Abstract—Reduction of end-to-end network delay is an optimization task with applications in multiple domains. Low delays enable improved information flow in social networks, quick spread of ideas in collaboration networks, low travel times for vehicles on road networks, and increased rate of packets in the case of communication networks. Delay reduction can be achieved by both improving the propagation capabilities of individual nodes and adding additional edges in the network. One of the main challenges in such network design problems is that the effects of local changes are not independent, and as a consequence, there is a combinatorial search-space of possible improvements. Thus, minimizing the cumulative propagation delay requires novel scalable and data-driven approaches. We consider the problem of network delay minimization via node upgrades. We show that the problem is NP-hard and prove strong inapproximability results about it (i.e. APX-hard) even for equal vertex delays. On the positive side, probabilistic approximations for a restricted version of the problem can be obtained. We propose a greedy heuristic to solve the general problem setting which has good quality in practice, but does not scale to very large instances. To enable scalability to real-world networks, we develop approximations for Greedy with probabilistic guarantees for every iteration, tailored to different models of delay distribution and network structures. Our methods scale almost linearly with the graph size and consistently outperform competitors in quality. We evaluate our approaches on several real-world graphs from different genres. We achieve up to 2 orders of magnitude speed-up compared to alternatives from the literature on moderate size networks, and obtain high-quality results in minutes on large datasets while competitors from the literature require more than 4 hours.

Index Terms—Shortest Paths, Network Design, Sampling

1 INTRODUCTION

Given a communication network, how can one minimize the end-to-end communication delay by upgrading networking devices? Similarly, how to minimize the travel time on an airline network by increasing the personnel and infrastructure at key airports? How to recruit users who can quickly re-post updates enabling fast global propagation of information of interest in a social network? There is a common network design problem underlying all the above application scenarios: for a large network with associated node delays, identify a set of nodes (within budget) whose delay reduction will minimize the path delays between any pair of nodes.

Network design problems, including planning, implementing and augmenting networks for desirable properties, have a wide range of applications in communication, transportation and information networks as well as VLSI design [1], [2], [3], [4], [5], [6]. Challenges in this area are posed by the rapidly growing sizes of real-world networks, leading to the need for scalable, data-driven approaches. In particular, some network design problems involve local changes to an existing large network such as adding/modifying links or nodes as a means to improve its global properties [3], [5], [7], [8], [9], [10]. In this paper we address a problem from the above category, namely, minimizing the overall end-to-end network delay.

- S. Medya and A. Singh are with the Dept. of Computer Science, University of California, Santa Barbara. E-mails: {medya,ambuj}@cs.ucsb.edu
- P. Bogdanov is with the Dept. of Computer Science, University at Albany—SUNY. E-mail: pbogdanov@albany.edu

Fig. 1: Airports with maximum impact on the overall network delay in the American Airlines network as discovered by our methods. If airline-caused delays are removed in these airports, the overall network delay decreases by 96% and 55% when the accumulated airline-caused flight delays (Total) and Average airport flight delays are considered respectively (data from US Dept. of Transportation).

The end-to-end delay in a network affects propagation speeds and is a function of the network link connectivity and the throughput capabilities of individual nodes. The majority of previous work focuses on delay minimization by augmenting network edges [3], [6], [10], [11], [12]. Less attention has been devoted to the complementary, but algorithmically non-equivalent setting in which the propagation capabilities of individual nodes are “upgraded” under budget [7]. In this paper we address the node-version of the delay minimization problem. A toy example instance and possible solutions for the problem are presented in Fig. 2. All nodes start with a delay of 1. The objective is to select a set of two nodes whose delay reduction will minimize the overall network delay (sum of pairwise delays). For instance, selecting nodes c, d (Fig. 2b) is a better solution than selecting nodes a, f (Fig. 2a).

Depending on the domain, low end-to-end delay en-
ables improved information flow in social and collaboration networks \cite{13}, reduced travel time for airline and road networks \cite{14} and increased throughput for communication networks \cite{9}. Consider, for example, the air transportation network of a major US carrier presented in Fig. 1 where edges correspond to flights offered by the carrier between endpoint cities. Based on historical information on past flights one can associate airports with airline-caused delays. An important question for an airline is then how to minimize overall delays by improving the number of personnel and available infrastructure (e.g. luggage handling) in problematic airports that affect multiple routes. In Fig. 1 we show the airports with highest delay-reduction potential, determined based on both historical delays and their position in the network. When the cumulative historical delays are considered (Total), hub airports like Chicago, Dallas and Miami constitute the best solution, while “fringe” airports make it to the list when the Average delay is considered \cite{11].

Another important application comes from online social networks where user behavior—activity and interest in a specific topic—determines the node delay for information propagation. In this domain, the objective is to speed up the global propagation of information by decreasing individual response time \cite{15}. A social media strategst of an election campaign, for example, would be interested in recruiting social network users who can re-post campaign updates immediately, enabling faster propagation of relevant campaign information. Both the position in the network and the current delay in propagating information should be taken into account in selecting recruits. While information and influence propagation are traditionally modelled as diffusion processes (i.e., using all possible paths) \cite{16}, multiple recent approaches (including the current work) focus on the most probable (shortest) paths in order to allow scalable solutions \cite{17}, \cite{18}.

Given a network with node delays, our goal is to identify a set of nodes whose delay reduction will minimize the sum of shortest path delays between a given set of pairs of nodes. We term this problem the Delay Minimization Problem (DMP), and demonstrate that it is NP-hard in a general network, even when the initial node delays are equal. Intuitively, the challenge stems from the fact that the global effect of a single node upgrade is dependent on the remaining nodes in the solution. We prove strong inapproximation result that DMP with initial equal delays remains APX-hard. On the positive side, we show approximation guarantees for other variants of the problem based on Vapnik-Chervonenkis theory \cite{19}.

Due to the strong inapproximation results of the DMP, we propose a Greedy heuristic which is optimal for restricted graph structures (e.g. trees) and has a good quality in practice for general graphs. However, Greedy does not scale to large networks due to its high computational complexity. Hence, we develop sampling-based algorithms that make similar selection of node upgrades with high probability using knowledge of only a small fraction of the network. The underlying model of delay, uniform or arbitrary delays, plays an important role in the design and solution quality of our corresponding sampling schemes. We provide a summary of the new material is as follows:

- Theory: A more general problem formulation including non-trivial and in-depth approximability analysis, sampling algorithm in a restricted setting and its complexity analysis, omitted (from short version) and new proofs.
- Methods: An exact solution for the new formulation based on mixed integer programming as well as relevant quality comparisons.
- Experiments: (a) comparing Greedy method with the sampling counterparts, (b) varying different parameters and (c) comparing against the approximation results from Theorem \cite{8}.
- Related work: Significantly extended.

### 2 Problem Definition and Complexity

A network is modeled as an undirected graph $G(V,E,l)$, where $V$ and $E$ are sets of vertices and edges respectively and $l$ is a function $l : V \rightarrow \mathbb{R}_{\geq0}$ over $V$ that specifies the delay/latency $l(v)$ of individual nodes. The delay (or length) of a path is defined as the cumulative delay of the vertices along the path, excluding that of the destination. More formally, if $P_{s,t} = (v_s, v_1, v_2, ..., v_r, v_t)$ is a path from vertex $v_s$ to $v_t$, its length is defined as $l(v_s) + \sum_{i=1}^{r} l(v_i)$. Delay at the destination node in a path is excluded since our targeted applications consider information/traffic flow and the destination node does not add any delays. The shortest

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definitions and Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(V,E)$</td>
<td>A graph with a vertex set $V$ and edge set $E$</td>
</tr>
<tr>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>$</td>
<td>E</td>
</tr>
<tr>
<td>$l(v)$</td>
<td>Delay value of node $v$</td>
</tr>
<tr>
<td>$d(s,t)$</td>
<td>Shortest path (s.p.) distance between $s$ and $t$</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>Candidate set of possible vertices for delay reduction</td>
</tr>
<tr>
<td>$k$</td>
<td>A budget of number of vertices to improve / upgrade</td>
</tr>
<tr>
<td>$T \subseteq V$</td>
<td>Selected target set of vertices to improve, $</td>
</tr>
<tr>
<td>$\text{DS}(G,T)$</td>
<td>Reduction in total s.p. distance after improving nodes in $T$</td>
</tr>
<tr>
<td>$\text{RS}(s</td>
<td>T)$</td>
</tr>
</tbody>
</table>

Given that nodes in $S$ are already upgraded / improved

### Related work: Significantly extended.
path between vertices \( s \) and \( t \) is that of minimum length (delay) among all such paths and its length is denoted as \( d(s,t) \). By convention, \( d(s,s) = 0 \) for all \( s \in V \). Let \( \Omega \) be a given set of pairs of vertices. We define the *shortest path delays* (SPD(\( \Omega \))) as the sum of shortest path lengths between the pairs of vertices in \( \Omega \). i.e., \( SPD(\Omega) = \Sigma_{(s,t) \in \Omega} d(s,t) \).

