Exploring Local Optima in Schematic Layout

Daniel Chivers
Department of Computer Science
The University of Kent, Canterbury
Email: dc355@kent.ac.uk

Peter Rodgers
Department of Computer Science
The University of Kent, Canterbury
Email: P.J.Rodgers@kent.ac.uk

***Abstract*—In search-based graph drawing methods there are typically a number of parameters that control the search algorithm. These parameters do not affect the fitness function, but nevertheless have an impact on the final layout. One such search method is hill climbing, and, in the context of schematic 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. Although we cannot characterize schematics completely and so cannot yet automatically assign parameters for diagrams, we observe that when parameters are set to values that increase the search space, they also tend to improve the final layout. We come to the conclusion that hill-climbing methods for schematic layout are more prone to reaching local optima than had previously been expected and that a wider search, as described in this paper, can mitigate this, so resulting in a better layout.**

I. Introduction

Search-based methods for graph drawing have been successfully used for a number of years. 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 where a number of aesthetics are present, including the 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 [5]. As a result 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 and mixed integer programming. The results from hill-climbing methods for schematic layout have been positively compared to some published maps [8], and allow easy implementation of layout criteria. Current hill-climbing methods also provide plenty of room for performance improvements on criteria calculations. Mixed integer programming techniques are able to escape local minima, producing high-quality schematics at the cost of optimisation time [6]. However, it is hard to implement new criteria and there is less potential for optimization.

Parameter optimization in search has been widely studied [1] and relates to the problem of metaoptimization [7]. Search-based methods for the general graph drawing problem include simulated annealing [4] and genetic algorithms [2]; 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 performance improvement potential 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. We were interested in seeing if we could increase the search of the problem space by varying these parameters. As they 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. As a consequence, we hoped to improve the layout of schematic diagrams drawn by search-based methods.

The method for calculating metrics described in this paper is considerably faster than previous techniques, hence permitting more alternatives to be examined and so widening the search of the problem space. 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 single search using a single parameter set.

In the remainder of this paper, Sect. II provides an overview of the algorithm used to layout schematics, along with a description of the performance improvements and the algorithm parameters. Section III provides the results from the testing process, as well as our interpretation. Finally, Sect. IV gives our conclusions and outlines possible future work.

II. Overview of the Method

Inspired by the methods developed in [8], our automated layout algorithm uses a multicriteria hill-climbing search

TABLE I. MAPS USED

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

TABLE II. PARAMETERS AND VALUES USED

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

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 an 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 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 method uses a series of criteria to calculate the aesthetic fitness. Each of these criterion has its own metric for calculating a value on the current schematic representing how well it adheres to the criterion. Using this value we can then 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 required values. 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 $45°$.
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.

TABLE III. OVERALL TIME IMPROVEMENTS (IN 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 IV. LAYOUT TIME IMPROVEMENTS BY CRITERIA (IN 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

   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; *North*, *North-East*, *East*, *South-East*, *South*, *South-West* and *West*.

The initial implementation of the search method in [3] was not optimized for performance, making the layout slow and so attempting the type of experimentation here was 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 is 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 were affected; these are then summed with the unaffected, previously calculated, values.

In some cases, in particular edge crossings and occlusion, detecting items that have been affected by a node movement is not trivial as a change in the position of a single edge can affect 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 I 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 III 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 IV

TABLE V. VIENNA RESULTS

Rank	Map ID	Start Distance	Iterations	Grid Spacing	Fitness
1	46	15	20	10	0.972
2	78	17	20	10	0.981
3	58	16	16	10	0.986
4	62	16	20	10	0.986
5	74	17	16	10	0.986
6	42	15	16	10	1.037
7	30	14	20	10	1.135
8	70	17	12	10	1.180
9	54	16	12	10	1.195
10	10	13	16	10	1.317
...					
40	53	16	12	8	2.174
...					
80	68	17	8	14	4.233

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 runs on a 1GHz NVIDIA Tegra 2 with 1GB of RAM.

To test how changing the parameters impacts on the final diagram, we developed a test rig that would allow us to explore a variety of settings for the chosen parameters. The range of values for the parameters were decided by informal testing and are given in Sect. III. 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 best fitness) from those generated to be our output schematic.

