
# Beyond Political Boundaries: Constructing Network Models for Megaregion Planning

Dr. Stephen Boyles, Priyadarshan Patil, Venktesh Pandey, and Cesar Yahia

December, 2018

A publication of the USDOT Tier 1 Center:
**Cooperative Mobility for Competitive Megaregions**
At The University of Texas at Austin

# Technical Report Documentation Page

| | | |
|---|---|---|
| 1. Report No. CM2-11 | 2. Government Accession No. | 3. Recipient's Catalog No.<br>ORCID: 0000-0002-5414-5438 |
| 4. Title and Subtitle<br>Beyond Political Boundaries: Constructing Network Models for Megaregion Planning | | 5. Report Date<br>December 2018<br>6. Performing Organization Code |
| 7. Author(s)<br>Stephen Boyles, Priyadarshan Patil, Venktesh Pandey, Cesar Yahia | | 8. Performing Organization Report No.<br>CM2-11 |
| 9. Performing Organization Name and Address<br>The University of Texas at Austin<br>School of Architecture<br>310 Inner Campus Drive, B7500<br>Austin, TX 78712<br><br>The University of Texas at Austin<br>Department of Civil, Architectural, and Environmental Engineering<br>301 E. Dean Keeton St. Stop C1761<br>Austin, TX 78712 | | 10. Work Unit No. (TRAIS)<br>11. Contract or Grant No.<br>USDOT 69A3551747135 |
| 12. Sponsoring Agency Name and Address<br>U.S. Department of Transportation<br>Federal Transit Administration<br>Office of the Assistant Secretary for Research and Technology, UTC Program<br>1200 New Jersey Avenue, SE<br>Washington, DC 20590 | | 13. Type of Report and Period Covered<br>Technical Report conducted October 2017-October 2018<br>14. Sponsoring Agency Code |
| 15. Supplementary Notes<br>Project performed under a grant from the U.S. Department of Transportation's University Transportation Center's Program. | | |
| 16. Abstract<br>The scale of urban planning is now focusing on megaregions in addition to metropolitan areas and states. The traffic assignment problem (TAP), used to study traffic flow patterns on networks, is a crucial step in urban planning. Megaregional networks transcend planning agency jurisdictions, challenging current network models and computational resources. This study aims to solve TAP on a megaregional scale by applying an algorithm based on the decomposition approach for the static TAP (DSTAP) that uses network decompositions based on network geography. In the first part of this research, we compare two partitioning algorithms for finding network partitions for megaregions by minimizing the number of subnetwork boundary nodes and the time required to solve DSTAP. The flow-based spectral partitioning generates flow balanced subnetworks which reduce the per iteration computation time and lead to faster convergence compared to the agglomerative partitioning algorithm. In the second part of this research, we propose a decomposition heuristic for large scale networks, allowing parallelization of TAP. The heuristic reduces the computational time for DSTAP by simplifying interactions within the subnetwork. For the uncongested Texas network, the proposed heuristic led to marginal 5% savings in computational time than state-of-the-art TAP methods, while for the congested scenario, the heuristic observed about 70% savings in computation time for the same | | |

| 1. Report No. CM2-11 | 2. Government Accession No. | 3. Recipient's Catalog No. ORCID: 0000-0002-5414-5438 | |
|---|---|---|---|
| convergence level. However, the heuristic leads to a lower bound on the relative gap value at termination (called heuristic gap) which ranges between 9E-3 and 5E-4 for the experiments conducted on the Texas statewide network. | | | |
| 17. Key Words<br>Megaregions, Traffic assignment problem, Partitioning | 18. Distribution Statement<br>No restrictions. | | |
| 19. Security Classif. (of report)<br>Unclassified | 20. Security Classif. (of this page)<br>Unclassified | 21. No. of pages<br>44 | 22. Price |

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

# Acknowledgements

We would like to thank Cooperative Mobility for Competitive Megaregions (CM2), a United States Department of Transportation (USDOT) Tier-1 University Transportation Center (UTC), for funding of this research. We are grateful for the support.

Finally, we would like to thank Ehsan Jafari for sharing his original code for the DSTAP algorithm and all the members of the Sparta lab at the University of Texas at Austin for their support and cooperation during the shared use of the computational resources in the lab.

# Table of Contents

# Chapter 1. Introduction

Megaregions cross state and political boundaries today and are characterized by trade and infrastructural connections. The key characteristic which makes megaregions unique is their size and complexity. A significant portion of the United States of America can be characterized in 11 megaregions (*1*,*2*,*3*). The megaregions span multiple states and represent over 75% of the total population. For example, the north-eastern megaregion spans the states of Connecticut, Rhode Island, New York, Massachusetts, Pennsylvania, Delaware, New Jersey and Virginia, and contains as much as 17% of the current US population.

This size of geographical and economic interaction has introduced a new scale of planning beyond the existing scale of models used at the state level or the county/city level maintained by state Departments of Transportation (DOTs) and Metropolitan Planning Organizations (MPOs) within the state, respectively. With increasing intra-megaregion trade and traffic (*4*,*5*), these statewide or county/city wide models are not appropriate for the megaregion scale. This mismatch of network model scale impacts long-range network planning process, specifically traffic assignment on the network. For example, consider the Texas Triangle which includes the cities of Austin, San Antonio, and Dallas. A change in the Austin arterial network impacts the traffic in the city, which in turns impacts the freight demand using the freeway and the truck routes taken to and from San Antonio and Dallas.

The traffic assignment problem (TAP) is a well-studied problem in the traffic engineering community. TAP assigns vehicles to network routes as the last step of the four-step planning process. The resulting link flows provide metrics for planning (for instance, travel times, or total vehicle-miles traveled) and alternatives analysis. Despite advances in computation power and solution algorithms, solving TAP on a megaregion with many urban areas remains challenging. Obtaining the data needed to construct such models is also challenging, given the multiple jurisdictions involved.

The primary motivation of this research is to address this deficiency in the scalability of traditional models. The new models and methods should not only focus on the interactions between multiple

cities and states, but also be computationally solvable and tractable, given current resources. This research aims to address this gap and develop methods to balance realism and accuracy of megaregion traffic assignment models.

In the recent literature, a decomposition algorithm for solving the traffic assignment problem (DSTAP) on large scale networks has been proposed (*6*). The proposed decomposition is based on network geography which makes it convenient for extending this algorithm for megaregions. Additionally, experiments on the Austin network have shown 35-70% reduction in computation time for solving TAP using DSTAP in comparison with using the traditional algorithms like gradient projection algorithm. Aligned with the motivation discussed above, this research investigates the usefulness of DSTAP for improving the state-of-the-art of traffic assignment models for megaregions by answering following questions:

1. How to partition a large megaregional network into smaller subnetworks for efficient parallel computation and reduction in computation time?
2. How to quantify the interactions between these subnetworks allowing for changes in one subnetwork to impact the others and develop heuristics which simplify these interactions?

We address these questions by comparing two partitioning algorithms on different real-world networks on their performance against different objectives and implementing DSTAP and its variant used a heuristic for solving traffic assignment on large networks.

The rest of the report is organized as follows. Chapter 2 compares partitioning algorithms and demonstrates their performance for different transportation networks. Chapter 3 implements the DSTAP algorithm and its variant by modeling interactions between subnetworks differently than DSTAP. This heuristic is proposed to achieve computation time savings. The chapter also compares the partitioning algorithms from Chapter 2 on Texas Statewide Analysis Model network. The relevant literature review is presented as sections in Chapters 2 and 3 which are self-contained. Chapter 4 summarizes the work and discusses the directions for future work.

# Chapter 2. Partitioning

## 2.1. Background

The traffic assignment problem is used to predict route choice and link flows for a given travel demand. The static version of this problem can be formulated as a convex program and solved efficiently using modern specialized algorithms (*7*,*8*,*9*). However, there are computationally demanding problems that require solving TAP multiple times or solving TAP on a large network. Those problems include bi-level mathematical programs with equilibrium constraints, solving TAP on statewide or national network models, and Monte Carlo simulations (*6, 10*).

Methods for parallelizing the traffic assignment problem to decrease computation time have been studied recently e.g. (*6, 11, 12*). The DSTAP algorithm was developed to decrease computation time by solving the traffic assignment problem in parallel on partitions of the full network (*6*). This approach creates subproblems for each partition and a master problem that equilibrates traffic across subnetworks. The master problem also includes regional traffic that has an origin or a destination outside a certain subnetwork or in two different subnetworks. To find equilibrium in this master-subproblem framework, the DSTAP algorithm exploits the equilibrium sensitivity analysis method developed in Boyles (*13*) to generate artificial links that represent paths between network nodes. DSTAP is shown to converge to the global equilibrium solution for a general network and its computation time is stated to depend on the subnetwork partitions (*6*).

The objective of this chapter is to identify and test partitioning algorithms that can improve the performance of a decomposition approach for solving the static traffic assignment problem. We seek to generate partitions that minimize the number of boundary nodes and the inter-flow between subnetworks. These requirements minimize the interactions between subnetworks, which influences the time needed to converge to a global equilibrium in a framework such as DSTAP. In addition, we seek partitions that minimize the computation time needed to solve the traffic assignment problem in parallel for the subnetworks. This refers to the per iteration lower level subproblems in DSTAP. With this motivation and objective in place, we test two algorithms in our analysis. The first algorithm is proposed by Johnson et al. (*14*) for objectives similar to those

required in this chapter. The second algorithm is based on flow weighted spectral partitioning. We compare the performance of the algorithms on real-world networks against the stated objectives.

The remainder of this chapter describes methods to parallelize TAP and evaluates the performance of the partitioning algorithms. Section 2.2 reviews current methods for solving TAP and partitioning networks. Section 2.3 presents the algorithms evaluated and their use in the DSTAP framework. Section 2.4 presents demonstrations for different transportation networks. Section 2.5 concludes the chapter.

## 2.2. Literature review

This section summarizes existing literature in the following areas: the latest advancements in methods for solving the traffic assignment problem, a parallelization approach to the traffic assignment problem, and the need for efficient network partitioning algorithms.

Algorithms for solving TAP are generally classified as either link-based, path-based, or bush-based. Link based methods work in the space of link flows and require less operational memory than path-based methods, but are much slower to converge (*15, 16*). Bush-based methods exploit the acyclic nature of paths that are used by origins at equilibrium (*8, 9, 17, 18*). Recent work also includes \epsilon-optimal improved methods for solving TAP on large problems (*19*). Although recent advancements have improved the state-of-the-art for solving TAP efficiently, there is still a need for faster methods. Computationally demanding instances include solving TAP on large-scale statewide models and solving TAP iteratively in network design problems with equilibrium constraints (*11, 20*).

To address the computational demands of large scale or iterative traffic assignment problems, methods that aim to parallelize TAP have been developed. Bar-Gera (*11*) describes a parallelization approach based on the paired-alternative segments. The algorithms proposed in Chen and Meyer (*21*) and Lotito (*22*) also parallelize TAP by decentralizing the computations for each OD pair. DSTAP algorithm developed by Jafari et al. (*6*) parallelizes TAP by network geography instead of the traditional decomposition approach by OD pairs. This chapter aims to

identify partitioning algorithms that minimize the computation time per iteration of DSTAP and the total computation time required to reach convergence. Proofs of convergence and correctness of the algorithm are provided in Jafari et al. (*6*).

The literature on network partitioning algorithms is extensive. These algorithms can be broadly classified into agglomerative/divisive heuristics, integer programming based approaches, and spectral partitioning algorithms. Integer programming formulations for the partitioning problem are proven to be NP-hard (*23*) and approximation heuristics have been proposed (*23, 24*).
Heuristics for generating partitions based on agglomerative and divisive clustering have been recently used in various transportation related applications. Saedmanesh and Geroliminis (*25*) used an agglomerative clustering heuristic for generating partitions based on “snake" similarities for applications of the macroscopic fundamental diagram. Etemadnia et al. (*23*) developed similar heuristics for distributed traffic management. Johnson et al. (*14*) developed another heuristic for decentralized traffic management. This heuristic aims to minimize boundary nodes in subnetworks and to create subnetworks of similar size. Their heuristic performed better than the METIS algorithm proposed in (*26*).

Spectral partitioning is an alternative approach for partitioning a graph (*27, 28, 29, 30, 31*). Bell (*32*) applied a capacity-weighted form of the spectral partitioning methods to investigate network vulnerability. Other transportation applications include air traffic control and urban traffic signal control systems (*33, 34*). The partitioning mechanism is based on the eigenvalues associated with the graph Laplacian. The partitions that result from spectral partitioning have low inter-cluster similarity (*28*). Additionally, using the normalized Laplacian generates graphs that are balanced by weight. This is an important feature since ignoring the balance requirement results in cuts that isolate a small number of peripheral nodes. For example, the minimum cut program that aims to minimize the weight between resulting partitions will often result in separating one node from the rest of the network (*30*). However, incorporating balance requirements causes cut problems to become NP-hard. Spectral partitioning is an approximate method for obtaining a cut with minimal cut cost while satisfying balance requirements (*27, 30, 31*).

