Sensor Placement for Learning on Networks

Arnav Burudgunte
ab141@rice.edu
Rice University
Houston, Texas, USA

Arlei Silva
arlei@rice.edu
Rice University
Houston, Texas, USA

ABSTRACT

Large infrastructure networks (e.g. for transportation and power distribution) require constant monitoring for failures, congestion, and other adversarial events. However, assigning a sensor to every link in the network is often infeasible due to placement and maintenance costs. Instead, sensors can be placed only on a few key links and machine learning algorithms can be leveraged for the inference of missing measurements (e.g. vehicle speeds, power flows) across the network. This paper investigates the sensor placement problem for networks. We first formalize two versions of the problem under flow conservation and smoothness assumptions and show that it is NP-hard to optimally place a fixed set of sensors. Next, we propose efficient and adaptive greedy heuristics for sensor placement that scale to large networks. Our experiments, using datasets from real-world application domains, show that the proposed approaches enable more accurate inference than existing alternatives from the literature. We demonstrate that considering even imperfect or incomplete ground-truth estimates can vastly improve the prediction error, especially for a small number of sensors.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; • **Theory of computation** → *Network flows.*

KEYWORDS

network flows, semi-supervised learning, graphs, labels

ACM Reference Format:
Arnav Burudgunte and Arlei Silva. 2023. Sensor Placement for Learning on Networks. In *Proceedings of (KDD Undergraduate Consortium '23).* ACM, New York, NY, USA, 8 pages. https://doi.org/X

1 INTRODUCTION

This paper addresses the problem of estimating edge measurements in networks. Given a value at each edge, our goal is to choose a set of edges to monitor in order to infer values at unmonitored edges.

Versions of our problem are motivated by infrastructure networks, such as traffic [15], water [19], and power [27] networks. We assume that each edge in the network has some associated value (e.g. traffic flow, water flow, electrical current) that must be measured to track congestion, anomalies, and other events. Owing to the size of the network and measurement cost, we often cannot place a sensor at every edge. A common solution is to place sensors at a small subset of the edges, and estimate the rest using semi-supervised learning [15, 34, 39]. Because the choice of this subset can seriously alter the prediction quality, our goal is to efficiently select a subset that yields the most accurate estimate.

We consider two versions of the sensor selection problem depending on the measurement of interest. *Flow conservation* concerns values such as vehicle counts in a road network or power flow in electrical networks [15], which follow the principle of flow conservation, meaning the sum of flows into a vertex is (approximately) equal to the sum of flows out of it. The *smoothness* version concerns smooth measurements such as traffic speeds and electrical voltage [2] in which values are not necessarily conserved, but where the measurement at a given edge are expected to be similar to the measurements at nearby edges.

The sensor placement problem is related to active learning, in which a machine learning model iteratively makes requests for specific observations to be labeled. Active learning algorithms generally assume that unlabeled observations are never known unless directly queried; models must choose the next labeled observation based only on the network topology [15] and previously-queried observations. Such algorithms begin with no information about the ground-truth labels and slowly learn more about the network as they label the data. In practice, however, we often have outside information about the edge labels before placing any sensors. For example, urban planners often have detailed predictions for traffic flow based on traffic demand models, spatial and temporal data, and other factors [1, 20, 23]. Such estimates can be extremely useful for inferring labels with a small number of sensors because they significantly constrain the possible solutions—there might be many possible predictions that satisfy the flow conservation or smoothness assumptions, but only a few will correctly predict the volume of traffic at major roads. As we will show, even an imperfect estimate of edge values can generate a more effective choice of sensors than a naive algorithm which makes predictions based solely on flow conservation or smoothness assumptions. The question we address in this paper is *how to use existing knowledge of edge values (or flows) to improve our predictions.*

Our work combines knowledge of the network topology and ground-truth values to choose an optimal placement of sensors. We summarize the main contributions of our work as follows:

- We provide a formal statement of the sensor placement problem under both flow conservation and smoothness, with a proof that both problems are NP-complete;
- We propose greedy heuristics as efficient solutions for both versions of our problem;
- We present experimental results showing our approach's improvement over existing baselines.

KDD Undergraduate Consortium '23, August 7, 2023, Long Beach, CA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/X

2 RELATED WORK

Graph-Based Semi-Supervised Learning. Semi-supervised learning (SSL), in which models learn from both labeled and unlabeled data, is motivated by applications where the availability of labeled data is limited [32, 37]. Graph-based SSL is a special case of SSL where observations can be encoded as vertices and edges represent relations between them [39]. Label propagation [37, 38] is the most popular technique for reconstructing smooth values. More recently, flow-based SSL [15] was proposed to infer conserved flows on the edges rather than smooth vertex labels. However, both the conservation- and smoothness-based solutions can be highly dependent on the choice of labeled edges and the nature of the data [15, 37], which is a motivation for our work. We focus on the problem of selecting edges to be labeled (or sensor locations, in the case of infrastructure networks), which is a special case of the active learning problem [28]. With the exception of [15], combining active learning with SSL has been studied from the smoothness perspective, largely as a graph coverage [10] or sampling (signal processing) [9] problem. In these scenarios, the choice of nodes or edges is made using only the network topology, and thus cannot be optimized using the ground-truth values [10, 15]. Here, we assume that edges are chosen by optimizing a reconstruction loss. We show that this approach allows us to match the performance of existing baselines using far fewer sensors.