The DMP asks for a subset of vertices whose upgrade, i.e., reducing their delay, minimizes the overall SPD. In the process, from a given candidate set of vertices \( T \), the delay of a fixed (small) number of vertices \( T \subset \Gamma \) is reduced to a small value \( \psi \geq 0 \). We call this subset \( T \) a *Target Set* (TS) and its size \( |T| \leq k \) the budget. The upgrade of the TS reduces the lengths of shortest paths in the network. We denote the resulting (effective) shortest path length between \( s \) and \( t \) given the upgrade of \( T \) as \( d(s,t|T) \). Our goal is to find a \( T \) that minimizes \( \Sigma_{(s,t) \in \Omega} d(s,t|T) \).

**Definition 1. Delay Minimization Problem (DMP):** Given a network \( G = (V,E,I) \), a candidate set of vertices \( \Gamma \), a set of pairs of vertices \( \Omega \) and a budget \( k \), find a target set \( T \subset \Gamma \), such that \( |T| \leq k \) and \( \Sigma_{(s,t) \in \Omega} d(s,t|T) \) is minimized.

As defined above, the general design problem considers a subset of vertices \( \Gamma \subset V \) that can be upgraded and a subset of pairs \( \Omega \subset V \times V \) whose pairwise delays are of interest for the minimization. This general definition allows focused design in a subnetwork of interest, however, for simplicity we will present our algorithms in the context of the global version of the problem, i.e. \( \Gamma = V \), \( \Omega = V \times V \) and \( \psi = 0 \) (unless specified otherwise). All results and algorithms are applicable to the general case.

Figure 2 shows two possible TS solutions of size \( k = 2 \) for a small network. Initially all vertices have a delay of 1 corresponding to an SPD of \( \Sigma_{(s,t) \in V \times V} d(s,t) = 58 \). The reduction due to any TS \( T \) is defined as the difference between the initial and the upgraded SPD, i.e. \( \Sigma_{(s,t) \in V \times V} d(s,t) - \Sigma_{(s,t) \in V \times V} d(s,t|T) \). An optimal TS maximizes the reduction (and minimizes the upgraded SPD). Thus, Figure 2 shows a sub-optimal TS \( \{a,f\} \) with reduction of 10, while Figure 2 shows an optimal TS \( \{c,d\} \) with maximum SPD reduction of 38. Our goal is to minimize the SPD by finding the optimal TS of budget size at most \( k \).

2. Reduction by units of delay can be approached with simple changes in our algorithms.

---

**2.1 Hardness and Inapproximability**

In this section we study the complexity of node delay minimization (DMP) under two models for delay distribution in the network. Under the general model, node delays can be arbitrary non-negative values, while the restricted uniform model assumes equal delays (for simplicity, delay of 1) associated with all nodes. We show that DMP is NP-hard in the special case of the uniform model, and hence, NP-hard under the general model as well. To show this hardness result we reduce the Set Cover problem to DMP.

**Theorem 1.** DMP is NP-hard even if the delay of all vertices is 1, i.e. under the uniform model.

**Proof.** We outline a reduction from the Set Cover problem. Consider an instance of the NP-complete Set Cover problem, defined by a collection of subsets \( S_1, S_2, ..., S_m \), for a universal set of items \( U = \{u_1, u_2, ..., u_n\} \). The problem is to decide whether there exist \( k \) subsets whose union is \( U \). To define a corresponding DMP instance, we construct an undirected graph with \( n + m + mp \) nodes: there are nodes \( i \) and \( j \) corresponding to each set \( S_i \) and each element \( u_j \) respectively, and an undirected edge \( (i,j) \) whenever \( u_j \in S_i \). Every \( S_i \) is connected to \( S_j \) when \( i \neq j \) and \( i,j \in 1,2, ..., m \). Every \( u_i \) is connected to other \( u_j \) when \( i \neq j \) and \( i,j \in 1,2, ..., m \). There are \( p \) vertices (with degree 1) attached to each \( S_i \). The \( j \)-th vertex (among these \( p \) vertices) attached with every \( S_i \) makes the set \( A_j \). All vertices have delay of 1. Intuitively, the construction makes the vertices in set \( S \) more likely to be chosen in TS.

Vertices in \( A_j \) will not be in the target set TS, as they are of degree 1. Next we prove that the minimum reduction (say quantity \( A \)) by any vertex from \( S \) is larger than the maximum reduction (say quantity \( B \)) by vertices from \( U \). As the delays are 1, the reduction depends on the number of shortest paths that pass through a vertex. The maximum number of shortest paths (quantity \( B \)) that pass through any vertex in \( U \) after being chosen in TS is less than the minimum (quantity \( A \)) of the same through any vertex in \( S \). Quantity \( B \) is exactly \((n-1)(m+pm)\), while \( A = p(m+n+p(m-1)) \). A choice of \( p = mn \) makes \( A \) larger than \( B \). Hence a choice of nodes from \( S \) is always preferable. Tables 2 and 3 show SPD computation between nodes in the sets \( S, U, A_j \) in two different cases. The quantities are as follows: \( W_1 = k(2(m-k)+m+k-1) + (m-3k)(3m-k-1+2k) \), \( W_2 = k(2(m-k)+k-1+k+1) + (m-3k)(3m-k-1+2k+2) \), \( W_3 = kn + 2(m-k)n \), \( W_4 = k(m-k) + (m-k)(1+k+1+2m-k-1) \), \( W_5 = km + 2(m-k)(m-k-1) + k(m-k) \).

The Set Cover problem is equivalent to deciding if there is a set of \( k \) vertices whose upgrade leads to \( SPD \leq X \) (where \( X = (m+n-1)(-k+m+n)+mp^2(3m-2k)+p(k(4-3m)+m(4m-5)) + 2pt(2mn-kt) \), sum of all the elements in Table 2). For a “yes”-instance of the Set Cover problem, we show that all \( k \) TS vertices correspond to selected sets in the set cover and we achieve the effective SPD of \( X \). For a “no”-instance of the corresponding Set Cover problem, the argument is as follows. For the TS we choose \( k \) nodes from set \( S \) as we have already proved, no vertex from set \( U \) or \( A \), can be in TS. For a “no”-instance Table 3 shows the desired SPD. Comparing both two tables, it is evident that the SPD in Table 3 is greater than \( X \). If the corresponding Set Cover problem has a set cover \( \leq k \), then only the SPD is reduced to \( X \). Hence the claim is true and the problem is NP-hard. In conclusion, DMP is NP-hard under the general model.

\[ \square \]
Table 2: Sum of shortest path delays when the size of the set cover is \( \leq k \) and delays of nodes in the set cover are reduced to 0. The cell \([K, L]\) denotes the sum of shortest path delays between nodes of set \( K \) and those of set \( L \).

Table 3: Sum of shortest path delays when size of set cover is > \( k \) and delays of arbitrary k nodes from \( S \) set to 0. The cell \([K, L]\) denotes the sum of shortest path delays between nodes of set \( K \) and those of set \( L \).

Theorem 1 establishes that the problem is NP-hard and finding a brute-force optimal TS of size \( k \) would involve considering all \( O(|V|^k) \) subsets of \( V \).

Since DMP is NP-hard, we next investigate the existence of approximations with guarantees. The following theorem shows that the general DMP is also APX-hard under uniform model. More specifically, it is NP-hard to approximate within a factor greater than \( 1 - \frac{1}{e} \).

**Theorem 2.** DMP under the uniform model is APX-hard and it is NP-hard to approximate within a factor greater than \( 1 - \frac{1}{e} \).

