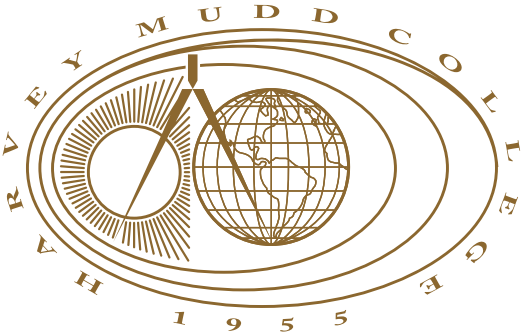
2008

Characteristics of Optimal Solutions to the Sensor Location Problem

David Morrison
Harvey Mudd College



Characteristics of Optimal Solutions to the Sensor Location Problem

David R. Morrison

Susan Martonosi, Advisor

Kimberly Tucker, Reader

May, 2008

HARVEY MUDD COLLEGE

Department of Mathematics

Abstract

Congestion and over-saturated roads pose significant problems and create delays in every major city in the world. Before this problem can be addressed, we must know how much traffic is flowing over the links in the network. We transform a road network into a directed graph with a network flow function, and ask the question, "What subset of vertices (intersections) should be monitored such that knowledge of the flow passing through these vertices is sufficient to calculate the flow everywhere in the graph?" To minimize the cost of placing sensors, we seek the smallest number of monitored vertices. This is known as the Sensor Location Problem (SLP). We explore conditions under which a set of monitored vertices produces a unique solution to the problem and disprove a previous result published on the problem. Finally, we explore a matrix formulation of the problem and present cases when the flow can or cannot be calculated on the graph.

Contents

List of Figures

Acknowledgments

I would like to thank my advisor, Prof. Susan Martonosi, for her willingness to let me tackle this project, and for all of her support in my research. I also want to thank Dr. Kimberly Tucker for agreeing to be my second reader even though she had never met me. Additionally, I would like to thank Claire Connelly for her technical knowledge and help. Finally, I would like to thank my best friend and fianceé, Karen Brown, for her help with my pictures and her tremendous support in all of the work I've done.

Chapter 1

Introduction

The issue of traffic congestion is a significant problem in today's society. Almost every city in the world has to deal with clogged roads and rush-hour traffic, and chances are good that even if you don't own a car, you've been in a car during a complete stand-still on a major thoroughfare. The issue is more than just an annoyance; delays caused by heavy traffic conditions play a major role in the loss of time and money on the part of both individuals and corporations. According to the Urban Mobility Report, published by the Texas Transportation Institute, the average American spent 38 hours waiting in congested traffic in 2005. Furthermore, the average person wasted 26 gallons of fuel in 2005 due to poor traffic conditions. All of this adds up to $710 per person per year wasted sitting in traffic, or $78.2 *billion* wasted nationwide (Schrank and Lomax, 2007). Thus, it is not in the least surprising that understanding why poor traffic conditions occur is one of the most prominent issues faced by urban scientists and engineers today.

However, in order to fully understand these and related issues, we must first be able to answer the question of which roads are congested. This could give rise to improved data for online utilities such as SigAlert, which provides real-time traffic data for the freeway system in southern California. In addition, many portable GPS systems now have the capability to display up-to-date information about traffic conditions in a road network. As these systems are only useful when the data that drives them is accurate, we turn our attention to the problem of determining the distribution of cars in a given road network.

As it turns out, this question is non-trivial. We can gain information (locally) about this distribution by placing sensors on roads or intersections. These sensors fall into two categories. Passive sensors provide information

only about the number of cars travelling over a section of road, whereas active sensors can record vehicle type, speed, or other determining characteristics. In either case, these sensors only provide counts at a given position or in some cases, limited information on the vehicle's origin and destination (for example, automated payment sensors on toll roads (Gentili and Mirchandani, 2005)). We seek a procedure for transforming this local information about traffic counts into a global traffic distribution. Clearly, we can gain complete information about traffic conditions by placing sensors on every link in the network; however, because even passive traffic sensors are costly and require maintenance, this solution quickly becomes prohibitively expensive.

Fortunately, we note that many of the sensors in the above case would be completely redundant: if some of the sensors were removed, the traffic flow over those roads could still be calculated from the remaining information. In order to minimize expenses, we want to minimize the number of such sensors placed, while still being able to determine the distribution of cars in the network. We ask the question, "In a particular road network, where should the minimum number of sensors be located in order to determine traffic conditions everywhere?"

We transform this problem into a problem in graph theory, and apply results from the field to aid in obtaining a solution. The road network is turned into a directed graph, with vertices representing intersections, and directed edges between the vertices representing roads. The traffic flow over the roads is described by a network flow function on the edges of the graph. The question then becomes, "What is the smallest subset of edges such that knowledge of the flow along them uniquely determines the flow everywhere on the graph?"

A final piece of information that aids in solving this problem comes from the knowledge of *turning ratios* at all intersections in the network. The turning ratio for a given intersection is simply the percent of incoming traffic flow that leaves in a particular direction. Most city planning committees or other administrative offices keep records of these turning ratios, and they can be easily determined by counting cars on a street corner for a few days. As it turns out, this assumption will be instrumental in finding a solution to the problem.

With these assumptions, we can define the Sensor Location Problem and attempt to determine a solution to the posed questions. In Chapter 2, we discuss earlier results and work done on this problem. Chapter 3 formally defines the Sensor Location Problem and analyzes some methods of determining if a given set of edges solves the problem. In this chapter, we

present a counterexample to some work done previously on the problem. We conclude in Chapter 4 by discussing open problems and plans for future work.

Chapter 2

Prior Work

Before attacking the problem of determining sensor location in a road network, we first explore some of the work done by other people on the problem. One of the first papers to discuss this subject is (Yang and Zhou, 1998). The paper focuses on a related problem, that of estimating origin/destination (O/D) matrices. O/D matrices are matrices with potential travel starting locations along the rows and possible destinations along the columns. The $(i, j)^{th}$ entry in the matrix is the total number of trips from origin i to destination j taken by all commuters in the network. Many cities keep track of these matrices to help direct traffic, but they are often poorly calculated, as they are based on incomplete knowledge of the network's traffic patterns. Naturally, determining these matrices is a very similar problem to calculating traffic flow. In fact, O/D matrices can be estimated well if the traffic flow is known everywhere, and traffic flow can be calculated more easily if an accurate O/D matrix is known.

As these problems are very closely related, we first state the sensor location problem in terms of origin/destination matrices: what is the minimum number of sensors needed (and where should they be located), to determine the value for each O/D pair (i, j) in the above matrix? Four general guidelines for locating sensors on a given road network are described in Yang and Zhou (1998):

1. The O/D Covering rule: some fraction of the trip for each O/D pair must be covered by at least one sensor.

2. The Maximal Flow Fraction rule: for a given O/D pair, the sensors should be located on the links with the largest fraction of flow for that O/D pair.

3. The Maximal Flow Intercept rule: given a set of links to (potentially) monitor, choose the ones that have the greatest number of O/D pairs traversing them.

4. The Link Independence Rule: monitor links whose flows are not dependent on each other.

These rules are based on common sense rather than mathematical principles. It is clear that in any distribution of sensors the first and fourth rules should always be satisfied. That is, if no sensor monitors any portion of some O/D trip, any amount of traffic could be flowing over it with no method of determining it. Likewise, if the information provided by one sensor is entirely dependent on the data provided by another, there is no reason to monitor both links (Yang and Zhou, 1998).

The other two rules are much more difficult to satisfy, however. In fact, they often come into conflict with each other, since the links with the largest fraction of flow for a given O/D pair very often will not be the same as the links that intercept many different O/D pairs. In other words, roads with high volumes of traffic will generally have most of the traffic going to the same place or lots of traffic going to many different places, but not both. Thus, rules two and three cannot always both be completely satisfied. Yang and Zhou (1998) use them as tunable parameters for a heuristic search function to find the best locations for sensors.

While we do not specifically incorporate these rules into our formulation of the problem, they are an excellent description of the properties that we would like any solution to the problem to have. As it turns out, these rules will be best satisfied as a side effect of our search for a solution to the Sensor Location Problem; rule four will play a particularly important role.

A more mathematical approach to the sensor placement problem is described in Gu and Jia (2005). This paper defines the concept of an "edge control set." This is a set of edges S in a graph such that if two network flow functions f_1 and f_2 are given, and $f_1(S) = f_2(S)$, then the functions are equal everywhere. That is to say, if the flow on every edge in S is known, the flow everywhere in the graph can be determined by simply finding a flow function on the graph that matches the values determined on S. An algorithm is described that can find a solution set S given two conditions on the graph:

i) All arcs of the graph lie on some directed cycle

ii) There are no sources or sinks in the graph

However, these conditions are fairly limited, since they do not model physical conditions very accurately.

These issues are addressed in a series of three papers on which most of our work is based. The first formal definition of the Sensor Location Problem (hereafter referred to as SLP) is presented in Bianco et al. (2001). In a follow-up paper, (Bianco et al., 2006), SLP is proven to be NP-complete by a reduction to a similar problem called the Dominating Paths Problem (DPP). Additionally, the paper describes a polynomial-time algorithm for finding a solution on paths, cycles, and combs. Finally, DPP is given a more in-depth analysis in Confessore et al. (2005), where an approximation algorithm is developed for the general case.

There are several things that should be noted about this series of papers. First, the papers assume that vertices are monitored instead of edges; a monitored vertex yields information about flow going into and out of it along all links incident to it, and thus is equivalent to monitoring all of the edges incident to it. There are certain cases when all of these edges do not need to be monitored, and the number of sensors can be reduced. However, unless otherwise mentioned, we proceed with the convention of monitoring vertices for the remainder of this paper.

Secondly, some of the authors' early results are incorrect, and thus much of the work described in the papers is based on faulty reasoning. We present some counterexamples to their work in Section 3.2. However, we believe that many of their results are accurate in spite of this; for example, the polynomial-time algorithm for paths and cycles described seems to be a correct algorithm, though the derivation is not correct. For this reason, we base much of our work off of these papers, and adopt many of their conventions and notation.

A number of other papers approach slightly different aspects of this problem. As already mentioned, Gentili and Mirchandani (2005) discuss the more challenging problem of locating active sensors on a network. In addition, Berman et al. (1995) discuss the problem of locating discretionary services (that is, gas stations, grocery stores, or similar services) on a road network to intercept the maximum number of potential customers. However, both of these problems require a significant amount of underlying knowledge about where to place passive sensors, and so we primarily focus on SLP instead of addressing these issues.

In the next chapter, we formally define SLP, and describe some approaches to determining when a given set of sensors can uniquely determine the flow on the network.