Traffic Forecasting. The problem of forecasting traffic flows, speeds, and other attributes is can be motivated by smart city applications. Early attempts at traffic prediction infer network flows from a set of origin-destination pairs [1] or vice versa [13]; the popular traffic simulator SUMO [23] operates using a similar principle. More recently, deep learning has been leveraged to predict traffic as a function of spatial and temporal data [20, 26, 30]. The relevant result for our work is that there are sophisticated, accurate models available for predicting traffic flow in graphs even without labels from sensors. We show that even when such estimates are noisy or imperfect, they provide useful information about ground-truth flows. We focus on the spatial dimension of the problem and use simpler inference models such as label propagation, but our algorithms could be generalized to use any forecasting model.

Sensor Placement and Influence Maximization. Previous work has studied sensor placement for other objectives such as monitoring spatial phenomena [17] and detecting contaminants in water distribution networks [18, 19]. Similar to these problems, we study sensor placement as a combinatorial optimization problem, though we minimize a prediction error rather than a fixed penalty function. Closely related to the sensor placement problem is the problem of influence maximization, which asks for the best subset of nodes to target for influence such that after some diffusion process, the maximum number of nodes have been influenced [16]. Solutions often depend on the diffusion model, which predicts the number of nodes affected by the chosen seed set [21]. Instead of using a process, we apply a machine learning model to make predictions based on a validation set (i.e. set of edge observations). Moreover, we notice that, although we apply similar greedy algorithms as those previously used for sensor placement and influence maximization, our objective functions are not submodular [24].

3 PROBLEM DEFINITION

3.1 Preliminaries

We represent a network as an unweighted graph $G = (V, E)$ with a set V of n vertices and set E of m edges. The graph is represented by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{ij} = 1$ if an edge exists between vertices i and j and $\mathbf{A}_{ij} = 0$ otherwise.

We are interested in the value given by vector $\mathbf{x} \in \mathbb{R}^m$ where $\mathbf{x}_i$ is the value (either conserved or smooth) at edge i. We represent the values of a subset $E' \subset E$ as a vector $\mathbf{x}_{E'} \in \mathbb{R}^{|E'|}$. In the rest of this paper, we use $\mathbf{f}$ and $\mathbf{x}$ to denote conserved and smooth values, respectively. Thus a set of labeled (sensor) edges S is represented by the corresponding value $\mathbf{x}_S \in \mathbb{R}^{|S|}$, and a set of unlabeled (target) edges T is represented as $\mathbf{x}_T \in \mathbb{R}^{|T|}$.

Though previous work [37] has considered the smoothness assumption as a property of vertices, we view both versions as edge problems. We show in Section 4.1 how the smoothness assumption can be translated to edges.

3.2 General Sensor Placement Problem

Prediction of conserved and smooth values can be represented within the same two-part framework. The first part is predicts the values of unlabeled edges based on a given set of sensors and the assumption. The second is choosing a set of sensors that yields the prediction with the lowest error. We formalize both parts here.

3.2.1 Prediction. Given a labeled set of sensors S ($|S| = k$) and corresponding observed smooth values $\mathbf{x}_S \in \mathbb{R}^k$, we produce an estimate $\hat{\mathbf{x}}$ for $\mathbf{x}$ via graph-based semi-supervised learning. The same holds for conserved values $\mathbf{f}_S \in \mathbb{R}^k$.

3.2.2 Sensor Placement. Given a set T of target vertices, a set C of candidate vertices, and budget k, our problem is to choose the subset of k vertices in C that yields the best prediction for T:

$$\begin{aligned} S^* = \underset{S \subseteq C, |S|=k}{\arg\min} \quad & ||\hat{\mathbf{x}}_T - \mathbf{x}_T||_2^2 \\ \text{s.t.} \quad & \hat{\mathbf{x}}_T = \phi(\mathbf{x}_S, S) \end{aligned} \tag{1}$$

where ϕ is the prediction model for smooth values (see Section 4.1). The same problem can be defined for conserved flows $\mathbf{f}$.

3.3 Hardness

The conserved and smooth versions of our problem are both NP-complete. We give the proof for conservation here.

Definition 3.1 (The Sensor Placement Problem). Given a graph G, a candidate set of edges $C \subseteq E$, a target set of edges $T \subseteq E$, a budget k, and an error ϵ, *SENSOR*(G, C, T, k) consists of determining whether there exists a set of edge labels $S \subseteq C$ such that $|S| = k$ and the edge predictions $\hat{\mathbf{f}}$ for S (see Equation 1) have an error $||\hat{\mathbf{f}}_T - \mathbf{f}_T|| \leq \epsilon$.

Theorem 3.2. *The Sensor Placement Problem (SENSOR) is NP-complete.*

Proof. Given a certificate $S \subseteq C$ of edges selected as sensors, we can clearly compute $\hat{\mathbf{f}}$ and check the error in polynomial time. Let *SUM*(X, t) be an instance of the subset sum problem for a finite set $X \subseteq \mathbb{Z}_+$ and $t \in \mathbb{Z}_+$. The problem is to find a subset $X' \subset X$ such that $\sum_{x \in X'} x = t$. This problem is NP-hard [5].

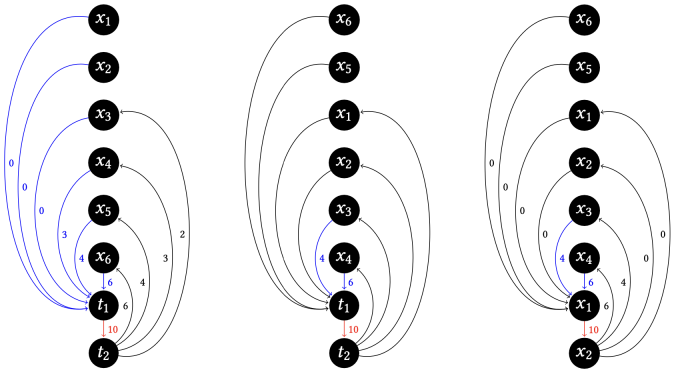


Figure 1: Reduction from *SUM* to *SENSOR*, with $X = \{3, 4, 6\}$ and $t = 10$. The candidate set $C = X$ is in blue, and the target $T = \{(t_1, t_2)\}$ in red. Choosing the two edges that sum to t for S generates a perfect prediction for T.