**Proof.** We use Maximum Coverage (MSC) problem for reduction. Given a collection of subsets \( S_1, S_2, ..., S_m \) for a universal set of items \( U = \{u_1, u_2, ..., u_n\} \), the problem is to choose at most \( k \) sets to cover as many elements as possible. We show an \( L \)-reduction from the MSC problem, parameterized by the multiplicative factors \( x \) and \( y \), such that the following two equations are satisfied:

\[
OPT(I_{DMP}) \leq xOPT(I_{MSC})
\]

\[
OPT(I_{MSC}) - s(T^M) \leq y(OPT(I_{DMP}) - s(T^D))
\]

where \( I_{MSC} \) and \( I_{DMP} \) are corresponding problem instances, and \( OPT(Y) \) is the optimal value for an instance \( Y \). \( s(T^M) \) and \( s(T^D) \) denote any solutions of the MSC and DMP instances respectively. If the conditions hold and DMP has an \( \epsilon \) approximation, then MSC has a \((1 - \epsilon)OPT(1 - \epsilon)\) approximation. However, MSC is NP-hard to approximate within a factor greater than \( 1 - \frac{1}{e} \). It then follows that \((1 - \epsilon)OPT(1 - \epsilon) < (1 - \frac{1}{e})\), or \( \epsilon < (1 - \frac{1}{e}) \). So, if the conditions are satisfied, DMP is NP-hard to approximate within a factor greater than \( 1 - \frac{1}{e} \).

To define a corresponding DMP instance, we construct an undirected graph with \( m + n + 2 \) nodes: a node \( i \) corresponds to each set \( S_i \) in MSC and similarly a node \( j \) corresponding to each element \( u_j \). There is an undirected edge \((i, j)\) whenever \( u_j \in S_i \). We add two extra nodes \( a \) and \( b \) connected to each other and \( b \) is connected to all nodes \( i \) where \( S_i \in S \), i.e. edges \((b, i)\) are added to the graph. Consider the DMP instance of the graph constructed as described above. Let the pairs of interest to upgrade be

### Fig. 3: Example of reduction from MSC to DMP, where \(|U| = 4\) and \(|S| = 3\). \( \Omega = \{(a, v)|v \in U\} \) and candidate set of vertices \( \Gamma = S \).

\( \Omega = \{(a, j)|u_j \in U\} \) and the candidate set of possible vertices to upgrade be \( \Gamma = \{i|S_i \in S\} \). The delays associated with all vertices are 1. Figure 3 illustrates the construction with \(|S| = 3\) and \(|U| = 4\).

Let the solution of \( I_{DMP} \) be \( s(T^D) \) by choosing \( k \) nodes from \( \Gamma = S \). It is easy to see that the achieved reduction of delay will be \( s(T^D) \) in \( \Omega \) as the delays on the nodes are 1. Hence, \( s(T^D) = s(T^M) \), where \( s(T^M) \) is the solution for MSC problem. It follows that both the conditions are satisfied when \( x = y = 1 \). As a result, DMP is NP-hard to approximate within a factor greater than \( (1 - \frac{1}{e}) \).

As the theorem suggests, it is computationally infeasible to get approximation within a factor greater than \( (1 - 1/e) \). It is an open question whether DMP has an approximation of any constant smaller than \( (1 - 1/e) \). Maximizing a non-negative, monotone and submodular function using a greedy approach leads to a well known constant time approximation of \( (1 - 1/e) \) [23]. The underlying objective function in DMP under uniform model does not have the submodular property.

**Lemma 1.** The objective function in DMP is monotone but not submodular, even under the uniform model.

**Proof.** The objective function \( f(T) \) in DMP is “delay reduction” defined as \( f(T) = \sum_{x \in T} d(s, t) - \sum_{x \in T} d(s, |T|) \), where \( T \) is the target set. The function \( f(T) \) is monotone in the size of \( T \). To prove non-submodularity, we consider the example of a ring graph \( G \) of six vertices with unit delays: vertex \( x_1 \) is connected to \( x_2, x_2 \) to \( x_3 \) and so on, and finally \( x_6 \) is connected to \( x_1 \). The intuition is the following: a super-set of nodes as TS might force more shortest paths through the newly added vertex than its sub-set as TS. Let set \( A = \phi \), \( B = \{x_2, x_3\} \). In our example, \( f(B \cup \{x_3\}) = 54 - 21 = 33 \), \( f(B) = 54 - 34 = 20 \), \( f(A \cup \{x_3\}) = 54 - 43 = 11 \), \( f(A) = 0 \). So, \( f(B \cup \{x_3\}) - f(B) > f(A \cup \{x_3\}) - f(A) \). So, \( f(\cdot) \) is not submodular.

### 2.2 Approximability for “long” paths

In theorem 2, we establish an inapproximability result about uniform (equal) weights DMP and as a result the general (arbitrary) weights setting is at least as hard. This leads us to explore the existence of approximation for DMP in restricted or special cases. One such direction is: Can we reduce with guarantees the most latent (longest) paths? We show that there exists an approximation for a restricted variant of DMP under the uniform model targeting long paths for delay reduction. Focusing on long paths, as opposed to all lengths, is useful in applications where delays up to a
Lemma 2. \( \epsilon \)-net \cite{24}: For any set system with bounded VC-dimension \( d \), a randomly drawn sample of size \( O\left(\frac{d \log d + \frac{1}{2} \log \frac{1}{\delta}}{\epsilon}\right) \) is an \( \epsilon \)-net with probability \( \delta \).

We analyze the VC dimension of the shortest paths in a graph. Given a graph \( G = (V, E) \), a shortest path can be uniquely defined by a source vertex \( s \) and a destination vertex \( t \). We consider set system \( (V, R) \) where \( V \) is the set of vertices and each element of \( R \) corresponds to the set of vertices on a shortest path. We refer to such collection of sets for graphs with unique shortest paths as USP systems \cite{25}. This set is defined as follows:

Definition 2. **USPS** \cite{25}: Given a graph \( G(V, E) \) and a collection \( R \) of shortest paths from \( G \), we say \( R \) is a unique shortest path system (USPS) if: any vertex pair \( u \) and \( v \) is contained in two shortest paths \( p_{u,t}, p_{v,t} \in R \), then \( u \) and \( v \) are linked by the same path, i.e., \( p_{u,v} = p'_{u,v} \), where \( p_{u,v} (p'_{u,v}) \) is the subpath of \( p_{u,t}, p_{v,t} \).

Next, we introduce two important lemmas:

Lemma 3. **Dimension** \cite{26,27}: For a graph \( G = (V, E) \), the set system \( (V, R) \), where \( R \) is a unique shortest path system, has a VC-dimension of 2.

Lemma 4. **Existence of USPS** \cite{25}: There exists a unique shortest path system for every graph.

We next show an approximation for “long” paths. First we define “long” paths as the following:

Definition 3. **Long path**: A long path (shortest path) is defined as the path that has a delay of \( cn \) or higher, where \( |V| = n \) and \( 0 < \epsilon < 1 \).

Our optimization objective for “long” paths is \( SPD'(G) = \sum_{s,t \in E \times V, d(s,t) \geq \epsilon n} d(s,t) \). Let \( R_{opt}(k) \) represent the reduction in \( SPD' \) by the optimal TS of size \( k \) and \( R_{rand}(kb) \), the reduction due to \( kb \) randomly chosen vertices. The relationship between \( k \), \( b \) and \( \epsilon \) is captured in the following theorem. The intuitive sketch of the proof is the following: if the set system \( (V, R) \) has bounded (constant) VC-dimension and the shortest paths are long enough (inside \( R \) then the size of the \( \epsilon \)-net (the set of nodes) reside on those paths are bounded by Lemma 2.

Theorem 3. Given a confidence parameter \( \delta \), \( \frac{R_{opt}(k)}{R_{rand}(kb)} \leq k \) with probability \( \delta \), where \( kb = (\frac{\epsilon}{c} \log \frac{1}{\epsilon} + \frac{1}{2} \log \frac{1}{\delta}) \).