Chapter 3

The Sensor Location Problem

3.1 Definitions

In an effort to model traffic patterns over a road network, we transform the network into a directed graph $D = (V, E)$ by replacing intersections with vertices and roads with directed edges between them. It is natural to consider "two-way" road networks—that is, networks in which all the streets have traffic flowing in both directions. This will simplify some results. However, this is not a restrictive assumption, and we believe that all results should be extensible to the general case.

To model this assumption, we assume that the edge relation on our directed graph is symmetric: that is, if $u, v \in V$ and $uv \in E$, then $vu \in E$ as well. However, it is important to note that the graph is *not* undirected; this is because the flow on edge uv in general will not be the same as the flow on edge vu. To highlight this distinction, we refer to the graph D as a **two-way directed graph**.

We represent the traffic flowing over the roads by a network flow function $f : E \to \mathbb{R}$ that satisfies the flow conservation law at each vertex $v \in V$:

$$\sum_{e \in v^-} f_e - \sum_{e \in v^+} f_e + S_v = 0, \tag{3.1}$$

where v^- is the set of arcs with head at v, and v^+ is the set of arcs with tail at v. S_v is the **balancing flow** at vertex v. These satisfy the balancing flow law:

$$\sum_{v \in V} S_v = 0 \tag{3.2}$$

The vertices with non-zero balancing flows are the places where mo-

torists are coming from and going to. If the balancing flow at a vertex is positive, the vertex is a **source**; if it is negative, it is a **sink**. These vertices are so important that we give them a special name:

Definition 3.1.1 (Bianco et al. (2006))**.** *If the balancing flow of a vertex v is non-zero, we call v a* **bound vertex***; the set of all bound vertices is denoted B.*

The remainder of the vertices in the graph, for which $S_v = 0$ are called **transport vertices***.*

For example, in Figure 3.1, $B = \{b, d, e, f\}$, and the set of transport vertices is $\{a, c\}$.

In order to gain information about the network flow function f, sensors are placed at various locations in the road network. We assume that sensors are placed at intersections (vertices), and that if an intersection is monitored, we know the number of cars entering and leaving the intersection along each road connected to the intersection. We refer to the vertices corresponding to intersections with sensors as **monitored vertices**, and denote the set of monitored vertices M. It will also be useful to consider the neighbor set of

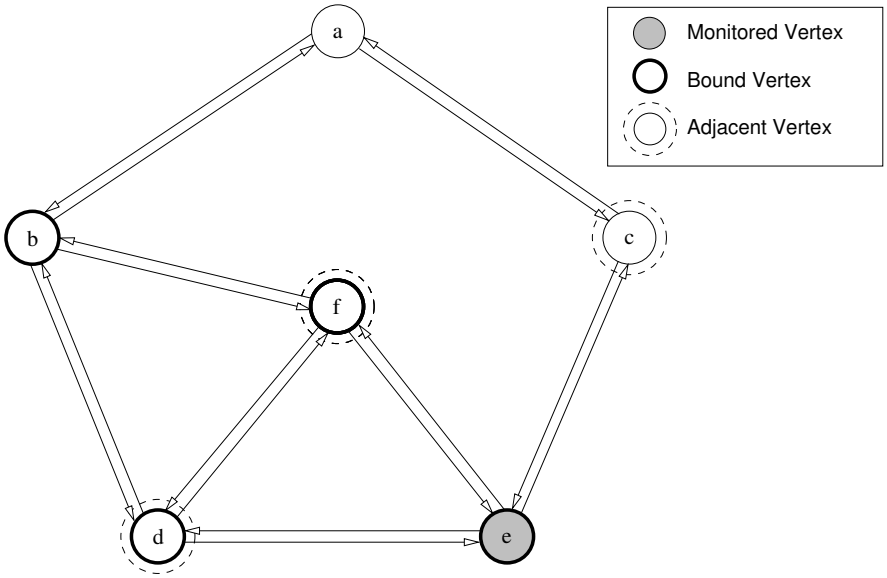


Figure 3.1: In this two-way directed graph, vertices b, d, e, and f are bound; vertices a and c are transport vertices. We put a sensor at vertex e, which gives the adjacent vertices to M as c, d, and f.

the monitored vertices:

Definition 3.1.2. The neighbor set of M, *denoted* $A(M)$ *is the set of vertices that are adjacent to vertices in* M *in* D. *That is,*

$$A(M) = \{v \in V \mid vm \in E \text{ for some } m \in M\}$$

In Figure 3.1, if we let $M = \{e\}$, then $A(M) = \{c, d, f\}$.

We finally assume knowledge of the turning ratios at every intersection in the network. For every vertex v, we associate with each outgoing arc vu a real number $c_{vu} \in [0, 1]$, which is the fraction of the total outgoing flow from v that leaves on arc vu. That is,

$$f_{vu} = c_{vu} \sum_{e \in v^-} f_e. \tag{3.3}$$

Then, we can write the flow of all outgoing arcs from v in terms of a single known outgoing arc:

$$f_{vu} = \frac{c_{vu}}{c_{vw}} f_{vw} \tag{3.4}$$

for any $vu \in v^+$, and f_{vw} known.

The ratio of the c's in equation (3.4) will be used often, so we introduce the following definition:

Definition 3.1.3. *The* **turning factor** *of edge* vu *with respect to some edge* vw, *denoted* α_{vu}, *is the ratio of the turning ratio of edge* vu *to edge* vw *(in general it will be clear from the context what edge the ratio is taken with respect to):*

$$\alpha_{vu} = \frac{c_{vu}}{c_{vw}}$$

Then equation (3.4) becomes

$$f_{vu} = \alpha_{vu} f_{vw} \tag{3.5}$$

for some known flow f_{vw}. The turning factors, therefore, are simply the amount by which we have to multiply a known outgoing arc to determine the flow on an unknown outgoing arc.

We note that, by definition, if we know the flow over some set of monitored vertices M, we can then apply knowledge of the turning factors to determine the flow over all outgoing edges of $A(M)$.

The Sensor Location Problem (SLP) can then be defined in terms of the above notation:

Definition 3.1.4 (The Sensor Location Problem, Bianco et al. (2006))**.** *Given a two-way directed graph $D = (V, E)$ together with a network flow function f and a set of bound vertices B, what is the smallest set M of monitored vertices such that knowledge of all turning ratios and the values of f on M uniquely determines f everywhere on D?*

In particular, note that the problem does not just seek the size of M; it also asks which vertices of the graph belong in M. Since this question is highly dependent on the structure of the graph, it is desirable to find an easy condition by which we can check to see if M uniquely determines f. This is the issue to which we will direct the majority of our attention.

3.2 Determining Uniqueness on Small Examples

The natural question to which we now turn our attention is, "When does a set M yield a unique solution to the flow on a graph?" We note that if two vertices in $A(M)$ are adjacent, then we know all of the flow on the edges between them. This is due to the fact that we know the flow over one outgoing edge from each of them, and thus by the turning ratios can determine the outgoing flow on the edges between them. These edges provide no further information, so we give them a name and then proceed to throw them away:

Definition 3.2.1 (Bianco et al. (2001))**.** *The* **combined cutset of** M*,* C_M*, is the set of edges in the subgraph of D induced by $M \cup A(M)$.*

That is to say, the combined cutset of M is the set of edges between all vertices in M, between M and $A(M)$, and between all vertices in $A(M)$. For example, Figure 3.2 shows the graph in Figure 3.1 with the combined cutset and M removed.

Given any set of monitored vertices M, we would like to know if we can determine the flow on the graph from the knowledge gained at these vertices. The combined cutset of M seems to be a natural place to start; since we know the flow over all edges in C_M, as well as at every vertex in M, they provide no additional information. Thus, removing C_M and M from the graph should remove no necessary information from the problem. We can then use the remaining flow coming out of $A(M)$ to determine the flow everywhere else in the graph.

Removing the edges in C_M and the vertices in M from the graph produces a subgraph D' that is often, but not always, disconnected. Let the

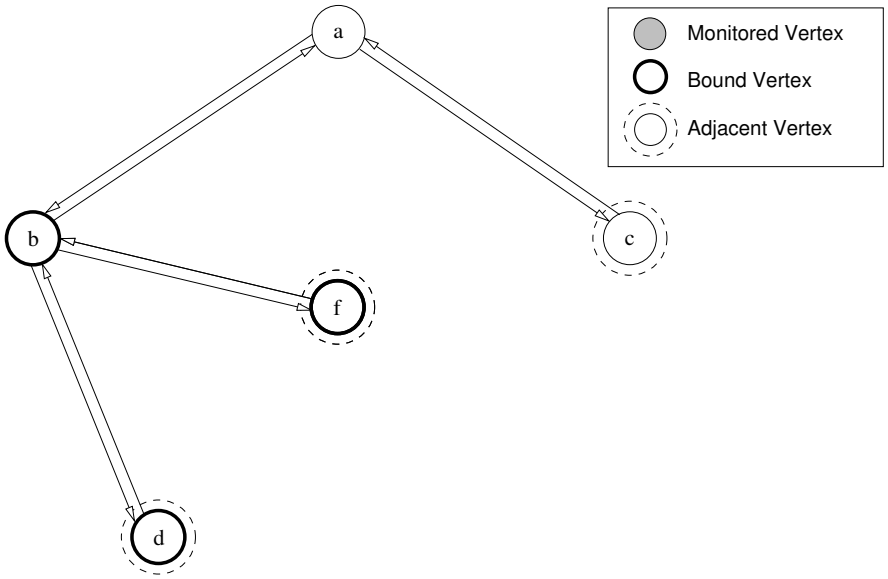


Figure 3.2: The graph from Figure 3.1 shown with the combined cutset C_M and monitored vertex set removed.

i^{th} component of the subgraph be labeled D'_i, the set of bound vertices in the component be labeled B'_i, and the set of (originally) adjacent vertices in the component be labeled $A'_i(M)$. Bianco et al. (2001) present a proof of the following theorem, which stems from the removal of C_M. We will show that this theorem is incorrect.

Theorem 3.2.2 (Bianco et al. (2001))**.** *Given a set of monitored vertices M, the flow on a digraph D can be uniquely determined everywhere if and only if for every connected component of $D' = (V - M, E - C_M)$,*

$$|B_i| \leq |A_i(M)|.$$

We present two examples, one in which we can calculate the flow everywhere in the graph, and one in which the conditions of the theorem are satisfied, but the flow cannot be calculated. For both examples, we assume that the turning ratios of unknown vertices are evenly distributed among the outgoing edges. That is, if $\delta^+(i)$ is the outgoing degree of vertex i, the turning ratio $c_{ij} = 1/\delta^+(i)$.

