Graph partitioning for network problems

Keith Ruddell and Andrea Raith
Department of Engineering Science
University of Auckland
New Zealand
k.ruddell@auckland.ac.nz

Abstract

Network models are used in numerous applications, including transportation, telecommunications and utilities. In such models one often wishes to suppress detailed local structure in order to emphasise global or long-distance behaviour which is most important to strategic decision-making. One desires a simpler network which preserves the topological and metric relations between regions. This may be achieved by judiciously aggregating nodes – treating several nodes as one. A scheme for aggregating nodes can be described by a partition of the graph.

We describe two of the many methods for partitioning graphs: spectral partitioning and an adaptation of k-means to a discrete metric space. We then discuss an application of partitioning and aggregation to the traffic assignment problem which resulted in often substantial reductions in computation time for urban transport models.

Key words: Graph Partitioning, Clustering, Network Aggregation, Traffic Assignment.

1 Introduction

Graphs and networks appear in a wide range of modelling applications, from the finite element method to the analysis of real-world utility and transportation networks.

Due to the computational complexity of many algorithms that operate on graph models, there is an incentive to use simpler graphs if the savings in computational cost can be said to outweigh the loss of accuracy. Several disciplines have developed algorithmic methods for partitioning graphs in line with varying objective functions. However, one may not know in advance which properties of the full graph one would like to keep, so there is no clear way of choosing an objective function to minimise over possible simplifications of the graph.

While researching new ways to initialise the traffic assignment problem, we used nearly a dozen graph partitioning methods to determine the aggregation of nodes in the traffic network. By solving the aggregated version of the problem and then carrying out a disaggregation procedure we could arrive at a better initial solution for the full problem in the sense that it required less overall computation time. We describe two of the graph partitioning methods and how they were applied to this aggregation method for traffic assignment.

For the purposes of this paper, *network* is a pair $(\mathcal{N}, \mathcal{A})$, consisting of a node set and arc set. Given a network $(\mathcal{N}, \mathcal{A})$, the underlying graph (V, E) — with vertices and edges — is formed by taking $V = \mathcal{N}$ and $E := \{\{u, v\} : (u, v) \in \mathcal{A}\}$, *i.e.* by forgetting the direction on the arcs. There are often parameters such as length and capacity attached to the arcs of a network. We will discuss the partitioning of graphs but it is easy to see how the same ideas extend to networks.

2 Two approaches to partitioning

One can classify graph partitioning methods into two broad approaches depending on whether local proximity or global connectedness is the main concern.

The first approach is to group nodes together based on their distance from one another. This could be called a 'bottom-up' approach. An example is the k-means algorithm, a fixed-point heuristic that gives a Voronoi-type partition which chooses a centre for each 'cluster' (part of the partition) and then minimises the mean distance from a zone to its nearest centre.

The second approach is to look for natural divisions of the network into tightly internally connected subregions with few external links. This is a 'top-down' approach where a natural first step is to divide the graph into two large regions. This approach is highly dependent on the large-scale topology of the graph in question. An objective that entails this approach is to partition the nodes into regions with as few inter-region links as possible. This is the k-way cut problem (or one of its variants) and can be solved approximately by repeated bisection (either through a minimum cut algorithm like Edmonds-Karp or through spectral partitioning) and adjustment (Kernighan-Lin algorithm).

2.1 The Voronoi problem on a graph

We define the *network Voronoi node diagram* on a graph as a subset $\mathcal{I}$ of the vertices, called the *generating set*, together with the set of *Voronoi cells* defined by

$$\mathcal{V}(p) := \{v \in \mathcal{N} \ : \ \mathrm{d}(p, v) \leq \mathrm{d}(q, v) \ \forall q \in \mathcal{I}\} \quad \text{for } p \in \mathcal{I}, \tag{1}$$

where d is the shortest path metric (using edge lengths if these are defined). If a tie-breaking rule is used to place nodes in $\mathcal{V}(p) \cap \mathcal{V}(q)$ in one side or the other, then $\mathcal{V}(\mathcal{I})$ forms a partition of the vertices.