## 2.3. Network partitioning for decentralized traffic assignment

We consider a directed network $G$ defined by a set of nodes $N$ and set of edges $A$. Let $M$ be the node-node adjacency matrix for the network. $M$ is an $|N| \times |N|$ matrix, with elements $m_{ij}$ equal to 1 if there is a link connecting node $i$ to $j$ and zero otherwise. We also define the weighted adjacency matrix $M_G^D$ with elements $m_{(i,j)}^{G,D}$ equal to $w_{(i,j)}$ if $(i,j) \in A$ and zero otherwise, where $w_{(i,j)}$ is the weight assigned to link $(i,j) \in A$. In this chapter, we assume $w_{(i,j)}$ to be the flow on link $(i,j)$. To construct a graph Laplacian, we use an undirected version of $M_G^D$, denoted by $M_G$, defined as the sum of $M_G^D$ and its transpose. The elements of $M_G$ are $m_{(i,j)}^G$. The graph diagonal matrix $D_G$ is defined as a diagonal matrix with principal diagonal elements in row $i$ as the sum of elements in row $i$ of $M_G$: $d_{ii} = \sum_j m_{(i,j)}^G$. The graph Laplacian is defined as $L_G = D_G - M_G$.

### 2.3.1. Decomposition approach to the static traffic assignment problem (DSTAP)

We aim to partition a large-scale network into subnetworks such that the DSTAP algorithm is solved efficiently. In order to properly define the objectives of the partitioning algorithms, we review the main elements of the DSTAP algorithm developed by Jafari et al. (*6*).

DSTAP is an iterative aggregation-disaggregation algorithm consisting of two levels, a master problem and a set of lower level subproblems corresponding to the respective subnetworks. A subproblem corresponds to solving the traffic assignment problem for a specific subnetwork. The master problem is used to model interactions between the subproblems. In the master problem, the subnetworks are aggregated using first order approximation methods based on equilibrium sensitivity analysis (*13, 35*). This results in artificial links representing the subnetworks in the master level problem. The algorithm proceeds by solving the subproblems in parallel, aggregating the subnetworks using artificial links, shifting flow towards equilibrium in the simplified master level network, obtaining subnetwork boundary flow from the master level iteration, and then proceeding to disaggregate the flow on subnetworks and solving the subproblems in parallel again. This procedure is repeated until convergence to a global equilibrium as shown in Figure 1.

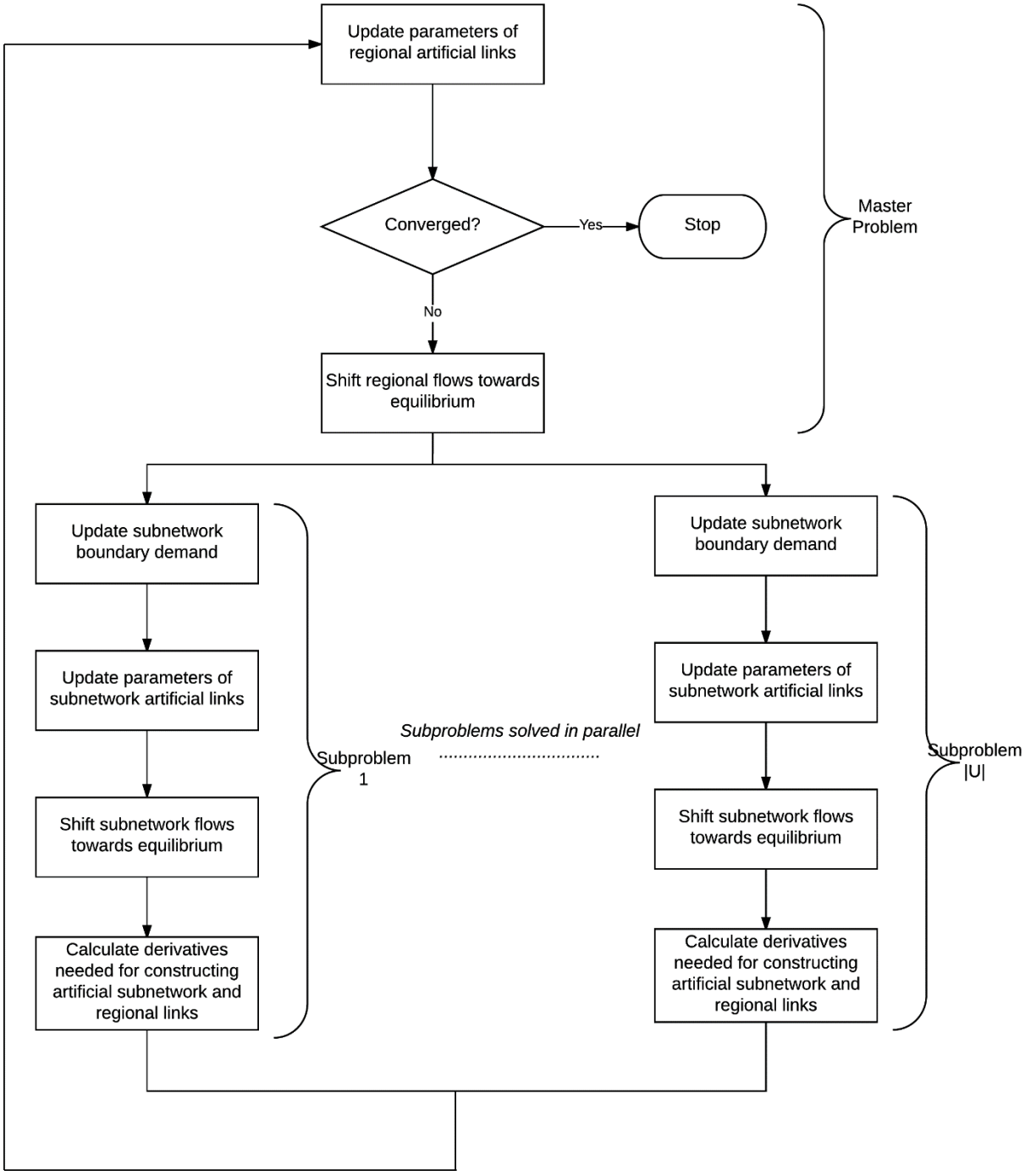



*Figure 1 Algorithm for the decomposition approach to the static traffic assignment problem*

The computational performance of DSTAP at each iteration depends on the number of artificial links. These links need to be updated at each iteration using equilibrium sensitivity analysis to

incorporate the latest information on travel costs. To reduce the number of artificial links generated, the number of boundary nodes associated with the subnetworks needs to be minimized. In addition to the regional artificial links that approximate subnetworks at the master level, there are subnetwork artificial links generated for each subproblem to represent flow that originates from a subnetwork then traverses other subnetworks before returning to the subnetwork. To reduce subnetwork artificial links, the flow traversing multiple subnetworks needs to be minimized.

The computational performance at each iteration is also influenced by the time needed to solve the traffic assignment problem in parallel for the subnetworks. This represents solving the K lower level subproblems in Figure 1. The computation time needed to solve the subproblems in parallel is dominated by the subproblem that requires the greatest computational cost. Therefore, to reduce this computation time, the subproblems need to be balanced in size. This can be achieved by balancing the flow distribution across subnetworks, as opposed to having few subnetworks containing most of the travel demand.

Consider the maximum excess cost termination criteria defined as the greatest difference between the longest used path and the shortest path for each OD pair, it was shown in Jafari et al. (*6*) that the maximum excess cost for the full network $\epsilon_{OD}$ is bounded by the total number of boundary points across subnetworks $\tilde{B}$ multiplied by the sum of the maximum excess cost for the master level regional network $\epsilon_{OD}^{r}$ and the maximum excess cost for all subnetworks $\epsilon_{OD}^{s}$ as shown in Equation (1). Therefore, to reach convergence faster, we need to increase the rate at which the bound in Equation (1) tightens. The subproblem maximum excess cost $\epsilon_{OD}^{s}$ can be reduced by solving the subproblems to a low gap level. After approximating the subnetworks with artificial links, the master level maximum excess cost $\epsilon_{OD}^{r}$ could be obtained. We note that if the inter-flow between subnetworks is minimized, then the artificial links representing the subnetworks will have a similar cost structure across successive iterations since the influence of external flows on subnetwork equilibrium is reduced. Therefore, the least cost path in the master level regional network would be relatively invariant across iterations. This implies that $\epsilon_{OD}^{r}$ could be reduced at a higher rate. In the extreme case where the master level least-cost path is completely dominated by constant costs on artificial links, the maximum excess cost could be reduced to zero by placing all the regional flow on the path with the least cost artificial links. Thus, faster convergence could

be reached by minimizing the inter-flow between subnetworks. Convergence rate can also be increased by minimizing the number of boundary nodes $\tilde{B}$ as shown in Equation (1).

$$\epsilon_{OD} \leq 2\tilde{B}(\epsilon_{OD}^{r} + \epsilon_{OD}^{s}) \quad (1)$$

### 2.3.2. Partitioning algorithms

We test the performance of two algorithms that aim to partition the network such that the computation time for a decomposition approach to solve traffic assignment is minimized.

*Domain Decomposition Algorithm*

The first heuristic algorithm tested is the shortest domain decomposition algorithm (SDDA) proposed in Johnson et al. (*14*). This algorithm works in an agglomerative fashion and constructs a given number of partitions such that the number of boundary nodes between the subnetworks is minimized (primary objective) and the partitions are balanced in size (secondary objective). SDDA only depends on the topological properties of the graph. This feature is desirable when limited information is available on link costs, flow between OD pairs, or other data that could form the basis of a partitioning algorithm. The computation time per DSTAP iteration is reduced by minimizing the boundary nodes and generating balanced subnetworks. Minimizing the number of boundary nodes would also improve the convergence rate.

The sequential steps of the algorithm are shown in Algorithm 1. The algorithm constructs the partitions by identifying source nodes which are “far" from each other given a distance measure. The number of links on a breadth-first search tree between two nodes is used as the distance measure. This distance measure indicates the extent of separation of two nodes and is used to determine association of a node to the source nodes of the partitions. The reader is referred to Johnson et al. (*14*) for more information on this algorithm.

**ALGORITHM 1 Shortest domain decomposition algorithm *(14)***

**Step 1:** Initialization
Let $n_s$ be the number of subnetworks/partitions to be generated
Set $R_s^n := \text{MAX}$
**Step 2:** Determining first source node
Set the rank of each node as the sum of the number of incoming and outgoing links

Choose the node with lowest rank $s_1$ as the first source node

**Step 3:** Updating the rank and determining other source nodes

**for** $i$ in $2: n_s$

Perform breadth-first search from every source node, $s_j \quad \forall 1 \leq j < i$

Determine the rank of node $n$ as a $(i-1)$-dimensional vector whose elements are the distance of node $n$ from source nodes $s_j$ where $1 \leq j < i$

Choose the node which has the highest total rank (sum of all elements in the rank vector)

Resolve ties in favor of nodes which have minimum value of the sum of pair-wise difference between each element of the rank vector

Assign the chosen node as the $i$-th source node $s_i$

**Step 4:** Populate subdomain associated with each source node For each node, assign it to the source node to which it has the minimum distance

**Step 5:** Identify system boundary nodes and allocate the subnetworks

for $(i, j) \in A$ do

if $i$ and $j$ are assigned to different source nodes then

Add $i$ and $j$ to the set of boundary nodes.

Stop

*Spectral Partitioning*

Spectral graph theory is used to study network properties using the graph Laplacian. The eigenvalues and eigenvectors of the Laplacian matrix can be used to identify low cost graph cuts. The cost of a cut is defined as a ratio of the weights on cut links to the size of the smaller subnetwork separated by the cut (*28, 30*).

The eigenvalues of an undirected graph Laplacian are real since the matrix is symmetric (*27*). Let $\varphi$ represent the eigenvectors and $\lambda$ the eigenvalues. The relation between the eigenvectors and eigenvalues for the graph Laplacian is shown in Equation (2). According to Spielman (*27*), the eigenvalues can be defined using Equation (3), where $S$ is a vector space of dimension $i$, and $i$ is the index of eigenvalue $\lambda_i$ arranged in an ascending order. The eigenvector for the corresponding eigenvalue can be found using Equation (4).

$$L_G \varphi_i = \lambda_i \varphi_i \tag{2}$$

$$\lambda_i = \min_{S\,of\,dim\,i} \max_{x \in S} \frac{x^T L_G x}{x^T x} \tag{3}$$

$$\varphi_i = \operatorname{argmin}_{S\,of\,dim\,i} \max_{x \in S} \frac{x^T L_G x}{x^T x} \tag{4}$$

The Laplacian matrix $L_G$ is also positive definite and thus the eigenvalues are non-negative. The second smallest eigenvalue and associated eigenvector obtained from Equations (3) and (4) can be used to partition the graph. The resulting partition is an approximation of the cut that minimizes the ratio cut in Equation (5), where $cut(A, \tilde{A})$ is the sum of the weights on the links separating the subnetworks $A$ and $A$ that are generated from the cut. The denominator of the ratio cut is the size of the smaller subnetwork $A$, where the size is determined by the number of nodes in $A$. Minimizing the ratio cut aims to find a cut with minimal weights on the links separating the subnetworks, and to maintain a balance in size of the generated subnetworks (*30*).

$$\text{ratio cut} = \frac{cut(A, \tilde{A})}{|A|} \tag{5}$$

To improve the efficiency of DSTAP, we use a flow weighted version of the Laplacian such that the cut cost in Equation (5) represents the inter-flow between subnetworks. This will improve the convergence rate of DSTAP. We also normalize the Laplacian matrix using Equation (6) similar to methods in the literature (*27, 29, 30, 31*). This normalization will generate partitions that are balanced by the total flow within the partitions instead of the number of nodes in Equation (5). In the DSTAP framework, balancing the partitions by flow would reduce the per iteration computation time needed to solve the subproblems in parallel.

$$L_{symm} = D_G^{-1/2} L_G D_G^{-1/2} \tag{6}$$

After calculating the second smallest eigenvalue and associated eigenvector of the normalized Laplacian, the nodes of the network are sorted based on the magnitude of the corresponding element in the eigenvector. The sorted list of nodes is then divided into two parts based on the signs of the corresponding eigenvector elements. This will generate the required partitions *(31, 32)*. The full algorithm for the flow weighted spectral partitioning is shown in Algorithm 2.