We can reduce an arbitrary instance of *SUM* to *SENSOR* by constructing graph G as follows. Create a vertex for each element in X and two additional vertices t_1 and t_2. Add a set of edges $C = \{(x, t_1) : x \in X\}$, each with a flow equal to s; an edge (t_1, t_2) with flow t; and set of edges $\{(t_2, x) : x \in X\}$ each with flow x. Finally, add $|X|$ vertices with 0 flow into t_1.

There exists a solution to $SENSOR(G, C, \{(t_1, t_2)\}, k)$ with prediction error 0 if and only if a solution exists for $SUM(X, t)$. The equivalence is straightforward: if some set X' sums to t, then choose edges $\{(x', t_1) : x' \in X'\}$, whose flows also sum to t, as sensors. If $|X'| < k$, choose edges with 0 flow for the remaining sensors. Since there is only edge out of t_1, that edge must have flow t. Conversely, if k sensors correctly predict the flow on edge (t_1, t_2), their edge flows must sum to t, and thus provide a solution to *SUM*. □

A similar construction can be used to prove that the smooth version of the problem is also NP-complete. This motivates us to investigate efficient heuristics for our general problem.

4 METHODS

4.1 Graph-Based Semi-Supervised Learning

We first describe the prediction algorithms used for both the conservation and smoothness versions of the problem.

4.1.1 Graph-Based SSL with Conservation [15]. For flow conservation, we consider the divergence on each vertex i, defined as the difference between flows out of i and flows into i:

$$\text{div}(i) = \sum_{e \in E: e \text{ out of } i} \mathbf{f}_e - \sum_{e \in E: e \text{ into } i} \mathbf{f}_e \tag{2}$$

Flows for missing edges are estimated by minimizing the sum-of-squares divergence given by

$$||\mathbf{Bf}||^2 = \sum_{i \in V} (\text{div}(i))^2 \tag{3}$$

where the incidence matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ is defined as

$$\mathbf{B}_{ij} = \begin{cases} 1 & \text{if edge } e_j \text{ enters node } i \\ -1 & \text{if edge } e_j \text{ leaves node } i \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

For undirected graphs, we first choose an arbitrary orientation for each edge before constructing $\mathbf{B}$. We then minimize the sum of the divergence and a regularization term parameterized by $\lambda \in \mathbb{R}_+$:

$$\begin{aligned} \hat{\mathbf{f}}^* = \arg\min_{\hat{\mathbf{f}} \in \mathbb{R}^m} & ||\mathbf{B}\hat{\mathbf{f}}||^2 + \lambda^2 \cdot ||\hat{\mathbf{f}}||^2 \\ \text{s.t. } & \hat{\mathbf{f}}_S = \mathbf{f}_S \end{aligned} \tag{5}$$

where λ guarantees that the solution is unique. For directed graphs, we additionally impose the constraint that $\hat{\mathbf{f}}_i \geq 0, \forall i$.

The resulting optimization problem can be rewritten as a regularized least squares problem for computation. Define $\mathbf{f}^0 \in \mathbb{R}^m$ such that $\mathbf{f}^0_i = \mathbf{f}_i$ if $i \in S$ and $\mathbf{f}^0_i = 0$ otherwise. Moreover, let $\mathbf{H} \in \mathbb{R}^{m \times (m-k)}$ be a matrix (map) such that $\mathbf{H}_{ij} = 1$ if flow $\mathbf{f}_i$ maps to $(\mathbf{f}_T)_j$ (i.e., they correspond to the same edge). The regularized least-squares formulation is

$$\mathbf{f}^*_T = \arg\min_{\mathbf{f}_T \in \mathbb{R}^{m-k}} ||\mathbf{BHf}_T - \mathbf{Bf}^0||^2 + \lambda ||\mathbf{f}_T||^2 \tag{6}$$

where the vector $\mathbf{f}_T$ is constrained to non-negative entries in the case of directed graphs [4].

4.1.2 Graph-Based SSL with Smoothness [37]. For edge smoothness, we turn the original graph G into a line graph $L(G) = (V', E')$, where edges in G become vertices in V' and adjacent edges in G are connected by edges in E'. Smoothness is based on edge gradients:

$$grad(i, j) = \mathbf{x}_i - \mathbf{x}_j \tag{7}$$

where $(i, j) \in E'$.

Missing values are estimated by minimizing the squared differences between values for each vertex and its neighbors in $L(G)$:

$$\mathbf{x}^T \mathbf{Lx} = \sum_{(i,j) \in E'} (grad(i, j))^2 \tag{8}$$

where $\mathbf{L} \in \mathbb{R}^{m \times m}$ is the Laplacian of $L(G)$:

$$\mathbf{L}_{ij} = \begin{cases} \deg(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and nodes } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

As in the conservation case, the above optimization problem can be formulated via least-squares. Let $\mathbf{D}$ be the degree matrix, which is a diagonal matrix with $\mathbf{D}_{ii} = deg(i)$. Then $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ is a transition matrix and $\mathcal{L} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{P}$ is a normalized Laplacian matrix. Moreover, let $\mathbf{H}$ and $\mathbf{x}^0$ be defined in the same way as $\mathbf{H}$ itself and $\mathbf{f}^0$ were defined in the previous section. The resulting least-squares formulation is

$$\mathbf{x}^*_T = \arg\min_{\mathbf{x}_T \in \mathbb{R}^{m-k}} ||\mathbf{H}^\mathbf{T} \mathcal{L} \mathbf{H} \mathbf{x}_T - \mathbf{H}^\mathbf{T} \mathbf{P} \mathbf{x}^0||^2 \tag{10}$$

The similarity between the SSL predictors under conservation and smoothness has several interesting implications. For instance, it suggests an iterative SSL algorithm for conserved flows that is similar to label propagation [37]. Moreover, it supports the design

of efficient iterative sensor placement algorithms based on repeated solutions of least-squares problems as new sensors are added. This is the topic of the next section.