Problem 1 (Centroidal Voronoi problem). Given a graph (V, E) and a natural number k, find a generating set $\mathcal{I} \subset V$ to minimise

$$\sum_{i \in \mathcal{I}} \sqrt{\sum_{v \in \mathcal{V}(i)} \mathrm{d}(i, v)^2}, \tag{2}$$

subject to

$$|\mathcal{I}| = k. \tag{3}$$

This network version of the problem has received less attention than the Euclidean-space versions (Okabe et al. 2000). The k-means method (also known as Lloyd's

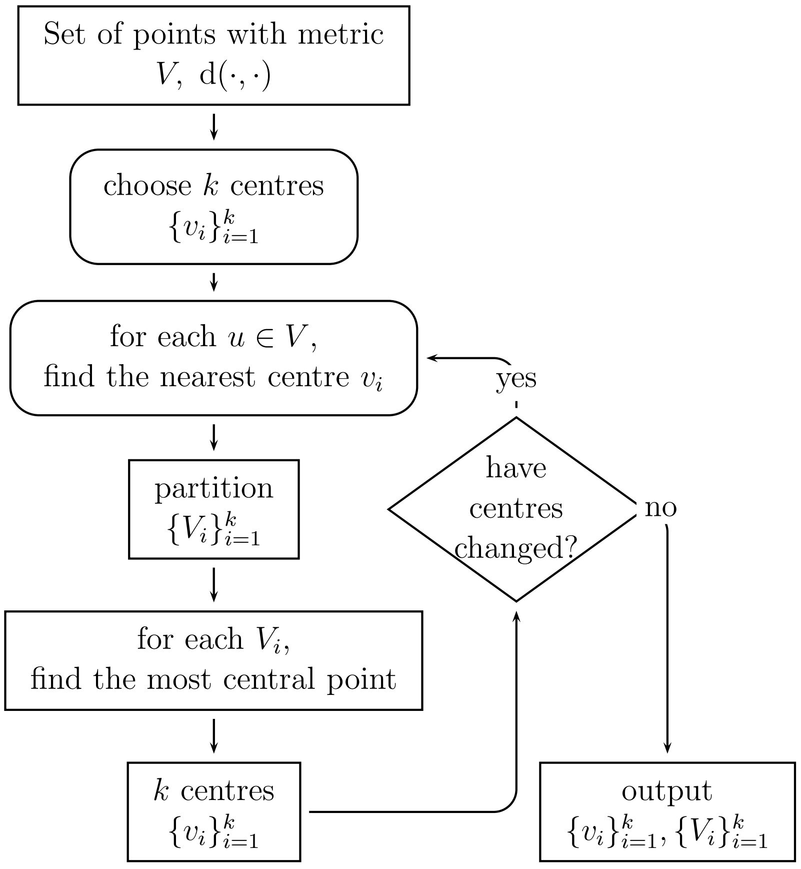


FIGURE 1: Flow chart of the k-means algorithm

algorithm), is a fast method for finding local minima of (2). Figure 1 shows the steps in the algorithm.

The fixed point that it finds is a local minimum of the objective (2), but not necessarily a global minimum. In a discrete space the algorithm should find a fixed point, provided it doesn't oscillate (a judicious choice of tie-break function for nodes equidistant from two centres should assure no oscillations). Our algorithm (fig. 1) is an adaptation from the classical k-means algorithm in a continuous space. The textbook example is where V is a bounded subset of $\mathbb{R}^n$ and $\mathrm{d}(\cdot,\cdot)$ is the Euclidean metric (Tan, Steinbach, and Kumar 2005, p. 496). In that situation the algorithm only reaches a fixed point in the limit, but it does converge reasonably fast (Okabe et al. 2000); in our tests we found that our network adaptation of k-means converged in about a dozen iterations.

In order to avoid a poor local optimum, we run the algorithm many (in the traffic assignment example 1000) times from randomly chosen initial centres.

2.2 Spectral Partitioning

The balanced minimum k-way cut problem is the standard model for balanced parallel computing problems, and also an example of the connectivity-focused approach to graph partitioning. If for instance one wishes to divide up a finite element method computation between multiple parallel processors, then one wants parts of roughly the same size but with as few interconnections as possible.

Problem 2 (Balanced k-way cut problem). Given a graph (V, E) and two integers k and b, find k non-empty subsets $V_i \subset V$ to minimise

$$\sum_{i=1}^{k} \deg_{\text{out}}(V_i), \tag{4}$$

subject to the balancing constraint

$$\left| |V_i| - \frac{|V|}{k} \right| \leq b \quad \forall i. \tag{5}$$

The objective (2) is the k-way generalisation of the objective of the classical minimum cut problem. To preclude the trivial solutions where all but one of the V_i are singletons containing low degree vertices, constraint (5) is imposed. A solution to this problem can be approached by first cutting the graph in two, and then bisecting the parts until there are k of them.