Since the spectral partitioning method proposed is based on link flows, a few implementation issues need to be considered. The use of the second smallest eigenvalue as the basis for partitioning requires the graph to be connected. Specifically, the weighted adjacency matrix $M_G$ should result in a connected graph. Otherwise, the second smallest eigenvalue will be zero. To ensure that the component being partitioned is connected, a preprocessing stage precedes the spectral analysis. In this stage, the links with zero flow are identified. If those links separate the network into components such that each component has positive intra-flow, the spectral partitioning is performed for each component separately. However, in transportation networks, it is more likely to observe multiple components where only one component has flow. In our analysis, this occurred due to the existence of peripheral links that do not have any flow but are included in the network geometry. In this case, those links are ignored since they are not used, and should not influence the partitioning of the main component.

Another consideration is the availability of flow values for the links in the network. In the case where the traffic assignment problem should be solved multiple times, solving the full network once to obtain link flows is worthwhile since the flows could be used to partition the network in subsequent iterations. If the flow values on the links change each time TAP is solved, the partitions could be updated iteratively. Alternatively, an approximate link flow solution could be obtained by solving centralized traffic assignment to a high gap value.

**Algorithm 2 Flow-weighted spectral partitioning**

**Step 1:** Pre-process the network to remove links with zero flow
If removing zero flow links creates multiple components with positive flow, then partition each component separately
**Step 2:** Calculate the flow weighted graph Laplacian
**Step 3:** Normalize the graph Laplacian using Equation (6)
**Step 4:** Get the eigenvector to be used for partitioning
**Step 5:** Order the nodes of the graph based on the eigenvector

**Step 6:** Partition the network by dividing the ordered node list based on the sign of the corresponding eigenvector elements

Decide if the obtained partitions should be divided further

If further partitioning is needed, then repeat the algorithm for each subnetwork

## 2.4. Demonstrations

We compare the performance of algorithms on a hypothetical network consisting of two copies of the Sioux Falls network and on three standard test networks: Anaheim, Austin, and Chicago sketch (*36*). Considering the previous discussion on the required computation time in the DSTAP section, we divide our analysis into a section on the computation time per iteration and another section on the DSTAP convergence rate. We note that computation time needed for partitioning is insignificant for both SDDA and flow weighted spectral partitioning (less than 1 second on a 3.3 GHz machine with 8 GB RAM) and is thus not included in the analysis.

### 2.4.1. Computation time per DSTAP iteration

As mentioned in the section on the decomposition approach for static traffic assignment, the computation time per iteration of DSTAP is dominated by the number of artificial links created and the time required to solve the subproblems in parallel.

The number of regional artificial links created is determined by the number of boundary nodes in the subnetworks. Therefore, we compare the number of boundary nodes generated by each algorithm. Note that the primary objective of the SDDA algorithm is to reduce the number of boundary nodes between the subnetworks.

The computation time required to solve the subproblems in parallel could be reduced by balancing the size of the subproblems. The flow weighted spectral partitioning method aims to minimize the flow balanced cut cost. If the cut cost is always equal to 1, the flow weighted spectral partitioning method will divide the flow equally among the subnetworks. This reduces the computation time needed to solve the subproblems in parallel. The SDDA algorithm creates subnetworks that are balanced by number of nodes as a secondary objective.

Table 1 shows the results for number of subnetwork boundary nodes generated by the algorithms and the computation time needed to solve the subproblems in parallel. Unless mentioned otherwise, we generate two subnetworks from each network. In terms of minimizing the number of boundary nodes SDDA performed better than the flow-weighted spectral partitioning method for the Austin and Anaheim networks. This result is expected since the objective of the flow-weighted spectral partitioning method is to minimize the balanced inter-flow while SDDA minimizes the boundary nodes. This implies that the number of regional artificial links generated by an SDDA partition will be lower.

**Table 1 Comparison of Network Partitioning Algorithms**

| **Network** | **Boundary nodes** | **Subnet computation time (s)** | **Inter-flow** |
|---|---|---|---|
| Austin (SDDA) | 174 | 632.83 | 186161 |
| Austin (Spectral) | 329 | 746.81 | 137940 |
| Austin (4 subnets, SDDA) | 296 | 290.97 | 368718 |
| Austin (4 subnets, spectral) | 440 | 82.50 | 296870 |
| Anaheim (SDDA) | 46 | 0.10 | 81991 |
| Anaheim (Spectral) | 48 | 0.13 | 56539 |
| Chicago sketch (SDDA) | 74 | 9.79 | 154791 |
| Chicago sketch (Spectral) | 50 | 7.42 | 201603 |

In terms of creating balanced subproblems, the flow weighted spectral partitioning method performed better than SDDA. The importance of balancing subproblems by flow is demonstrated by the partitioning of the Austin network into 4 subnetworks. The computation time needed to solve the subproblems in parallel using the SDDA partitions was approximately 3.5 times the corresponding time resulting from flow weighted spectral partitioning algorithm. Figure 2 shows the partitions generated for the Austin network. Subnetwork 1 in the SDDA partition contains 65% of the flow. The computation time associated with this subnetwork determines the computation time needed to solve the lower level subproblems at each iteration of DSTAP. The maximum share of network flow within a subnetwork resulting from the flow weighted spectral partitioning algorithm is 39%. For the Chicago sketch network, SDDA also creates heavily imbalanced

subnetworks with one subnetwork containing 90% of the flow. If this network was larger, the difference in subnetwork computation time would be significant.

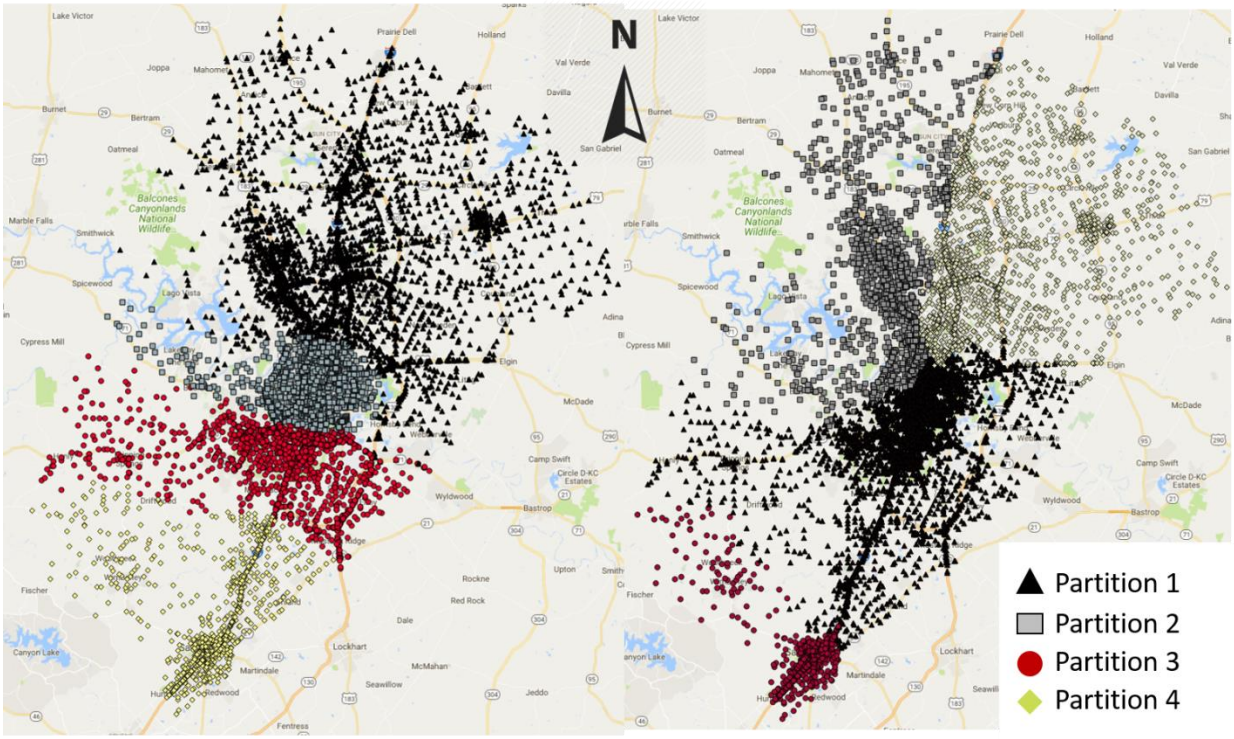



*Figure 2 Partitioning of Austin regional network into four partitions. Left: flow weighted spectral partitioning. Right: SDDA.*

### 2.4.2.DSTAP convergence rate

We also measure the rate at which the DSTAP algorithm converges towards a global equilibrium given the subnetworks generated from a specific partitioning procedure. We test this convergence rate using a hypothetical network with two copies of Sioux Falls. The network was created by replicating the Sioux Falls network and adding artificial demand between the two copies as shown in Figure 3. The artificial demand was kept low at 1.5% of the total demand within each network. Figure 3 also shows the subnetworks generated by the flow weighted spectral algorithm and by SDDA. The generated partitions demonstrate the importance of flow weighted spectral partitioning for networks which have intuitive geographic concentrations such as networks in statewide planning models with concentrated flow density in each city. The flow weighted spectral partitioning method was able to identify each Sioux Falls network as a separate component, as

opposed to partitions generated by SDDA. In terms of subnetwork boundary nodes, both partitions are equivalent.

In the DSTAP section, we showed that faster convergence could be achieved if the inter-flow between subnetworks is minimized. The results in Table 1 and in Figure 3 indicate that the flow weighted spectral partitioning method is superior to the SDDA algorithm in terms of minimizing inter-flow. The only exception is for the Chicago sketch network. However, the partition generated by SDDA for the Chicago sketch network was heavily imbalanced with one partition containing 90% of the flow. We expect the spectral partitioning method to avoid such cuts due to the flow balancing requirement.

Figure 4 shows the convergence rate of DSTAP for the hypothetical double Sioux Falls network when partitioned using the flow weighted spectral partitioning method and SDDA. DSTAP converges to the global equilibrium solution after approximately 135 iterations using partitions generated from the flow weighted spectral partitioning algorithm. As for the SDDA partitions, the convergence rate of the DSTAP algorithm was low. This demonstrates the importance of minimizing the inter-flow between the subnetworks.

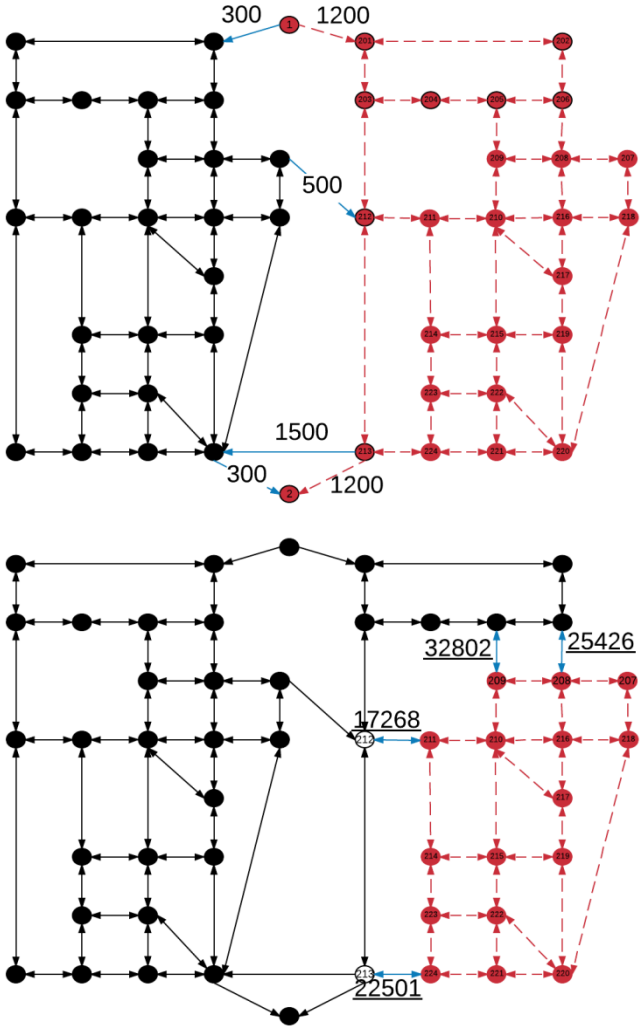



*Figure 3 Partitioning of double Sioux Falls hypothetical network: Top: flow weighted spectral partitioning. Bottom: SDDA. The line type defines different partitions.*