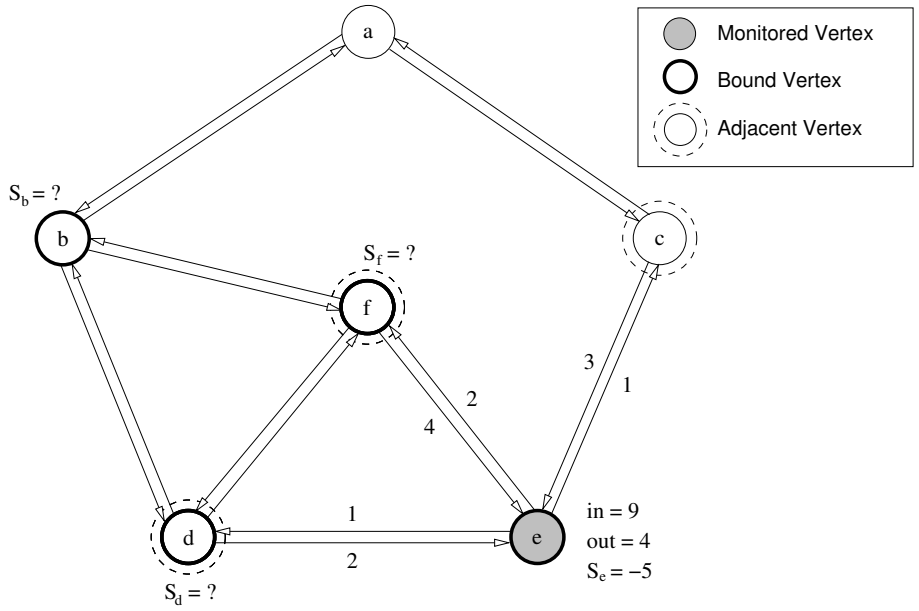


Figure 3.3: The graph from Figure 3.1, analyzed in Example 1. One can check that removal of the combined cutset C_M (shown in Figure 3.2) satisfies the conditions in Theorem 3.2.2, and we can calculate the flow everywhere in the graph (see Figure 3.4).

Example 1

Consider again the network shown in Figure 3.1. Suppose that by monitoring vertex e, we determine the flow values over edges incident to e to be those shown in Figure 3.3. We now wish to calculate the flow on the remainder of the graph. We note first that $A'(M) = \{c, d, f\}$ and that $B' - M = \{b, d, f\}$. Examining the only connected component of D' shows that $|A'(M)| = |B' - M| = 3$, so by Theorem 3.2.2, there should be a unique solution to the flow f.

To gain some intuition about the problem, we propagate flow along edges until we get stuck, and examine the remaining system of equations. First, consider the flow at vertex c. Since we are assuming even distribution of turning ratios, we get that $f_{ca} = 3$ and $f_{ce} = 3 = \frac{1}{2}(1 + f_{ac})$. Solving yields $f_{ac} = 5$. Because the turning ratios are evenly split at vertex a, this shows that $f_{ab} = 5$, as well. We then calculate $f_{ba} = 7$ to satisfy equation (3.1) at vertex a.

Next, we apply equation (3.4) at vertices d, f, and b. Since the turning ratios are evenly distributed and $f_{de} = 2$, this implies that f_{df} and f_{db} are 2 as

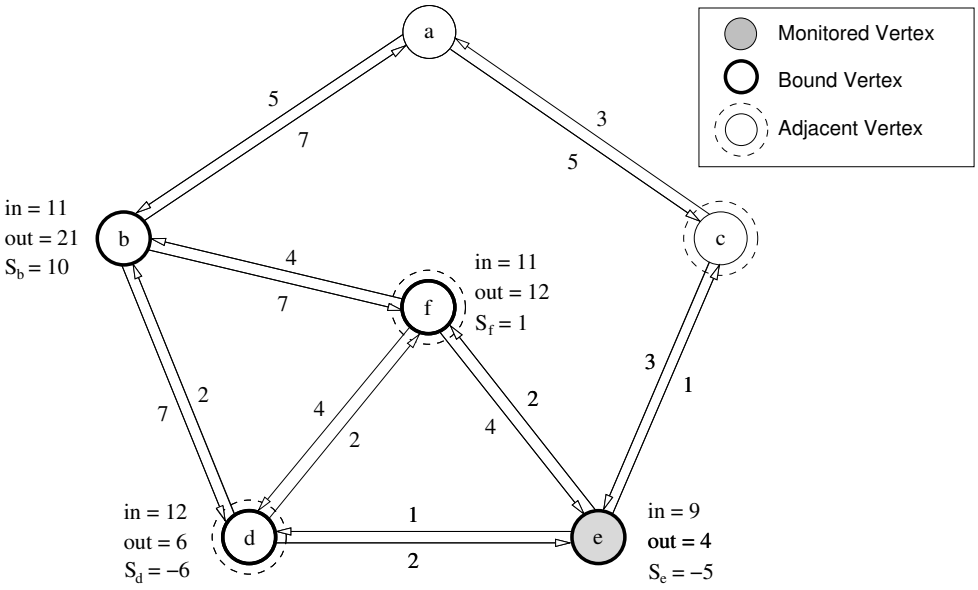


Figure 3.4: The final flow calculated on the graph for Example 1.

well. Similarly, we find that $f_{fd} = f_{fb} = 4$, and $f_{bd} = f_{bf} = 7$. This produces the final graph shown in Figure 3.4. It is easy to check that equation (3.1) is satisfied at every vertex, and the balancing flows satisfy equation (3.2).

Alternatively, we could write down the system of equations by applying equation (3.1) at every vertex, and substituting in for the known flow values at e. By choosing a "canonical" outgoing edge from each vertex, we can use the turning factors with respect to this canonical edge (as in equation (3.5)) to reduce the number of unknown variables. As it turns out, for this example, the system of equations is linearly independent. (The idea of choosing a canonical edge for each vertex is important, and will reappear in Section 3.3).

Example 2

In the next example that we consider (shown in Figure 3.5), the set of monitored vertices M satisfies the conditions presented in Theorem 3.2.2, but does not produce a linearly independent system of equations. By monitoring vertex a, we note that in the only connected component of D', $|A'(M)| = |B' - M| = 2$; thus, by Theorem 3.2.2, we should be able to determine f uniquely on all edges.

If we observe 4 units of flow along edges ab, ba, ad, and da, however, by following the above procedure, we cannot calculate the flow on edges ed and fd. In particular, notice that if $f_{ed} = x \in [0, 8]$, then setting $f_{fd} = 8 - x$ yields a solution that satisfies equations (3.1) and (3.2), and thus the solution to the network flow function is not unique. In fact, if one writes out the system of equations given by monitoring vertex a, it quickly becomes apparent that the system is not linearly independent; therefore, the graph in Figure 3.5 is a counterexample to Theorem 3.2.2.

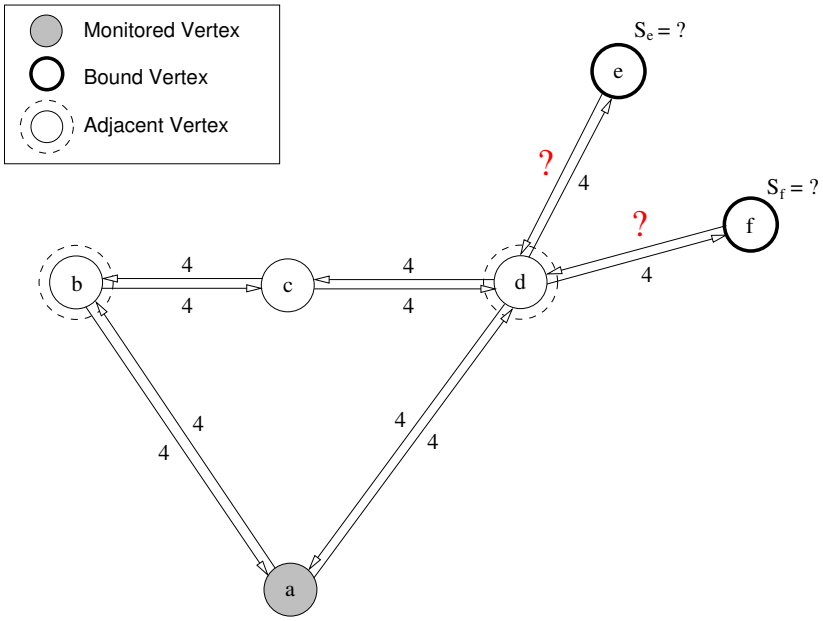


Figure 3.5: The graph analyzed in Example 2. We monitor vertex a in the above graph; then $C_M = \{ab, ba, ad, da\}$, and it is easy to check that the graph with C_M and M removed satisfies the conditions in Theorem 3.2.2. However, we cannot calculate f_{ed} or f_{fd} from the known information.

In fact, there are many such counterexamples. As it turns out, the theorem is incorrect even when the graph is a tree, or when the inequality in the theorem is strict (these examples are not hard to find). The proof of Theorem 3.2.2 relies on the fact that there exists a system of equations with fewer unknown variables than equations. However, it neglects to take into account the possibility that these equations may not be linearly independent. In terms of the conditions presented at the beginning of Chapter 2, the proposed solution does not satisfy condition four.

In an attempt to better understand the circumstances under which Theorem 3.2.2 fails, we next examine the problem as matrix equation problem, where the entries in the matrix represent the adjacencies within the graph.

3.3 SLP and Invertible Matrices

There are two common ways of representing a graph concisely as a matrix. The first is with the **adjacency matrix**, **A**. This matrix encodes all of the adjacencies in a graph as follows:

Definition 3.3.1. *The* **adjacency matrix** *of a directed graph D is a $|V| \times |V|$ matrix where the $(i,j)^{th}$ entry is* 1 *if there is an edge from vertex i to vertex j, and* 0 *otherwise.*

The other very common matrix representation is the **incidence matrix**, denoted **M**:

Definition 3.3.2. *The* **incidence matrix** *of a directed graph D is a $|V| \times |E|$ matrix where the $(i,j)^{th}$ entry is -1 if the tail of edge j is at vertex i,* 1 *if j's head is at vertex i, and* 0 *if j is not incident to i.*

It is clear by inspection that the incidence matrix is nothing more than the coefficient matrix of the system of flow conservation laws:

$$\mathbf{M}\,\mathbf{f} + \mathbf{s} = \mathbf{0}, \tag{3.6}$$

where $\mathbf{f}$ is the $|E|$-length flow vector and $\mathbf{s}$ is the $|V|$-length vector of balancing flows.