A balanced two-way cut can be achieved by spectral analysis of a special matrix. There are several matrices associated with graphs and networks. The *adjacency matrix* $\boldsymbol{A}$ of a graph on n vertices is a $n \times n$ symmetric binary matrix where

$$\boldsymbol{A}_{ij} = \begin{cases} 1 & \text{if there is an edge from vertex } i \text{ to vertex } j \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The *degree matrix* of a graph (V, E) is a diagonal matrix with entries $\boldsymbol{\delta}_{ii} = \deg(v_i)$. The *graph Laplacian* is defined by

$$\boldsymbol{L} = \boldsymbol{\delta} - \boldsymbol{A}, \tag{7}$$

following Newman (2010).

Problem 3 (Minimum cut problem, integer program formulation). Given a graph with Laplacian $\boldsymbol{L}$ and an integer b, find a vector $\boldsymbol{s} \in \mathbb{R}^n$ to minimise

$$\boldsymbol{s}^T \boldsymbol{L} \boldsymbol{s}, \tag{8}$$

subject to

$$|s_i| = 1 \qquad \forall i \text{ and} \tag{9}$$

$$\boldsymbol{1}^T \boldsymbol{s} \leq b. \tag{10}$$

If the constraint (9) is relaxed, then the problem can be solved analytically, by spectral analysis (hence the name of the method). The optimal solution of the relaxed problem is an eigenvector of $\boldsymbol{L}$, and the objective is proportional to the associated eigenvalue λ, thus we seek the smallest non-zero eigenvalue of $\boldsymbol{L}$ and one of its eigenvectors. To obtain a feasible solution to Problem 3, we project the eigenvector onto the set $\{+1, -1\}^n$, while respecting constraint (10).

Once we have obtained a k-way balanced cut, it can then be refined by the Kernighan-Lin algorithm. The Kernighan-Lin (Kernighan and Lin 1970) is a hill-climbing heuristic for improving the cut-value of a k-way balanced cut. It works by examining all pairs of nodes (u, v) where u and v are on opposite sides of the cut, and considering the change in the cut-value if u and v were swapped. At each iteration it swaps a sequence of nodes that improves the cut value.

3 Partitioning and Aggregation Applied to Traffic Assignment

The traffic assignment problem forms an indispensable part of many transportation planning models (Patriksson 1994). It models the distribution of traffic on a regional transport network as an equilibrium between fixed demand for travel and the increasing cost (in time) of travel imposed by congestion. Given a traffic network subject to congestion and given demand for travel between origins and destinations within the network, a user equilibrium is a pattern of flow that satisfies this demand and where no user can reduce their travel time by taking a different path. This equilibrium condition can be framed as a minimisation problem with non-linear objective — the Traffic Assignment Problem (TAP). Our formulation is adapted from Patriksson's (1994) book.

In a traffic network, traffic flows to and from a special set of z-many non-traversable nodes called *zone centroids*, denoted $\mathcal{Z}$. The *demand matrix* or *trip table* is a $z \times z$ matrix $\mathbf{D}$ with non-negative entries. The *arc cost* functions (also known as volume-delay or link-flow functions) $t_a : \mathbb{R}^+ \to \mathbb{R}^+$ are positive, non-decreasing and convex for all $a \in \mathcal{A}$. These functions may represent travel time, cost or a combination of these. The variables in our formulation are the route flows h_r. We write $\boldsymbol{h}$ for the vector of all route flows.

Example 4. Figure 2*A* shows a very simple traffic network. Nodes are black spots. Links are straight lines that meet at nodes. The letters are zone centroids and the dashed lines are *centroid connectors*, artificial arcs connecting the zone centroids to the rest of the network.

We may formulate TAP as the minimisation problem below.

