# Network Partitioning Algorithms for Solving the Traffic Assignment Problem Using a Decomposition Approach

Cesar N. Yahia
Graduate Research Assistant
Department of Civil, Architectural and Environmental Engineering
The University of Texas at Austin
301 E. Dean Keaton St. Stop C1761
Austin, TX 78712-1172
Ph: 779-804-4407, FAX: 512-475-8744
Email: cesaryahia@utexas.edu
(*Corresponding author.*)

Venktesh Pandey
Graduate Research Assistant
Department of Civil, Architectural and Environmental Engineering
The University of Texas at Austin
301 E. Dean Keaton St. Stop C1761
Austin, TX 78712-1172
Ph: 737-222-8473, FAX: 512-475-8744
Email: venktesh@utexas.edu

Stephen D. Boyles
Associate Professor
Department of Civil, Architectural and Environmental Engineering
The University of Texas at Austin
301 E. Dean Keaton St. Stop C1761
Austin, TX 78712-1172
Ph: 512-471-3548, FAX: 512-475-8744
Email: sboyles@mail.utexas.edu

**Word count:**
4448 words text+
188 words abstract+
580 words references+
5 tables/figures $\times$250 words (each) = 6466 words

**Submission date: August 1, 2017**

## ABSTRACT

Recent methods in the literature to parallelize the traffic assignment problem consider partitioning a network into subnetworks to reduce the computation time for large-scale networks. In this article, we seek a partitioning method that generates subnetworks minimizing the computation time of a decomposition approach for solving the traffic assignment (DSTAP). We aim to minimize the number of boundary nodes, the inter-flow between subnetworks, and the computation time when the traffic assignment problem is solved in parallel on the subnetworks. We test two different methods for partitioning. The first is an agglomerative clustering algorithm heuristic that decomposes a network with the objectives of minimizing subnetwork boundary nodes. The second is a flow weighted spectral clustering algorithm that uses the normalized graph Laplacian to partition the network.

We assess the performance of both algorithms on different test networks. The results indicate that the agglomerative heuristic generates subnetworks with a low number of boundary nodes, which reduces the per iteration computation time of DSTAP. However, the partitions generated may be heavily imbalanced leading to significantly higher computation time when the subnetworks are solved in parallel separately at a certain DSTAP iteration. For the Austin network partitioned into 4 subnetworks, the agglomerative heuristic requires 3.5 times more computational time to solve the subnetworks in parallel. We also show that the spectral partitioning method is superior in terms of minimizing the inter-flow between subnetworks. This leads to a faster convergence rate of the DSTAP algorithm.

***Keywords—*** Network partitioning; Spectral partitioning; DSTAP; Regional modeling; Decentralized traffic assignment; Network contraction

## 1 INTRODUCTION

The traffic assignment problem (TAP) 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 *(1)* and solved efficiently using modern specialized algorithms *(2, 3)*. However, there are computationally demanding problems that require solving TAP multiple times or on a large scale. Those problems include bi-level mathematical programs with equilibrium constraints *(4)*, statewide or national modeling of the transportation network, and Monte Carlo simulations to account for forecasting errors of TAP parameters *(5)*.

Methods for parallelizing the traffic assignment problem to decrease computation time have been studied recently e.g. *(5, 6, 7)*. The decomposition approach to the static traffic assignment problem (DSTAP) was developed to decrease computation time by solving partitions of the transportation network in parallel *(5)*. 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 *(8)* to generate artificial links that represent all paths between certain network nodes. DSTAP is shown to converge to equilibrium for a general network and its computation time is stated to depend on the subnetwork partitions *(5)*.

The objective of this study 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 separately and 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 the one proposed in Johnson et al. *(12)* for the objectives similar to those required in this paper. The second algorithm is based on flow weighted spectral clustering. We compare the performance of the algorithms on real-world networks against the stated objectives.

The remainder of this paper describes methods to parallelize TAP and evaluates the performance of partitioning algorithms when applied to decomposition methods for solving TAP. Section 2 reviews current methods for solving TAP and partitioning networks. Section 3 presents the algorithms evaluated and their use in the DSTAP framework. Section 4 presents demonstrations for different transportation networks. Section 5 concludes the paper.