4.2 Sensor Placement Algorithms

We propose greedy algorithms for the sensor placement problem under the assumptions of flow conservation and smoothness.

4.2.1 Algorithm for Conserved Flows. Given our hardness result (see Section 3.3), we propose a greedy algorithm that iteratively selects the sensor that minimizes the prediction error in Equation 1. The pseudocode is given in Algorithm 1. At each of the k iterations (lines 2-11), the algorithm selects the best new sensor s^* based on its resulting prediction error ϵ (lines 6-10).

The flow prediction problem is not submodular [24]—a single badly placed sensor can sometimes generate a worse prediction than no sensor (e.g. if many edges have near-zero flow). This limits our ability to prove worst-case approximation results for our algorithm. In practice, however, it is almost always possible to add a sensor that decreases the prediction error and thus the problem exhibits (empirical) diminishing return (see Figure 2). We believe that this property enables our algorithm to work well in practice.

In terms of complexity, at each of the k iterations, our algorithm evaluates the benefit of every not yet chosen edge in C. This requires $O(m)$ solutions of the least-squares problem from Equation 6. Each solution can be computed iteratively using LSMR [8] in $O(m^2)$ time, giving an overall running time of $O(m^4)$.

This exhaustive search at each iteration of our algorithm is costly for large values of m. We can reduce the number of required estimates by taking advantage of the problem's "near-submodularity"—as more sensors are placed, the benefit from not chosen edges rarely increases. This allows us to evaluate these benefits lazily [19]. Before choosing any sensors, we compute the benefit of each edge and store the results in a heap. At each iteration, we recompute the benefit of the best remaining edge. If it remains the best, then it is highly unlikely any other edge can overtake it, and we can choose that edge without reevaluating any other benefits. This saves up to $|C|$ evaluations per iteration.

4.2.2 Algorithm for Smoothness. The simple greedy approach described in the previous section does not work as well for smoothness. To see why, recall that label propagation is equivalent to iteratively estimating each vertex in $L(G)$ as the average of its neighbors [37]. Thus a good sensor placement resembles a good clustering: it should choose representative vertices from k different neighborhoods of $L(G)$. As is often the case in clustering problems, the optimal solution is quite different for different values of k. The same property holds for our problem.

Our approach selects each sensor conditionally on the remaining sensors. We use K-means clustering [12] as inspiration for the smooth sensor placement approach with centroids behaving as sensors. Our solution is described by Algorithm 2. The algorithm starts with a random initial cluster assignment S (line 1) and, at each iteration, tries to replace a sensor s by one of its neighbors $u \in N(s)$ (lines 5-12). The replacement is successful whenever the reconstruction error decreases (lines 10-12). When no further improvement is possible, the algorithm terminates.

Algorithm 1: Sensor Placement for Conserved Flows

Input: $G = (V, E)$; $C, T \subseteq E$; $k \leq |C|$, $\mathbf{f} \in \mathbb{R}^m$, $\lambda \in \mathbb{R}_+$
Output: $S \subseteq C$

1 $S \leftarrow \emptyset$;
2 **for** $i = 1, ..., k$ **do**
3 $\epsilon_{min} \leftarrow \infty$;
4 **foreach** $s \in C - S$ **do**
5 $S' = S \cup \{s\}$;
6 $\mathbf{f}_T = \phi(\mathbf{f}_{S'}, S', \lambda)$;
7 $\epsilon = ||\hat{\mathbf{f}}_T - \mathbf{f}_T||_2^2$;
8 **if** $\epsilon < \epsilon_{min}$ **then**
9 $\epsilon_{min} \leftarrow \epsilon$;
10 $s^* = s$;
11 $S = S \cup \{s^*\}$;
12 **return** S;

Algorithm 2: Sensor Placement for Smooth Labels

Input: $G = (V, E)$; $C, T \subseteq E$; $k \leq |C|$, $\mathbf{x} \in \mathbb{R}^m$
Output: $S \subseteq C$

1 $S \leftarrow k$ randomly chosen edges in C ;
2 $\mathbf{x}_T = \phi(\mathbf{x}_S, S)$;
3 $\epsilon_{min} = ||\hat{\mathbf{x}}_T - \mathbf{x}_T||_2^2$;
4 **while** *not converged* **do**
5 **foreach** $s \in S$ **do**
6 **foreach** $u \in N(s)$ **do**
7 $S' = S/s \cup \{u\}$;
8 $\mathbf{x}_T = \phi(\mathbf{x}_{S'}, S')$;
9 $\epsilon = ||\hat{\mathbf{x}}_T - \mathbf{x}_T||_2^2$;
10 **if** $\epsilon' < \epsilon_{min}$ **then**
11 $S = S'$;
12 $\epsilon_{min} = \epsilon$;
13 **return** S;

The hill-climbing approach eventually converges because moving a sensor always decreases the prediction error, but it may only find a local minimum. The performance can be improved by considering a larger set of candidates for each swap (line 6), such as the p-hop neighbors of s for $p > 1$. In principle, this neighborhood could be expanded to span the entire graph, but the total cost of evaluating swaps for a large neighborhood is often prohibitive.