However, in its current form, equation (3.6) is not overly useful. For one thing, there are unknown variables in both the $\mathbf{f}$ and $\mathbf{s}$ vectors. Additionally, we note that this equation does not account for our assumption that we know the turning ratios in the graph, and thus contains many more unknown variables than are necessary. Therefore, we construct a modified incidence matrix taking into account this additional (known and unknown) information. We denote this matrix **E**, and create it in the following manner (for the remainder of section, we focus on the graph shown in Figure 3.3 as a running example):

1. We index the vertices of a two-way directed graph arbitrarily, and associate with each vertex $i \in V$ an arbitrary outgoing edge e_i; this edge will be the "canonical" edge representative for vertex i discussed at the end of Example 1 in Section 3.2. Since we know the turning

ratios of the graph, the flow over all other outgoing edges from i can be written in terms of f_{e_i}. Note that, using the turning factors with respect to e_i, we can write the flow over edge ij as $f_{ij} = \alpha_{ij} f_{e_i}$. If $ij = e_i$, then $\alpha_{ij} = 1$.

For the example graph in Figure 3.3, we choose the following edges to be our canonical representatives for each vertex: ab, ba, ca, db, ed, and fb. We also assume that all turning ratios are equal except at vertex e, since monitoring has revealed the turning ratios here are different. That is to say that $\alpha_{ij} = 1$ for all $i, j \in V$ except when $i = e$. Since the flow from e to f is twice that of the flow over e's canonical edge (ed), we have that $\alpha_{ef} = 2$; however, since the flow over ec equals f_{ed}, $\alpha_{ec} = 1$.

2. We first take into account our knowledge of the turning ratios to reduce the number of columns in the incidence matrix from $|E|$ to $|V|$. To do this, let $\mathbf{E}^*$ be a $|V| \times |V|$ matrix. We label the rows of the matrix with the vertex labels, and label the columns with the canonical edge representatives for each vertex. Then, we have that the $(i, jk)^{th}$ entry of $\mathbf{E}$ is given by

$$\mathbf{E}^*_{i,jk} = \begin{cases} \alpha_{ji} & \text{if } e_j = jk \text{ and } ji \in E \\ -\sum \alpha_{i\ell} & \text{if } e_i = jk \text{ (note that this implies that } i = j). \\ 0 & \text{if } i \text{ and } j \text{ are not connected} \end{cases}$$

The first condition accounts for the flow over incoming edges to the vertices in the graph; the second condition says that the total outgoing flow relative to the vertex's canonical edge is the negative sum of all the α's at that vertex.

To further illustrate the reasoning behind this matrix, we fill in two rows of $\mathbf{E}^*$ for the graph of our example, and then present the completed matrix:

i) First, consider the row corresponding to vertex a. The canonical edge for vertex a is ab, and $\alpha_{ab} = \alpha_{ac} = 1$. Therefore, $\mathbf{E}^*_{a,ab} = -2$. Additionally, since there are incoming edges to a from b and c, we have that $\mathbf{E}^*_{a,ba} = \alpha_{ba} = 1$, and $\mathbf{E}^*_{a,ca} = \alpha_{ca} = 1$. There are no other connections from or to vertex a, so all other entries in the row are 0.

ii) Next, consider the row corresponding to vertex d. The canonical edge for vertex d is db, and $\alpha_{db} = \alpha_{de} = \alpha_{df} = 1$ (that is to say,

the flows over all outgoing edges from vertex d are equal). Thus, $\mathbf{E}^*_{d,db} = -3$. Also, to account for the incoming edges to vertex d, we see that there is an edge from vertex b to d, and that the flow over bd is the same as the flow over b's canonical edge; that is, $\alpha_{bd} = 1$ (similarly for ed and fd). Since no other vertices are connected to d, all other entries in the matrix are 0.

Here is the completed matrix for our example:

$$\mathbf{E}^* = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} ab & ba & ca & db & ed & fb \end{array} \\ \begin{pmatrix} -2 & 1 & 1 & 0 & 0 & 0 \\ 1 & -3 & 0 & 1 & 0 & 1 \\ 1 & 0 & -2 & 0 & 1 & 0 \\ 0 & 1 & 0 & -3 & 1 & 1 \\ 0 & 0 & 1 & 1 & -4 & 1 \\ 0 & 1 & 0 & 1 & 2 & -3 \end{pmatrix} \end{array}$$

At this point, notice that we have simply "collapsed" the incidence matrix $\mathbf{M}$ into a square matrix by taking advantage of the turning ratios. Additionally, we note that if the turning ratios are split evenly (that is, $c_{ij} = c_{ik}$ for all vertices i, j, and k), the matrix is nothing more than the adjacency matrix $\mathbf{A}$ with the negative outdegree of the vertices along the matrix diagonal.

It is important to note that $\mathbf{E}^*$ has linearly dependent rows; the sum of all the rows is 0 (this can be seen by examining any given column uv, and noting that the sum of all positive entries is simply $\sum \alpha_{uk}$). However, we note that any $|V| - 1$ of the rows are linearly independent. This will be proved in Theorem 3.3.4.

3. Finally, we absorb the unknown balancing flows in the vector $\mathbf{s}$ into this matrix; Add $|B|$ columns to $\mathbf{E}^*$, such that each column corresponds to the balancing flow at a single bound vertex i, and the column has a 1 in the i^{th} row and 0's everywhere else. Note that the dimensions of the matrix are now $|V| \times (|B| + |V|)$. We update the vector $\mathbf{f}$ to contain the additional unknown S_i variables. Now note that the i^{th} row of this matrix corresponds to equation (3.2) at vertex i. We denote this matrix $\mathbf{E}$.

Below we have constructed the $\mathbf{E}$-matrix for the graph presented in Figure 3.3.

$$\mathbf{E} = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccccccc} ab & ba & ca & db & ed & fb & S_b & S_d & S_e & S_f \end{array} \\ \begin{pmatrix} -2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -3 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 2 & -3 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

We can now update equation (3.6) to be (slightly) more useful:

$$\mathbf{E}\,\mathbf{g} = \mathbf{x}, \tag{3.7}$$

where $\mathbf{g}$ is a vector containing all of the unknown variables (both the flow over edge representatives and balancing flows), and $\mathbf{x}$ is a vector containing all of the known information about the system (determined by the monitored vertices).

For our example, we have that $\mathbf{g}^T = (f_{ab}, f_{ba}, f_{ca}, f_{db}, f_{ed}, f_{fb}, S_b, S_d, S_e, S_f)$. At this point, since we are monitoring no vertices, $\mathbf{x} = \mathbf{0}$, but that will change momentarily. However, the values contained in $\mathbf{x}$ are not important for our purposes, since they do not affect the solvability of the system.

In any case, the system as it stands in equation (3.7) is clearly unsolvable, since the matrix $\mathbf{E}$ contains fewer rows than columns. Once we are given a set M of monitored vertices, we can then adjust the matrix and check to see if a solution exists (i.e., if the rank of the new matrix is at least the number of columns—this will imply that all of the unknown variables can be solved for). We therefore rewrite equation (3.7) once more to take into account the knowledge provided by the monitored vertices. We construct the flow calculation matrix $\mathbf{E}_M$ as follows:

4. Since we know the flow over all incoming edges to any vertex $m \in M$, as well as the total outgoing flow from vertex m, we can remove row m from the matrix. Additionally, for each vertex i such that m and i are adjacent, we know the flow along the edge from m to i; therefore, we can also remove column e_m from $\mathbf{E}$.

   In order to maintain the system of equations in equation (3.7), we must also remove f_{e_m} from $\mathbf{g}$, remove the m^{th} entry from $\mathbf{x}$, and for each vertex i adjacent to m, add $-\alpha_{ij} f_{mv}$ to the i^{th} entry of $\mathbf{x}$. Thus, the vector $\mathbf{x}$ is updated with the known flow values determined by M.

Finally, we remove any columns corresponding to balancing flows that are known due to monitoring. At this point, we have done nothing more than substitute known outgoing flow values into the system of flow balance equations, and removed trivial equations (that is, equations that have no unknowns in them).

5. We now must take into account the knowledge of the incoming flow to vertices in M. For each vertex in $A(M)$, we note that we know the flow over its canonical edge, since the flow value is just a (known) multiple of the flow value from $A(M)$ to M.

   Thus, for each vertex $a \in A(M)$, we add a row to the matrix that contains a 1 in column e_a and a 0 in every other column. We also update the a^{th} entry of $\mathbf{x}$ to contain the flow value over a's canonical edge (f_{e_a}). This represents an equation of the form $f_{e_a} = \frac{1}{\alpha_{am}} f_{am} = x$.

6. We call this completed matrix the **flow calculation matrix**, and denote it $\mathbf{E}_M$.

We can now write equation (3.7) in terms of the additional knowledge afforded by the monitored vertices:

$$\mathbf{E}_M \mathbf{g}' = \mathbf{x}', \tag{3.8}$$

where $\mathbf{g}'$ and $\mathbf{x}'$ have been updated to reflect the known flows. If we can solve equation (3.8), then we can uniquely determine the flow everywhere on the graph. For example, here is the flow calculation matrix and corresponding equation for the graph in Figure 3.3, with $M = \{e\}$:

$$\begin{pmatrix} -2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -3 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} f_{ab} \\ f_{ba} \\ f_{ca} \\ f_{db} \\ f_{fe} \\ S_b \\ S_d \\ S_f \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \\ -1 \\ -2 \\ 3 \\ 2 \\ 4 \end{pmatrix}$$

It is easy to check that $\text{rank}(\mathbf{E}_M) = 8$ for the above matrix, and thus the columns are linearly independent; this implies that equation (3.8) is solvable

for the graph in Figure 3.3. Indeed, plugging the system into a computer algebra system easily gives the same results as presented in Section 3.2.

Notice that $\mathbf{E}_M$ has a *local* block structure that will be useful when referring to various parts of the matrix ($\mathbf{E}_M$ also has a global block structure based on the partitions of the graph. This will be discussed in more detail later; at that time we will define exactly what we mean by "local" and "global" block structures. For now, it suffices to know that this block structure exists over the whole matrix):

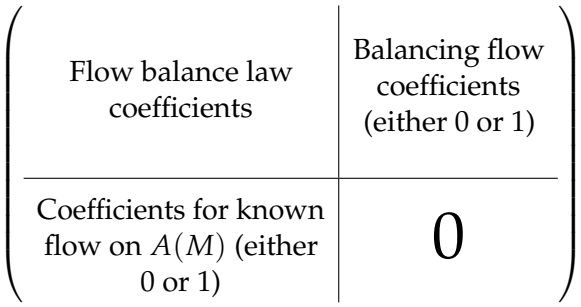


Figure 3.6: The local block structure present in the matrix $\mathbf{E}_M$.