## 2 LITERATURE REVIEW

This section summarizes existing literature in the following areas: the latest advancements in the traffic assignment problem methods, 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 *(13, 14)*. Bush-based method exploit the acyclic nature of paths that are used by origins at the equilibrium *(2, 3, 15, 16)*. Recent work also includes $\epsilon$-optimal improved methods for solving TAP on large problems *(17)*. Although the recent advancements have improved the state-of-the-art for solving TAP fast and efficiently, there is still a need for faster methods. These are several instances which require TAP to be solved on large scale networks or solved repeatedly. These include running large-scale statewide models for evaluating multiple projects, solving TAP multiple times for network design problems with equilibrium constraints *(4, 18)*, to name a few.

To address the computational demands of large scale or iterative traffic assignment problems, methods that aim to parallelize TAP have been developed. Bar-Gera *(6)* describes a parallelization approach based on the paired-alternative segments. The algorithms proposed in Chen and Meyer *(19)* and Lotito *(20)* also parallelize TAP by decentralizing the computations for each OD pair. A decomposition approach for static traffic assignment (DSTAP) developed by Jafari et al. parallelizes TAP by network geography instead of the traditional decomposition approach by OD pairs *(5)*. This article 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. *(5)*.

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 *(9)* and approximation heuristics have been proposed *(9, 21)*.

Heuristics for generating partitions based on agglomerative and divisive clustering have been recently used in various transportation related applications. Saedmanesh and Geroliminis *(11)* used an agglomerative clustering heuristic for generating partitions based on "snake" similarities for applications of the macroscopic fundamental diagram. Etemadnia et al. *(9)* developed similar heuristics for distributed traffic management where a greedy agglomerative strategy was adopted for combining adjacent nodes with high flow. Johnson et al. *(12)* 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 *(22)*.

Spectral partitioning is an alternative approach for clustering and partitioning graph e.g. *(23, 24, 25, 26, 27)*. Bell *(10)* applied a capacity-weighted form of the spectral partitioning methods to investigate vulnerability. Other transportation applications include air traffic control and urban traffic signal control systems *(28, 29)*. The partitioning mechanism is based on the eigenvalues associated with the graph Laplacian. The partitions that result from the spectral partitioning have low inter-cluster similarity *(24)*. Additionally, using the normalized Laplacian generates graphs that are balanced by the weight chosen. 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 *(26)*. However, incorporating balance requirements causes the cut problems to become NP-hard. Spectral partitioning is an approximate method for obtaining a cut with minimal inter-flow while satisfying balance requirements *(23, 26,*

*27).*

## 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 article, we assume $w_{(i,j)}$ to be the flow on link $(i,j)$ at the equilibrium. 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 $L_G = D_G - M_G$.

### 3.1 Decomposition Approach to the Static Traffic Assignment Problem (DSTAP)

We aim to partition a large scale network into subnetworks such that an algorithm based on the decomposition approach to the static traffic assignment problem (DSTAP) is solved efficiently. In order to properly define the objectives of the partitioning algorithms evaluated, we review the main elements of the DSTAP algorithm developed by Jafari et al. *(5)*.

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 *(8, 30)*. 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.

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. Therefore, 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 balanced in size. This can be achieved by balancing the flow distribution across subnetworks instead of having few subnetworks with the majority of flow.

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. *(5)* 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 $\epsilon^r_{OD}$ and the maximum excess cost for all subnetworks $\epsilon^s_{OD}$ 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 $\epsilon^s_{OD}$ 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 $\epsilon^r_{OD}$ 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 $\epsilon^r_{OD}$ 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 link. Thus, faster convergence could be reached by minimizing the inter-flow between subnetworks.

$$\epsilon_{OD} \leq 2\tilde{B}(\epsilon^r_{OD} + \epsilon^s_{OD}) \tag{1}$$

## 3.2 Partitioning Algorithms

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

### *3.2.1 Domain decomposition algorithm*

The first heuristic algorithm used is the shortest domain decomposition algorithm (SDDA) proposed in Johnson et al. *(12)*. This algorithm works in an agglomerative fashion and constructs a given number of partitions of a network such that the number of boundary nodes between the subnetworks is minimized (primary objective) and the partitions are balanced in size (secondary objective). Achieving the algorithm objectives would minimize the computation time per iteration associated with DSTAP. Minimizing the boundary nodes reduces the number of artificial links created, and generating balanced subnetworks would reduce the time needed to solve the subproblems in parallel. In addition, this algorithm 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 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 mea-