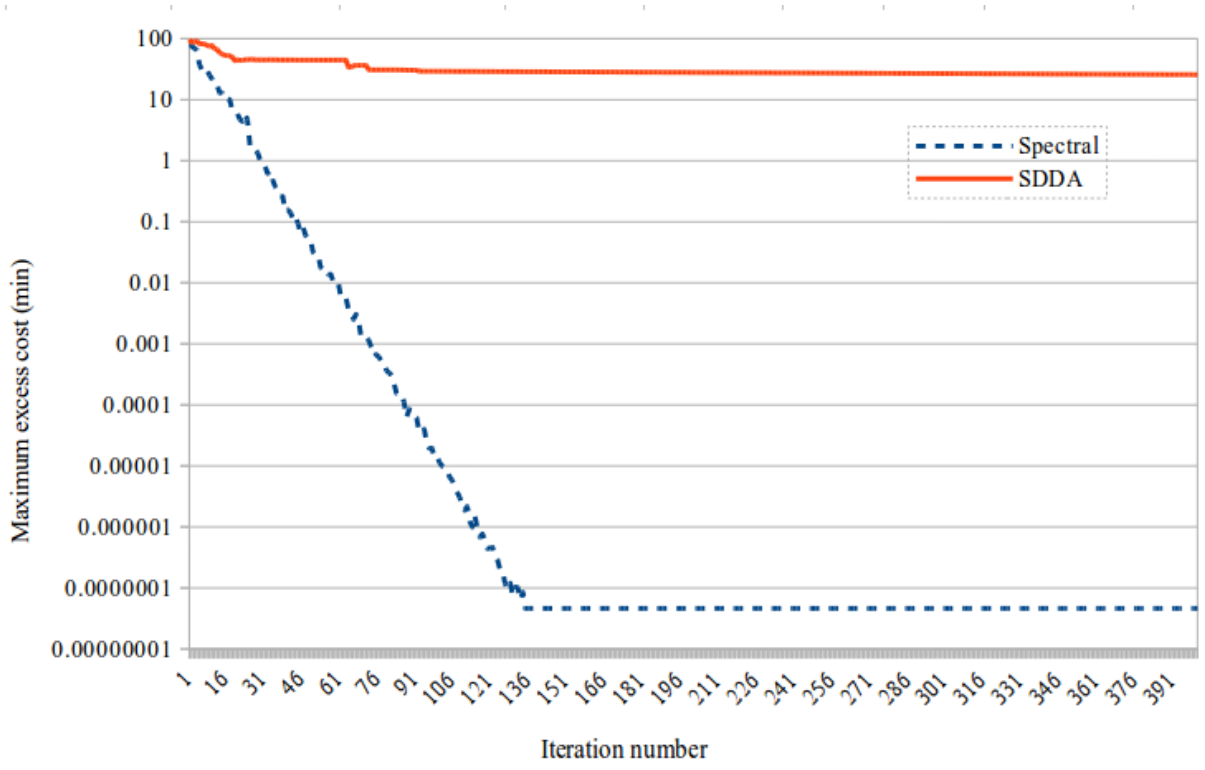



*Figure 4 Iterative change of the maximum excess cost of the DSTAP algorithm when used with flow weighted spectral partitions and SDDA partitions of the hypothetical double Sioux Falls network.*

## 2.5 Conclusion

This chapter evaluated the performance of different partitioning algorithms used for spatial parallelization of the static traffic assignment problem. The partitioning objective is to minimize the computation time needed to solve the static traffic assignment using a decomposition approach. The computation time per DSTAP iteration could be reduced by minimizing the number of subnetwork boundary nodes and the time required to solve the traffic assignment problem for the subnetworks in parallel. The convergence rate of DSTAP depends on the inter-flow between subnetworks.

We tested two different methods for partitioning. The first approach is an agglomerative clustering algorithm developed by Johnson et al. (*14*) to minimize the number of boundary nodes between the subnetworks and to create partitions that are balanced in size. The second approach developed

is based on flow weighted spectral partitioning. The results indicate that the agglomerative clustering algorithm generates subnetworks that have a low number of boundary nodes. However, the subnetworks generated from this method may be heavily imbalanced as shown for the Austin and Chicago sketch networks. This leads to higher computation time for solving the DSTAP subproblems in parallel. The flow weighted spectral partitioning method generates flow balanced subnetworks which reduce the per iteration computation time. In addition, the inter-flow between subnetworks is minimized by the spectral partitioning algorithm, which leads to a faster convergence rate of the DSTAP algorithm.

# Chapter 3 Decomposition Algorithm and Heuristic

## 3.1. Background

In this chapter, we consider a heuristic derived from the DSTAP algorithm where the subnetwork artificial links are ignored while solving each subnetwork. That is, we only include the interactions between the subnetworks as part of the master network and ignore the interactions between the subnetworks while directly solving TAP for each subnetwork.

The chapter primarily contributes to the literature in two ways. First, the proposed heuristic offers an improvement in computation time over the existing state-of-the-art algorithms for solving traffic assignment on megaregions using a decomposition approach. This will be useful for developing efficient planning models in the future. Second, we motivate the need for developing efficient partitioning algorithms for decomposing large megaregions that can improve the heuristic gap and thus the efficiency of the algorithm.

The rest of the chapter is structured as follows: Section 3.2 reviews existing research and provides an overview of the advances in the TAP solution methods and solving TAP for large scale networks. Section 3.3 presents a decomposition algorithm for static TAP (DSTAP), while Section 3.4 presents application of DSTAP heuristic, by modeling artificial links differently to achieve computation time savings. Section 3.5 discusses partitioning algorithms, providing a comparison of two state-of-the-art algorithms. Finally, we conclude by summarizing the discussions and providing directions for future advances in Section 3.6.

## 3.2. Literature review

The static traffic assignment problem is an optimization problem that minimizes the Beckmann function, which is the sum of the integral of arc costs at given flows (*7*). Due to the convex nature of the problem, solution algorithms work iteratively by improving the quality of the solution with each iteration until the convergence is reached. Two convergence criteria used in practice are relative gap and average excess cost. The relative gap metric measures the difference between the current total system travel time and the total system travel time if each user was assigned to the

current shortest path, as a ratio. The average excess cost metric is the difference between the total system travel times averaged over the flow. This can be interpreted as the average delay faced by each user on the network. While relative gap has no such interpretation, it has a well documented convergence level for select link analysis or network design problem, which is network independent, and is usually accepted to be $10^{-6}$ (*37*).

Link-based methods (*15*) for solving the TAP emerged as the initial algorithms of choice and were based on Beckmann's formulation (*7*) of TAP. These algorithms prioritized low computational needs over quick convergence. As computational power increased, improved path-based methods like the gradient projection algorithm were proposed (*16*). Research then focused toward a smarter approach which shifts flows between paths without explicitly enumerating them, paving the way for bush-based methods (8*, 9, 17, 18*). Further advances have been made in solving TAP on large networks using methods that successively refine $\epsilon$-optimal flows on the network (*19*). A detailed overview of the field can be found in Patriksson (*38*). More recently, borrowing advances from the field of artificial intelligence, machine learning methods have been applied to the traffic assignment problem (*39*).

Despite these advances, computational power and memory remains a bottleneck for traffic assignment, due to two factors. First, data collection methods have improved, allowing for higher resolution data about networks and yielding significantly more detail about the trip and network data. This includes non-traditional sources like mobile phone data (*40, 41, 42*) or GPS data (*43, 44*), or even advanced vehicle identification (*45*). Secondly, more urban regions have expanded and coalesced into megaregions, consisting of several metropolitan areas, tied together by economic, social, demographic, and other factors, often crossing multiple county and state boundaries. This scale of planning naturally leads to huge networks being studied, testing the limits of current computational resources. In the recent years, multiple studies have concluded that existing proprietary software as well as researchers' own implementations have been unable to process these networks and test scenarios in their entirety (*46, 47, 48, 49*). These scenarios include planning for evacuation, capacity expansion, or project evaluation, to name a few. Researchers have either broken down the scenarios into multiple parts or simplified the scenarios for an approximate assessment.

To address such large networks, research has also focused on parallelizing TAP. Bar-Gera (*11*) proposes an approach incorporating parallelization which depends on paired alternative segments. Some studies (*21*, *22*) attempt to speed up computations by decentralizing and parallelizing computations for all origin-destination pairs. The network simplex problem has been exploited for large scale networks, as proposed by Zheng (*50*). Other studies have adapted origin-based assignment to speed up TAP convergence (*51*, *52*). Jafari et al. (*6*) have proposed the DSTAP algorithm, decomposing the problem by network geography and solving the subnetworks in parallel, which significantly improves the computation time per iteration. This chapter discusses DSTAP, proposes exclusion of subnetwork artificial links from consideration to achieve computation time savings on megaregional networks, and compares the results with the existing state-of-the-art TAP solutions.

## 3.3. DSTAP as algorithm

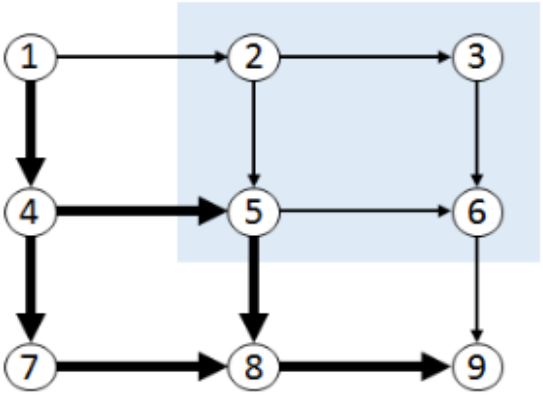
Proposed in Jafari et al. (*6*), DSTAP is a decomposition algorithm that solves traffic assignment on networks which can be partitioned into multiple subnetworks. It is an iterative aggregation-disaggregation algorithm which consists of iteratively solving two problems, called the master problem and the subproblem. The subproblem solves traffic assignment on each subnetwork while the master problem solves traffic assignment on a master network derived from all subnetworks. The master network consists of artificial links which represent an aggregation of the routes within each subnetwork. We call these regional artificial links. The subnetworks consist of artificial links which represent an aggregation of routes between two nodes in the subnetwork passing through other subnetworks. We call these subnetwork artificial links. The parameters of these artificial links are derived using first order approximation methods based on bush-based sensitivity analysis (*13*). Figure 1 shows a flowchart of the DSTAP algorithm. The algorithm starts with an initialization of the artificial link parameters in the master network, then solves the master network to an equilibrium which gives the flow on each artificial link. The flows on these links are used to update the demand while solving the subnetworks. The subnetworks are then solved in parallel with the updated demand. At the end of each iteration, bush-based sensitivity analysis is used to

update the parameters of artificial links. The process is repeated until the relative gap on the complete network drops below a threshold.

### 3.3.1 DSTAP example

We show the steps in one iteration of DSTAP on an example network. Figure 5 shows a nine-node grid network. We choose this simple network to illustrate the concepts of DSTAP; the computational savings on the network may not be relevant.

This network is acyclic and the arrow direction represents the direction of link flow. The links with thicker line width have a free-flow travel time of 9 minutes and a capacity of 500 vehicles per hour, while the other links have a free-flow travel time of 4.5 minutes and a capacity of 250 vehicles per hour. We construct an artificial subnetwork partition where nodes 2, 3, 5, and 6 belong to subnetwork 1 while the other nodes belong to subnetwork 2. The demand between different nodes in the network is also shown in Figure 5. Figure 6 shows the master network and subnetworks constructed from the complete grid network.




| Origin | Destination | Demand |
|---|---|---|
| 1 | 6 | 5000 |
| 1 | 9 | 2000 |
| 2 | 6 | 2000 |
| 4 | 8 | 2000 |