Each iteration of our algorithm makes $O(k|C|)$ least-squares predictions. As in the flow conservation case, the least squares problem in Equation 10 can be solved in $O(m^2)$ time, giving a total running time of $O(m^4)$ per iteration. We find experimentally that that the algorithm converges in a small number of iterations and that the degree $|N(s)|$ of each edge s is usually much lower than m. However, as the solutions for different values of k are not nested, computing placements for different budgets is still not as efficient as the flow conservation solution presented in the previous section.

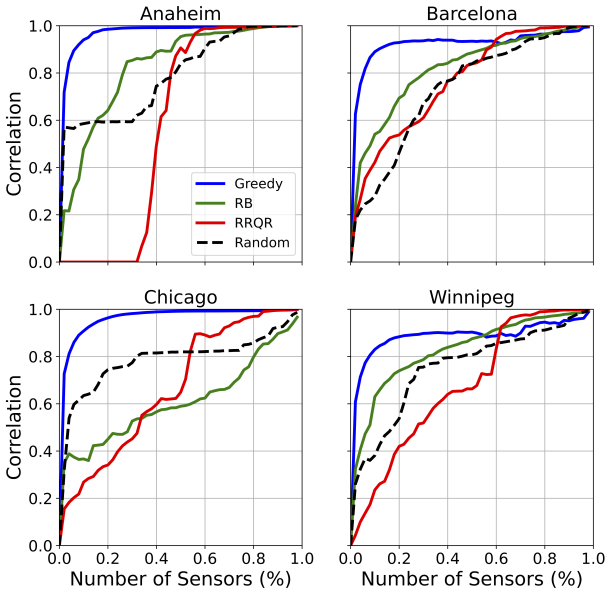


Figure 2: Conserved flow prediction results when flows are fully-observed for validation purposes (optimal scenarios). The plots show the correlation between the prediction $\hat{\mathbf{f}}$ and ground truth flow $\mathbf{f}$. Our greedy heuristic (Greedy) outperforms all three baselines in all datasets.

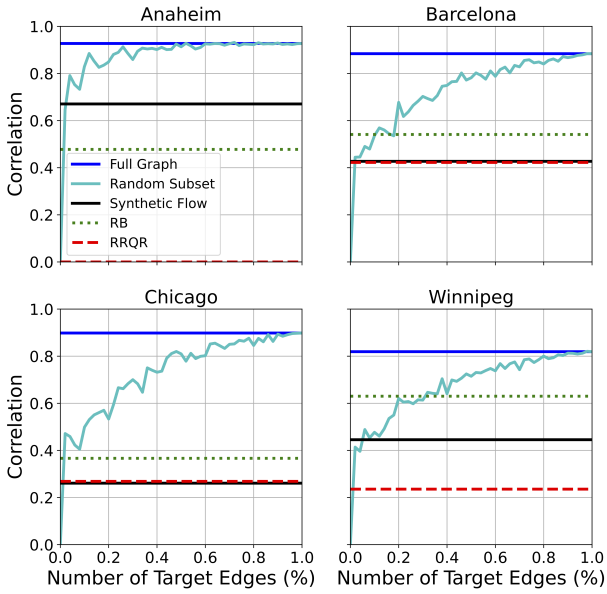


Figure 3: Conserved flow prediction results for our method using fully-observed (Full Graph), synthetic (Synthetic Flow), and partially observed (Random Subset) flows compared to the baselines. The number of sensors (k) is fixed at 10% of the edges. Synthetic flows are not able to effectively guide the sensor placement but a target set with just 20% of edges is enough for our method to outperform the baselines.

5 EXPERIMENTAL RESULTS

We test our hypothesis that optimizing sensor placement based on flow values is more effective than choosing sensors based only on the network topology using real-world traffic networks and flows. Experimental settings and metrics are discussed in the appendix.

5.1 Flow Conservation

We first evaluate our approach that assumes flow conservation. We consider two settings. In the first setting, we assume that our method is able to fully observe flow values to place the sensors, which is the ideal scenario. Next, we consider a more realistic setting where flows are unknown—requiring our method to use proxy values—or only partially known.

5.1.1 Fully-observed flows. In this setting, $T = C = E$ in Algorithm 1. We compare our approach to three topology-based baselines: random selection (Random), recursive bisection (RB) [15], and rank-revealing QR factorization (RRQR) [3, 15]. Details for all baselines are provided in the appendix.

Figure 2 shows the correlation between predicted and ground-truth flows for a varying number of sensors (as a percentage). The results show that our approach significantly outperforms the baselines, especially when the number of sensors is small (20% or less). The results also show that there is no clear best baseline.

5.1.2 Unknown, Partially Unknown, and Noisy Flows. A possible objection to the previous setting is that, in real-world applications, the complete flows are rarely fully-observed. In this section, we apply our algorithm without such a strong assumption.

Synthetic Flows. We generate synthetic flows under the conservation assumption (see Appendix A.1) [15]. The greedy heuristic is computed using the prediction error on the synthetic flows, and the resulting sensors are tested on the true flows for the four traffic networks. The sensor placements based on synthetic flows do not always outperform the baselines (see Figure 3). This is evidence that the synthetic flows are not an effective proxy for the real flows.

Partial Data. Here, we assume that we are able to observe a fraction of edges selected uniformly at random in the network (e.g. by placing additional temporary sensors). We then apply our method to place sensors by optimizing the prediction error over these partially observed edges, in effect restricting the target set T. Results from Figure 3 show that, in most cases, the greedy algorithm outperforms the topology-based baselines using a small percentage of labeled edges. This result motivates data-driven schemes for sensor placement, such as the ones proposed in this paper.