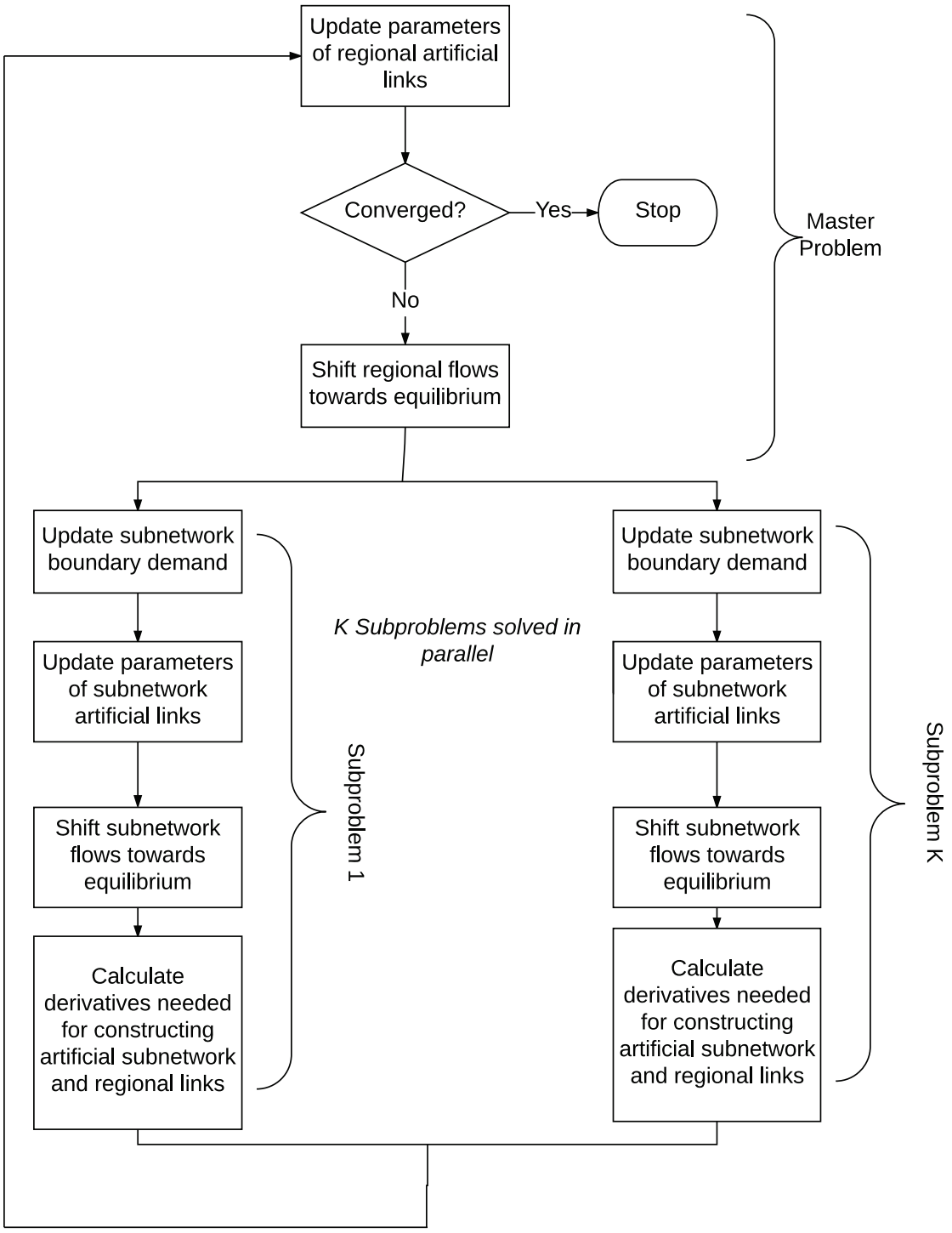


**FIGURE 1** Algorithm for the decomposition approach to the static traffic assignment problem *(5)*

sure. We use the number of links on a breadth-first search tree between two nodes as the distance measure between the nodes instead of the P-LCA algorithm with unit costs proposed by Johnson et al. *(12)*. 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. *(12)* for more information on this algorithm.