*Figure 5 Grid network and associated demand*

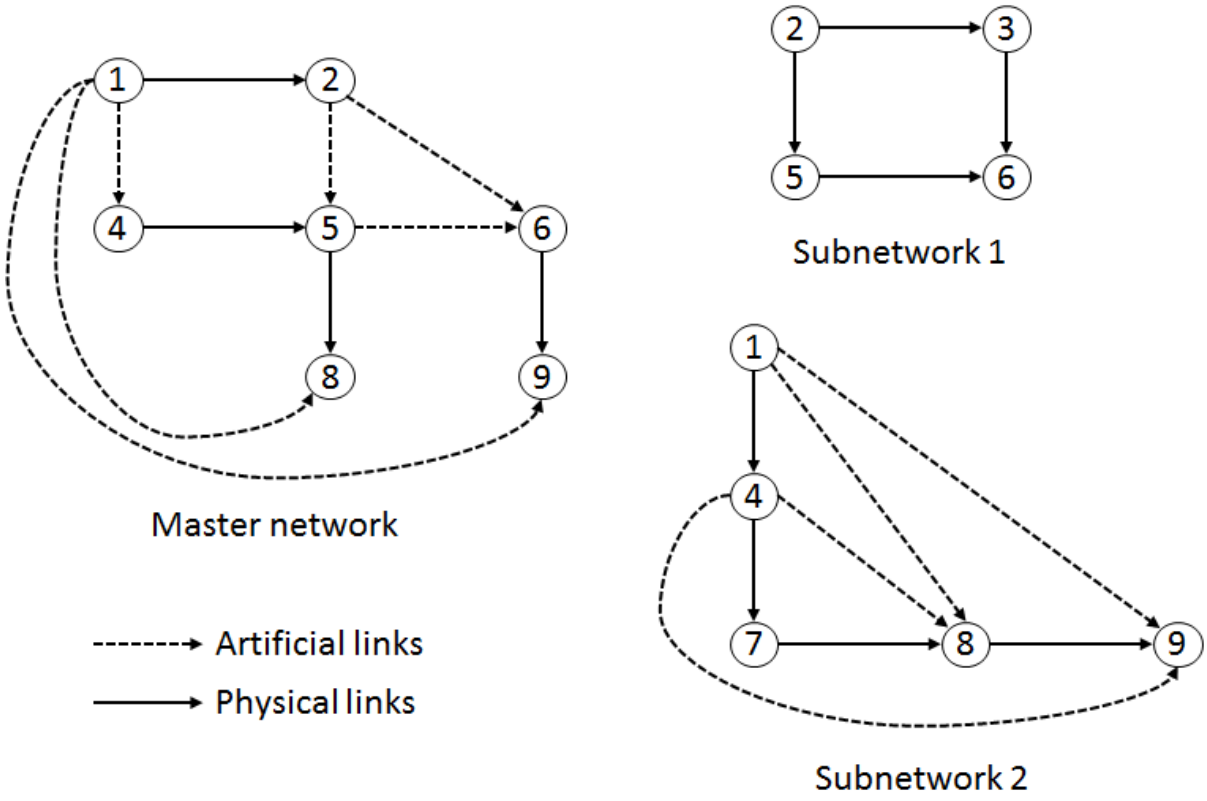



*Figure 6 Master network and subnetworks for DSTAP*

The master network consists of nodes at the boundary of each subnetwork, defined as the nodes on either end of a link cut by the subnetwork boundary. The physical links are the links in the complete network which have head and tail nodes in different subnetworks. The artificial links in the master network represent all paths which are completely contained in any subnetwork which connect two nodes. For example, link (1,9) in the master network represents path [1,4,7,8,9] contained in subnetwork 2. The master network consists of origin-destination pair $1 \rightarrow 6$ as this is the only pair whose nodes are in different subnetworks.

The subnetworks are comprised of the nodes contained in the boundary of each subnetwork, physical links connecting any nodes within each subnetwork, and artificial links between nodes representing all paths connecting two nodes which pass through other subnetworks. For example, link (1,9) in subnetwork 2 represents paths [1,2,3,6,9] and [1,2,5,6,9] which pass through subnetwork 1. Subnetwork 1 contains origin-destination pair $2 \rightarrow 6$, while subnetwork 2 has the origin-destination pairs $1 \rightarrow 9$ and $4 \rightarrow 8$.

The DSTAP algorithm for this network proceeds as illustrated in Figure 1. Table 2 shows the values of flow updates for two consecutive iterations. In iteration 18, we first assign the master network demand from 1 to 6 onto paths [1,4,5,6] and [1,2,6]. Then, the flow along the artificial links in the master network is used to update the demand within the subnetworks. This demand is added to the original demand between the subnetwork nodes. For example, for origin-destination pair $2 \rightarrow 6$ in subnetwork 1, we update the demand as the sum of original demand between 2 and 6 (2000 units of flow) and the demand coming into the network from the master network (1863.5 units of flow).

Then, each subnetwork is solved in parallel. We add an extra 732.31 units of flow to the demand between nodes 2 and 6 in subnetwork 1, as this is the flow on the artificial link (1,9) within subnetwork 2 in the previous iteration which is assigned as a demand because the paths represented by the artificial link (1,9) in subnetwork 1 pass through these nodes.

Next, the algorithm updates the parameters for artificial links in master network and the subnetworks. These parameters are calculated using bush-based sensitivity analysis and represent the change in travel time between two nodes with variation in the demand between two nodes. For example, the parameters for link (1,9) in subnetwork 2 are calculated by observing the change in the travel times between nodes 2 and 6 if the flow between those nodes were to change by amount $\Delta x$. The new travel time function is given by 18915.29 + 30.94*(x-714.52), which suggests that if the flow on the link increases from its current value of 714.52 units of flow, then the travel time will increase by the rate of 30.94 time units for every unit increase in the flow.

In the last step, the algorithm maps the flows obtained by solving the master network and the subnetworks onto the full network. This mapping is done by first finding an equivalent full-network path for each path in the master or subnetworks; the flow on that path is then assigned in proportion to the flow between the nodes. For example, the flow on path [1,2,6] in the master network is mapped to paths [1,2,3,6] and [1,2,5,6] and the assigned flow is proportional to the flow split at node 2.

*Table 2 Sample iteration data for DSTAP on example grid network*

**Iteration 18**

**Master network assignment paths**

| OD pair | Demand | Paths | Path flow |
|---|---|---|---|
| 1 --> 6 | 2000 | [1,4,5,6] | 136.5 |
| | | [1,2,6] | 1863.5 |

**Updating subnetwork OD demand**

| | OD pair | Demand |
|---|---|---|
| Subnetwork 1 | 1-->4 | 136.5 |
| Subnetwork 2 | 2-->6 | 1863.5 + 2000 = 3863.5 |
| | 5-->6 | 1863.5 |

**Subnetwork assignment paths**

**Subnetwork 1**

| OD pair | Demand | Paths | Path flow |
|---|---|---|---|
| 2 --> 6 | 3863.5 + 732.31 | [2,5,6] | 1474.00 |
| | | [2,3,6] | 3121.81 |
| 5 --> 6 | 136.5 | [5,6] | 136.50 |

**Subnetwork 2**

| OD pair | Demand | Paths | Path flow |
|---|---|---|---|
| 1 --> 9 | 5000 | [1,4,9] | 213.20 |
| | | [1,9] | 714.52 |
| | | [1,4,8,9] | 942.15 |
| | | [1,4,7,8,9] | 3130.13 |
| 1 --> 4 | 136.5 | [1,4] | 136.5 |
| 4 --> 8 | 2000 | [4,8] | 2000 |

**Update artificial link parameters**

| Network | Link | Travel time as function of flow x |
|---|---|---|
| Master net | (1,8) | 12432.8 + 12.77x |
| | (1,9) | 18382.1 + 18.61x |
| | (1,4) | 8267.9 + 7.47 (x - 136.50) |
| | (2,6) | 11275.89 + 9.39 (x - 1863.5) |
| | (5,6) | 7176.77 + 11.3 (x - 136.5) |
| Subnetwork 2 | (1,8) | 13233.5 + 21.29x |
| | (1,9) | 18915.29 + 30.94 (x - 714.52) |
| | (4,8) | 4172.86 + 5.28 (x - 2942.15) |
| | (4,9) | 9854.63 + 14.94 (x - 213.2) |

**Complete network flows**

| OD pair | Path | Path flow |
|---|---|---|
| 1 --> 9 | [1,2,5,6,9] | 714.52 |
| | [1,4,5,6,9] | 213.2 |
| | [1,4,7,8,9] | 3130.13 |
| | [1,4,5,8,9] | 942.15 |
| 2 --> 6 | [2,3,6] | 1236.96 |
| | [2,5,6] | 763.04 |
| 4 --> 8 | [4,7,8] | 1.00E-10 |
| | [4,5,8] | 2000 |
| 1 --> 6 | [1,2,3,6] | 1152.54 |
| | [1,4,5,6] | 136.5 |
| | [1,2,5,6] | 710.96 |

**Full network gap = 0.0214**

**Iteration 19**

**Master network assignment paths**

| OD pair | Demand | Paths | Path flow |
|---|---|---|---|
| 1 --> 6 | 2000 | [1,4,5,6] | 120.28 |
| | | [1,2,6] | 1879.72 |

**Updating subnetwork OD demand**

| | OD pair | Demand |
|---|---|---|
| Subnetwork 1 | 1-->4 | 120.28 |
| Subnetwork 2 | 2-->6 | 1879.72 + 2000 = 3879.72 |
| | 5-->6 | 1879.72 |

**Subnetwork assignment paths**

**Subnetwork 1**

| OD pair | Demand | Paths | Path flow |
|---|---|---|---|
| 2 --> 6 | 3879.72 + 714.52 | [2,5,6] | 1490.53 |
| | | [2,3,6] | 3103.71 |
| 5 --> 6 | 120.28 | [5,6] | 120.28 |

**Subnetwork 2**

| OD pair | Demand | Paths | Path flow |
|---|---|---|---|
| 1 --> 9 | 5000 | [1,4,9] | 230.67 |
| | | [1,9] | 697.53 |
| | | [1,4,8,9] | 941.49 |
| | | [1,4,7,8,9] | 3130.3 |
| 1 --> 4 | 120.28 | [1,4] | 120.28 |
| 4 --> 8 | 2000 | [4,8] | 2000 |

**Update artificial link parameters**

| Network | Link | Travel time as function of flow x |
|---|---|---|
| Master net | (1,8) | 12439.58 + 12.77x |
| | (1,9) | 18386.0 + 18.61x |
| | (1,4) | 8273.66 + 7.47 (x - 120.28) |
| | (2,6) | 11270.02 + 9.39 (x - 1879.72) |
| | (5,6) | 7180.29 + 11.3 (x - 120.28) |
| Subnetwork 2 | (1,8) | 13219.51 + 21.27x |
| | (1,9) | 18912.0 + 30.93 (x - 697.53) |
| | (4,8) | 4173.28 + 5.28 (x - 2941.49) |
| | (4,9) | 9865.77 + 14.94 (x - 230.67) |

**Complete network flows**

| OD pair | Path | Path flow |
|---|---|---|
| 1 --> 9 | [1,2,5,6,9] | 697.53 |
| | [1,4,5,6,9] | 230.68 |
| | [1,4,7,8,9] | 3130.3 |
| | [1,4,5,8,9] | 941.49 |
| 2 --> 6 | [2,3,6] | 1231.63 |
| | [2,5,6] | 768.37 |
| 4 --> 8 | [4,7,8] | 1.00E-10 |
| | [4,5,8] | 2000 |
| 1 --> 6 | [1,2,3,6] | 1157.56 |
| | [1,4,5,6] | 120.28 |
| | [1,2,5,6] | 722.16 |

**Full network gap = 0.0205**

Figure 7 shows the variation in relative gap values with iteration number for the master network, each subnetwork, and the full (combined) network. As observed, even though the relative gap measures for master network and subnetworks oscillate, the full network gap decreases steadily with each iteration and drops below 1E-5 after 46 iterations.

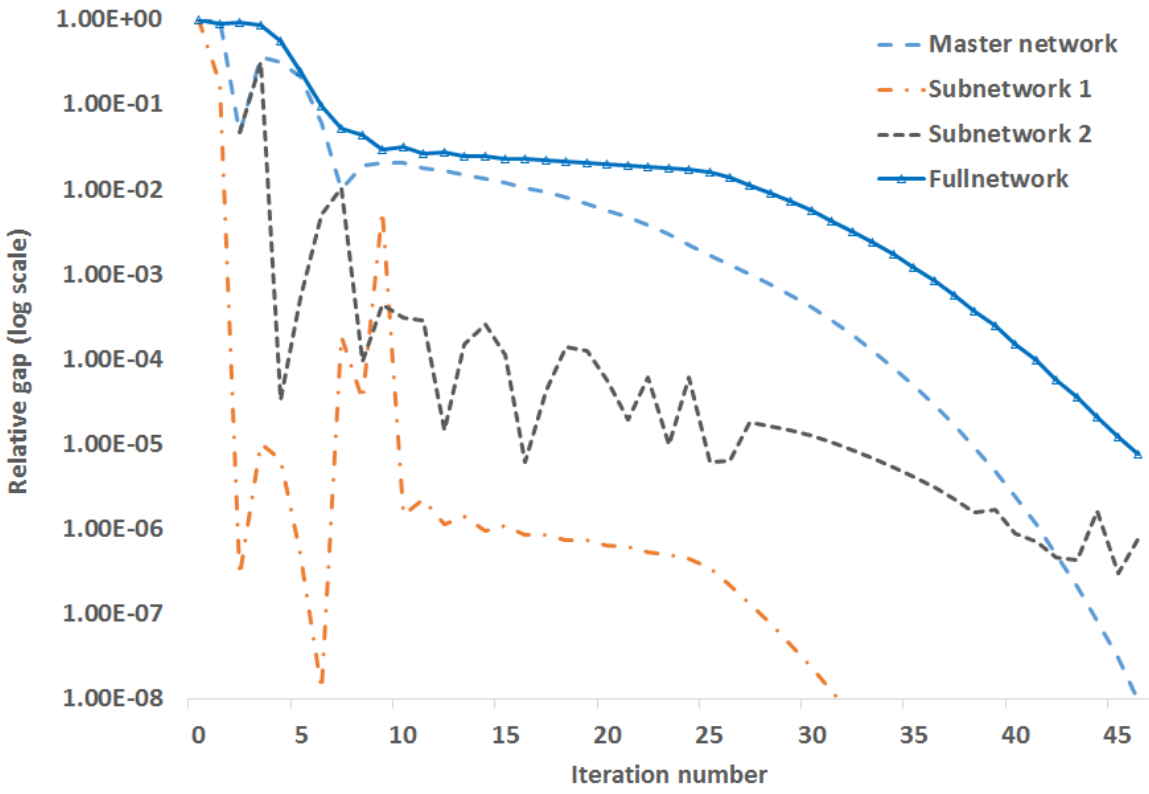



*Figure 7 Progression of relative gap for the components of DSTAP*

## 3.3.2 DSTAP for Megaregions

Solving traffic assignment on large-scale networks is time-intensive. DSTAP can be used to reduce the computation time on larger networks by decomposing a large network into subnetworks and solving the subnetworks in parallel. An instance of a large-network implementation of DSTAP is presented in Jafari et al. (*6*), where computational savings ranging between 35-70% are obtained on the Austin regional network. As noted in the study, the efficiency of the DSTAP algorithm depends on multiple factors, including the number of boundary nodes, size of the regional origin-destination (O-D) matrix, and interaction between subnetworks. Thus, developing appropriate

subnetworks from a complete network is crucial in determining the reduction in computation time. Yahia et al. (*53*) in their experiments on an artificially-constructed double Sioux Falls network show that poorly-chosen subnetworks may lead the algorithm to not converge within a reasonable time. In the next section, we explore the possibility of using DSTAP as a heuristic for certain subnetworks.