Proof. We develop an upper bound for \( R_{opt}(k) \). In the best case scenario, all \( k \) nodes of the TS are present in a set \( S \) of all different shortest paths. So, the maximum reduction is \( k|S| \) (as every vertex has a delay of 1). From Lemma 3 and \( |S| \) the USPS of a graph always exists and its VC-dimension is 2. A random sample of \( b'(\frac{\epsilon}{c} \log \frac{1}{\epsilon} + \frac{1}{2} \log \frac{1}{\delta}) \) vertices will be an \( \epsilon \)-net with probability \( \delta \) by Lemma 2 (\( b' \) is the constant in the asymptotic bound in Lemma 2 and \( 1/b' = b \)). In other words, at least one vertex from every shortest path of (length \( \geq cn \)) of the USPS will be included in the sample. If these vertices are selected as a target set, the resulting reduction will be \( |S'| \), where \( |S'| \) is the set of shortest paths of length \( \geq cn \) in USPS. It is clear that \( |S'| \geq |S| \). So, \( \frac{R_{opt}(k)}{R_{rand}(kb)} = \frac{k|S|}{|S'|} \leq k. \)

The theorem shows that the problem of minimizing the delay of long paths under the uniform model has an approximation of \( 1/k \). As the bound on path length \( cn \) increases, we need a smaller number of samples to “cover” every path. Through experiments, we compare our proposed algorithm PCS (Algorithm 3) against the theoretical upper bound on the restricted metric involving only long paths. Table 4 summarizes the hardness results.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Model</th>
<th>NP-hard?</th>
<th>APX-hard?</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMP</td>
<td>Uniform</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DMP</td>
<td>General</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DMP (Long paths)</td>
<td>Uniform</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>DMP (on Tree)</td>
<td>General</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

**TABLE 4:** Summary of hardness results.

In summary, we have established that it is infeasible to get a constant factor approximation for the general DMP beyond a constant (Thm. 2) and that approximations exist for only restricted variants (Thm. 3) and special graph structures (Lemma 5 proved in Section 5). Due to the inapproximability result and inspired by solutions in restricted scenarios, we next focus on a practical greedy heuristic to solve DMP and sampling-based variants that scale to large problem instances.

### 3 Algorithms

We present a greedy approach for DMP that selects the vertex that minimizes the SPD in each iteration. Such an approach is optimal for \( k = 1 \). It also produces optimal results for networks with simple structures (Lemma 5) and works well in practice for general instances. It is, however, expensive as it requires re-computation of all shortest paths at every iteration. To make the approach scalable, we employ sampling techniques and introduce probabilistic approximations for every single step of Greedy for different delay models.
**Algorithm 1: Greedy (GR)**

<table>
<thead>
<tr>
<th>Require:</th>
<th>Network $G = (V, E, L)$, Vertex delays $d(v)$, Budget $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure:</td>
<td>A subset of $k$ nodes, $T$</td>
</tr>
<tr>
<td>1:</td>
<td>Initialize Matrix $A$ with 0 and $T$ as $\emptyset$</td>
</tr>
<tr>
<td>2:</td>
<td>Compute all pair shortest paths</td>
</tr>
<tr>
<td>3:</td>
<td>Store $d(s, t)$ in Matrix position $A_{s,t}$</td>
</tr>
<tr>
<td>4:</td>
<td>while $</td>
</tr>
<tr>
<td>5:</td>
<td>for $v' \in V$ do</td>
</tr>
<tr>
<td>6:</td>
<td>Compute $RS(v'(T))$ when $l(v') &gt; 0$</td>
</tr>
<tr>
<td>7:</td>
<td>end for</td>
</tr>
<tr>
<td>8:</td>
<td>$v \leftarrow \max_{v \in V} {RS(v'(T))}$ and then set $R(v)$ as 0</td>
</tr>
<tr>
<td>9:</td>
<td>Update $d(s, t)$ for $s, t \in V$ as $l(v)$ becomes 0</td>
</tr>
<tr>
<td>10:</td>
<td>$T \leftarrow T \cup {v}$</td>
</tr>
<tr>
<td>11:</td>
<td>end while</td>
</tr>
<tr>
<td>12:</td>
<td>Return $T$</td>
</tr>
</tbody>
</table>

### 3.1 Greedy Construction of the Target Set

While finding the optimal TS is NP-hard, in the case of only one target vertex, an exact solution can be obtained by computing the reduction of all individual nodes in polynomial time. Therefore, a greedy algorithm, which selects a vertex that optimally reduces SPD at each step, is a natural approach to solve DMP. Before presenting the algorithm, we introduce some additional notation. We define the delay Reduction ($RS$) by a target set $S$ as:

$$RS(S) = \sum_{s \in T} d(s, t) - \sum_{s \in T} d(s, t|S).$$

We further define $RS$ by a vertex $v$, given that a subset $S$ has already been included in TS (assuming $v \notin S$) as:

$$RS(v|S) = \sum_{s \in V} d(s, t|S) - \sum_{s \in V} d(s, t|S \cup \{v\}).$$

The reduction of adding vertex $v$ to a set $S$ in TS can be expressed as $RS(S \cup \{v\}) = RS(v|S) + RS(S)$. The $RS$ of a vertex depends on: (i) its delay and (ii) the number of unique shortest paths passing through it after removing its delay. Maximizing $RS(v|S)$ takes both these properties into account. Next, we present an algorithm which iteratively selects the vertex of maximum reduction $RS(v|S)$.

GR (Alg. 1) is a greedy TS selection strategy. It takes a network $G$ ($|V| = n$ and $|E| = m$) and a budget $k$ as input. First it pre-computes all pairs of shortest paths and stores them in a $n \times n$ matrix $A$ (steps 2-3). Then it computes the TS of $k$ vertices in $k$ iterations. In each iteration, it selects the vertex with maximum $RS$ conditioned on the current TS (step 5-8). When probing each vertex, the algorithm assumes its delay as 0, updates the stored distances accordingly and estimates the reduction of the vertex. It chooses the vertex of maximum $RS$, makes its delay permanently 0, and adds it to the TS. GR also updates the stored shortest path distances accordingly.

**Example:** We provide a running example of GR in Fig. 2(b). The first selected vertex is either $c$ or $d$ as $RS(c) = RS(d) = 19$ and the $RS$ of any other vertex is 5. Assuming that GR chooses $c$ at the first step, the next best vertex is $d$ as $RS(d|c) = 19$. $RS(v|c)$, when $v$ is any other vertex, remains 5. In the example, GR produces the optimal TS as the network structure is a tree. GR is optimal for certain families of networks with simple structure. The following lemma outlines such families.

**Lemma 5.** Greedy (Alg. 1) produces an optimal TS in restricted structures such as trees, cliques and complete bipartite graphs under the general model.

Shortest paths between any pair of nodes in trees are unique and, hence, they do not change after upgrading the delay of any vertex. Intuitively, this fact about trees helps the greedy algorithm (Algorithm 1) to produce an optimal TS of size $k$. In a clique, since there is an edge between any pair of vertices, selecting $k$ vertices in descending delay order produces an optimal result and this is exactly the selection of GR. In a complete bipartite graph, if the delay of one vertex is updated then all vertices from the opposite partition will use this vertex to reach other vertices in their own partition, hence GR will again produce an optimal solution.

**Complexity:** GR runs in time $O(kn^3)$ which is dominated by the computation of shortest paths in steps 2, 6 and 9. Finding the next “best” vertex by evaluating the reduction of all possible vertices requires $O(n^3)$ time, where $n$ is the number of vertices. Moreover, updating the distances after a vertex is included in TS takes $O(n^2)$. The space complexity of computing all pairs shortest paths is $O(n^2)$. The high complexity of GR introduces a scalability challenge, rendering the algorithm infeasible for large real-world networks. Hence, we develop sampling-based versions of GR for large graphs and provide approximation guarantees w.r.t. every step of GR.

### 3.2 General Model: Approximate Target Set

The main drawback of GR is that it is not scalable. We address its computational and storage bottlenecks using a sampling scheme. The main idea behind our approach is as follows: instead of computing and optimizing the sum of distances between all pairs of vertices, we can estimate it based on a small number of sampled vertex pairs.

In what follows, we bound the difference in quality of our sampling solution GS (presented in Alg. 2) and Greedy (Alg. 1). In this case, the absolute value of the reduction $RS$ is not a suitable metric as the initial sum of shortest path distances (SPD) varies across input graphs. Hence, we choose Relative Reduction ($RR$) as a quality metric where we normalize $RS$ by the initial SPD. We define the measure $RR$ of a set $S$ as $RR(S) = \frac{RS(S)}{SPD}$. The $RR$ of a vertex $v$ given a set $S$ comprising the current TS is defined in a similar manner, $RR(v|S) = \frac{RS(v|S)}{SPD}$. As part of GS, we sample uniformly with replacement a set of ordered vertex pairs $P$ of size $p$ ($|P| = p$) from the set of all vertex pairs $U = \{(s, t) | s \in V, t \in V, s \neq t\}$, $|U| = n(n - 1)$. The samples can be viewed as random variables associated with the selection of a pair of vertices and the distance between a sampled pair is the value of the random variable. When uniform random sampling is used, each pair is chosen with probability $\frac{1}{n(n - 1)}$ and the choice of one sample does not affect that of any other sample. Thus, the samples are independent and identically distributed random variables.