Notice that the columns of balancing flow coefficients on the right half of the matrix were added in step 3 of the formation of the matrix. The rows of known flow coefficients for $A(M)$ on the bottom half of the matrix were added in step 5.

Because we are concerned about the linear dependencies (or lack thereof) in E_M, it will be useful to prove some results about when rows are linearly independent, and which rows are linearly independent. The first result dictates when rows from the top half of $\mathbf{E}_M$ are linearly independent. We first show that removing one row from the matrix $\mathbf{E}^*$ produces a set of rows that are linearly independent (recall that all of the rows in $\mathbf{E}^*$ combine to 0, and thus are not linearly independent). This is proved using a result found in Bertsimas and Tsitsiklis (1997) about the incidence matrix of a graph:

Theorem 3.3.3. *Let* $\mathbf{M}$ *be the incidence matrix of a connected digraph* D*. The truncated incidence matrix* $\tilde{\mathbf{M}}$ *formed by removing the last row of* $\mathbf{M}$ *has linearly independent rows.*

The result follows from arranging the rows in $\tilde{\mathbf{M}}$ to form an upper-triangular matrix, and then observing that the determinant of the matrix is

non-zero. The same result holds for $\mathbf{E}^*$:

Theorem 3.3.4. *Let $\tilde{\mathbf{E}}^*$ be the matrix formed by removing the last row from the matrix $\mathbf{E}^*$. Then $\tilde{\mathbf{E}}^*$ has linearly independent rows.*

Proof. Let $|V| = n$, and label the vertices of the graph from $1, ..., n$. Now, suppose that $\tilde{\mathbf{E}}^*$ did not have linearly independent rows. Let $\mathbf{E}^*_1, ..., \mathbf{E}^*_n$ be the rows of $\mathbf{E}^*$. Then there exists some set of (not all zero) coefficients $b_1, ..., b_{n-1}$ such that $b_1\mathbf{E}^*_1 + ... + b_{n-1}\mathbf{E}^*_{n-1} = \mathbf{0}$.

Multiplying both sides of this equation by the flow variable vector $\mathbf{f}^*$ (containing only the variables corresponding to the flow over canonical edges) produces $b_1\mathbf{E}^*_1\mathbf{f}^* + ... + b_{n-1}\mathbf{E}^*_{n-1}\mathbf{f}^* = 0$, and multiplying out yields the following equation:

$$\begin{gathered} b_1\mathbf{E}^*_{1,e_1}f_{e_1} \ + \ ... \ + \ b_1\mathbf{E}^*_{1,e_n}f_{e_n} \ + \\ b_2\mathbf{E}^*_{2,e_1}f_{e_1} \ + \ ... \ + \ b_2\mathbf{E}^*_{2,e_n}f_{e_n} \ + \\ ... \\ b_{n-1}\mathbf{E}^*_{n-1,e_1}f_{e_1} \ + \ ... \ + \ b_{n-1}\mathbf{E}^*_{n-1,e_n}f_{e_n} \ = \ 0 \end{gathered}$$

We note that each term $\mathbf{E}^*_{i,e_j}f_{e_j}$ where $i \neq j$, by definition becomes $\alpha_{ji}f_{e_j} = f_{ji}$. Similarly, each term of the form $\mathbf{E}^*_{i,e_i}f_{e_i}$ becomes $-(\sum \alpha_{ik})f_{e_i}$, where the sum is over all vertices adjacent to i. The sum simplifies further to $\sum -f_{ik}$ by applying the turning factor definition. Now, we can collect terms in the above expansion to get

$$\begin{gathered} (\pm b_1 \ \pm b_2)f_{1,2} \ + \ (\pm b_1 \ \pm b_3)f_{1,3} \ + \ ... \ + \ (\pm b_1 \pm b_{n-1})f_{1,n-1} \ + \\ (\pm b_2 \ \pm b_1)f_{2,1} \ + (\pm b_2 \ \pm b_3)f_{2,3} \ + \ ... \ + (\pm b_2 \ \pm b_{n-1})f_{2,n-1} \ + \\ ... \\ (\pm b_{n-1} \pm b_1)f_{n-1,1} \ + \ (\pm b_{n-1} \pm b_2)f_{n-1,2} \ + \ ... \ + \ (\pm b_{n-1} \pm b_{n-2})f_{n-1,n-2} \ + \\ (\pm b_1 f_{1,n} \pm b_n f_{n,1}) \ + \ (\pm b_2 f_{2,n} \pm b_n f_{n,2}) \ + \ ... \ + \ (\pm b_2 f_{n-1,n} \pm b_n f_{n,n-1}) \ = \ 0, \end{gathered}$$

where the sign of the first element in each pair of coefficients is opposite that of the second, but arbitrary based on the choice of canonical edges, and where $f_{i,j}$ is 0 if i and j are not connected. Since this equation must be true for all non-zero values of the flow function, we see that each of the coefficients of the above equation must be 0. By inspection, however, we see that each coefficient of the non-zero flow values is a linear combination of the entries

in that arc's column of the truncated incidence matrix $\tilde{\mathbf{M}}$. This implies that each column of $\tilde{\mathbf{M}}$ sums to 0 when multiplied with $(b_1, ..., b_{n-1})$. Conversely, if multiplying each column of $\tilde{\mathbf{M}}$ with $(b_1, ..., b_{n-1})$ produces 0, the above equation will be true. Thus, these statement are equivalent, and we have produced a set of coefficients for which the first $n-1$ rows of the graph's incidence matrix sum to 0. By Theorem 3.3.3, however, this is impossible; thus, the first $n-1$ rows of $\mathbf{E}^*$ must be linearly independent. ■

The addition of the $|B|$ extra columns in step 3 clearly does not affect any linear dependencies in the matrix, since each column contains exactly one non-zero entry. In other words, if there is a 1 in the i^{th} row of an added column, and there exists some set of coefficients for which the first $n-1$ rows of $\mathbf{E}$ combine to 0, all such coefficients b_i must be 0 (since there are no other non-zero entries in those columns to cancel out the 1 in row i), and the remaining coefficients must also be 0, since otherwise the rows of $\mathbf{E}^*$ would be linearly dependent. Therefore, removing a row from $\mathbf{E}$ creates a matrix with linearly independent rows.

It is clear, then, that the top half of the matrix in Figure 3.6 has linearly independent rows, since this matrix is nothing more than $\mathbf{E}^*$ with a few rows and columns removed. We now need to know under what circumstances the remaining rows of $\mathbf{E}_M$ can be added while retaining linear independence:

Theorem 3.3.5. *Let $W = \{\vec{v}_1, ..., \vec{v}_n\}$ be a set of vectors whose span has dimension m, with $m < n$. Also, suppose that $\vec{v}_1, ..., \vec{v}_k$ are linearly independent vectors in W with $k < m$; then there exists a linearly independent subset of W, denoted S, such that $\vec{v}_1, ..., \vec{v}_k \in S$ and $|S| = m$.*

Proof. We wish to show that without loss of generality, if we are forced to exclude $\vec{v}_1$ from S, the assumptions of the theorem are false. Therefore, Suppose that for every linearly independent set of $m-1$ vectors excluding $\vec{v}_1$, there exists some linear combination of these vectors that combines to form $\vec{v}_1$. Since the dimension of span(W) is m, and n is strictly greater than m, there exists at least one linearly independent set of vectors of size m that excludes $\vec{v}_1$. That is, there exist coefficients $c_1, ..., c_{m-1}$ and $d_2, ..., d_m$ (not all zero), as well as a set of vectors $\vec{u}_1, ..., \vec{u}_m$ such that $\vec{u}_i \neq \vec{v}_1$ and

$$\vec{v}_1 = c_1\vec{u}_1 + ... + c_{m-1}\vec{u}_{m-1} = d_2\vec{u}_2 + ... + d_m\vec{u}_m \tag{3.9}$$

Without loss of generality, assume that c_1 is the first non-zero coefficient for the $\vec{u}$'s. Then equation (3.9) implies that

$$c_1\vec{u}_1 = (d_2 - c_2)\vec{u}_2 + ...(d_{m-1} - c_{m-1})\vec{u}_{m-1} + d_m\vec{u}_m \tag{3.10}$$

Since not all coefficients can be the same or zero, this shows that $\vec{u}_1$ is a linear combination of the other vectors, contradicting our assumption that $\{\vec{u}\}$ is linearly independent. Therefore, there must exist some linearly independent set of $m-1$ vectors that does not combine to form $\vec{v}_1$, which implies that adding $\vec{v}_1$ to this set retains the independence. As this logic can be applied for all k vectors, we therefore have shown that some set S exists such that $\vec{v}_1, ..., \vec{v}_k \in S$ and $|S| = m$. ■

This theorem has an important corollary that will be quite useful in the proof of Theorem 3.4.3:

Corollary 3.3.6. *If* $\text{rank}(\mathbf{E}_M) = \#\{\textit{columns of } \mathbf{E}_M\}$*, then there exists a set of rows forming a square matrix such that the first* $|V| - |M|$ *rows corresponding to the flow balance laws in Figure 3.6 are contained in this set.*

Proof. Either $\mathbf{E}_M$ itself is square (in which case this statement is trivial), or $\mathbf{E}_M$ has more rows than columns. However, the rows of the matrix correspond to the set W and the dimension of their span is the number of columns of $\mathbf{E}_M$, since $\text{rank}(\mathbf{E}_M) = \#\{\text{columns of } \mathbf{E}_M\}$. By Theorem 3.3.5, there exists a set of linearly independent rows that contain the first $|V| - |M|$ rows and form a square matrix. ■

Thus, we see that if the matrix $\mathbf{E}_M$ has linearly independent columns (equivalently, if $\text{rank}(\mathbf{E}_M)$ equals the number of columns), we can calculate all of the unknown variables, and M determines the flow on the graph uniquely. On the other hand, if $\mathbf{E}_M$ does not have linearly independent columns, the flow cannot be uniquely determined. We note that $\mathbf{E}_M$ has $|V| - |M| + |B - M|$ columns, and $|V| - |M| + |A(M)|$ rows. This implies that a necessary condition for calculating flow is that $|A(M)| \geq |B - M|$; however, this is not a sufficient condition, as can be seen in Example 2 from Section 3.2. This is the flow calculation matrix for the graph in Figure 3.5:

$$\begin{array}{c} \\ b \\ c \\ d \\ e \\ f \\ f_{ba} \\ f_{dc} \end{array} \begin{array}{c} \begin{array}{ccccccc} ba & cb & dc & ed & fd & S_e & S_f \end{array} \\ \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

It is easy to check that the rank of this matrix is 6; note that with coefficients $2, 1, 3$, and -1 respectively, the first two rows and the last two rows combine to zero.