Noisy Estimates. Now we consider the setting where our model has access to noisy estimates of the ground-truth flows (e.g. based on the macro traffic demand model for a city [33, 36]). For each edge e_i, we simulate a noisy estimate $\mathbf{f}_i + \epsilon$ with randomly generated noise $\epsilon \sim N(0, r\sigma)$ where σ is the standard deviation of $\mathbf{f}$ and $r \in \mathbb{R}$ controls the amount of noise. Figure 4 shows the correlation between the predicted and original ground-truth flows for varying noise levels. Even with large amounts of noise, our approach (Noisy Flows) outperforms all baselines.

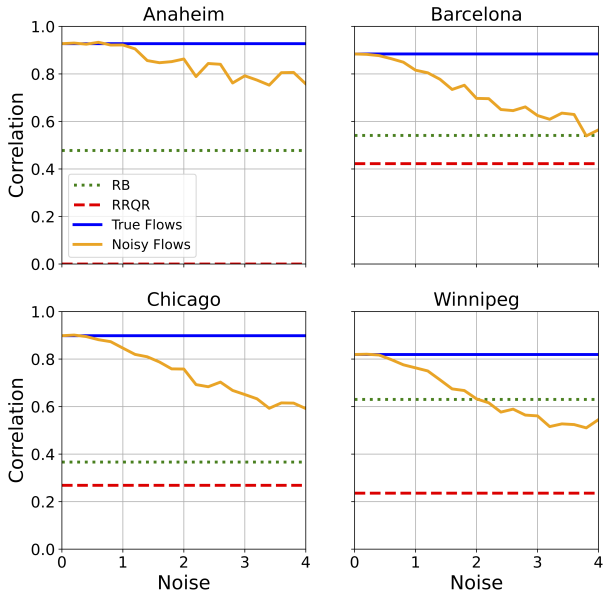


Figure 4: Conserved flow prediction results for sensor selection based on a noisy estimate of ground-truth flows for varying noise levels. The number of sensors placed (k) is fixed at 10% of the edges. The sensor placement is quite robust to noise target values, outperforming the baselines under noise levels of up to 2× the standard deviation of the flows.

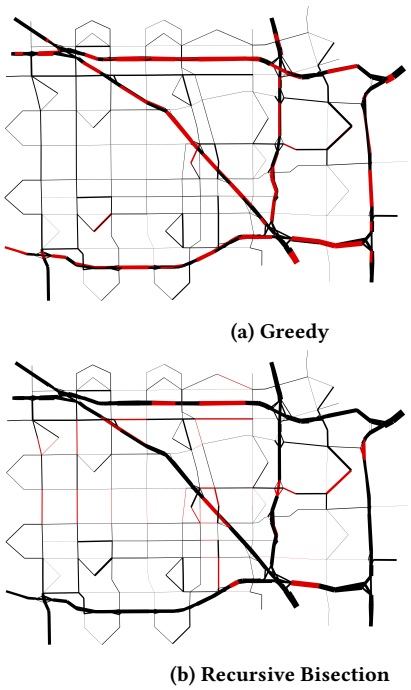
5.2 Flow Smoothness

We test the sensor selection algorithm under the smoothness assumption using synthetic smooth flow values. Here, we apply the second-smallest eigenvector of the Laplacian matrix $\mathbf{L}$, which is known to be smooth but not constant [39]. We then run Algorithm 2 to identify the best placement of sensors. When available, we apply the flow counts at each edge to weight the prediction error. As discussed in Section 4.2.2, the algorithm for smooth flows is much slower than the one for conserved ones, so our results are based on smaller networks. The results from Figure 6 show that our method outperforms the baselines for all datasets.

6 CONCLUSION AND ONGOING WORK

We have proposed an approach for sensor placement for semi-supervised learning in networks that accounts for ground-truth values. Our framework accounts for both conserved and smooth flow values. We show that choosing the optimal solution is NP-hard and provide effective heuristics for the problem. Our experiments show that our methods significantly outperform baselines that account only for the graph topology.

This paper is based on an ongoing project. We will incorporate more real-world datasets (for traffic flows and power transmission). Moreover, we will investigate how to speed up the placement evaluation using a recursive least-squares algorithm [7, 14]. Finally, we will investigate how solve the conservation and smoothness versions of our problem jointly to capture complex patterns [22].



(a) Greedy

(b) Recursive Bisection

Figure 5: Example of sensor placements (in red) using our heuristic (Greedy) and the recursive bisection baseline for the Anaheim road network. Edge traffic counts are represented by edge thickness. Unlike the baseline, our approach targets a few high-traffic paths.

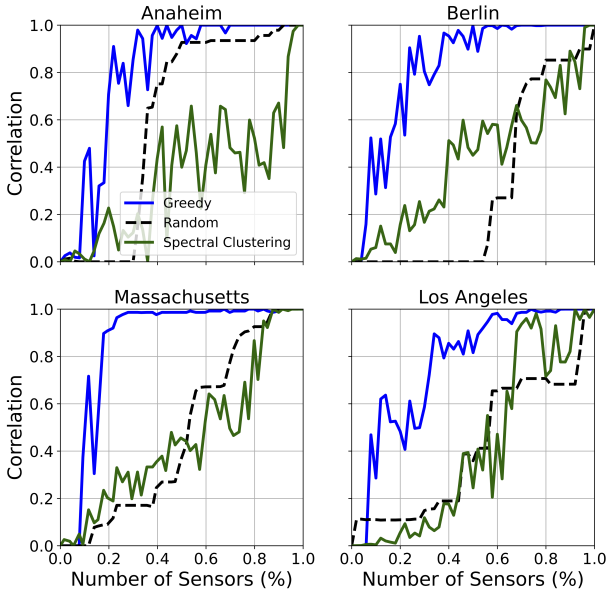


Figure 6: Smooth flow prediction results for sensor selection on synthetically generated smooth values. For Anaheim, the correlation is weighted by edge traffic counts; for other networks, the correlations are unweighted. Our approach (Greedy) consistently outperforms the baselines.