We first show that the estimate of SPD based on samples is unbiased. Namely, for any target set of nodes $S$, the average of the sum of distances between pairs in $P$ is an unbiased estimate of that between all pairs of vertices, the latter being defined as $\mu = \sum_{s \in V, t \in V \setminus S} d(s, t)$. The vertex whose inclusion in TS optimizes this estimate is chosen in each step of GS.
Lemma 6. Given a sample of node pairs \( P, |P| = p \), the expected average distance among the sampled pairs is an unbiased estimate of the average of all-pair distances \( (\mu) \):
\[
E[\frac{1}{|P|} \sum_{i=1}^{p} X_i] = \mu
\]
where \( X_i \) represents the distance between the \( i \)-th pair of vertices in the sample.
Proof. The random variable, \( X_i \) is the the distance between the \( i \)-th pair of vertices in the sample. The probability of a pair in the selection is \( \frac{1}{n(n-1)} \cdot E[X_i] = \frac{1}{n(n-1)} \cdot \Sigma_{s,t \in V} d(s, t) \). We sample pairs independently with replacement. So, the variables, \( X_i \)'s are i.i.d. Now, \( E[\frac{1}{|P|} \sum_{i=1}^{p} X_i] = \frac{1}{p} \cdot \frac{1}{n(n-1)} \cdot \Sigma_{s,t \in V} d(s, t) = \mu \).

We employ Hoeffding’s inequality [28] to bound the error produced by our sampling method in a single greedy step. Hoeffding’s inequality provides a sample-size dependent bound for the difference between the estimated mean (based on samples) and the actual mean of a population. The requirement for the applicability of Hoeffding’s inequality is that the summed variables are chosen independently from the same distribution, which is the case in our setting. Similar independent node pair sampling analysis using Hoeffding’s inequality has been previously employed by Yoshida et al. [29] to estimate the group betweenness of vertices, whereas we estimate the reduction in the sum of shortest paths upon node upgrades. In what follows, we demonstrate that the estimate has low error with high probability, thus requiring only a small number of samples. Furthermore, we show the same quality guarantee with even smaller number of samples in small-world networks.

**Theorem 4.** Given a target set \( S \) and a sample \( P \) of size \( p \), if \( v_g \) and \( v_a \) are the next vertices chosen by GR and GS respectively, the difference in delay reduction due to these choices is bounded as follows:
\[
Pr[|RR(v_g) - RR(v_a)| < \varepsilon] > 1 - \frac{1}{n^2},
\]
where \( p = O(\frac{\varepsilon^2 \log n}{\varepsilon.SPD}), c = \frac{\text{diam}}{l_{\text{min}}} \) diam and \( l_{\text{min}} \) are the diameter and minimum delay respectively.
Proof. Let \( M_g = \Sigma_{s,t \in V} d(s, t, S \cup \{v_g\}) \) and \( M_a = \Sigma_{s,t \in V} d(s, t, S \cup \{v_a\}) \). Let also \( \mu_g \) and \( \mu_a \) denote the corresponding mean distances and \( Y_g \) and \( Y_a \) be the corresponding expected means computed using the samples.

Since the samples provide an unbiased estimate (Lemma 6) and are i.i.d., we can use Hoeffding’s inequality [28] to bound the error of the mean estimates:
\[
Pr[|Y_g - \mu_g| \geq \beta] \leq \delta
\]
where \( \delta = 2 \exp(-\frac{2\beta^2}{2}) \), \( X_i \) represents the distance between the \( i \)-th pair of vertices in the sample, \( a_i \leq X_i \leq b_i \), and \( \Lambda = \frac{p}{2}(b_i - a_i)^2 \). Similarly, \( Pr[|Y_a - \mu_a| \geq \beta] \leq \delta \).

Applying union bound, \( Pr[|Y_g - \mu_g| \geq \beta] \cup (|Y_a - \mu_a| \geq \beta) \leq 2\delta \). By construction, \( \mu_g \geq \mu_a \) as GR selects the best next vertex at each step. On the other hand, since GS selects \( v_a \), it must be that \( Y_a \geq Y_g \). As, the sampled best node is probabilistic, we need to apply union bound over \( n \) possible nodes. As a consequence, we get \( Pr[|\mu_g - \mu_a| \geq 2\beta] \leq 2n\delta \), or alternatively

Algorithm 2: Greedy with Sampling (GS)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Choose ( p = O(\varepsilon.SPD) ) pairs of vertices in ( P )</td>
</tr>
<tr>
<td>2</td>
<td>( T = \emptyset )</td>
</tr>
<tr>
<td>3</td>
<td>while (</td>
</tr>
<tr>
<td>4</td>
<td>for ((s, t) \in P ) do</td>
</tr>
<tr>
<td>5</td>
<td>Compute ( d(s, t') ) and ( s, target[t'] \to d(s, t') \forall t' \in V )</td>
</tr>
<tr>
<td>6</td>
<td>Compute ( d(s', t') ) and ( s, source[s'] \to d(s', t') \forall s' \in V )</td>
</tr>
<tr>
<td>7</td>
<td>end for</td>
</tr>
<tr>
<td>8</td>
<td>for ( v' \in V ) do</td>
</tr>
<tr>
<td>9</td>
<td>if (</td>
</tr>
<tr>
<td>10</td>
<td>( R_{v'} \to \Sigma_{s,t \in V} d(s, t') - \Sigma_{s,t \in V} d(s, t) \cup {v'} )</td>
</tr>
<tr>
<td>11</td>
<td>end if</td>
</tr>
<tr>
<td>12</td>
<td>end for</td>
</tr>
<tr>
<td>13</td>
<td>( v \to \max_{v' \in V} {R_{v'}} )</td>
</tr>
<tr>
<td>14</td>
<td>( t(v) = 0 ) and ( T \to T \cup {v} )</td>
</tr>
<tr>
<td>15</td>
<td>end while</td>
</tr>
<tr>
<td>16</td>
<td>Return ( T )</td>
</tr>
</tbody>
</table>

Note that, in the theorem, we assume \( l_{\text{min}} > 0 \) without loss of generality. If \( l_{\text{min}} = 0 \), one can delete any node of zero delay, add all possible edges among its neighbors and consider the resulting network as an input. For small-world networks (where the diameter is \( \leq l_{\text{max}}.\log n \)), a property exhibited in many domains, we show that the number of samples needed to obtain the same quality is much smaller.

Corollary 5. Given a small-world network in which \( \text{diam} \leq l_{\text{max}}.\log n \), the error of GS using \( p = O(\frac{\varepsilon^2 \log n}{\varepsilon.SPD}) \) samples can be bounded as:
\[
Pr[|RR(v_g) - RR(v_a)| < \varepsilon] > 1 - \frac{1}{n^2}
\]
Proof. Let the network have a small-world property [30], \( \text{diam} \leq l_{\text{max}}.\log n \), where \( l_{\text{max}} \) is the maximum delay. Now \( c \) in above theorem can be replaced by \( l_{\text{max}}.\log n \), where \( l_{\text{min}} \) is the minimum delay. \( l_{\text{min}} \) is assumed to be constant.

GS (Alg. 2) takes as input a network \( G \), a target approximation error \( \varepsilon \), a sampling factor \( c \) and a budget \( k \). The algorithm outputs a target set of vertices constructed based on optimizing the sum of the distances between each of the sampled pair paths. The approximation error, \( \varepsilon \), defines the difference between the approximate and the optimal reduction at each step. The number of samples \( p \) depends on the number of vertices \( n \), the error \( \varepsilon \), and the sampling factor \( c \). In theory, \( c \) should be chosen as shown in the theorem based on the input graph \( G \). But in practice, we use a small constant \( c \), requiring small number of samples (see Sec. 4).
The algorithm first samples pairs from the population of all pairs (step 1). Note that, although we present the algorithm with sampling pairs once, sampling new $p$ pairs in each iteration does not change the quality bounds or running time of the algorithm. The algorithm runs for $k$ iterations. It computes the sum of distances between each of the sampled pairs and selects the best vertex which reduces this sum the most. To achieve this, in each iteration, we compute the desired shortest path distances and store them (step 5-6). Next we select the vertex with maximum reduction in the sum of distances of the sampled pairs (step 8-12). The selection is conditioned on already selected vertices in TS.