## 3.4. Modified DSTAP as a heuristic

DSTAP, as an algorithm, is proven to converge to the TAP solution. But for applications such as megaregional TAP, where computation time is a significant constraint, the algorithm can be modified to act as a heuristic, cutting down on computation time. Subnetwork artificial links, representing flows assigned for an O-D pair but which pass through subnetworks not containing the origin and the destination, can be excluded from the implementation of DSTAP. For DSTAP without subnetwork artificial links, this flow is assigned to other routes in the pathset or, potentially, a new path or paths. This allows DSTAP to reduce computation time, as discussed later. A heuristic gap may be introduced, representing the aggregated reduction in pathsets and their impact on the solution accuracy. The heuristic gap is the minimum relative gap that the heuristic can achieve. With an appropriate choice of subnetworks, subnetwork artificial links do not come into play, allowing for 100% convergence without any heuristic gap. For other partitions, there exists a non-zero heuristic gap. The heuristic will asymptotically approach the heuristic gap, but the network relative gap can never fall below the heuristic gap.

We conduct experiments of DSTAP as a heuristic on the Texas Statewide Analysis Model (SAM) network, which entirely encompasses the Texas Triangle megaregion. Provided by Texas Department of Transportation, the network has 122,658 links and 44,796 nodes. The overall file contains the road network, along with airline routes, train routes and waterways. For the purpose of this experiment, we use only the road network. The dataset provides all basic characteristics of the network, as well as additional future forecasts and tolls. Figure 8 shows the network under consideration.

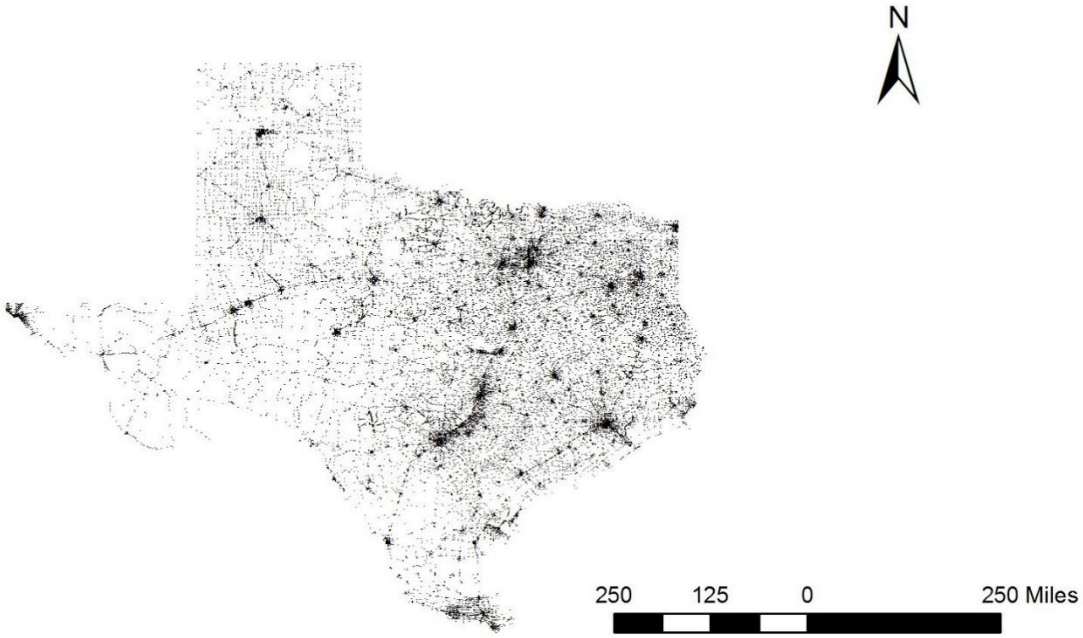



*Figure 8 Visualization of the Texas SAM network*

Figure 9 shows the computational performance of the DSTAP heuristic, contrasted with gradient projection, a path-based algorithm *(16)*. We solve the Texas network to a relative gap of $10^{-4}$ for two demand levels. The demand is reduced to 40% of the original to solve an uncongested network. For the reduced-demand scenario, computational performance is similar, with both heuristics dropping to $10^{-4}$ relative gap within almost identical computation time. Being in an uncongested state, this occurs because most (>95%) traffic is on the shortest path, thus leading both heuristics to achieve the target within a few iterations. For the full-demand scenario, we observe that the DSTAP heuristic drops to a gap of 0.01 in 1907 seconds, while gradient projection takes 6939 seconds to achieve the same relative gap, an increase of 21% in computation time. However, the gap for the DSTAP heuristic does not drop below 0.009 after subsequent iterations, indicating a heuristic gap value of 0.009, which is reasonable for large scale networks. This gap value is achieved in ~4,000 seconds.

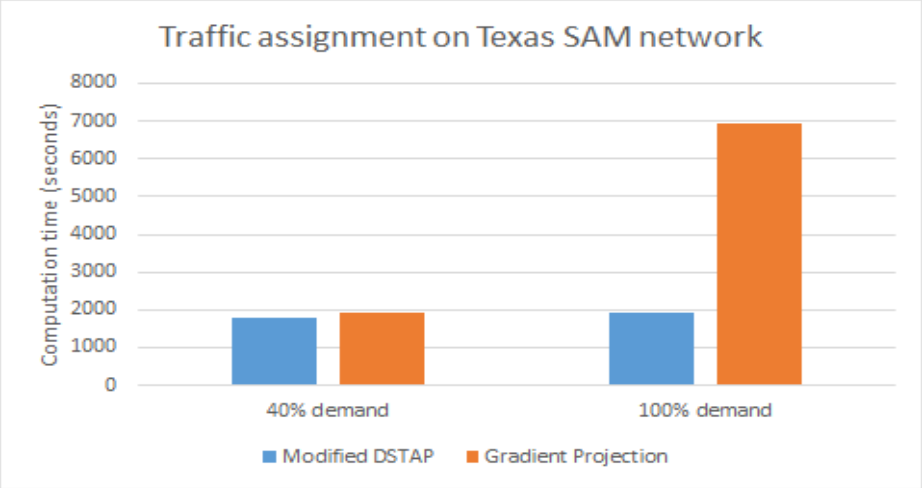



*Figure 9 Computational comparison of DSTAP heuristic and Gradient Projection*

Based on the results, the proposed heuristic saves computation time to the same gap level, especially for congested network flows. Despite being based on subnetworks not explicitly designed to minimize the heuristic gap, there is a noticeable performance gain. The traffic assignment problem is, at a minimum, non-linear in nature. Splitting a large network of size *N* into subnetworks, solving them in parallel, and then solving a sketch of the master network thus allows for reduction in computational operations. As a heuristic, we reduce the problem size further by not creating additional links, thus reducing the number of operations needed and resulting in a faster solution. The downside of this method is the introduction of the aforementioned heuristic gap, which restricts some flows that are never assigned to the right paths.

The heuristic gap is caused by paths which "zig-zag" between multiple partitions; that is, the path starts in a subnetwork, passes through one or more distinct subnetworks (including potentially returning to the original subnetwork) before terminating in the original subnetwork. All demand from these paths is forcibly assigned to paths which are contained within the original subnetwork. We can reduce the heuristic gap by choosing appropriate subnetworks. Consider the Austin network shown in Figure 10. If the network is divided along the river, resulting in a northern and southern subnetwork, the DSTAP heuristic has a heuristic gap lower than $10^{-6}$ *(6)*. This is to be expected from Austin topography, with bridges on the river being far apart; the equilibrium path for a traveler will not cross the river at a bridge, drive to another bridge, and cross back to the original side of the river to reach their destination. These examples lead to the logical question

about the best choice of subnetworks for a given network. Section 3.5 discusses partitioning algorithms and compares them with respect to DSTAP requirements.

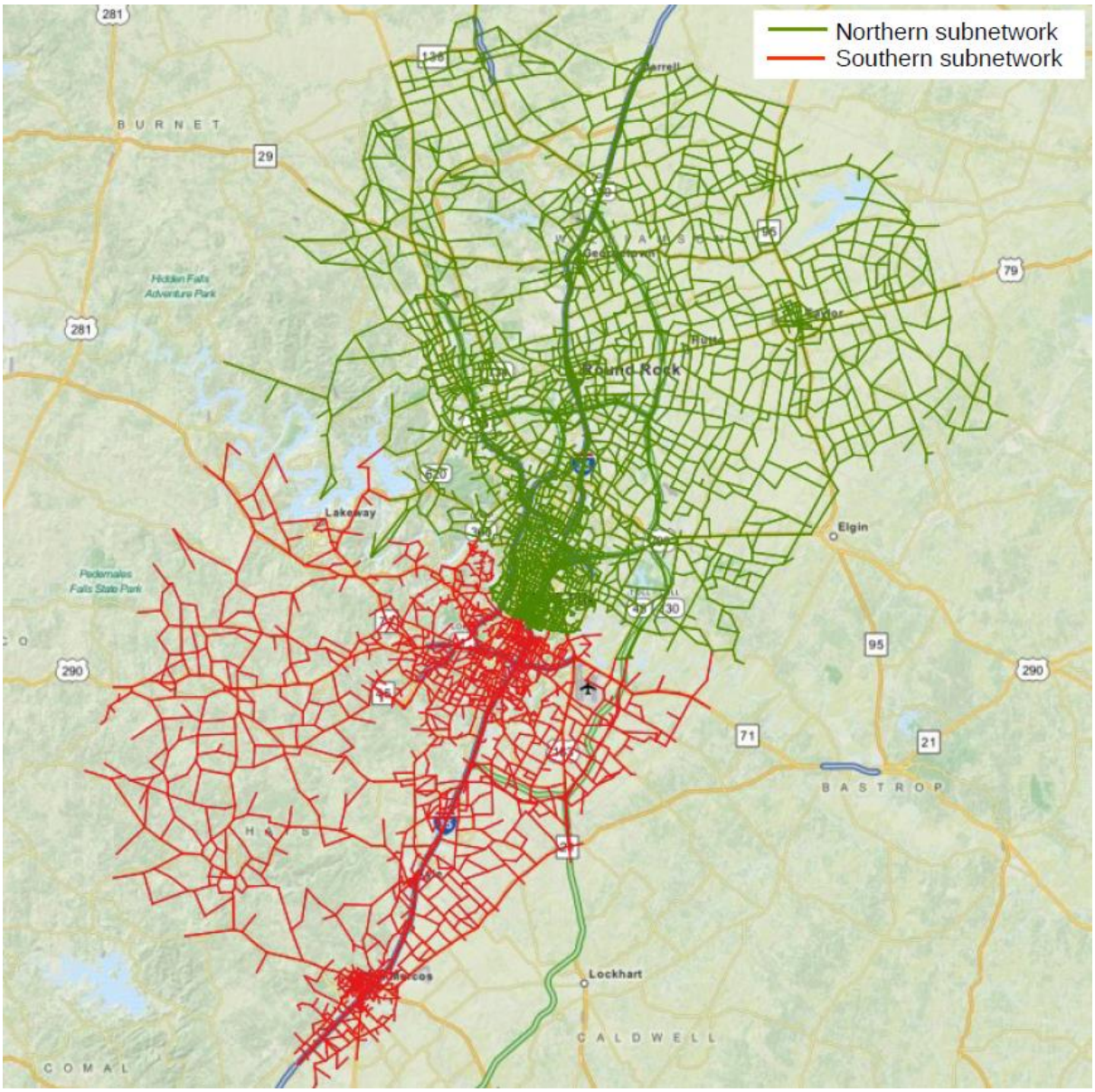



*Figure 10 North and south subnetworks for the Austin network (Source- (6))*

## 3.5. Comparison of partitioning algorithms

The aim of a partitioning algorithm is to divide a network graph into two or more subnetworks, typically based on flow balance or minimizing the number of boundary nodes while balancing the

subnetwork size. Network partitioning methods for transportation networks typically fall into one of three categories: agglomerative clustering heuristics, integer programming-based methods, and spectral partitioning (SP) algorithms. Agglomerative heuristics involve methods based on greedy heuristics focusing on flow (*23*) or "snake" similarities (*25*). Johnson et al. (*14*) proposed a heuristic focusing on creating subnetworks of similar size by reducing the number of boundary nodes. Integer programming methods have been proven to be NP-hard (*23*). Researchers have proposed approximations to these formulations to reduce computation time (*22, 24*). Spectral partitioning methods are based on the eigenvalues of the Laplacian matrix, balancing the subnetworks by the chosen weight (*27, 28, 29, 32, 34*).