REFERENCES

[1] Hillel Bar-Gera. 2002. Origin-Based Algorithm for the Traffic Assignment Problem. *Transportation Science* 36, 4 (2002), 398–417. http://www.jstor.org/stable/25769124

[2] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label Propagation and Quadratic Criterion. In *Semi-Supervised Learning*. MIT Press. https://doi.org/10.7551/mitpress/9780262033589.003.0015 arXiv:https://academic.oup.com/mit-press-scholarship-online/book/0/chapter/353093119/chapter-ag-pdf/44419226/book_41571_section_353093119.ag.pdf

[3] Tony F. Chan. 1987. Rank revealing QR factorizations. *Linear Algebra Appl.* 88-89 (1987), 67–82. https://doi.org/10.1016/0024-3795(87)90103-0

[4] Donghui Chen and Robert J Plemmons. 2010. Nonnegativity constraints in numerical analysis. In *The birth of numerical analysis*. World Scientific, 109–139.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition* (3rd ed.). The MIT Press.

[6] Joaquim F. Pinto da Costa. 2011. *Weighted Correlation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1653–1655. https://doi.org/10.1007/978-3-642-04898-2_612

[7] Soura Dasgupta and Yih-Fang Huang. 1987. Asymptotically convergent modified recursive least-squares with data-dependent updating and forgetting factor for systems with bounded noise. *IEEE Transactions on information theory* 33, 3 (1987), 383–392.

[8] David Chin-Lung Fong and Michael Saunders. 2011. LSMR: An Iterative Algorithm for Sparse Least-Squares Problems. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2950–2971. https://doi.org/10.1137/10079687X arXiv:https://doi.org/10.1137/10079687X

[9] Akshay Gadde, Aamir Anis, and Antonio Ortega. 2014. Active Semi-Supervised Learning Using Sampling Theory for Graph Signals. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 492–501. https://doi.org/10.1145/2623330.2623760

[10] Andrew Guillory and Jeff A Bilmes. 2009. Label Selection on Graphs. In *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Eds.), Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2009/file/90794e3b050f815354e3e29e977a88ab-Paper.pdf

[11] Andrew Guillory and Jeff A Bilmes. 2009. Label selection on graphs. *Advances in Neural Information Processing Systems* 22 (2009).

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA.

[13] Martin L Hazelton. 2001. Inference for origin–destination matrices: estimation, prediction and reconstruction. *Transportation Research Part B: Methodological* 35, 7 (2001), 667–676. https://doi.org/10.1016/S0191-2615(00)00009-6

[14] Syed Aseem Ul Islam and Dennis S Bernstein. 2019. Recursive least squares for real-time implementation [lecture notes]. *IEEE Control Systems Magazine* 39, 3 (2019), 82–85.

[15] Junteng Jia, Michael T. Schaub, Santiago Segarra, and Austin R. Benson. 2019. Graph-Based Semi-Supervised & Active Learning for Edge Flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 761–771. https://doi.org/10.1145/3292500.3330872

[16] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 137–146.

[17] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. 2006. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*. 2–10.

[18] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. 2008. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management* 134, 6 (2008), 516–526.

[19] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-Effective Outbreak Detection in Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Jose, California, USA) *(KDD '07)*. Association for Computing Machinery, New York, NY, USA, 420–429. https://doi.org/10.1145/1281192.1281239

[20] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4189–4196.

[21] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. 2018. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1852–1872.

[22] Michael James Lighthill and Gerald Beresford Whitham. 1955. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229, 1178 (1955), 317–345.

[23] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2575–2582. https://elib.dlr.de/127994/

[24] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14 (1978), 265–294.

[25] OpenStreetMap contributors. 2023. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

[26] Hao Peng, Hongfei Wang, Bowen Du, Md Zakirul Alam Bhuiyan, Hongyuan Ma, Jianwei Liu, Lihong Wang, Zeyu Yang, Linfeng Du, Senzhang Wang, et al. 2020. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences* 521 (2020), 277–290.

[27] Ananth Narayan Samudrala, M. Hadi Amini, Soummya Kar, and Rick S. Blum. 2019. Optimal Sensor Placement for Topology Identification in Smart Power Grids. In *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*. 1–6. https://doi.org/10.1109/CISS.2019.8692792

[28] Burr Settles. 2009. Active learning literature survey. (2009).

[29] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905. https://doi.org/10.1109/34.868688

[30] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 914–921.

[31] Ben Stabler, Hillel Bar-Gera, and Elizabeth Sall. 2016. Transportation Networks for Research. https://github.com/bstabler/TransportationNetworks.

[32] Amarnag Subramanya and Partha Pratim Talukdar. 2014. *Graph-Based Semi-Supervised Learning*. Morgan & Claypool Publishers.

[33] OZ Tamin and LG Willumsen. 1989. Transport demand model estimation from traffic counts. *Transportation* 16 (1989), 3–26.

[34] Yuichi Tanaka, Yonina C. Eldar, Antonio Ortega, and Gene Cheung. 2020. Sampling Signals on Graphs: From Theory to Applications. *IEEE Signal Processing Magazine* 37, 6 (2020), 14–30. https://doi.org/10.1109/MSP.2020.3016908

[35] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[36] Luis G Willumsen. 1981. Simplified transport models based on traffic counts. *Transportation* 10, 3 (1981), 257–278.