The above matrix framework suggests that we only need to figure out the balancing flows at every vertex in the graph in order to solve the above system. We note that, by knowing *a priori* which vertices are bound and which are transport vertices, we have determined S_v at some fraction of the vertices in the graph. That is, we have reduced the number of unknown variables from $2|V|$ to $|V| + |B|$, since $S_v = 0$ at all $v \notin B$.

This hints at an important related problem, that of determining which vertices in a network are bound. Almost any intersection could conceivably be a destination for a few motorists; unfortunately, in order to take advantage of the above observations, the number of bound vertices needs to be significantly less than the total number of vertices. Furthermore, if the number of bound vertices is large, the number of required sensors will in general also be large. A final complication is that some vertices may be sources during one portion of the day, and sinks during another (e.g., an office building would be a destination in the morning, and an origin in the afternoon).

As we shall see, though, determining the flow on a graph at a fixed point in time is dependent only on the location of the bound vertices, not on the sign of their balancing flows. That is, whether or not a particular vertex is a source or sink at a given time does not matter when calculating the flow at that time. Thus, we do not consider the issue of time-dependence of bound vertices. Likewise, we do not further address the issue of determining the location of bound vertices in a network in this paper, except to observe that any vertex that is only a destination for a few motorists can probably be treated as a transport vertex for the purposes of a simplifying model.

3.4 Characterization of Valid Solutions

Now that we have successfully turned SLP into a matrix problem, we can use this formulation to attempt to answer the question "When does a given set M produce a matrix $\mathbf{E}_M$ such that the rank of $\mathbf{E}_M$ equals the number of columns?" In order to (partially) answer this question we introduce the following definition:

Definition 3.4.1. *A* **B-path** *is a path starting at some bound vertex and ending at a vertex in* $A(M)$.

Based on some initial observations on small examples, it seemed that the following conjecture would prove to be correct:

Conjecture 3.4.2. *The flow associated with a digraph D and a set of bound vertices B can be uniquely determined everywhere from a set of monitored vertices M if and only if there exists a set $\mathcal{P}$ of $|B - M|$ disjoint B-paths.*

For instance, note that in Example 1 from Section 3.2, a set of 3 disjoint B-paths exists (shown in Figure 3.7), and as we have already shown, we can calculate the flow in this case. However, for Example 2 from Section 3.2, any set of 2 B-paths will be forced to intersect at vertex d, as in Figure 3.8. Thus, the number of disjoint B-paths is smaller than $|B - M|$ and we are unable to calculate the flow.

Unfortunately, it turns out that Conjecture 3.4.2 is only half-correct. If the number of disjoint B-paths is too low, then we will show that the flow cannot be calculated on the graph. However, there are a number of circumstances in which a set of at least $|B - M|$ vertex-disjoint B-paths does exist, but the flow on the graph still cannot be calculated. It appears that this occurs

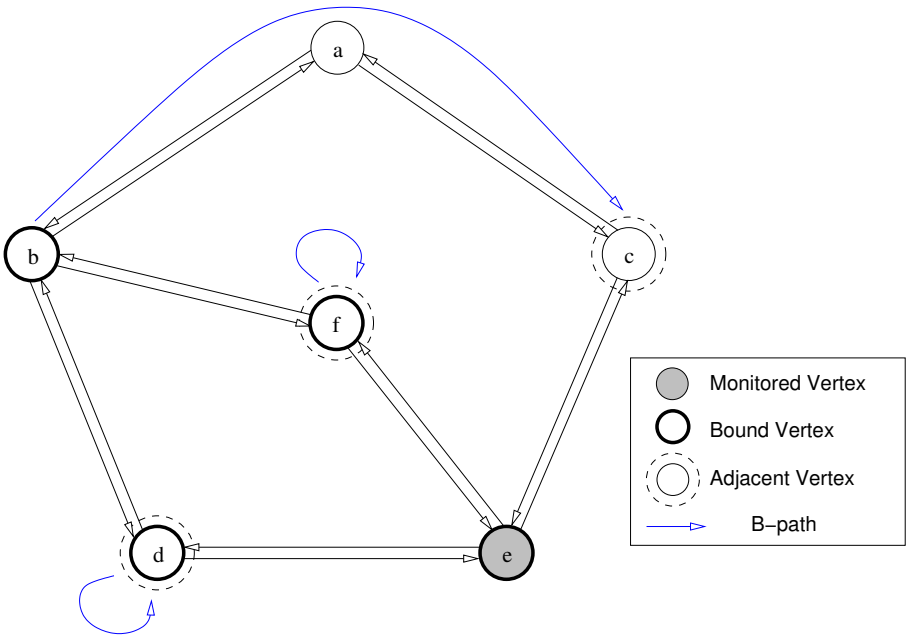


Figure 3.7: The graph for Example 1 in Section 3.2, shown with a set of disjoint B-paths (cf. Figure 3.4).

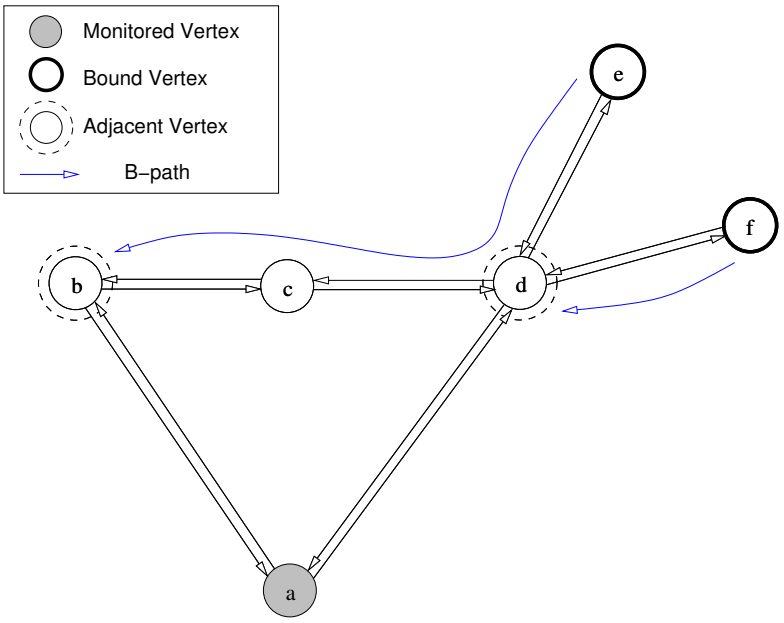


Figure 3.8: The graph for Example 2, Section 3.2, together with a set of B-paths. However, any 2 B-paths must pass through vertex d, so there is no set of $|B - M|$ disjoint B-paths associated with M (cf. Figure 3.5).

when there is some degree of symmetry of vertices in the graph around the monitored vertex set, but thus far we have not been able to figure out a condition on either the graph or the set M for when this occurs. Below we present two examples of when this occurs, and then spend the remainder of this section proving the reverse direction of Conjecture 3.4.2.

The first case that we consider is shown in Figure 3.9. Notice that there exist a number of automorphisms of the graph that leave the overall system alone (for example, vertices a and b could be switched, as could c and d). However, by adding an additional vertex (f) in Figure 3.10, we remove these automorphisms. We suspect that this may be a key to solving the problem, but initial efforts in this direction have yielded no results.

To see why these automorphisms may play some role in preventing determination of the flow, this is the flow calculation matrix for the graph shown in Figure 3.9:

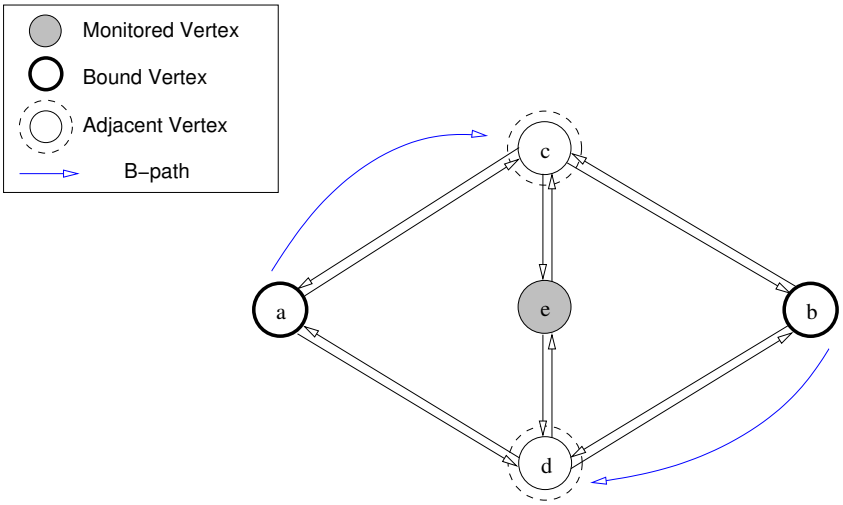


Figure 3.9: In this graph, we note that monitoring vertex e creates 2 disjoint B-paths; yet, the $\mathbf{E}_M$ matrix does not have linearly independent columns, and thus we cannot calculate the flow on the graph.