Problem 5 (Traffic Assignment Problem (TAP)). Given a traffic network $(\mathcal{N}, \mathcal{Z}, \mathcal{A}, \boldsymbol{t})$ and a demand matrix $\mathbf{D}$, find a path flow vector $\boldsymbol{h}$ to minimise

$$T(\boldsymbol{f}) := \sum_{a\in\mathcal{A}} \int_0^{f_a} t_a(s)\ ds \tag{11}$$

subject to

$$f_a(\boldsymbol{h}) = \sum_{r\in\mathcal{R}:a\in r} h_r \qquad \forall a \in \mathcal{A}, \tag{12}$$

$$\sum_{r\in\mathcal{R}_{pq}} h_r = \mathbf{D}_{p,q} \qquad \forall (p,q) \in \mathcal{Z}^2 \text{ and} \tag{13}$$

$$h_r \geq 0 \qquad \forall r \in \mathcal{R}. \tag{14}$$

All algorithms for the solution of TAP are iterative. Starting from an initial feasible set of route flows, traffic is shifted so as to reduce (11) incrementally. We focus on path-based algorithms that explicitly record route flows throughout the procedure. The algorithm we will use to test the effectiveness of centroid aggregation and disaggregation is path equilibration, due to Leventhal, Nemhauser, and Trotter (1973). Traditionally, the initial feasible solution calculated by routing all traffic along the path that is shortest in uncongested conditions, the all-or-nothing assignment.

As even the best TAP algorithms can take hours to solve on models of large cities, we would like to find ways to reduce the computational cost of TAP so that more detailed networks can be used. This is where graph partitioning, in the form of our centroid aggregation method, comes in.

3.1 The centroid aggregation method

The centroid aggregation (CA) method works by first solving a simplified traffic assignment problem with some zone centroids grouped together — all zones in a group treated as part of a larger *super-zone*. The partition of $\mathcal{N}$, which of course we have calculated using one of the above graph partitioning methods, determines which zones are grouped together. Under this grouping, we obtain a simplified network and trip matrix.

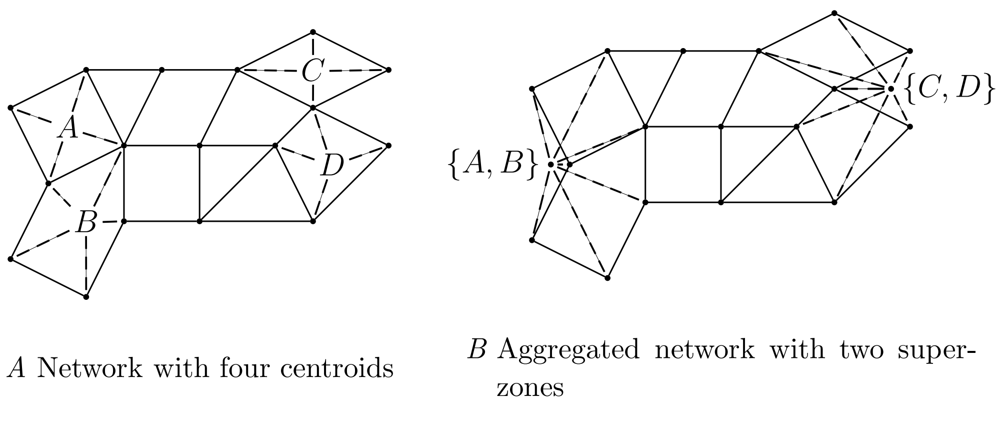
To find an equilibrium solution in the aggregated network we use all-or-nothing assignment for an initial solution followed by path equilibration. Finally, a solution in the full network is derived from the aggregated problem solution. This last step is called disaggregation.

3.2 Aggregation

To simplify the traffic assignment it is only necessary to aggregate the zones where flows originate and terminate. Having decided on a partition of the network, we identify all zones that fall in the same part, *i.e.* if X is a partition of $\mathcal{N}$ then the aggregated zone set is defined by $\hat{\mathcal{Z}} := \{Y \cap \mathcal{Z} \ : \ Y \in X\}$.

The demand matrix for the aggregated problem is computed by

$$\hat{\mathbf{D}}_{\hat{p},\hat{q}} := \sum_{\substack{p\in\hat{p}\\ q\in\hat{q}}} \mathbf{D}_{p,q} \quad \forall(\hat{p},\hat{q}) \in \hat{\mathcal{Z}}^2. \tag{15}$$



A Network with four centroids

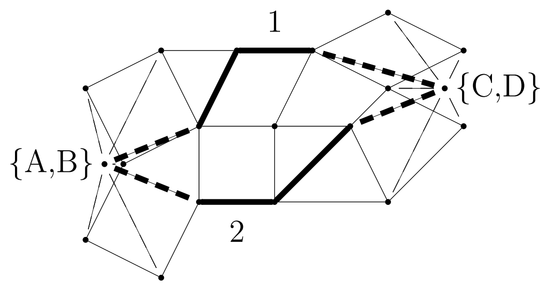
B Aggregated network with two super-zones

FIGURE 2: Zone aggregation

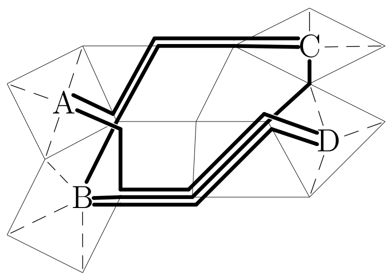
By way of example, figure 2*A* shows a network with four zones. Centroid connectors are shown as dashed lines. In figure 2*B*, the zones have been aggregated under the partition $\{\{A, B\}, \{C, D\}\}$.

