Towards Scalable Network Delay Minimization

Sourav Medya
University of California, Santa Barbara
medya@cs.ucsb.edu

Petko Bogdanov
University at Albany - SUNY
pbogdanov@albany.edu

Ambuj Singh
University of California, Santa Barbara
ambuj@cs.ucsb.edu

***Abstract*—Reduction of end-to-end network delays 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 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 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.**

In this paper, we consider the problem of network delay minimization via node upgrades. Although the problem is NP-hard, we show that probabilistic approximation for a restricted version can be obtained. We design scalable and high-quality techniques for the general setting based on sampling that are targeted to different models of delay distribution. Our methods scale almost linearly with the graph size and consistently outperform competitors in quality.

I. Introduction

Given a communication network, how can one minimize the end-to-end communication delay by upgrading networking devices? 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, 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 [7], [8], [9], [10], [3], [5]. In this paper we address a problem from the above category, namely, minimizing the overall end-to-end network delay.

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 edges [11], [10], [3], [6]. Less attention has been devoted to the complementary, but algorithmically non-equivalent setting in which the propagation capabilities of nodes are "upgraded" under budget [9]. In this paper, we consider the node-version of the delay minimization problem.

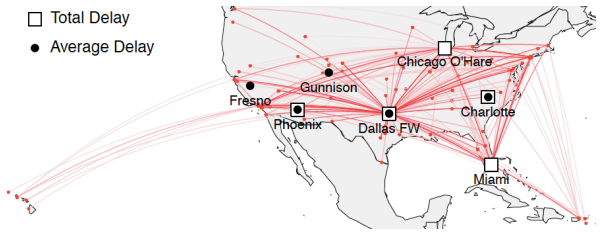


Fig. 1: Airports with maximum impact on the overall network delay in the *American Airlines* network as discovered by our methods. If airline-caused delays are removed in these airports, the overall network delay decreases by 96% and 55% when the accumulated airline-caused flight delays (*Total*) and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

Depending on the domain, low end-to-end delay enables improved information flow in social and collaboration networks [12], reduced travel time for airline and road networks [13] and increased throughput for communication networks [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[1].

Another important application comes from 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 [15]. A

[1]Extended discussion of the findings in this data is available in the Experimental section and in the extended version of this paper [14].

social media strategist of an election campaign, for example, would be interested in recruiting social network users who can re-post relevant 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 a diffusion process (i.e., using all possible paths) [16], multiple recent approaches (including the current work) focus on the most probable (shortest) paths in order to allow scalable solutions [17], [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 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 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.

Our contributions in this paper include:

- We consider the node delay minimization problem and show that it is NP-hard even for uniform delays. We develop an approximation guarantee using VC dimension theory and analyze the complexity for a restricted problem formulation.
- We propose high-quality sampling-based algorithms that scale almost linearly with the network size. In million-node networks we obtain high-quality solutions within one hour, while non-trivial alternatives are infeasible.

II. Problem Definition

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 $L : V \rightarrow \mathbb{R}_{\geq 0}$ 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 $l(P_{s,t}) = l(v_s) + \Sigma_{i=1}^{r} l(v_i)$. 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 delay. The shortest 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 $s \in V$. We define the *all pair shortest path delays (SPD)* as the sum of shortest path lengths between all pairs of vertices, i.e., $SPD(G) = \Sigma_{s,t \in V} d(s, t)$.

The DMP asks for a subset of vertices whose upgrade (delay reduction) minimizes the overall *SPD*. In the process, the delay of a fixed (small) number of vertices $T \subset V$ is reduced to 0[2]. We call this subset T a *Target Set (TS)* and its size $|T| = k$, the budget. The upgrade of the TS, T results in reduction of 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 $\Sigma_{s,t \in V} d(s, t|T)$.

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

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

The proofs of all the theorems (and lemmas) in the paper are presented in the extended version [14].

Complexity: We consider two different models for the distribution of the delays in a network and characterize the problem complexity. Under the *general model*, node delays can be arbitrary non-negative values, while the *uniform model* assumes equal delays (for simplicity, delay of 1) on all nodes. We show that DMP is NP-hard in the special case of the *uniform model*, and hence it is in the same complexity class as the *general model*. To show this hardness result we reduce the Set Cover problem to our problem. However, for restricted network structures such as trees, finding an optimal TS takes polynomial time.

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

Theorem 1 establishes that the problem is NP-hard. However, finding an optimal TS in trees takes polynomial time 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 a simple greedy algorithm (formally defined as Algorithm 1) to produce an optimal TS of size k.