III. Results and Analysis

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 II lists the values used for each parameter. Running a map from Table I with all possible parameter configurations from Table II results in a total of 80 layouts. Table V 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 in 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; however, there are no patterns evident that would allow the derivation of an ideal grid spacing for a particular map.

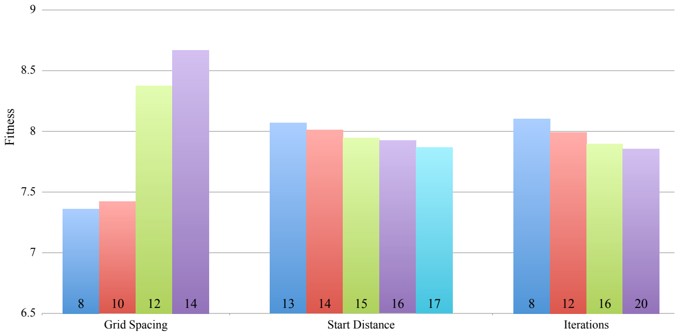

Fig. 1. Cumulative mean fitness value for all maps combined. Note: The Y-Axis starts from 6.5

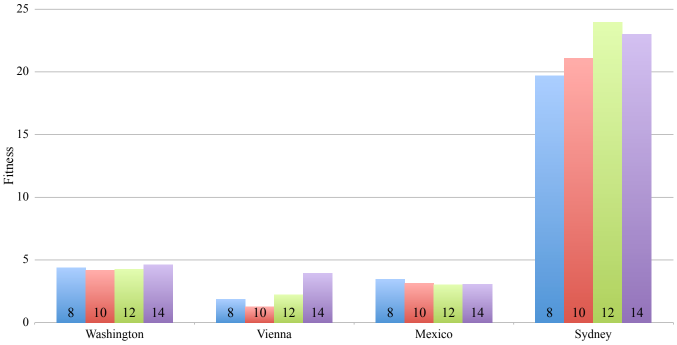

Fig. 2. Mean grid spacing against fitness value

From Table V 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 3 shows the original geographic layout of Vienna, used as the starting point by the algorithm. Figures 4, 5 and 6 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 the median Vienna diagram, Fig. 5. For example, the median diagram has generally worse line straightness than the best diagram, Fig. 4, which can be seen in nearly all lines. Many more cases of local optima can be seen in the worst diagram, Fig. 6, which along with much poorer line straightness, has multiple edges which break the octilinearity criterion.

Besides Vienna, we have also included the geographic and best layout of Sydney from the tests, shown in Figures 7 and 8 respectively. An abridged table of results for Sydney can be found in the Appendix. 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