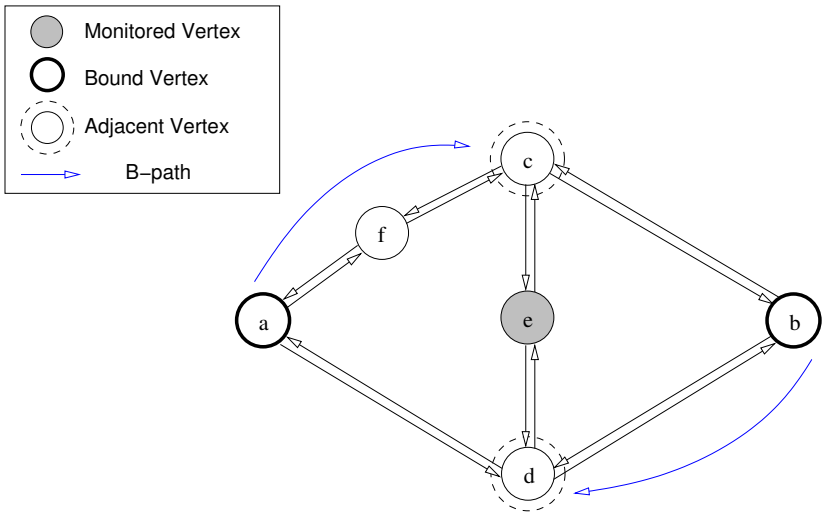


Figure 3.10: This graph is the same graph shown in Figure 3.9 with an additional vertex spliced into edge ac. The addition of this vertex allows the problem to be solved with the same monitored vertex set as before.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ f_{cb} \\ f_{da} \end{array} \begin{array}{c} \begin{array}{cccccc} ad & bc & cb & da & S_a & S_b \end{array} \\ \begin{pmatrix} -2 & 0 & 1 & 1 & 1 & 0 \\ 0 & -2 & 1 & 1 & 0 & 1 \\ 1 & 1 & -3 & 0 & 0 & 0 \\ 1 & 1 & 0 & -3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Notice that because there are connections between vertex c and both a and b, as well as vertex d and both a and b, we can take a linear combination of the last four rows of the above matrix to produce the **0** vector. Also note that any other vertex in the graph could be chosen as a monitored vertex set of size 1 that would allow the flow on the graph to be calculated everywhere. However, as shown in Figure 3.10, if we add a single vertex to the graph, we destroy the symmetry in the matrix, allowing the flow to be calculated by monitoring vertex e. This is the flow calculation matrix for the graph in Figure 3.10. Note that it has full rank:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ f \\ f_{cb} \\ f_{da} \end{array} \begin{array}{c} \begin{array}{ccccccc} ad & bc & cb & da & fa & S_a & S_b \end{array} \\ \begin{pmatrix} -2 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & -2 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & -3 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & -3 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

It is unclear at this time what about the structure of the graph causes this, or how to modify Conjecture 3.4.2 to correct for this problem. However, we can show that (at a minimum) the reverse direction of Conjecture 3.4.2 is true, and it is to this task that we devote the remainder of this section. As it seems to be more applicable, we state the contrapositive:

Theorem 3.4.3 (Statement A)**.** *Let $D = (V, E)$ be a two-way directed graph with bound vertex set B, and let M be a set of monitored vertices. If the flow can be calculated on the graph, then there must exist a set of disjoint B-paths with size at least $|B - M|$.*

Our first task in proving this will be to translate the proof into the language of matrices. We have already established that ability to calculate

flow is equivalent to the flow calculation matrix having full rank. Since it is difficult to see and trace paths through $\mathbf{E}_M$, however, we also wish to translate the condition of Theorem 3.4.3 into a statement about disconnecting cuts. Specifically, we note that the number of vertex-disjoint B-paths cannot be larger than the size of a minimum disconnecting set C between B and $A(M)$. If we add a vertex s and add edges from s to all vertices in $A(M)$, and add a vertex t with edges from all vertices in $B - M$ to t, we can apply Menger's theorem to discover that the number of vertex-disjoint B-paths is exactly equal to the size of the minimum disconnecting set C. Therefore, we have the following restatement of Theorem 3.4.3:

Theorem 3.4.3 (Statement B)**.** *Let $D = (V, E)$ be a two-way directed graph with bound vertex set B, and let M be a set of monitored vertices. Finally, let C be a minimum disconnecting set between $A(M)$ and $B - M$. If the rank of the flow calculation matrix is equal to the number of columns (that is, the flow on the graph can be uniquely calculated), then the size of C is at least $|B - M|$.*

A final useful observation comes in noticing that the graph D can be partitioned into various subgraphs based on the positions of vertices in M: For each bound vertex b in $B - M$, consider every vertex that can be reached from b without passing through any vertex in M. Let this set of vertices (which may include other bound vertices) be one such partition in the graph. Let $[B_b]$ be the equivalence class denoting this partition. By taking all such equivalence classes, together with the set M, and the set of all vertices unreachable from any bound vertex without passing through M, we have accounted for all vertices in the graph.

It is a simple matter to note that the flow calculation matrix can be rearranged into block form by collecting rows and columns corresponding to vertices in the partitions. These blocks are completely separate from each other, since (by construction) there are no paths from one to another except through M, and all rows and columns corresponding to vertices in M have been removed from the matrix (this is what we refer to as the "global" matrix block structure in Section 3.4. Each of these blocks can be further broken down into the block structure shown in Figure (3.6)). Thus, we can prove Theorem 3.4.3 for each of these partitions independently of the others simply by considering the block corresponding to each partition as a separate matrix:

Theorem 3.4.3 (Statement C)**.** *Let $D = (V, E)$ be a two-way directed graph with bound vertex set B, and let M be a set of monitored vertices. Partition D as described above; then, for each partition $[B_b]$ containing at least one bound vertex b,*

if the rank of the matrix corresponding to $[B_b]$'s block in $\mathbf{E}_M$ equals the number of columns in this block, then the size of the minimum disconnecting set C between $A(M)$ and $B - M$ in that partition is at least $|B - M|$.

Therefore, it is sufficient to prove the theorem for each partition of the graph:

Theorem 3.4.3 (Statement D)**.** *Let D, B, and M be as in Theorem 3.4.3, with the flow calculation matrix partitioned into blocks as described. For each block i, let C_i be the minimum vertex cut between $(B - M)_i$ and $A(M)_i$. If $\text{rank}(\mathbf{E}^i_M) = \#\{\text{columns of } \mathbf{E}^i_M\}$, then $|C_i| \geq |(B - M)_i|$.*

Proof. For ease of notation, we drop the subscript notation for the partitions of block i.

Let V_M be the set of vertices in V that are not in M or $A(M)$ that are connected to M by some path that does not pass through C; similarly, let V_B be the set of vertices not in $B - M$ that are connected to $B - M$ by some path that does not pass through C. Note that C, $A(M)$, and $B - M$ could all overlap, as in Figure 3.11; we label these intersections as shown, where $X_{A(M),C}$ corresponds to all vertices in both $A(M)$ and C, but $X_{A(M)}$ contains vertices only in $A(M)$. Note that since C is by definition a vertex cut between $B - M$ and $A(M)$, the set $X_{A(M),B-M}$ is empty, and so we disregard it henceforth.

We note that since $\text{rank}(\mathbf{E}_M)$ equals the number of columns of $\mathbf{E}_M$, Corollary 3.3.6 says that there exists a square invertible matrix $\hat{\mathbf{E}}_M$ containing all rows corresponding to the balancing flow laws (the upper half of Figure 3.6) and $|B - M|$ rows corresponding to known flows in $A(M)$ (the lower half of Figure 3.6). Note that any submatrix of this square invertible matrix must have full rank (that is, must have rank equal to the number of rows, since the submatrix has fewer rows than columns).

So, consider the submatrix $\mathbf{F}$ that contains only the rows corresponding to vertices in $X_{A(M)}$ and $V(M)$ (the shaded portions of Figure 3.11), as well as the $|B - M|$ rows corresponding to known flows on $A(M)$. Next, we compare the number of rows and non-zero columns of $\mathbf{F}$. In particular, for $\mathbf{F}$ to have full rank, we must have that

$$R \leq K - Z, \tag{3.11}$$

where R is the number of rows of $\mathbf{F}$, K is the total number of columns, and Z is the number of zero columns. By construction, we have that

$$R = |X_{A(M)}| + |V_M| + |B - M|$$

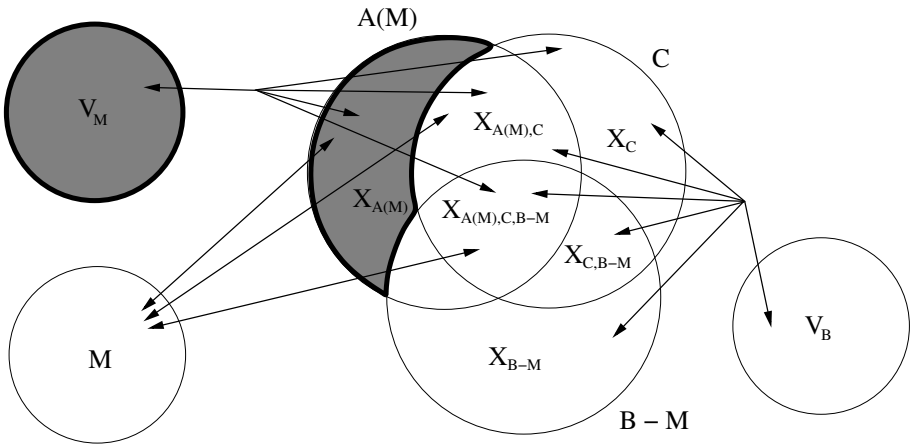


Figure 3.11: The partition of the vertex set for Theorem 3.4.3. V_M is the set of unaccounted-for vertices on the M side of the cut, and V_B is the set of unaccounted-for vertices on the $B - M$ side of the cut. Bold arrows indicate possible connections between sets. Some of these sets may be empty—in particular, note that by definition, there can be no vertices in $((B - M) \cap A(M)) \setminus C$, since C must separate $B - M$ and $A(M)$. The shaded-in regions correspond to the vertex rows included in the submatrix **F**.

and

$$K = |X_A(M)| + |X_{A(M),C}| + |X_C| + |V_B| + |V_M| + 2(|X_{A(M),C,B-M}| + |X_{C,B-M}| + |X_{B-M}|),$$

Now we need to determine Z. We note that since no vertices in the $|B - M|$ columns corresponding to balancing flows (the columns on the right in Figure 3.6) appear in $X_{A(M)}$ or V_M, all of these columns must be **0**. Additionally, there are no edges from X_{B-M} or V_B to $X_{A(M)}$ or V_M, since all edges from $B - M$ or V_B must pass through the cut (by definition). This means that the columns corresponding to those vertices are also **0** in the submatrix **F**, since there are non-zero entries only if the vertices are adjacent. Therefore, we have that $Z \geq |X_{A(M),C,B-M}| + |X_{C,B-M}| + |X_{B-M}| + |V_B| + |X_{B-M}|$. Applying equation (3.11) and canceling common terms, we see that $|X_{A(M),C,B-M}| + |X_{A(M),C}| + |X_C| + |X_{C,B-M}| = |C| \geq |B - M|$. ■

As an example, we walk through the construction of the **F**-matrix for the graph shown in Figure 3.5, and note that the number of zero columns in this matrix is too large. We have already calculated the flow calculation matrix for this graph in Section (3.3); we have as our minimum cut $C = \{d\}$, and we note that $V_M = \{c\}$. Therefore, we have $X_{A(M)} = \{b\}$, and following the

procedure outlined in the proof of Theorem 3.4.3, we get the following for the **F**-submatrix:

$$\mathbf{F} = \begin{array}{c} \\ b \\ c \\ f_{ba} \\ f_{dc} \end{array} \begin{array}{c} \begin{array}{ccccccc} ba & cb & dc & ed & fd & S_e & S_f \end{array} \\ \begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Notice that the last four columns of the matrix are 0, which means that the rank of this submatrix can't be any higher than 3. This implies that the rank of $\mathbf{E}_M$ cannot equal the number of columns, and thus the flow on the graph cannot be calculated.