**Algorithm 1** Shortest domain decomposition algorithm *(12)*

**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 $2 : n_s$ **do**
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

### *3.2.2 Spectral partitioning*

Spectral graph theory is used to study graph properties using the Laplacian matrix of the graph. This theory can be used to identify cuts through the graph which have low cost by computing the eigenvalues and eigenvectors of the Laplacian matrix. 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 *(24, 26)*.

The eigenvalues of an undirected graph Laplacian are real since the matrix is symmetric *(23)*. 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 *(23)*, since the matrix is symmetric the eigenvalues can be defined using equation (3), where $S$ is a vector space of dimensions $i$, where $i$ is the index of $\lambda_i$ eigenvalue arranged in an ascending order.

The eigenvector for the corresponding eigenvalue can be found using equation (4).

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

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

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

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 $cut(A, \bar{A})$ is the sum of the weights on the links separating the subnetworks $A$ and $\bar{A}$ 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 *(26)*.

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

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 *(23, 25, 26, 27)*. 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 *(10, 27)*. 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

## 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 *(31)*. 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.

### 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 reduced 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, the partition 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 this feature 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 partition 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.

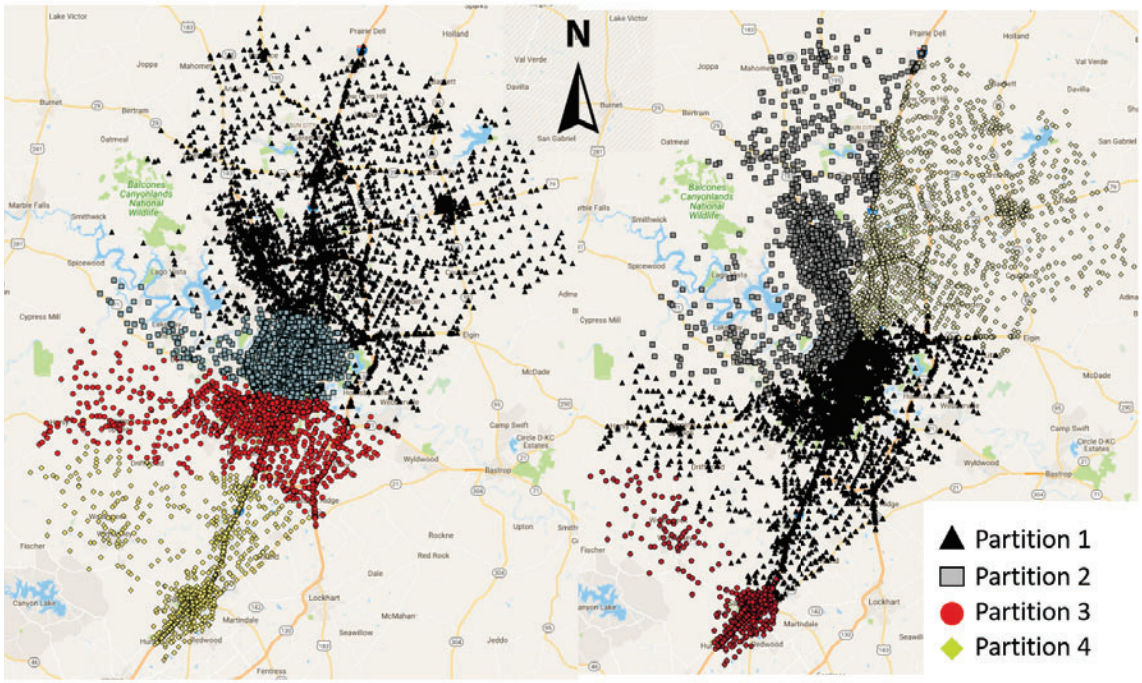


**FIGURE 2** Partitioning of Austin regional network into four partitions. Left: flow weighted spectral partitioning. Right: SDDA

## 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 usefulness of flow weighted spectral partitioning for networks which have intuitive geographic concentrations like 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. In terms of subnetwork boundary nodes, both partitions are equivalent.

In the DSTAP section, we showed that faster convergence could be achieved if the interflow 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 mini-

mizing 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 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.