**Complexity:** The running time of GS is dominated by the computation of shortest paths. Running Dijkstra’s algorithm for each sampled pair takes $O(m + \log n)$. The algorithm has $k$ iterations. This leads to the time complexity of $O(kp(m + n \log n))$. We need to store only the distances from the end vertices of the pair to all other vertices. This leads to a space complexity of $O(pm)$.

### 3.3 Uniform Model: Approximate Target Set

In some applications, instances of our design problem may feature uniform (equal) or close-to-uniform initial delays. For example, many routing devices in a computer network might have similar hardware configuration and hence feature comparable delays. Similarly, intersections with the same number of lanes within a road network allow for similar rate of cars to propagate during congestion periods. Such homogeneous instances offer more structure to the design problem and allow for a better (faster and high-quality) sampling scheme than our general-case algorithm GS. Hence, we develop and analyze a superior sampling-based method, called PCS (Path Count with Sampling), targeted for the uniform model.

We relate the delay reduction due to a vertex to the number of shortest paths passing through it. Let $\zeta^v(S)$ (or $\zeta^v$, we are omitting $S$ for simplicity) denote the number of shortest paths passing through a vertex $v$ assuming that $S$ is the target set.

**Theorem 6.** In the uniform model, for a given set $S$ and $v \notin S$, $RS(v|S) = \zeta^v + (n - 1)$.

**Proof.** There are three different cases to consider based on the kind of shortest paths. First, for shortest paths where $v$ is start vertex, updating its delay results in a reduction of $n - 1$. Second, for shortest paths that go through $v$ (but $v$ is not the start vertex), the reduction is $\zeta^v$. Finally, for shortest paths where $s$ is not on the path, the change in its delay does not result in any reduction (due to equal delays). Therefore, $RS(v|S) = \zeta^v + (n - 1)$.

With the above result, a greedy algorithm only needs to know the values of $\zeta$ for each vertex. The main bottleneck of computing $\zeta$ involves shortest path computation between all pairs of vertices. We address this complexity by a different sampling scheme. We estimate $\zeta$ for a vertex based on the shortest paths among $p$ pairs of vertices sampled independently with replacement. Let $X^v$ be a random variable denoting the number of times $v$ belongs to $SP_{s,t}$ for all sampled pairs $(s,t)$, where $SP_{s,t}$ $(s,t \notin SP_{s,t})$ denotes the set of vertices on the shortest path(s) between $s$ and $t$.

The expected value of the random variable is computed as follows:

**Lemma 7.** For any vertex $v$, $E[X^v] = \frac{p}{n(n-1)}\zeta^v$.

The lemma holds due to the additive property of expectation and the fact that the pairs are sampled independently. Next, we show that the difference in quality of GR and PCS is small with high probability in a single greedy step.

**Theorem 7.** Given a sample $P_i|P = p \leftarrow O\left(\frac{\log n}{\epsilon^2}\right)$, if $v_{g}$ and $v_{a}$ are the vertices chosen by GR and PCS respectively, then $Pr[|RR(v_{g})| - RR(v_{a})| < \epsilon] > 1 - \frac{1}{n^2}$.

**Proof.** If $X_1, X_2, ..., X_p$ are independent random variables in $[0, 1]$ and $\bar{X} = \frac{1}{p} \sum X_i$, then from Hoeffding’s inequality [28], $Pr[|\bar{X} - E[\bar{X}]| \geq \beta] \leq 2 \exp(-2\beta^2p)$. Using Lemma 7 and Hoeffding’s inequality, $Pr[|\zeta_{g}^v - \frac{1}{n(n-1)}X^v| \geq \beta] \leq 2 \exp(-2\beta^2p)$, and similarly $Pr[|\zeta_{a}^v - \frac{1}{n(n-1)}X^v| \geq \beta] \leq 2 \exp(-2\beta^2p)$. The optimal vertex chosen by PCS is $v_{a}$ and hence $X^v \geq X^v$. Since $X^v \geq X^v$, we get $\bar{X} \geq \bar{X}$. (by construction), we apply the same logic of union bound as in Theorem 4 to achieve $Pr[|\zeta_{g}^v - \frac{1}{n(n-1)}| < 2\beta] > 1 - 4n \exp(-2\beta^2p)$.

Now, we use this inequality to derive the following: $Pr[|RR(v_{g})| - RR(v_{a})| < \epsilon] = Pr[|RS(v_{g}) - RS(v_{a})| < \epsilon.SPD]$ $= Pr[|\zeta_{g}^v - \zeta_{a}^v| < \epsilon.SPD]$ (from Theorem 6) $= Pr[|\zeta_{g}^v - \zeta_{a}^v| < \epsilon.SPD]$ $> 1 - 4n \exp(-2\epsilon^2.NSPD^2)$ $> 1 - 4n \exp(-\frac{\epsilon^2}{2})$. Since $SPD > N$.

If we choose $p = \frac{2\log(4n^2)}{\epsilon^2}$, then $Pr[|RR(v_{g})| - RR(v_{a})| < \epsilon] > 1 - \frac{1}{n^2}$.

**Theorem 7** shows that the error of PCS w.r.t. GR is bounded by $\epsilon$ with probability $1 - \frac{1}{n^2}$ at a single step. The number of samples needed by PCS is $O(\log(n))$; this is a factor of $O(\log^2 n)$ less than the number of samples needed in GS for small-world networks, and a factor of $O(\frac{\log(n)}{\epsilon})$ less in general networks. Table 5 summarizes the number of samples varying different structures and delay distributions.

<table>
<thead>
<tr>
<th>Model/Algo.</th>
<th>Sample</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform/PCS</td>
<td>$p = O\left(\frac{\log(n)}{\epsilon^2}\right)$</td>
<td>$p(m + n)$</td>
</tr>
<tr>
<td>Small-world/GS</td>
<td>$p = O\left(\frac{\log(n)}{\epsilon}\right)$</td>
<td>$p(m + n \log(n))$</td>
</tr>
<tr>
<td>General/GS</td>
<td>$p = O\left(\frac{\log^2(n)}{\epsilon}\right)$</td>
<td>$p(m + n \log(n))$</td>
</tr>
</tbody>
</table>

**TABLE 5:** Samples and running times (at a single step) varying different delay distribution and structures. The error is within $\epsilon$ with probability $(1 - \frac{1}{n^2})$.

Algorithm [3] (PCS) computes TS based on estimates of number of shortest paths through each vertex. The approximation error $\epsilon$ bounds the difference between reduction by PCS and Greedy (GR) in each iteration. In each of the $k$ iterations, PCS first samples $p$ pairs of nodes from the population of all pairs (the proven approximation still holds...
when the samples are obtained before the iteration starts as is in Algorithm 2. It computes the shortest distances between the vertices of each pair in the sample and thus finds an approximate measure of number of shortest paths through each vertex. If both the vertices of a pair have delay 1, they both are present in edited graph (step 20). The edited graph is obtained by deleting the vertex with delay 0 and by adding all the edges between its neighbours. Computing BFS explores the vertices on all possible shortest paths. If one of them has delay 0, we compute BFS from the other vertex to its gateway vertex (For each such vertex v added to TS, we maintain a list of vertices, called gateway, where v.gateway = {u | d(v, u) = 0, l(u) = 1}). If both of them have delay 0, we compute BFS between each of the gateway vertices of them. We choose the pair(s) of minimum distance. The shortest paths between them explore the desired vertices (with delay 1). The vertex with the maximum ζ is chosen in each iteration. Next we explain that the overall complexity is $O(kp(m + n))$.

For steps 5, 9, 13, PCS performs BFS. The probability of picking vertices with delay 0 is $\frac{1}{d_m}$, where $x$ is number of vertices in the current TS. So, the expected time complexity for one of these steps is $p((1 - \frac{x}{d_m})(m + n) + \frac{x}{d_m}p(m + n)) = p(m + n)(1 + (g - 1)\frac{x}{d_m})$ where $g \leq \frac{xd_m}{2}$ is the maximum size of gateway list. $d_m$ is maximum degree among the vertices of the vertices in TS. Step 20 can take $O(d_m^2)$, where $d_m$ is maximum degree of vertex among the vertex and its neighbors. Step 21 takes $O(x^2d_m^2)$ as it accumulates vertices from gateway of the neighbors. So, the running time for one step when the TS has $x$ vertices is $O(p(m + n)(1 + \frac{x^2}{d_m^2}) + d_m^2 + x^2d_m^2)$. As $x < k$ is small, and the running times of step 20 and 21 are not tight, the running time is dominated by $p(m + n)$. So, the overall time complexity is $O(kp(m + n))$.