3.3 Disaggregation

Once we have solved TAP on the aggregated network, the next step is to derive a detailed path flow solution from the aggregated equilibrium solution.



A Active routes in equilibrium from $\{A, B\}$ to $\{C, D\}$



B Active routes between untangled OD pairs

Figure 3: Allocating flow to OD pairs and resulting routes in the full network

The problem of dividing the flow from one super-zone to another among the different origin-destination pairs can be expressed as an instance of the transportation problem. The transportation problem finds the cheapest way to transport commodities from sources to sinks. It assumes that the cost of transportation from a given source to a given sink is linear in the amount transported and independent of all other variables.

The sources are the routes in the simplified solution, supplying the flow that they carry. We define the cost of using route r to get from p to q as the distance from p to the first non-zone node along r plus the distance from the last non-zone node to q. Solving the transportation problem by the network simplex method gives a 'basic feasible solution' in which all but $m + n + 1$ source-sink pairs have zero allocation, where m and n are the numbers of sources and sinks (Bertsekas 1991).

Figure 3*A* shows the active paths in the equilibrium solution of the simplified network of figure 2 and figure 3*B* shows the resulting untangled routes in the original network.

Once we have applied the untangling procedure to every aggregated origin-destination pair, the only unsatisfied demands are between zones grouped in the same super-zone. This traffic does not appear in the aggregated network, so needs to be assigned from scratch — we use all-or-nothing assignment.

From our untangled solution an iterative TAP algorithm should converge much quicker than from a standard all-or-nothing initial solution.

3.4 Results

We test our algorithm on three medium-sized networks from the website of (Bar-Gera 2013), representing Barcelona, Chicago and Winnipeg. The computer used

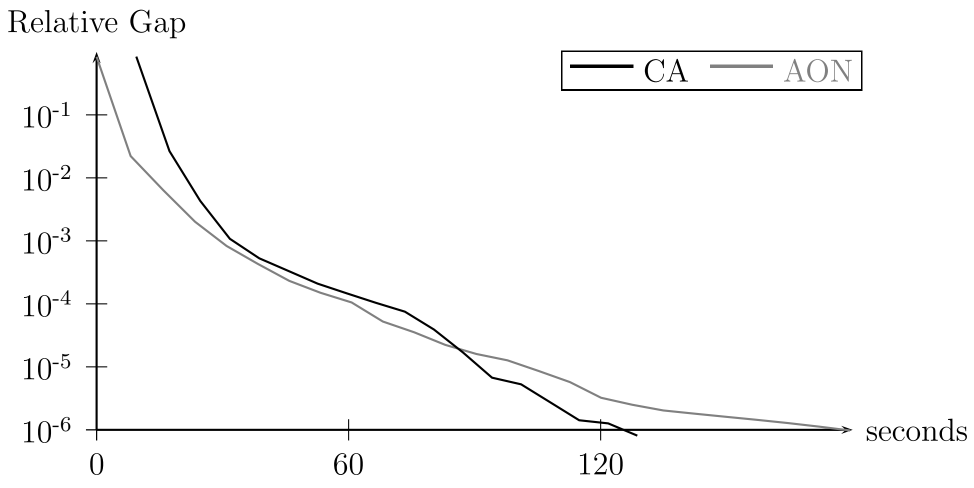


FIGURE 4: Relative gap vs CPU time, for Winnipeg test network, best case

had an Intel® Core$^{\text{TM}}$2 Duo CPU E8400 @ 3.00GHz $\times$ 2 processor with 4GB RAM running Ubuntu 12.04. All algorithms were implemented in C++. Our program automates the entire Centroid Aggregation method, including partitioning of the network, aggregation and disaggregation. The methods and results are more fully discussed in Ruddell's (2013) masters thesis.