For the purpose of this chapter, we compare two partitioning algorithms from Chapter 2 and discuss the properties of partitioning algorithms suited to traffic assignment. We study and compare the shortest domain decomposition algorithm (SDDA) and the spectral partitioning algorithm for generating subnetworks for the Texas network. We briefly review the algorithms below.

SDDA seeks to find partitions which minimize the number of boundary nodes generated between the partitions. For a given number of partitions n, it first finds a source node for each partition from which the entire partition is to be constructed. The process starts with the first source node, then finds the next source node by finding a node farthest away from the original source node using a distance metric, then repeats the process to find all source nodes. The metric can be modified to other variables, such as time or number of links on a breadth-first search (BFS) tree. After source node selection, all nodes are assigned a rank vector based on their location in the BFS tree originating at each source. The node is assigned to the subnetwork of the source with the lowest rank. The boundary nodes and links are then calculated and are stored for further use. The algorithm is shown in Algorithm 1. SDDA needs only the network and link characteristics as input, and provides a list of nodes in each subnetwork.

Spectral partitioning seeks to find partitions that minimize the inter-flow between subnetworks and generates subnetworks which are balanced in size. The partitioning algorithm, shown in Algorithm 2, requires network and link data, along with approximate flow values on each link. It

calculates the minimum-cost cut across the network by computing eigenvector and eigenvalues of the Laplacian matrix. A detailed mathematical explanation of the algorithm can be found in Yahia et al. (*53*). For this algorithm, two major concerns are the availability of flow data and network connectivity. The link flow data may not be available before start of the simulation and, if the network is disjoint, the algorithm may terminate prematurely. However, these concerns are minor for applications in megaregions planning. The transportation network for megaregions, by definition, forms a connected graph obviating the second concern. Next, traffic assignment needs to be solved multiple times for long term planning purposes ensuring that the partitions generated from flows obtained after the first run can be reused for the future applications of the DSTAP heuristic.

Figure 11 compares the plots of partitions generated by the SP and SDDA methods. As can be observed, the generated partitions are different from each other. In SDDA partitions, multiple urban areas (depicted in green) are separated into different partitions with the partition passing through the Austin city center. This type of partitioning may cause a high heuristic gap when solving the modified DSTAP on the generated subnetworks. The spectral partitioning algorithm does better at keeping the major cities with many inter-regional trips within one partition so that the interflow between the partitions is minimized.

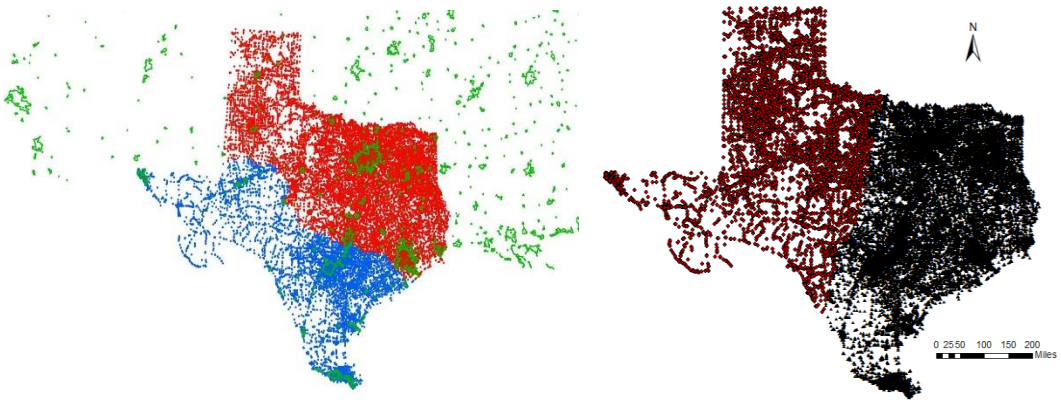



*Figure 11 SDDA (left) and spectral partitioning (right) outputs for the Texas network*

While only the two-partition case is explored in depth here, a few important observations are presented for higher numbers of partitions. SDDA selects source nodes sequentially, based on the

cumulative rank vector of all nodes. This implies that if a node is a source for the $k^{th}$ partition, for any number of partitions greater than *k*, that node continue to be a source. This results in issues at higher numbers of partitions, when two sources end up close to each other, resulting in one generating a partition with very few nodes. On the Texas network, the source nodes for the $7^{th}$ and $8^{th}$ partitions would generate partitions of eight nodes for all cases in which they were used, due to proximity with source node 1 and source node 3, respectively. Figure 12 shows these cases, with the partitions being visible near Texarkana, TX and Brownsville, TX. This static selection of source nodes can cause an imbalance in parallelizing TAP for many subnetworks. SDDA, due to the lack of flow data, was shown to divide multiple urban areas (e.g. Austin, San Angelo) into more than two partitions for higher number of partitions, as shown in Figure 13.

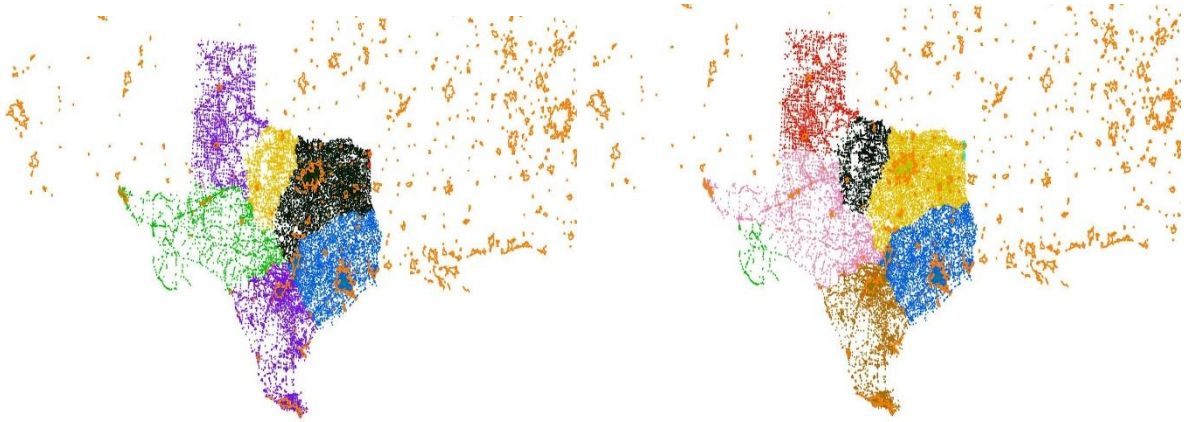


*Figure 12 Eight- and nine- partition case for the SDDA algorithm (each color denotes a different partition)*

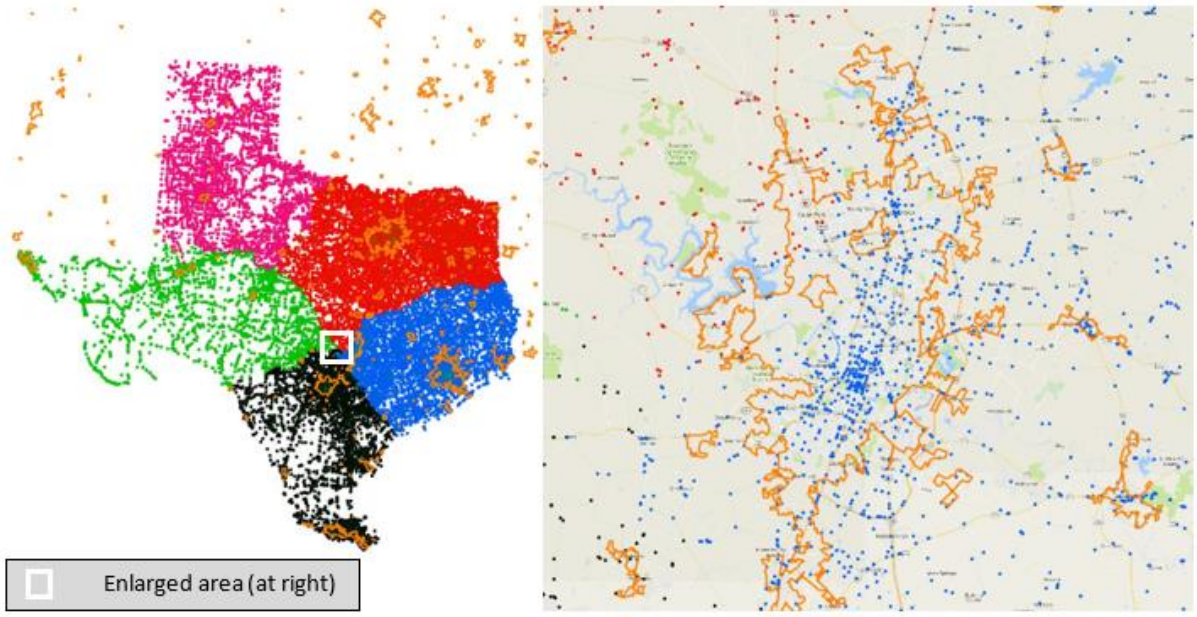



*Figure 13 Five-partition case for SDDA, with a focus on the Austin area*

Spectral partitioning, while providing subnetworks which are better suited for DSTAP, alters its partitions with changes in input flow. The important implication here is that approximate flows allow for partitions which are equally good when compared to flows at convergence. Figure 14 looks at the regions with flow difference for SP partitions at two different input flow convergence levels, 1E-2 and 1E-5. On the SAM network, 94.6% of the links had exactly identical flow for the two convergence levels. Of the 5.4% links where flow differed, 90% had a flow difference below 3% and 80% had a flow difference below 1.4%. The nodes in red are connected to links where flow differs. Figure 15 shows the SP algorithm output for 4 partition case, where it can be observed that urban regions have been left within one partition, unlike the SDDA algorithm partitions in Figure 13.

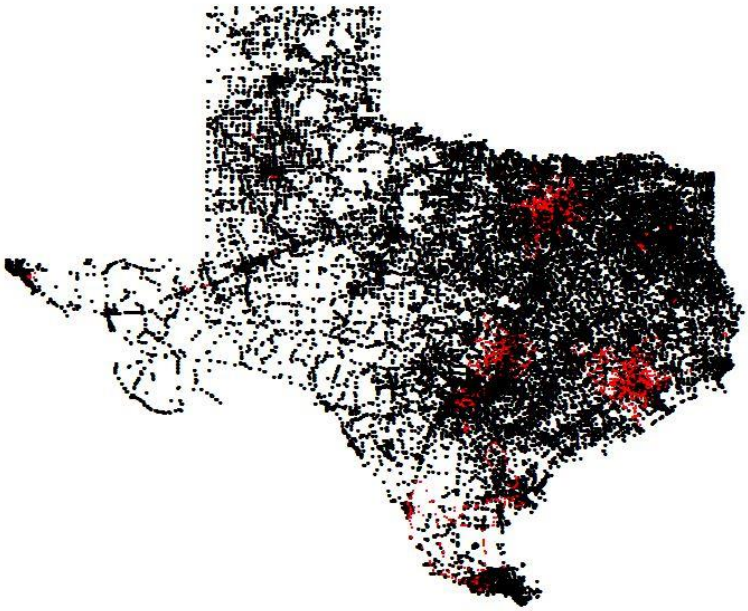


*Figure 14 Areas with flow difference between two different convergence levels*

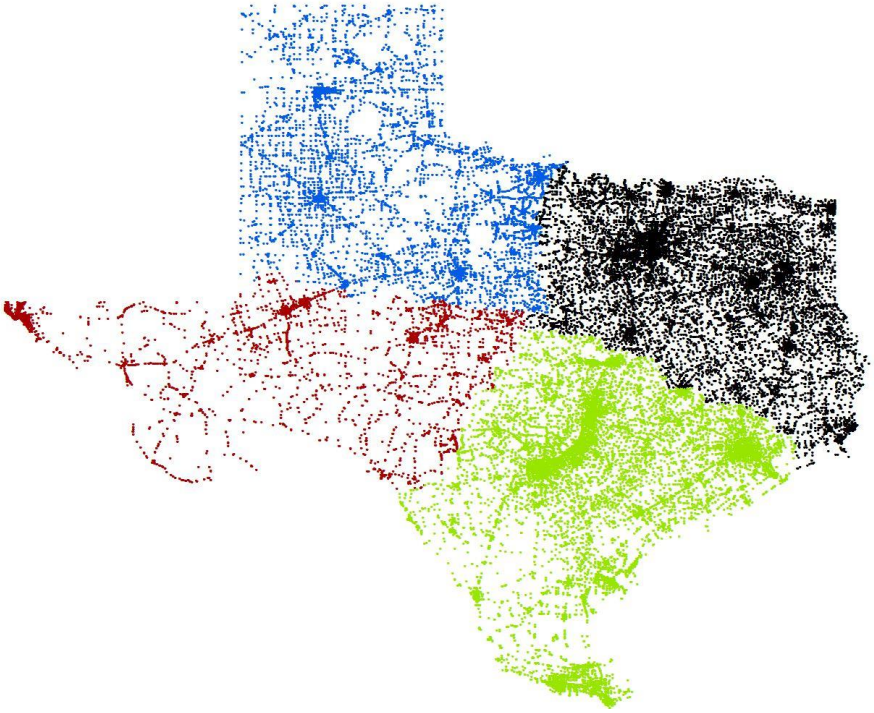


*Figure 15 SP partition for the 4-partition case*

## 3.6. Conclusion

This chapter focuses on network models for megaregions, and their application for the traffic assignment problem. DSTAP is shown to be an efficient algorithm for megaregion networks, suited to take advantage of the megaregional geography to aid in parallelizing TAP computations by decomposing the network. It solves TAP on the megaregion network by iteratively solving TAP on the sub-networks in parallel, then solving the master network and finally, stitching the flows. Removal of subnetwork artificial links turns DSTAP into a heuristic for megaregional TAP. The heuristic has simplified interactions between different subnetworks, achieving better computation times on megaregional geography. When contrasted with the gradient projection algorithm, it provided marginal computational savings (~5%) for uncongested networks for similar gaps, but significant time savings (~70%) for congested networks.

