, and Stephen G. Walker. Automatic metro map layout using multicriteria optimization. *IEEE Transactions on Visualization and Computer Graphics*, 17(1):101–114, 2011.
[20] Kozo Sugiyama and Kazuo Misue. Graph drawing by the magnetic spring model. *Journal of Visual Languages and Computing*, 6(3):217–231, 1995.
[21] Roberto Tamassia. *Handbook of Graph Drawing and Visualization*. CRC Press, 2013.
[22] Yu-Shuen Wang and Ming-Te Chi. Focus+context metro maps. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2528–2535, December 2011.
[23] Mark Ware and Nigel Richards. An ant colony system algorithm for automatically schematizing transport network data sets. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1892–1900, 2013.

Appendix

Table 10: Fitness statistics by map - Phase 1

Schematic	Mean	Median	Mode	SD	IQR
Washington	4.358	4.258	4.634	0.222	0.489
Vienna	2.327	2.197	2.221	1.043	0.601
Mexico	3.179	3.139	3.073	0.306	0.304
Sydney	21.924	21.926	23.232	1.681	2.675

Table 11: Statistics by Parameter Value - Phase 1

Parameters	Mean	Median	Mode	Standard Deviation	IQR
Grid Spacing					
8	7.354	4.116	19.730	7.223	5.471
10	7.421	3.879	4.144	8.004	6.143
12	8.361	3.624	2.221	9.090	6.461
14	8.653	4.434	4.634	8.353	5.907
Start Distance					
13	0.043	4.178	19.663	8.264	5.318
14	7.992	4.205	19.660	8.268	5.466
15	7.928	4.133	19.730	8.215	5.468
16	7.911	4.133	19.730	8.193	5.486
17	7.861	4.105	2.221	8.200	5.339
Iterations					
8	8.081	4.205	2.221	8.197	5.425
12	7.976	4.145	2.221	8.213	5.323
16	7.884	4.144	2.221	8.221	5.542
20	7.848	4.144	2.221	8.227	5.488

Table 12: Mexico Results - Phase 1

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	13	20	14	2.667
2	14	16	14	2.667
3	14	20	14	2.667
4	15	16	14	2.667
5	15	20	14	2.667
6	16	16	12	2.802
7	16	20	12	2.802
8	17	16	12	2.802
9	17	20	12	2.845
10	13	20	10	2.847
...				
40	17	12	12	3.073
...				
80	13	8	8	4.111

Table 13: Sydney Results - Phase 1

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	17	20	8	19.387
2	14	8	8	19.660
3	14	12	8	19.660
4	14	16	8	19.660
5	14	20	8	19.660
6	13	8	8	19.663
7	13	12	8	19.663
8	13	16	8	19.663
9	13	20	8	19.663
10	17	8	8	19.719
...				
40	14	12	10	21.237
...				
80	14	12	12	24.332

Table 14: Vienna Results - Phase 1

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	15	20	10	0.972
2	17	20	10	0.981
3	16	16	10	0.986
4	16	20	10	0.986
5	17	16	10	0.986
6	15	16	10	1.037
7	14	20	10	1.135
8	17	12	10	1.180
9	16	12	10	1.195
10	13	16	10	1.317
...				
40	16	12	8	2.174
...				
80	17	8	14	4.233

Table 15: Washington Results - Phase 1

Rank	Start Distance	Iterations	Grid Spacing	Fitness
1	17	8	10	4.074
2	15	16	10	4.116
3	15	20	10	4.116
4	16	12	10	4.116
5	15	16	8	4.122
6	15	20	8	4.122
7	16	12	8	4.122
8	17	8	8	4.135
9	13	16	10	4.144
10	13	20	10	4.144
...				
40	16	8	12	4.258
...				
80	16	8	8	4.777