4.EXPERIMENTAL RESULTS

We evaluate the quality and scalability of our algorithms in both synthetic and real-world networks. We conduct all experiments on 3.30GHz Intel cores with 30 GB RAM. All algorithms are implemented in Java.

4.1 Datasets

The real-world datasets for evaluation are listed in Table 6. The air transportation (http://www.rita.dot.gov) data consist of airline flight networks with delays at airports set according to historical flight delays due to circumstances within the airline’s control (e.g., maintenance or crew problems, aircraft cleaning, baggage loading, fueling, etc.). We consider average and total delay of flights originating from an airport in the period 01/13-09/15. Our road Traffic data is from the highway network of Los Angeles, CA [31], where the delay at an intersection is defined as the scaled inverse of the observed speed at a given point in time (1500 * 1/speed). According to this definition the delay values range between 15 and 80 (similar to that of the original speeds). The Twitter dataset is a social network in which edges correspond to follower relationships among users. We disregard the direction of edges for our analysis. Node delays in this network represent the average inter-arrival time between posts on a given topic. We experiment with different topics described in [32]. The vertices in the DBLP network are authors and the edges represent co-authorship on at least one paper. For DBLP, we assign delays randomly, with values uniformly distributed in multiples of ten between 10 to 100. Our goal is to evaluate the scalability of our algorithms on a large real-world network structure.

4.2 Quality of sampling compared to Greedy

We report the number of samples in PCS and GS as $c \cdot \log n$, where $c$ is related to the expected error $\epsilon$ in Thms. 4 and 7. Unless stated otherwise, we use $c = 10$. 

![Diagram](https://via.placeholder.com/150)
First, we compare our sampling schemes GS and PCS with Greedy (GR) in order to evaluate the effect of sampling on quality, which we theoretically analyze in Theorems. \[4\] \[7\] To enable the comparison, we use small real datasets due to the limited scalability of GR. The quality of the compared algorithms is quantified as the Relative Reduction (RR) of SPD, while efficiency—in terms of wall-clock time. We use 3.5\(\log n\) samples for GS and PCS in these experiments.

In all experiments, our sampling schemes achieve similar quality as that of Greedy (GR), while taking close to two orders of magnitude less time. In the uniform model, the difference in quality between our sampling scheme PCS and GR does not exceed .05% in the traffic dataset (Figure 4b), while PCS takes only 2% of the time taken by GR (Figure 4a). This trend persists in the case of the general delay model for which we employ our sampling-based Greedy (GS). We compare GS and GR on multiple snapshots of the Traffic dataset and report average completion times and quality in Figures 4c, 4d. GS is 200 times faster than GR and its solution’s RR is only 0.1% worse than that of GR Figures 4c, 4d.

Some non-central airports in the AA and DA networks have significant average delays and hence disregarding the network position results in a 10-fold worse quality of the High-Delay baseline. All other baselines do not exceed the quality of GS for varying budgets and airlines.

### 4.3 Comparison to baselines

Next we evaluate the performance of our algorithms in comparison to alternatives. We consider several baseline methods, listed in Table 8, along with their theoretical running times and those of our algorithms. Some baselines select TS vertices based on local properties: degree (Deg-Cen) or delay (High-Delay); while others—based on the product of global path centrality and delay (Path-Cen and It-Path-Cen [9]). It-Path-Cen updates the number of shortest paths through a vertex after each selection of a target vertex.

Fig. 5a presents the RR of competing techniques on the Traffic network with uniform delays using 50\(\log (n)\) samples for PCS. The baseline algorithm It-Path-Cen for the setting of uniform delays is equivalent to the exhaustive greedy GR and this comparison is already available in Fig. 4a. On this relatively small network, PCS produces at least 6% better RR than the best alternative Path-Cen. Note, that in this setting simple alternatives such as Random and Deg-Cen, although fast, have unacceptably low quality.

Next we associate the delays (general model) at road intersections (nodes) measured at different times, and compare with competing techniques. As the results on different snapshots are similar, we show a representative figure on quality (fig. 5b). Using 10\(\log (n)\) samples, GS produces higher RR than both Path-Cen and It-Path-Cen, with up to 1 and 2 orders of magnitude running time improvement respectively (plots omitted due to space constraint). Unlike It-Path-Cen, GS does not target nodes only based on the number of shortest paths through them, but estimates the improvement of nodes given those already in the target set and achieves a better quality.

In larger graphs computing the exact quality (reduction of SPD) has high computational cost as it requires computing all-pair shortest paths. Hence, in order to evaluate the competing techniques, we estimate RR based on a representative sample of pairwise shortest path lengths. We randomly sample 1000 pairs 10 times and average the quality results. We evaluate the competing techniques on DBLP and the Twitter datasets.

First we evaluate the running time in comparison to the best-quality competing techniques in Tab. 9. As expected based on their theoretical complexity, Path-Cen and It-Path-Cen [9] do not scale well for large datasets. Our algorithms...
TABLE 9: Running time comparison of our algorithms and those proposed in [9] (budget = 5).

<table>
<thead>
<tr>
<th>Data</th>
<th>Algo.</th>
<th>#Sample</th>
<th>Time(min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLP-100K (unif.)</td>
<td>PCS</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>Twitter-50K (gen.)</td>
<td>Deg-Cen</td>
<td>36</td>
<td>15</td>
</tr>
</tbody>
</table>

TABLE 10: Running times of PCS and GS(4T) with k = 5.

In Twitter (budget k = 5). In DBLP, Deg-Cen has similar quality as PCS since authors of high degree tend to be central.

The only parameter in our techniques is the number of samples which provides a natural trade-off between running time and quality. Our analysis shows that we usually need only small fraction of sampled pairs to match the performance in greedy in both real-world and synthetic data. Details of this analysis are available in the Appendix.

4.4 Experiments related to Theorem 3

Comparison of PCS and Upper Bound from Theorem 3 on “Long” Paths: Theorem 3 compares the performance of a random algorithm against the optimal algorithm for the restricted metric of sum of “long” paths (length ≥ k’ = c)n. Based on the theorem, k′ R′ rand(kb) ≥ R′ opt(k).

We can, thus, compare PCS(k) (PCS with budget k) against k′ R′ rand(kb) as a proxy for comparing against R′ opt(k).

Since the constant b is not known, we vary it to evaluate the quality of PCS(k).

We experiment with two different settings for kb (100, 200) in a 2,000-vertex subgraph of the DBLP data. We assume that the path length threshold and the budget are the same (i.e., k’ = k) and vary over the range 3–5. As in earlier plots, we compute relative reductions for the methods. Since k′ R′ rand(kb) is only an estimate, if this quantity exceeds the original value of the metric, we set it to the original value and its relative reduction to 100%. For simplicity, we refer to this quantity as UB.

RR* denotes relative reduction in the sum of “long” paths. Fig. 7a and 7b present UB and the relative reduction (RR*) for PCS. RR* by PCS is within 50% of UB. (As kb is unknown, PCS may in fact occasionally produce a higher RR* than UB, Fig. 7a for k’ = 3). Increasing the length threshold (k’ reduces the difference between UB and PCS.

Finally, we explore the reduction by PCS in the sum of long versus short paths. Fig. 7c compares the reduction: RR for the actual metric (sum of all paths) and RR* for the restricted metric (sum of paths of length ≥ k’). There is a higher chance for long paths to contain upgraded vertices.

As expected, the figure shows higher reduction when the metric includes only longer paths.

4.5 Effect of parameters

The most important parameter for our sampling schemes is the number of samples. In Figs. 7d and 7e we present the variation in quality and performance of GS with the increase of number of samples on the Twitter-Celeb data.
The running time of GS grows linearly with the sample constant \( c \), while the quality increases and then saturates confirming our main premise that not all SP need to be observed to make a good-quality selection in our problem.

We also study the effect of different inputs on our algorithms on synthetic and real networks and their scalability with the number of samples and for increasing budget. The delays in the general model are randomly distributed in \([500, 1000]\). Fig. 8a shows the scalability of our methods for increasing network size (Barabasi graphs, growth parameter 3). As expected, PCS scales better than GS (on networks with 0.1 million vertices PCS is 30 times faster), and both scale significantly better than the non-sampling alternative GR. This experimentally confirms the theoretical running times of Alg. 1, 2 and 3.