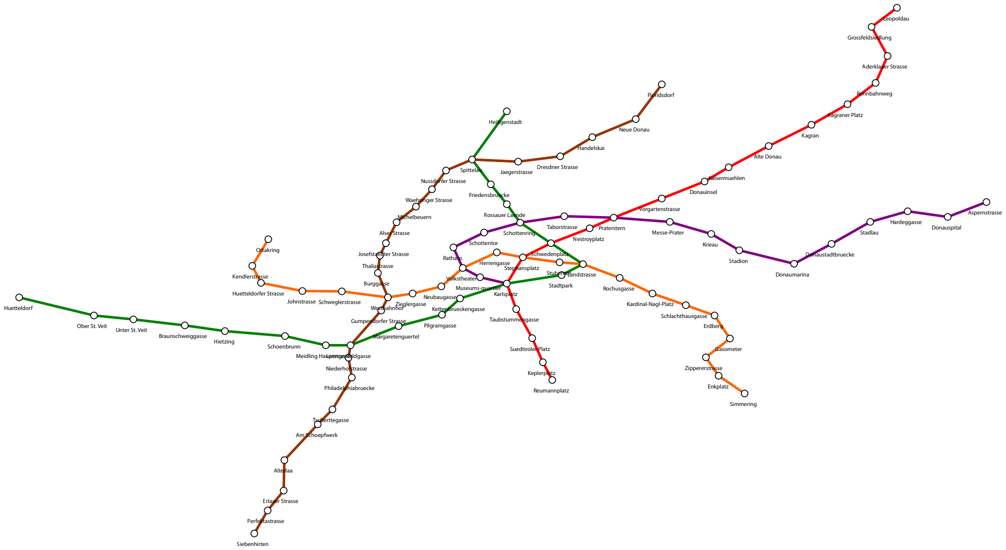

Fig. 3. Vienna - Geographic Map

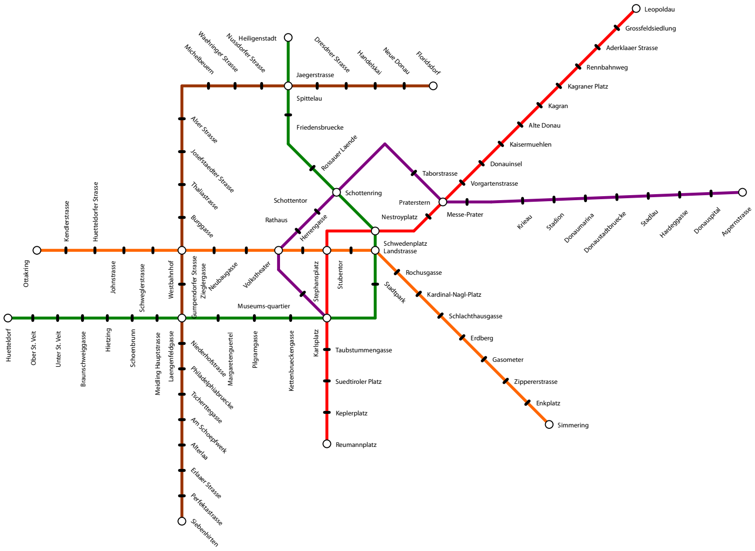

Fig. 4. Vienna - Rank 1 (Fitness = 0.972)

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 parameters, 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 1 shows the cumulative mean fitness for each parameter value. The graph indicates that, for number of iterations and starting distance, a larger parameter value is likely to lead to a layout with a better 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.

However, Fig. 2, 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 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

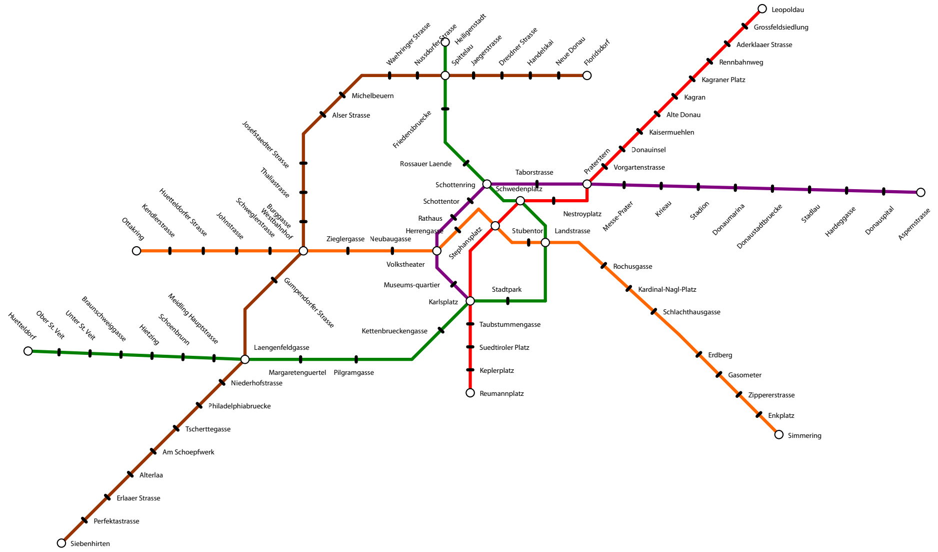

Fig. 5. Vienna - Rank 40 (Fitness = 2.174)