Now that we have characterized as much as possible when solutions are invalid, we finally turn our attention to finding an optimal solution to SLP. It turns out that even though we were unable to find a sufficient condition for calculating flow based on the structure of the graph, we can still say interesting things about finding such a solution; this is the topic of the next section.

3.5 SLP and NP-Completeness

The final result that we have determined about SLP is its complexity. We were unable to discover a condition on the structure of the graph and on M that allowed us to determine the flow uniquely. However, we are able to prove directly from the matrix formulation that SLP is NP-complete. First, however, it is necessary to state the decision problem version of SLP.

Definition 3.5.1 (SLP Decision Problem)**.** *Given a two-way directed graph $D = (V, E)$ together with a network flow function f and a set of bound vertices B, does there exists a set M of size k such that knowledge of f on M uniquely determines f everywhere on D?*

We prove the NP-completeness of SLP by a reduction from the Dominating Set (DS) problem, stated below:

Definition 3.5.2 (Dominating Set Problem)**.** *Given an undirected graph $G = (V, E)$, does there exist a set of vertices M such that for all $v \in V$, either $v \in M$ or v is adjacent to a vertex in M? (such a set is called a* **dominating set***.)*

Now we have everything we need to prove that SLP is NP-complete:

Theorem 3.5.3. *SLP is NP-complete.*

Proof. We first note that SLP is indeed in NP. That is to say, if we are given a set M of monitored vertices by an oracle, we can construct the matrix $\mathbf{E}_M$ in polynomial time. Additionally, we can calculate its rank in polynomial time, and compare it to the number of columns in constant time. Thus, SLP is in NP.

Now, suppose we are given an undirected graph G and asked to find a dominating set of size k. We show that if we can solve SLP in polynomial time, we can also solve the dominating set problem in polynomial time. We transform the graph $G = (V, E)$ into an instance of SLP by replacing every undirected edge with two directed edges (one in each direction), taking $B = V$, and assigning turning ratios arbitrarily. We now seek a monitored vertex set of size k. Clearly, this transformation can be done in polynomial time.

Next, we show that if we can find a monitored vertex set M with $|M| = k$, that set is precisely the set of dominating vertices for DS. Note that the matrix $\mathbf{E}_M$ has $|V - M| + |(B - M)|$ columns; additionally, there are $|V - M|$ rows in the matrix corresponding to the flow balance equations; in order for the rank of the matrix to equal the number of columns, we need at least $|B - M|$ vertices to be in $A(M)$. However, since $V = B$ and $A(M)$ does not contain any vertices in M, $A(M) = B - M$, and M is a valid dominating set.

Finally, we show that if we have a dominating set M of size k, this set of vertices is also a valid monitored vertex set. We note that every vertex in the graph is either in M or in $A(M)$. Because we know the flow on every outgoing edge of every vertex in the graph (by the turning ratios) we know the flow on *every* edge in the graph. Therefore, M is a valid monitored vertex set.

We have shown that $DS \leq_p SLP$, and since DS is NP-complete, SLP must be as well. ∎

This important result shows that even once a condition on M is found that describes the form of solutions, actually finding such solutions is quite difficult. Therefore, future work should focus on finding an approximation algorithm that works well for the general case. This is discussed briefly in the next chapter.

Chapter 4

Conclusion

We have seen a number of small examples of SLP, and cases in which it fails to be solvable. The next section discusses some open problems and areas for future research.

4.1 Future Work

The most immediate open question regarding SLP is to find a sufficient condition on M that will enable calculation of the flow everywhere on the graph. In nearly every example that we considered, having enough disjoint B-paths is all that is needed to successfully calculate the flow, except in cases when the graph is extremely symmetric about the monitored vertex set M. Thus, a natural first step would be to explore when this symmetry occurs and how it manifests itself in the flow calculation matrix. Another possible line of approach would be to determine conditions on M for calculability on various classes of graphs, such as trees or cycles.

Secondly, Bianco et al. (2006) list a number of cases in which a polynomial-time algorithm for solving SLP on a graph can be found. In particular, polynomial-time algorithms are described for paths, cycles, and combs (a type of tree that has a central "spine" and at most one leaf adjacent to each vertex in the spine). We believe that their algorithm for paths and cycles (at least) is still correct, even though its proof relies on Theorem 3.2.2. This algorithm monitors every third bound vertex along the path or cycle, and it is fairly intuitive to see that this will indeed be optimal, as this will allow you to determine the balancing flows at both the monitored vertex, and the balancing flow at the vertices to the left and right of the monitored vertex. However, proving this (and some of their other results) will be another

important step.

Additionally, we hypothesize that even though SLP is NP-hard in general, finding a solution on trees will be quite easy, since they are acyclic. Indeed, a number of related problems such as Vertex Cover, Dominating Set, and Independent Set all have polynomial-time algorithms on trees, leading us to believe that it is highly likely for a polynomial-time algorithm for SLP on trees to exist as well (Garey and Johnson, 1979; Valiente, 2002). Furthermore, we think that (if such an algorithm exists), we can use it to develop a good approximation algorithm for the general case of SLP, by simply finding a spanning tree of the graph (which is easy to do), determining a solution on that tree, and using that as an approximation to the optimal solution for the original graph.

If we are able to choose our spanning tree in a clever-enough way, we might be able to guarantee that we don't have to adjust the solution on the tree too much to create a valid solution on the graph. This will allow us to (hopefully) create a very tight upper bound for the worst-case analysis of the approximation algorithm.

Another important open question relates to the solvability of the matrix system in equation (3.8). We stated in Section 3.3 that the values contained in the vector $\mathbf{x}'$ do not matter for determining the solvability of the system. However, an important question that we did not consider was whether the solution to the system of equations always makes sense. That is to say, if we fill in values in $\mathbf{x}'$ based on observed values at M, will the solutions that we get always be consistent, and have non-negative flow? Both of these questions are important ones to consider.

A final open problem relating to SLP, alluded to briefly in Section 3.3, is that of determining the location of the bound vertices in the network. As stated before, this will have a significant impact on the efficiency of any algorithm developed to find solutions to SLP, and is an important thing to consider when applying our work to an actual road network.

A final observation should be made. Except in the case where a monitored vertex is a bound vertex, sensors do not need to be placed on all edges incident to some $m \in M$. In fact, if $m \notin B$, we note that we only need to place a sensor on one outgoing edge of each vertex $a \in A(m)$. The knowledge of the turning ratios gives us all other outgoing flows from these vertices, and the calculation of the flow depends only on these values. In the case where $m \in B$, we need only place enough sensors around m to determine S_m.

4.2 Final Thoughts

Solving the Sensor Location Problem is a critical next step in understanding and reducing the large number of congested streets and highways in the world today. We have seen a number of examples in which SLP is not solvable, as well as cases in which it is. However, many of the examples we have presented here are small and bear little or no resemblance to actual street patterns. Many cities have fairly regular arrangements of streets, and in some cases this regularity may, in fact, aid in finding a good solution to the problem. In other cases, however, cities have very irregular street layouts; thus, we desire a solution that will work well regardless of the arrangement of the streets.

In order to find this solution, we have explored conditions upon the location of sensors that will enable us to determine the traffic flow everywhere. We have presented a counterexample to some work done previously on the problem, as well as described a new matrix formulation of the problem that will aid in determining a solution. We were unable to find a condition on the sensor location that guarantees a solution to the traffic flow; however, we were able to find a condition that ensures that a unique solution does *not* exist. Finally, we have shown that the problem is, in fact, NP-complete, meaning that it is unlikely that a fast algorithm will ever be found.

Thus, we hope to be able to develop a fast approximation algorithm based on the principles described herein, and test the algorithm on a number of actual city street networks to see how well it performs. Some actual data have been collected in Eisenman et al. (2006) for the highway system between Baltimore and Washington, D.C. Simulations were run for both random sensor location, as well as sensor location based on the advice of traffic engineers. This data could then be compared against the results achieved with the algorithm that we develop.

It is our hope that understanding how to monitor traffic flow can, in the future, lead to a better understanding of how to design road networks to avoid congestion and heavy traffic. With the world's population moving towards an increasingly fast-paced and mobile society, this question should be one that is at the forefront of our minds and study. Such an understanding will not only save time and money on the part of individuals, corporations, and countries, but it will produce a more efficient means of travel that will waste less of our natural resources, and provide a drastic reduction on our impact to the natural world around us.

Bibliography

Berman, Oded, Dmitry Krass, and Chen Wei Xu. 1995. Locating discretionary service facilities based on probabilistic customer flows. *Transportation Science* 29(3):276–290.

Bertsimas, Dimitris, and John N. Tsitsiklis. 1997. *Linear Optimization*. Athena Scientific.

Bianco, Lucio, Giuseppe Confessore, and Monica Gentili. 2006. Combinatorial aspects of the sensor location problem. *Annals of Operations Research* 144(1):201–234.

Bianco, Lucio, Giuseppe Confessore, and Pierfrancesco Reverberi. 2001. A network based model for traffic sensor location with implications on O/D matrix estimates. *Transportation Science* 35(1):50–60.

Confessore, Giuseppe, Paolo Dell'Olmo, and Monica Gentili. 2005. Experimental evaluation of approximation and heuristic algorithms for the dominating paths problem. *Computers & Operations Research* 32(9):2383–2405.

Eisenman, Stacy M., Xiang Fei, Xuesong Zhou, and Hani S. Mahmassani. 2006. Number and location of sensors for real-time network traffic estimation and prediction—sensitivity analysis. *Network Modeling 2006* (1964):253–259.

Garey, Michael R., and David S. Johnson. 1979. *Computers and Intractability*. Bell Laboratories: Murray Hill, NJ.

Gentili, Monica, and Pitu B. Mirchandani. 2005. Locating active sensors on traffic networks. *Annals of Operations Research* 136(1):229–257.

Gu, Weizhen, and Xingde Jia. 2005. On a traffic control problem. In *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms, and Networks*.

Schrank, David, and Tim Lomax. 2007. 2007 Urban mobility report. Tech. rep., Texas Transportation Institute.

Valiente, Gabriel. 2002. *Algorithms on Trees and Graphs*. Springer-Verlag Berlin Heidelberg New York.

Yang, Hai, and Jing Zhou. 1998. Optimal traffic counting locations for origin-destination matrix estimation. *Transportation Research Part B—Methodological* 32(2):109–126.