[37] Xiaojin Zhu and Zoubin Ghahramani. 2003. *Learning from Labeled and Unlabeled Data with Label Propagation*. Technical Report CMU-CALD-02-107. School of Computer Science, Carnegie Mellon University.

[38] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (Washington, DC, USA) *(ICML'03)*. AAAI Press, 912–919.

[39] Xiaojin Zhu, Jaz Kandola, John Lafferty, and Zoubin Ghahramani. 2006. Graph Kernels by Spectral Transforms. In *Semi-Supervised Learning*. MIT Press. https://doi.org/10.7551/mitpress/9780262033589.003.0015 arXiv:https://academic.oup.com/mit-press-scholarship-online/book/0/chapter/353093119/chapter-ag-pdf/44419226/book_41571_section_353093119.ag.pdf

A REPRODUCIBILITY

Here we describe the setup and implementation details of our experiments presented in Section 5. Implementations of all algorithms, along with code for reproducing experiments, can be found at https://github.com/arnav-b/sensor-placement.

A.1 Flow Conservation

Datasets. We use real-world traffic flows on road networks from four cities, Anaheim, Barcelona, Chicago, and Winnipeg (see Table 1) [31]. The nodes in each network represent intersections and edges represent roads between them. Each network is represented as a directed graph where the direction of an edge corresponds to the direction of traffic flow.

Table 1: Road Networks Used for Experiments

Network	n	m	Flows	Speeds
Anaheim	416	914	Real	Synthetic
Barcelona	1020	2522	Real	–
Chicago	933	2950	Real	–
Winnipeg	1052	2836	Real	–
Berlin	361	766	–	Synthetic
Eastern Massachusetts	74	258	–	Synthetic
Los Angeles	1190	1212	–	Synthetic

Hyperparameters. We set $\lambda = 10^{-6}$ when solving the least-squares formulation in Equation 6.

Baselines. We compare our flow selection method with three baselines. First, random selection (Random) simply chooses the next edge uniformly at random from the currently unchosen edges. Second, recursive bisection (RB) [15] partitions the graph using spectral clustering and chooses edges crossing the new cut as the next sensors. The idea is to find "bottleneck" edges where major flows are concentrated, such as major highways. Recursive bisection has been found to work well when much of the flow is not conserved [15]. Finally, rank-revealing QR (RRQR) [3] exploits a known bound on the error of the flow conservation algorithm (Equation 5). If the SVD of the incidence matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ is given by $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^T$, then the $m - n + 1$ rightmost columns $\mathbf{V}_C$ of $\mathbf{V}$ give a basis for the cycle space of fully conserved flows. If S is a set of $m - n + 1$ linearly independent rows of $\mathbf{V}_C$, then the reconstruction error is bounded by

$$||\hat{\mathbf{f}} - \mathbf{f}|| \leq (\sigma_{\min}^{-1}(\mathbf{V}_{SC}) + 1) \cdot ||\delta|| \tag{11}$$

RRQR uses a greedy heuristic to minimize this bound by minimizing $\sigma_{\min}^{-1}(\mathbf{V}_{SC})$ [15].

Evaluation metrics. We report the Pearson's correlation coefficient between the predicted ($\hat{\mathbf{f}}$) and the ground-truth ($\mathbf{f}$) flows.

Synthetic Flows [15]. For the experiments in Section 5.1.2, we generate synthetic flows based on the singular vector decomposition of $\mathbf{B}$ as follows. In the SVD of $\mathbf{B}$ given above, $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times m}$ contains the singular values $\rho_1, ..., \rho_m$ of $\mathbf{B}$ along the main diagonal with $m - n$ extra columns of zeros, and $\mathbf{V} \in \mathbb{R}^{m \times m}$ contains the right singular vectors. These vectors $\mathbf{V}_1, ..., \mathbf{V}_m$ give a basis for the edge flow space. The synthetic flow $\mathbf{f}$ can be computed as

$$\mathbf{f} = \sum_{i=1}^{m} \frac{b}{\rho_i + \epsilon} \mathbf{V}_i \tag{12}$$

where $b \in \mathbb{R}$ controls the flow magnitude and $\epsilon \in \mathbb{R}_+$ controls the amount of non-conserved flow. Similar to [15], we set $b = 20$ and $\epsilon = 0.1$ for our experiments.

A.2 Smoothness

Datasets. Because the sensor selection heuristic for smooth labels (Algorithm 2) is much slower than the heuristic for conserved flows (Algorithm 1), we use the smaller Anaheim, Berlin, and Eastern Massachusetts networks [31], and a sub-network of the Los Angeles, CA, network from OpenStreetMap [25] in our experiments.

Baselines. We compare our selection method with random selection and spectral clustering [11, 29, 35]. The spectral clustering-based sensors are chosen by generating k clusters on the line graph $L(G)$ and choosing a vertex at random from each cluster. When one or more clusters are empty, we choose edges at random to make up for the difference.

Evaluation metrics. We report the weighted correlation between the predicted labels $\hat{\mathbf{x}}$ and the ground-truth $\mathbf{x}$. Weights are given by a vector $\mathbf{w} \in \mathbb{R}^m$ where $\mathbf{w}_i$ is the weight for edge i. Setting $\mathbf{w} = \mathbf{f}$ allows us to weight correlation by the flow at each edge. We then compute the weighted correlation coefficient between $\hat{\mathbf{x}}$ and $\mathbf{x}$ [6]. For the Anaheim network, where the ground truth flow is available, we set $\mathbf{w} = \mathbf{f}$. For all other networks, we set $\mathbf{w} = \mathbf{1}$, which is equivalent to unweighted correlation.