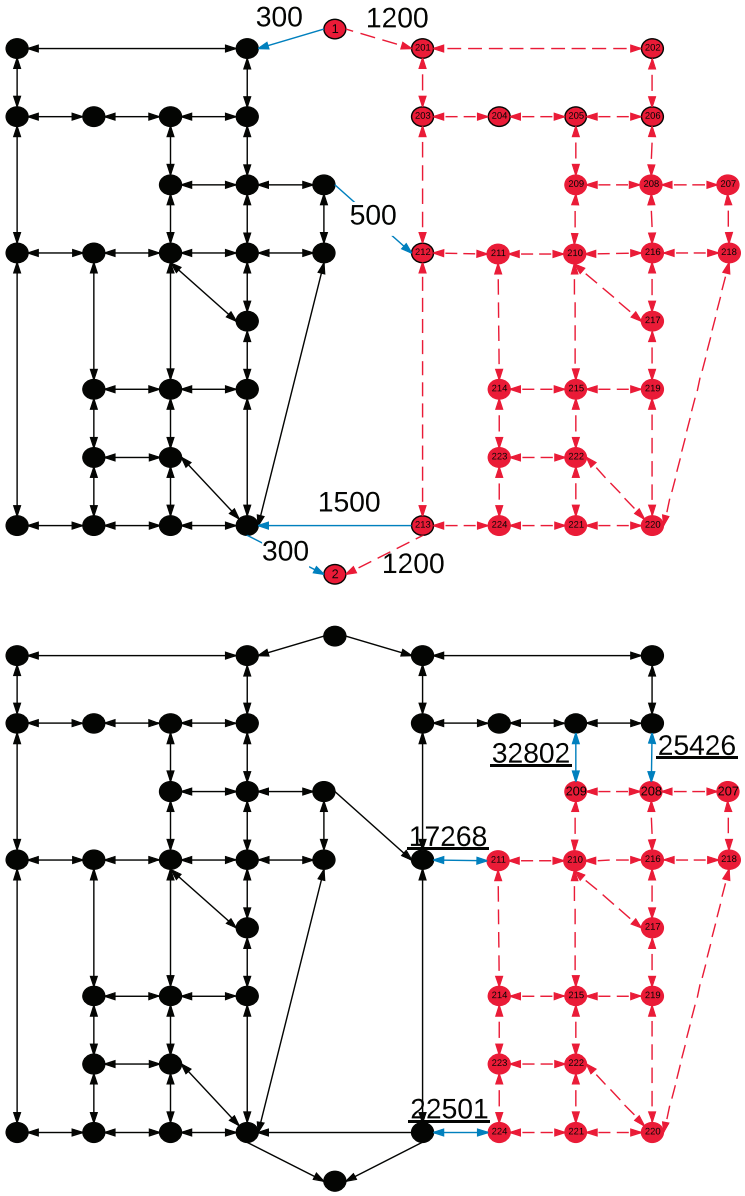


**FIGURE 3** Partitioning of double Sioux Falls hypothetical network: Top: flow weighted spectral partitioning. Bottom: SDDA. The line type defines different partitions.