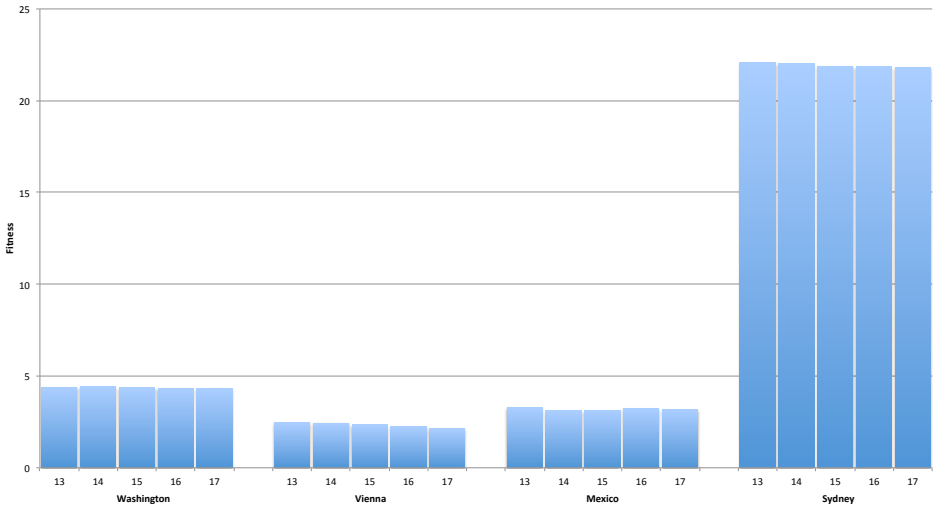


Fig. 19: Mean start distance against fitness value - Phase 1

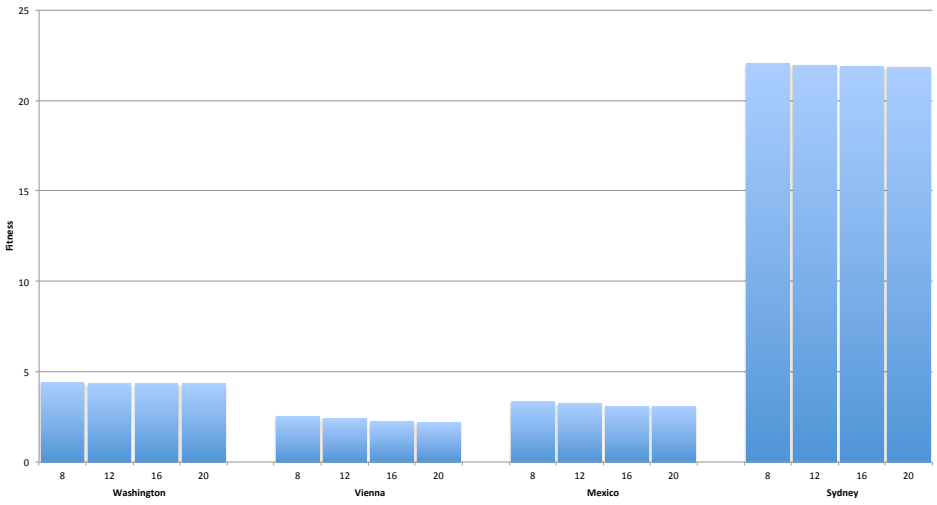


Fig. 20: Mean number of iterations against fitness value - Phase 1

Fig. 21: Washington - Geographic Map

Fig. 22: Washington - Phase 1 - Rank 1 (Fitness = 4.074)

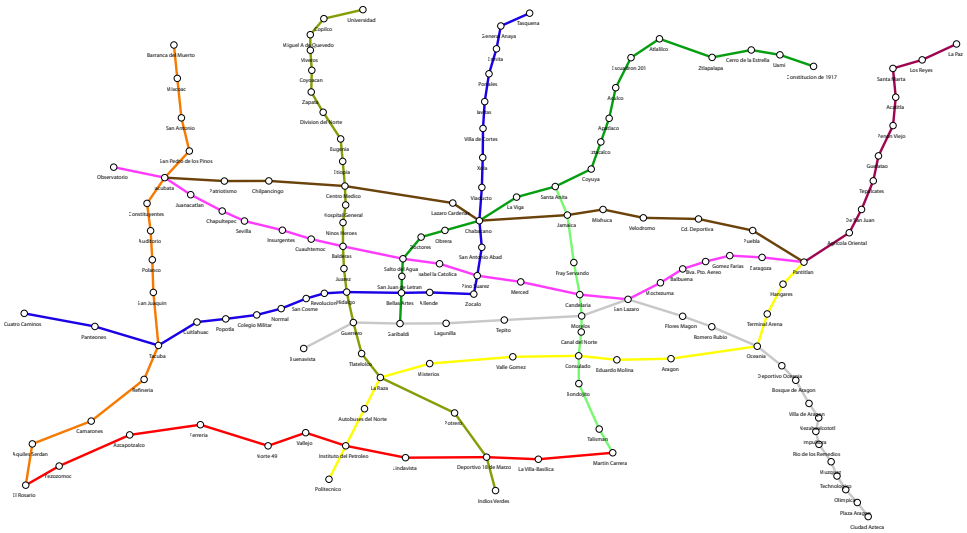


Fig. 23: Mexico - Geographic Map

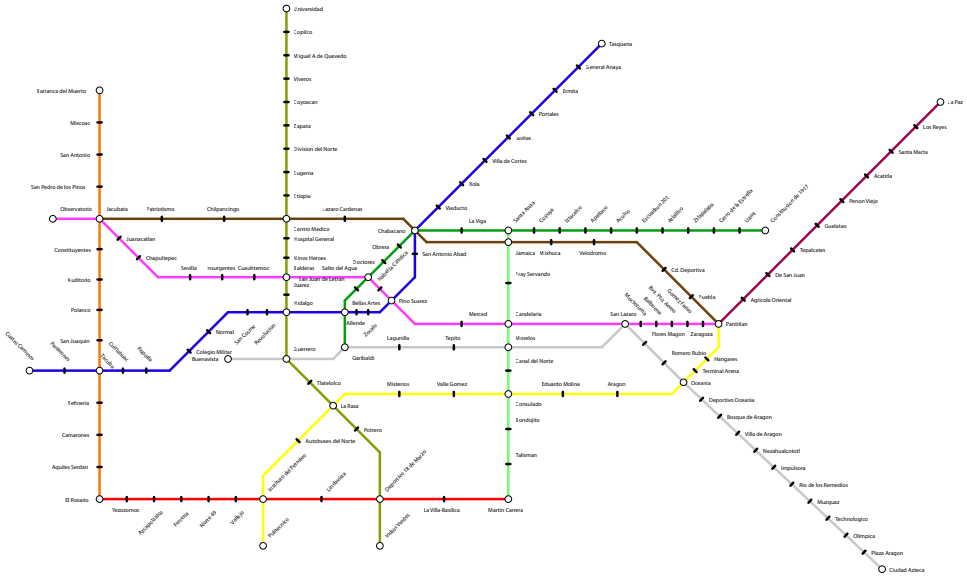


Fig. 24: Mexico - Phase 1 - Rank 1 (Fitness = 2.667)