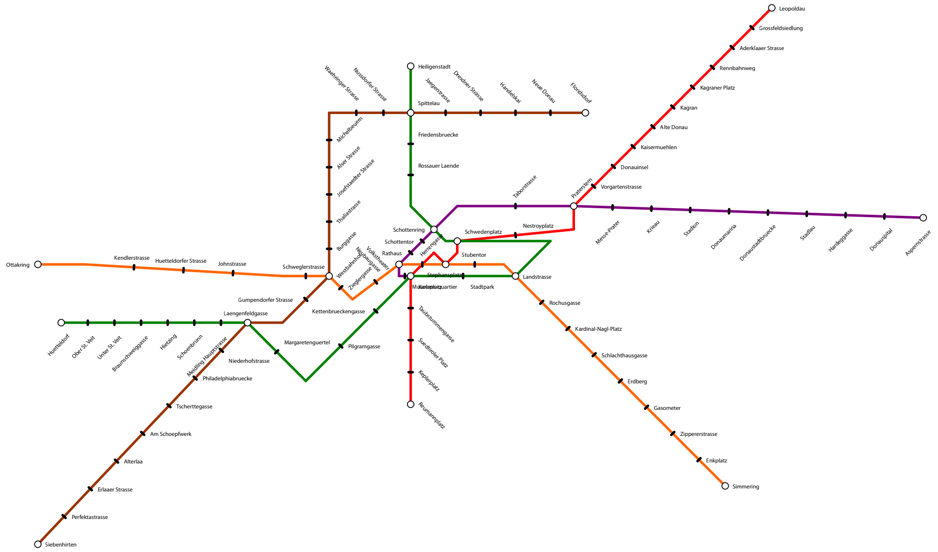

Fig. 6. Vienna - Rank 80 (Fitness = 4.233)

has a specific value at which the best layouts are produced. Interestingly, Mexico and Sydney, which are comparatively large maps with similar sizes, display opposite trends in terms of grid spacing, with Mexico favouring the larger spacing values and Sydney the smallest. Further investigation is required by examining a larger variation of grid spacing in order to try to identify any global trends.

IV. Conclusion

When implementing any multicriteria search method, many parameters are used to configure the algorithm, including those studied in this paper. With the algorithm we present here, the fitness of the best schematic layout is typically considerably better than a layout that might be found by an ad-hoc method for assigning parameter values. As a result, our algorithm generates better layouts than previous schematic search-based methods.

We have shown that for all four maps there is a slight trend towards better layouts being produced by more iterations and a greater start distance. This behaviour is expected, as increasing these parameters allows the system to evaluate more station positions, but at the potential cost of greater run time. Unlike number of iterations and start distance, grid spacing did not display a clear trend for improving fitness across all schematics, but results obtained did indicate that each schematic has a 'best' grid spacing value at which a lower fitness is more commonly produced. Although trends have been identified in all parameters, the best layouts are not when all the parameters are at their best settings as indicated by the trends; there is considerable variation.

We have begun work on examining parameters for improving layout and characterizing maps. However, much future work needs to be done. Firstly, other parameters may be investigated, for example, the number of bend points on diagrams and those that control clustering. Secondly, the current