DSTAP as a heuristic introduces a heuristic gap, representative of the differences between the actual and simplified interactions within subnetworks. The heuristic gap is heavily affected by the choice of subnetworks which dictate the level of interaction between subnetworks. Various partitioning algorithms were surveyed, with two being implemented: shortest domain decomposition algorithm and spectral partitioning. SDDA is an agglomerative clustering algorithm, requiring only the network information, and partitions the network aiming to create balanced partitions with a minimum number of boundary nodes. SP is a flow-weighted clustering algorithm, aiming to find the min-flow cut while balancing the partitions using total flow. Using the partitions from both algorithms as input for modified DSTAP, we observe that the heuristic gap was lower for the SP partitions (5E-4) as opposed to SDDA partitions (0.009). A few advantages and drawbacks of each algorithm are discussed, concluding that SP is better suited for transportation planning applications.

# Chapter 4. Conclusion and Recommendations

Motivated by the challenges in the scalability of traditional models used for solving TAP on large-scale megaregional networks, this research evaluated the usefulness of DSTAP algorithm which uses a decomposition approach to perform traffic assignment. The decomposition is performed based on network geography which makes this algorithm useful for solving TAP on megaregions, where each component of the megaregion can be modeled as a detailed subnetwork and the interactions between these components can be modeled using artificial links in a master network.

The research focused on two questions: how to determine appropriate partitions of large-scale networks and how to quantify and simplify interactions between subnetworks for faster computation time. For the first research question two different partitioning algorithms were tested on different large-scale networks. These include the SDDA algorithm that minimizes the number of boundary nodes between the subnetworks and creates partitions that are balanced in size and the flow-weighted spectral partitioning algorithm that generates partitions with less interflow. It was shown that compared to the SDDA algorithm, the partitions generated by spectral partitioning had higher number of boundary nodes while the average computation time per iteration for solving DSTAP was lower. The spectral partitioning also generated partitioning which were more balanced in size compared to the SDDA partition leading to faster convergence rate. For the second research question, subnetwork artificial links were removed turning DSTAP into a heuristic for solving TAP on large-scale networks. A heuristic gap was introduced representing the differences between the actual and simplified interactions within subnetworks. For the Texas SAM network, heuristic gaps of 0.009 and 5E-4 were achieved using SDDA and spectral partitions, respectively. The heuristic gap is shown to be affected by the choice of subnetworks which dictate the level of interaction between subnetworks. In terms of computation time savings, when contrasted with the gradient projection algorithm, the heuristic provided marginal computational savings (~5%) for uncongested networks for similar gaps, but significant time savings (~70%) for congested networks.

Based on the findings of this research, we make following recommendations. First, DSTAP algorithm *(6)* can be used for solving TAP on large-scale megaregional networks with different

subregions modeled as subnetworks. To determine the boundaries of the subnetworks, the spectral partitioning algorithm is recommended as it finds subnetworks such that the interflow between the subnetworks is minimized. Second, DSTAP heuristic proposed in Chapter 3 can be used as an approximation for solving TAP under limited computational resources. The heuristic gap depends on the choice of network partitions and thus different partitions should be evaluated before implementing the heuristic for planning purposes.

This research leads to several directions for the future work. First, other partitioning algorithms that aim to simultaneously minimize boundary nodes and inter-flow can be explored. Second, alternative approximation algorithms can be studied that reduce the number of artificial links generated by DSTAP and thus improving the convergence rate and the computational efficiency of the algorithm. Third, a theoretical analysis of the heuristic gap can be performing including questions like finding an upper bound on the heuristic gap given a partitioning algorithm and a network and finding appropriate partitioning algorithms that reduce this heuristic gap below a threshold. Last, based on the data availability, the transferability of the performance of DSTAP with a selected partitioning algorithm should be tested from one megaregion to the other.

# References


1. A. C. Nelson. Megaregion Projections 2015 to 2045 with Transportation Policy Implications. Transportation Research Record: Journal of the Transportation Research Board, (2654), 11-19, 2017.
2. Regional Plan Association, Megaregions, http://www.america2050.org/content/megaregions.html. Accessed on December 27, 2018.
3. A. Nelson and R. Lang. Megapolitan America. *Routledge*. 2011.
4. M. Dewar and D. Epstein. Planning for "megaregions" in the United States. *Journal of Planning Literature*, *22*(2), 108-124, 2007.
5. G. D. Nelson, and A. Rae. An economic geography of the United States: From commutes to megaregions. *PloS one*, *11*(11), e0166083, 2016.
6. E. Jafari, V. Pandey, and S. D. Boyles. A decomposition approach to the static traffic assignment problem. *Transportation Research Part B: Methodological*, 105, 2017.
7. M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.
8. H. Bar-Gera. Origin-based algorithm for the traffic assignment problem. *Transportation Science*, 36(4), 2002.
9. R. B. Dial. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B: Methodological*, 40(10), 2006.
10. B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1), 2007.
11. H. Bar-Gera. Traffic assignment by paired alternative segments. *Transportation Research Part B: Methodological*, 44(8-9), 2010.
12. K. Abdelghany, H. Hashemi, and A. Alnawaiseh. Parallel all-pairs shortest path algorithm: Network decomposition approach. *Transportation Research Record: Journal of the Transportation Research Board*, 2567, 2016.
13. S. D. Boyles. Bush-based sensitivity analysis for approximating subnetwork diversion. *Transportation Research Part B: Methodological*, 46(1), 2012.

14. P. Johnson, D. Nguyen, and M. Ng. Large-scale network partitioning for decentralized traffic management and other transportation applications. *Journal of Intelligent Transportation Systems*, 20(5), 2016.
15. M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics*,3, 1956.
16. R. Jayakrishnan, W. Tsai, J. Prasker, and S. Rajadhyaksha. A faster path-based algorithm for traffic assignment. *Transportation Research Record: Journal of the Transportation Research Board*, 1443, 1994.
17. Y. M. Nie. A class of bush-based algorithms for the traffic assignment problem. *Transportation Research Part B: Methodological*, 44(1), 2010.
18. G. Gentile. Local User Cost Equilibrium: A bush-based algorithm for traffic assignment. *Transportmetrica A: Transport Science*, 10(1), 2014.
19. H. Zheng and S. Peeta. Cost scaling based successive approximation algorithm for the traffic assignment problem. *Transportation Research Part B: Methodological*, 68, 2014.
20. M. Josefsson and M. Patriksson. Sensitivity analysis of separable traffic equilibrium equilibria with application to bilevel optimization in network design. *Transportation Research Part B: Methodological*, 41(1), 2007.
21. R. Chen and R. R. Meyer. Parallel optimization for traffic assignment. *Mathematical Programming*, 42(1), 1988.
22. P. A. Lotito. Issues in the implementation of the DSD algorithm for the traffic assignment problem. *European Journal of Operational Research*, 175(3), December 2006.
23. H. Etemadnia, K. Abdelghany, and A. Hassan. A network partitioning methodology for distributed traffic management applications. *Transportmetrica A: Transport Science*, 10(6), 2014.
24. N. Garg, V. V. Vazirani, andM. Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2), 1996.
25. M. Saeedmanesh and N. Geroliminis. Clustering of heterogeneous networks with directional flows based on “snake” similarities. *Transportation Research Part B: Methodological*, 91, 2016.
26. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1), 1998.

27. D. A. Spielman. Spectral graph theory and its applications. IEEE, 2007.
28. D. A. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, 421(2-3), 2007.
29. M. E. J. Newman. Spectral methods for community detection and graph partitioning. *Physical Review* E, 88(4), 2013.
30. U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.
31. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8), 2000.
32. M. G.H. Bell, F. Kurauchi, S. Perera, and W. Wong. Investigating transport network vulnerability by capacity weighted spectral analysis. *Transportation Research Part B: Methodological*, 99, 2017.
33. S.Martinez, G. Chatterji, D. Sun, and A.M. Bayen. A weighted-graph approach for dynamic airspace configuration. In *Proceedings of the AIAA Conference on Guidance, Navigation, and Control (GNC). American Institute of Aeronautics and Astronautics*, 2007.
34. Y. Ma, Y. Chiu, and X. Yang. Urban traffic signal control network automatic partitioning using laplacian eigenvectors. In *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference On Intelligent Transportation Systems*. IEEE, 2009.
35. E. Jafari and S. D. Boyles. Improved bush-based methods for network contraction. *Transportation Research Part B: Methodological*, 83, 2016.
36. H. Bar-Gera. Transportation network test problems. http://www.bgu.ac.il/bargera/tntp/, 2017.
37. D. Boyce, B. Ralevic-Dekic, and H. Bar-Gera. Convergence of traffic assignments: how much is enough?. *Journal of Transportation Engineering*, *130*(1), 49-55, 2004.
38. M. Patriksson. *The traffic assignment problem: models and methods*. Courier Dover Publications. 2015.
39. G. De Luca and M. Gallo. Artificial neural networks for forecasting user flows in transportation networks: Literature review, limits, potentialities and open challenges. In *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on* (pp. 919-923). IEEE, 2017

40. C. H. Chapman and O. B. Downs. *U.S. Patent No. 7,831,380*. Washington, DC: U.S. Patent and Trademark Office, 2010.
41. J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen. Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment. *Transportation Research Part C: Emerging Technologies*, *18*(4), 568-583, 2010.
42. Z. Wang, S. Y. He, and Y. Leung. Applying mobile phone data to travel behaviour research: A literature review. *Travel Behaviour and Society*, *11*, 141-155, 2018.
43. P. S. Castro, D. Zhang, and S. Li. Urban traffic modelling and prediction using large scale taxi GPS traces. In *International Conference on Pervasive Computing* (pp. 57-72). Springer, Berlin, Heidelberg, 2012.
44. D. Woodard, G. Nogin, P. Koch, D. Racz, M. Goldszmidt, and E. Horvitz. Predicting travel time reliability using mobile phone GPS data. *Transportation Research Part C: Emerging Technologies, 75*, 30-44, 2017.
45. S. Carrese, E. Cipriani, L. Mannini, and M. Nigro. Dynamic demand estimation and prediction for traffic urban networks adopting new data sources. *Transportation Research Part C: Emerging Technologies*, *81*, 83-98, 2017.
46. R. Harrison, D. Johnson, L. Loftus-Otway, N. Hutson, D. Seedah, M. Zhang, and C. Lewis. *Megaregion freight planning: A synopsis* (No. FHWA/TX-11/0-6627-1), 2012.
47. Z. Zhang, K. Spansel, and B. Wolshon. Megaregion network simulation for evacuation analysis. *Transportation Research Record: Journal of the Transportation Research Board*, (2397), 161-170, 2013.
48. Z. Zhang and B. Wolshon. *Gulf Coast Megaregion Evacuation Traffic Simulation Modeling and Analysis* (No. SWUTC/15/600451-00101-1). Southwest Region University Transportation Center (US), 2015.
49. Z. Zhang and B. Wolshon. *Analysis of Evacuation Clearance Time under Megaregion Disaster Threats* (No. SWUTC/16/600451-00114-1). Southwest Region University Transportation Center (US), 2016.
50. H. Zheng. Adaptation of network simplex for the Traffic Assignment Problem. *Transportation Science*, *49*(3), 543-558, 2015.

51. J. Xie and C. Xie. Origin-based algorithms for traffic assignment: algorithmic structure, complexity analysis, and convergence performance. *Transportation Research Record: Journal of the Transportation Research Board*, (2498), 46-55, 2015.
52. B. Javani and A. Babazadeh. Origin-destination-based truncated quadratic programming algorithm for traffic assignment problem. *Transportation Letters*, *9*(3), 166-176, 2017.
53. C. N. Yahia, V. Pandey, and S. D. Boyles. Network Partitioning Algorithms for Solving the Traffic Assignment Problem using a Decomposition Approach. *Transportation Research Record*, 0361198118799039, 2018.