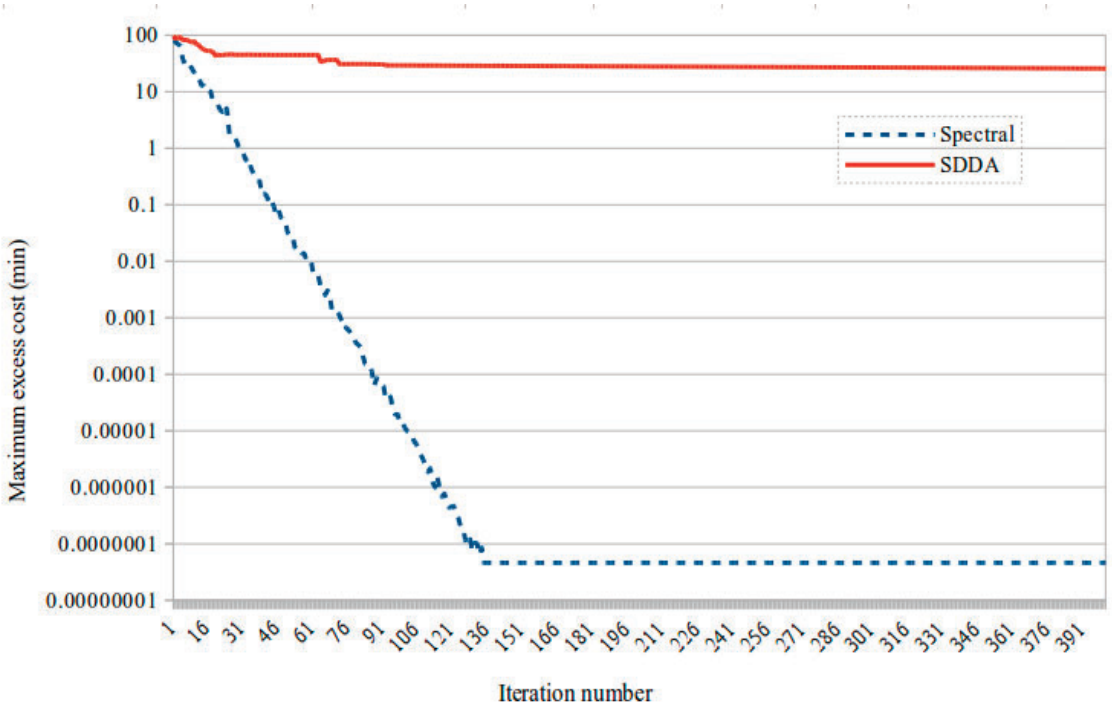


**FIGURE 4** Iterative change of the maximum excess cost of the DSTAP algorithm when used with flow weighted spectral partitions and SDDA partitions of the hypothetical double Sioux Falls network.

## 5 CONCLUSIONS

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

We test two different methods for partitioning. The first approach is an agglomerative clustering algorithm developed in Johnson et al. *(12)* 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 clustering. 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 the Chicago sketch networks. This leads to higher computation time for solving the DSTAP subproblems in parallel. The flow weighted spectral partitioning method leads to balanced subnetworks which reduces 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.

Future work will further assess the trade-offs between minimizing the per iteration computation time and maximizing the convergence rate. Partitioning methods that aim to simultaneously minimize boundary nodes and inter-flow will be explored. In addition, we will seek alternative approximations that reduce the number of artificial links generated by DSTAP.

## ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1254921. Partial support was also provided by the Data-Supported Transportation Planning and Operations University Transportation Center. The authors are grateful for this support.

## REFERENCES

[1] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[2] H. Bar-Gera. Origin-based algorithm for the traffic assignment problem. *Transportation Science*, 36(4), 2002.

[3] 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.

[4] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1), 2007.

[5] E. Jafari, V. Pandey, and S. D. Boyles. A decomposition approach to the static traffic assignment problem. *Transportation Research Part B: Methodological*, 105, 2017.

[6] H. Bar-Gera. Traffic assignment by paired alternative segments. *Transportation Research Part B: Methodological*, 44(8-9), 2010.

[7] 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.

[8] S. D. Boyles. Bush-based sensitivity analysis for approximating subnetwork diversion. *Transportation Research Part B: Methodological*, 46(1), 2012.

[9] H. Etemadnia, K. Abdelghany, and A. Hassan. A network partitioning methodology for distributed traffic management applications. *Transportmetrica A: Transport Science*, 10(6), 2014.

[10] 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.

[11] M. Saeedmanesh and N. Geroliminis. Clustering of heterogeneous networks with directional flows based on "snake" similarities. *Transportation Research Part B: Methodological*, 91, 2016.

[12] 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.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics*, 3, 1956.

[14] 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.

[15] Y. M. Nie. A class of bush-based algorithms for the traffic assignment problem. *Transportation Research Part B: Methodological*, 44(1), 2010.

[16] G. Gentile. Local User Cost Equilibrium: A bush-based algorithm for traffic assignment. *Transportmetrica A: Transport Science*, 10(1), 2014.

[17] H. Zheng and S. Peeta. Cost scaling based successive approximation algorithm for the traffic assignment problem. *Transportation Research Part B: Methodological*, 68, 2014.

[18] 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.

[19] R. Chen and R. R. Meyer. Parallel optimization for traffic assignment. *Mathematical Programming*, 42(1), 1988.

[20] 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.

[21] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2), 1996.

[22] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1), 1998.

[23] D. A. Spielman. Spectral graph theory and its applications. IEEE, 2007.

[24] D. A. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, 421(2-3), 2007.

[25] M. E. J. Newman. Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4), 2013.

[26] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8), 2000.

[28] 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.

[29] 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.

[30] E. Jafari and S. D. Boyles. Improved bush-based methods for network contraction. *Transportation Research Part B: Methodological*, 83, 2016.

[31] H. Bar-Gera. Transportation network test problems. `http://www.bgu.ac.il/bargera/tntp/`, 2017.