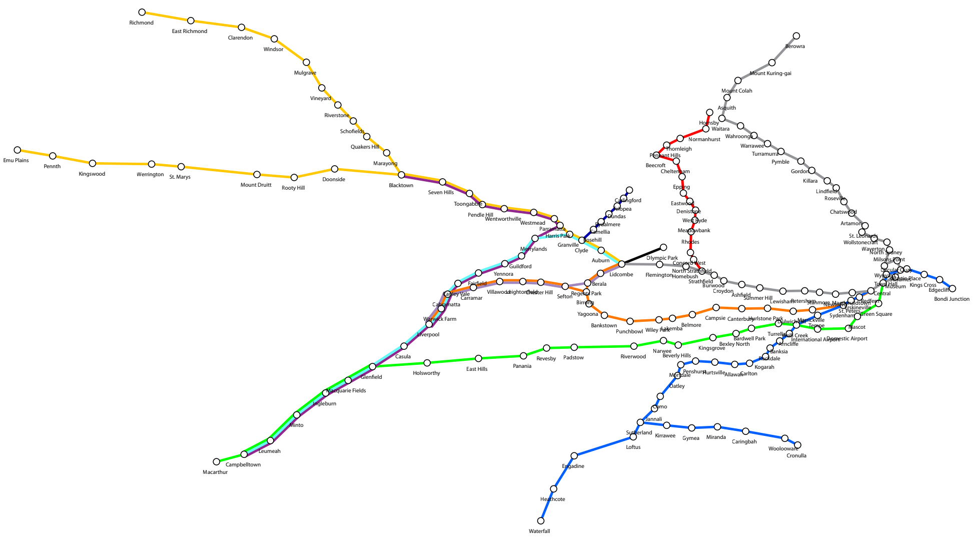

Fig. 7. Sydney - Geographic Map

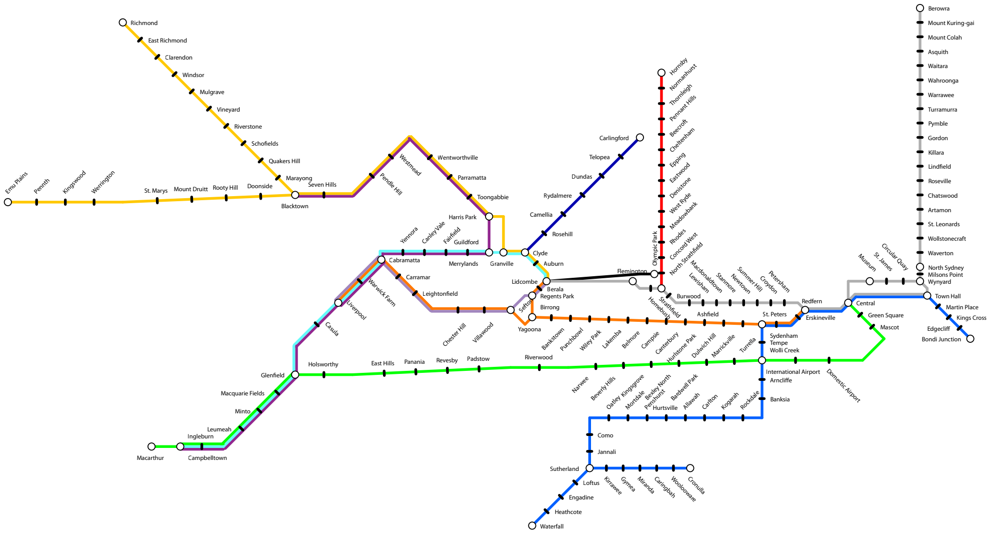

Fig. 8. Sydney - Rank 1 (Fitness = 19.387)

parameters have not been fully explored, including looking at the limits of the trends currently identified in this paper. We could also develop more sophisticated characterizations of diagrams such as geographic density variance, or examine if radial and orbital schematic characteristics have an effect and so can aid the choice of parameter value. Finally, we have evidence that the most optimal layout has not yet been reached with our current method, and so further investigation with a wider search is an important next step towards generating the best possible schematic layouts.

References

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

[2] 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.

[3] Daniel Chivers and Peter Rodgers. Gesture-based input for drawing schematics on a mobile device. In *Proceedings of Information Visualization*, pages 127–134, 2011.

[4] Ron Davidson and David Harel. Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331, October 1996.

[5] 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.

[6] 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.

[7] Ibrahim Osman and Gilbert Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623, October 1996.

[8] 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.

Appendix

TABLE VI. FITNESS STATISTICS BY MAP

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 VII. STATISTICS BY PARAMETER VALUE

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 VIII. WASHINGTON RESULTS

Rank	Map ID	Start Distance	Iterations	Grid Spacing	Fitness
1	226	17	8	10	4.074
2	202	15	16	10	4.116
3	206	15	20	10	4.116
4	214	16	12	10	4.116
5	201	15	16	8	4.122
6	205	15	20	8	4.122
7	213	16	12	8	4.122
8	225	17	8	8	4.135
9	170	13	16	10	4.144
10	174	13	20	10	4.144
...					
40	211	16	8	12	4.258
...					
80	209	16	8	8	4.777

TABLE IX. VIENNA RESULTS

Rank	Map ID	Start Distance	Iterations	Grid Spacing	Fitness
1	46	15	20	10	0.972
2	78	17	20	10	0.981
3	58	16	16	10	0.986
4	62	16	20	10	0.986
5	74	17	16	10	0.986
6	42	15	16	10	1.037
7	30	14	20	10	1.135
8	70	17	12	10	1.180
9	54	16	12	10	1.195
10	10	13	16	10	1.317
...					
40	53	16	12	8	2.174
...					
80	68	17	8	14	4.233