The METIS package (Karypis 2013) was used to provide spectral partitioning and Kernighan-Lin routines. We implemented the network k-means algorithm ourselves.

As the purpose of graph partitioning is here to save computation time by spending some small amount to produce a better initial solution than that given by the near-instantaneous all-or-nothing (AON) initialisation, we have recorded the times to equilibrate to within a relative gap of 10^{-6} using the path equilibration algorithm. Each partitioning method is tested across a range of coarseness parameters. The more zones in each super-zone the coarser the aggregation. For both of the partitioning methods described in this paper the coarseness parameter is the mean super-zone size, the average number of zones per super-zone.

Figure 4 shows the progression of relative gap over time for a typical run of path equilibration with CA and with standard all-or-nothing initialisation. Rather than starting with a small relative gap, the CA-initialised run stars with a larger relative gap but converges at a faster rate and more steadily than the run initialised by all-or-nothing assignment.

Using METIS, we tested spectral partitioning, refined by the Kernighan-Lin algorithm. In table 1, we see that this gave very good results for Barcelona under the path equilibration algorithm. Also, CA with spectral aggregation maps gave a reduction in computation time for Winnipeg and Chicago at all levels of coarseness.

Using our adaptation of k-means, table 2 shows the modest but consistent savings in computation time.

Overall we sometimes see significant improvement in run time and even in the worst cases, the increase in time required is minimal. Our disaggregation mapping belongs to a family of methods for projecting equilibrated solutions of similar traffic assignment problems onto good feasible starting points for equilibration. What we have shown is that aggregating zones and solving a simplified TAP can provide a better starting point for equilibration on the full network than all-or-nothing, even though the disaggregated solution may be far from equilibrium in relative gap.

Table 1: Time (s) to converge to relative gap $< 10^{-6}$
Spectral/Kernighan-Lin aggregation by METIS

		Mean super-zone size					
	AON	2	3	4	5	6	7
Barcelona	22.4	21.5	19.3	18.6	18.9	13.7	19.5
Chicago	180	160	179	155	153	145	144
Winnipeg	71	55	59	42	62	53	50

Table 2: Time (s) to converge to relative gap $< 10^{-6}$
k-Means partition with free-flow travel distance

		Mean super-zone size					
	AON	2	3	4	5	6	7
Barcelona	22.4	18.9	20.9	18.8	19.4	19.7	19.4
Chicago	180	147	148	167	167	180	151
Winnipeg	71.0	36	29	33	64	44	63

4 Conclusion

Graph partitioning is a family of problems with many different objective functions. If one has a network problem that may benefit from graph partitioning, it is difficult to decide which member of the family is most appropriate, especially when, as with centroid aggregation for TAP, there is not clear connection between the overall objective (providing a better initial solution) and the objectives of graph partitioning methods. As the number of possible partitions grows exponentially with the number of nodes, it may be best to set aside the search for an optimal partition and settle for a method that provides, at little cost, a partition that is bound to have some merit.

Acknowledgements

Many thanks to Olga Perederieieva for the use of her traffic assignment computer code.

References

Bar-Gera, Hillel. 2013. Transportation Test Problems. `http://www.bgu.ac.il/~bargera/tntp/`.

Bertsekas, Dimitri P. 1991. *Linear network optimization - algorithms and codes.* Cambridge, Mass: MIT Press.

Karypis, George. 2013. *METIS:A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 5.1.0.* Minneapolis: University of Minnesota, Department of Computer Science & Engineering.

Kernighan, B. W., and S. Lin. 1970. "An Efficient Heuristic Procedure for Partitioning Graphs." *The Bell system technical journal* 49 (1): 291–307.

Leventhal, T., G. Nemhauser, and L. Trotter. 1973. "A Column Generation Algorithm for Optimal Traffic Assignment." *Transportation Science* 7 (2): 168–176.

Newman, M. E. J. 2010. *Networks: An Introduction.* Oxford; New York: Oxford University Press.

Okabe, Atsuyuki, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. 2000. *Spatial tessellations: Concepts and applications of Voronoi diagrams.* 2nd. Probability and Statistics. New York: Wiley.

Patriksson, Michael. 1994. *The Traffic Assignment Problem: models and methods.* Netherlands: VSP.

Ruddell, Keith. 2013. "Hot-Starting the Traffic Assignment Problem by Zone Aggregation and Disaggregation." Master's thesis, University of Auckland.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. 2005. *Introduction to Data Mining.* Reading, MA: Addison-Wesley.