Figs. 8b and 8c present the running time for increasing \( c \) and budget=10 for PCS and GS respectively. Note that the number of samples used by our techniques is controlled by \( c \) #samples=\( c \cdot \log(n) \). As expected, GS and PCS scale linearly with \( c \). The same behaviour persists for increasing budget in Figs. 8a and 8c (\( c = 15 \)). These results also confirm the expected theoretical running time behavior.

5 Previous Work

There is a considerable amount of research in network design targeting various objectives and addressing problems on augmenting the network structure as well as modifying node and edge attributes. So, the whole set of problems differ in upgrade models and objective functions. In this section, we mainly focus on those problems which address optimizing different metrics related to shortest paths.

Network design: Paik et al. [11] first introduced a set of design problems in which vertex upgrades improve the delays of adjacent edges. Later, Krumke et al. [12] generalized this model assuming varying costs for vertex/edge upgrades and proposed to minimize the cost of the minimum spanning tree. Lin et al. [3] also proposed a delay minimization problem with weights associated with undirected edges. The authors in [33] proposed a stochastic formulation of minimizing shortest paths by adding edges. The above formulations are different from ours as in our case delays are associated with vertices.

The problems considered in Dilkina et al. [9] correspond to a variation of DMP and are closer to our setting. Note that they do not consider a candidate set \( \Gamma \), which is important in real-life scenarios as one might not have the opportunity to upgrade all nodes. Moreover, we show in our comparative evaluation that our methods are superior in both scalability and quality (Section 4.3).

Structural network design: Delay minimization and other global objectives (vertex eccentricity, diameter, all-pairs shortest paths etc.) have been previously addressed by edge addition [10], [34], [35], [36], [37]. Meyerson et al. [10] designed approximation algorithms for single source and all pair SP minimization. Demaine et al. [36] minimize a network diameter and node eccentricity by adding shortcut edges with a constant factor approximation algorithm. Prior work also considers eccentricity minimization in a composite network where a social node connectivity is improved by additional communication network edges [37]. All the problems, however, are based on adding new edges i.e., structural modification, and hence are complementary to our setting. In different applications, node-based and edge-based schemes could be adopted individually or in unison.

Centrality: Other related problems involve efficient computation of betweenness centrality. In [38], the authors compute top \( k \) nodes based on betweenness centrality via sampling. The group betweenness problem has been solved in almost linear time by Yoshida et al. [29] by a high quality probabilistic approximation algorithm.

6 Conclusions

In this paper, we studied and proposed solutions for the network design problem of node delay minimization. The problem has diverse applications in a variety of domains including social, collaboration, transportation and communication networks. We proved that the problem is NP-hard even for initial equal node delays. We also showed that the problem with equal delays is even APX-hard and cannot be approximated a factor greater than \( (1 - \frac{1}{e}) \). However, approximation guarantees for a restricted formulation via randomized schemes based on VC dimension theory was obtained. We proposed and evaluated high-quality methods for the problem based on sampling that scale to large million-node instances and consistently outperform existing alternatives. We evaluated our approaches on several real-world graphs from different genres. We achieved up to 2 orders of magnitude speed-up compared to alternatives from the literature on moderate size networks, and obtained high-quality results in minutes on large datasets while competitors from the literature require more than 4 hours.

Acknowledgments

Research was sponsored by the Army Research Laboratory and accomplished under Cooperative Agreement Number

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2792470, IEEE Transactions on Knowledge and Data Engineering
We formulate the DMP problem as a mixed integer program (MIP), in order to obtain exact solutions in both the general and uniform model. We use a multi-commodity flow formulation (similar to [9]) to compute the shortest path delay between a node pair \( p \in \Omega \). An undirected graph \( G \) is transformed into a directed graph \( G' \) as follows: a node \( v \) is replaced by two nodes \( v^- \) and \( v^+ \) with two additional parallel edges from \( v^- \) to \( v^+ \) with delays \( l(v) \) (original node edge, \( e_v \)) and 0 (upgraded node edge \( e'_v \)) respectively. If an edge \((u,v)\) is present in the original graph, there are two edges \((u^+,v^-)\) and \((v^+,u^-)\) with delays 0 in \( G' \). The variables used in the formulation are as follows: (1) \( x_{v} \): a flag for whether node \( v \) is to be upgraded, (2) \( d(p) \): the effective shortest path length delay between the nodes in pair \( p \), (3) \( \text{budget} \): the total number of upgraded nodes, (4) \( f_{pe} \): continuous variable indicating whether edge \( e \) is chosen to be on the shortest path for the pair \( p \), (5) \( f_{pv} \): the flow of the commodity \( p \) on edge \( e_v \), (6) \( f_{pv}' \): continuous variable that indicates the flow of the commodity \( p \) on edge \( e'_v \).

In an integral solution, \( f_{pv} \) and \( f_{pe} \) denote whether the original node and upgraded node respectively are chosen to be on the shortest path between the pair \( p \) in an integral solution. We use \( \delta^- (v^-) \) and \( \delta^+ (v^+) \) to denote the set of incoming and outgoing edges respectively. The full MIP formulation is as follows:

\[
\min \sum_{p \in \Omega} d(p) \tag{3}
\]

\[
s.t. f_{ps} + f'_{ps} = 1, f_{pt} + f'_{pt} = 1 \quad \forall p = (s,t) \in \Omega \tag{4}
\]

\[
\sum_{e \in \delta^- (s^-)} f_{pe} = 0 \quad \forall p = (s,t) \in \Omega \tag{5}
\]

\[
\sum_{e \in \delta^+ (s^+)} f_{pe} = f_{ps} + f'_{ps} \quad \forall p = (s,t) \in \Omega \tag{6}
\]

\[
\sum_{e \in \delta^- (t^-)} f_{pe} = f_{pt} + f'_{pt} \quad \forall p = (s,t) \in \Omega \tag{7}
\]

\[
\sum_{e \in \delta^+ (t^+)} f_{pe} = 0 \quad \forall p = (s,t) \in \Omega \tag{8}
\]

\[
\sum_{e \in \delta^+ (v^+)} f_{pe} = f_{pv} + f'_{pv} \quad \forall p = (s,t) \in \Omega, \forall v \neq s,t \in V \tag{9}
\]

The MIP formulation for DMP is shown in Eqs. (3 – 16). The constraints as Eqs. (4 – 10) are used to model the shortest path delay of each terminal pair as multicommodity flow. Constraints (4 – 8) enforce the nodes \( s \) and \( t \) to be the source and sink respectively with one unit of flow in each terminal pair \((s,t)\). The next two constraints (9, 10) ensure the flow conservation through the rest of the nodes. Constraints (11 – 12) enforce that the upgraded node edge \( e'_v \) will carry the flow instead of the original node edge \( e_v \) when the node \( v \) is upgraded. Similarly, the original node edge \( e_v \) carries the flow when the node \( v \) is not upgraded. Constraint (13) computes the total delay. Constraint (14) computes the total budget and sets the maximum as \( k \). Constraints 15 and 16 ensure that upgrade decision variables and flow variables are binary and non-negative respectively.

**Experimental results:** We implement MIP using CPLEX and validate on the traffic data (see Tbl. 5). In these experiments we use 100 terminal pairs. We compare the full MIP running time to that of our sampling techniques and to MIP_100 and MIP_30m: MIP versions for which we restrict the computation in CPLEX to 100 times that of our algorithm and also 30 minutes respectively. The results in Figure 8a show that our sampling algorithms (PCS and GS) obtain solutions much closer to that of the optimal than time-restricted MIP. We find the similar trend in synthetic data too. Note restricted-time MIP has significant quality gaps (up to 9%) compared to the optimal solutions unlike our algorithms which are persistently within 2% of the optimal. Our algorithms also are much faster than the restricted MIP versions.

<table>
<thead>
<tr>
<th>Uniform</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>PCS</td>
</tr>
<tr>
<td>3</td>
<td>1.1</td>
</tr>
<tr>
<td>10</td>
<td>1.7</td>
</tr>
<tr>
<td>15</td>
<td>2.3</td>
</tr>
</tbody>
</table>

**TABLE 11:** Running times (in seconds) of the different versions of MIPs and our algorithms.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2792470, IEEE Transactions on Knowledge and Data Engineering

TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING

Fig. 9: Comparison of MIPs on Traffic: (a) PCS in the Uniform Model; and (b) GS in the General Model.

REFERENCES