TABLE X. MEXICO RESULTS

Rank	Map ID	Start Distance	Iterations	Grid Spacing	Fitness
1	96	13	20	14	2.667
2	108	14	16	14	2.667
3	112	14	20	14	2.667
4	124	15	16	14	2.667
5	128	15	20	14	2.667
6	139	16	16	12	2.802
7	143	16	20	12	2.802
8	155	17	16	12	2.802
9	159	17	20	12	2.845
10	94	13	20	10	2.847
...					
40	151	17	12	12	3.073
...					
80	81	13	8	8	4.111

TABLE XI. SYDNEY RESULTS

Rank	Map ID	Start Distance	Iterations	Grid Spacing	Fitness
1	317	17	20	8	19.387
2	257	14	8	8	19.660
3	261	14	12	8	19.660
4	265	14	16	8	19.660
5	269	14	20	8	19.660
6	241	13	8	8	19.663
7	245	13	12	8	19.663
8	249	13	16	8	19.663
9	253	13	20	8	19.663
10	305	17	8	8	19.719
...					
40	262	14	12	10	21.237
...					
80	263	14	12	12	24.332

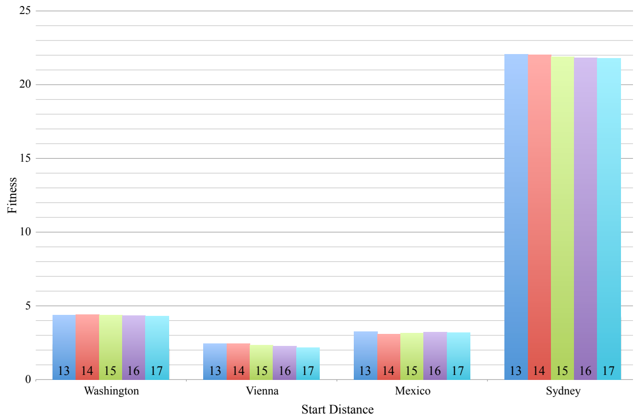

Fig. 9. Mean start distance against fitness value

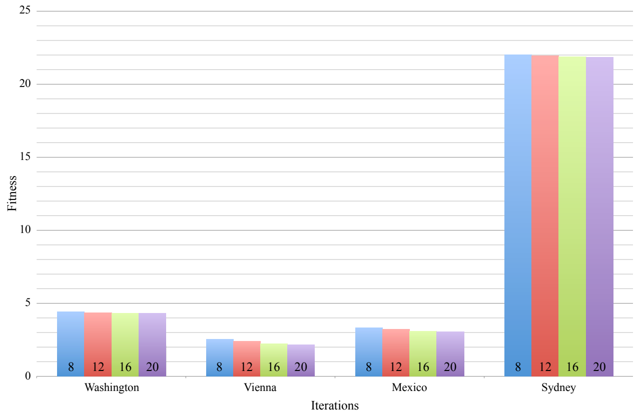

Fig. 10. Mean number of iterations against fitness value

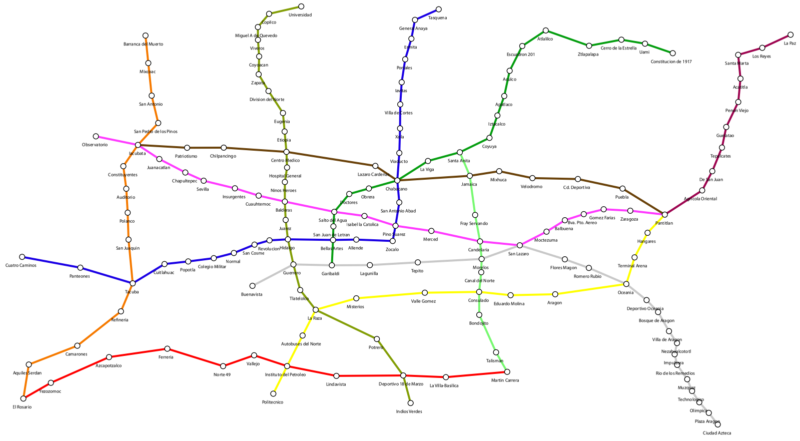

Fig. 12. Mexico - Geographic Map

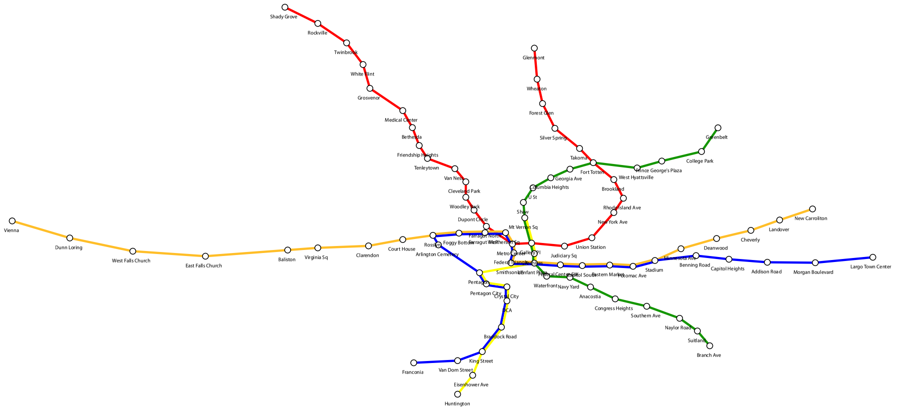

Fig. 11. Washington - Geographic Map

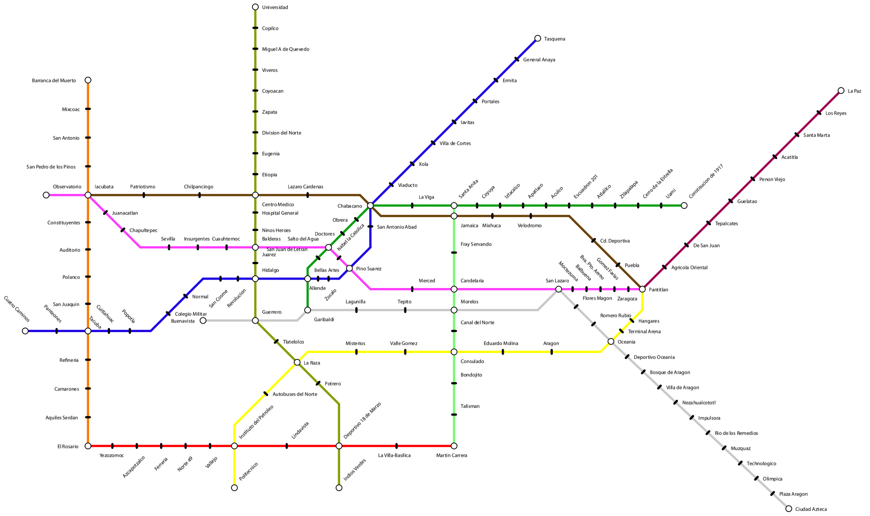

Fig. 14. Mexico - Rank 1 (Fitness = 2.667)

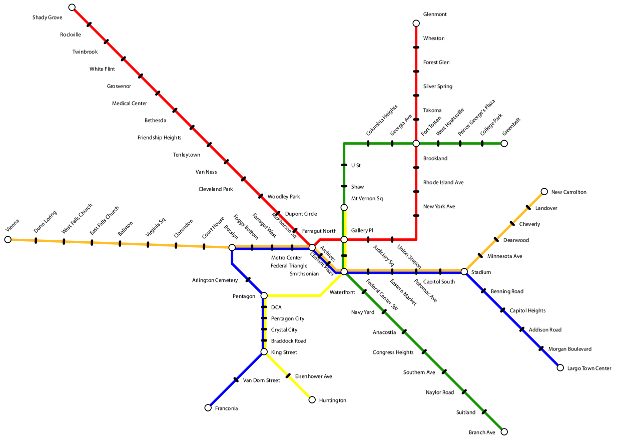

Fig. 13. Washington - Rank 1 (Fitness = 4.074)