Approximability: Since DMP is NP-hard, we explore the existence of approximations with guarantees. The underlying objective function in DMP is monotone as the SPD reduces after each upgrade. However, it does not have the submodular property. We next show an approximation for "long" paths of delay ϵn or higher, where $|V| = n$ and $0 < \epsilon < 1$. Our optimization objective for "long" paths is $SPD^{\epsilon}(G) = \Sigma_{s,t \in V, d(s,t) \geq \epsilon n} d(s, t)$. Let $R^{\epsilon}_{opt}(k)$ represent the reduction in SPD^{ϵ} by the optimal TS of size k and $R^{\epsilon}_{rand}(kb)$, the reduction due to kb *randomly chosen* vertices. The relationship between k, b and ϵ is captured in the following theorem. The details are discussed in the extended version of this paper [14].

Theorem 2. *Given a confidence parameter δ, $\frac{R^{\epsilon}_{opt}(k)}{R^{\epsilon}_{rand}(kb)} \leq k$ with probability δ, where $kb = (\frac{2}{\epsilon} log \frac{2}{\epsilon} + \frac{1}{\epsilon} log \frac{1}{\delta})$.*

III. Algorithms

We present a greedy approach for DMP that consecutively 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 1) 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 approximation algorithms for different delay models.

Algorithm 1: Greedy (GR)

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

A. 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 selecting 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: $RS(S) = \Sigma_{s,t \in V} d(s,t) - \Sigma_{s,t \in V} d(s,t|S)$.

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

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

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 $RS(v|S)$.

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

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. GR.

B. 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 $\mu = \frac{\Sigma_{s,t \in V} d(s,t|S)}{n(n-1)}$. The vertex whose inclusion in TS optimizes this estimate is chosen in each step of GS.

Lemma 2. *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 (μ): $E[\frac{1}{p}\sum_{i=1}^{p} X_i] = \mu$ where X_i represents the distance between the i-th pair of vertices in the sample.*

We employ Hoeffding's inequality [19] to bound the error produced by our sampling method in a single greedy step. The requirement for the applicability of Hoeffiding'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. [20] to estimate the group betweenness of vertices. In what follows, we demonstrate that the estimate has low error with high probability requiring only small number of samples. Furthermore, we show the same quality guarantee with even smaller number of samples in small-world networks.

Theorem 3. *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:*

$$Pr[|RR(v_g|S) - RR(v_a|S)| < \epsilon] > 1 - \frac{1}{n^2},$$

where p is $O(\frac{c^2 log n}{\epsilon^2})$, $c = \frac{diam}{l_{min}}$, $diam$ and l_{min} are the diameter and minimum delay respectively.

As mentioned before, the proof is in the extended ver-

Algorithm 2: Greedy with Sampling (GS)

Require: Network $G = (V, E, L)$, Approximation error ϵ, Sampling factor c, Budget k
Ensure: A subset of k nodes, Target Set
Choose $p = O(clogn/\epsilon^2)$ pairs of vertices in P
$T \leftarrow \Phi$
while $|T| \leq k$ **do**
  for $(s,t) \in P$ **do**
    Compute $d(s,t'|T)$ and $s.target[t'] \leftarrow d(s,t'|T)\ \forall t' \in V$
    Compute $d(s',t|T)$ and $t.source[s'] \leftarrow d(s',t|T)\ \forall s' \in V$
  end for
  for $v' \in V$ **do**
    if $l(v') > 0$ **then**
      $R_{v'} \leftarrow \Sigma_{(s,t)\in P} d(s,t|T) - \Sigma_{(s,t)\in P} d(s,t|T \cup \{v'\})$
    end if
  end for
  $v \leftarrow max_{v' \in V}\{R_{v'}\}$
  $l(v) \leftarrow 0$ and $T \leftarrow T \cup \{v\}$
end while
Return T

sion [14]. Note that, in the theorem, we assume $l_{min} > 0$ without loss of generality. If $l_{min} = 0$, one can delete any node of zero delay, add all possible edges among its neighbours and consider the resulting network as an input. For small-world networks (where the diameter is $\leq l_{max}logn$), a property exhibited in many domains, we show that the number of samples needed to obtain the same quality is much smaller.

Corollary 4. *Given a small-world network in which* $diam \leq l_{max}logn$*, the error of GS using* $p = O(\frac{log^3 n}{\epsilon^2})$ *samples can be bounded as:*

$$Pr[|RR(v_g|S) - RR(v_a|S)| < \epsilon] > 1 - \frac{1}{n^2}$$

GS (Alg. 2) takes as input a network G, a target approximation error ϵ, 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, ϵ, 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 ϵ, 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. IV). The running time of GS is $O(kpnlogn)$ and is dominated by the computation of shortest paths (for details of the algorithm and its complexity see the extended version [14]).

C. 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 higher-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 to 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 ζ^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 5. *In the uniform model, for a given set* S *and* $v \notin S$, $RS(v|S) = \zeta^v + (n-1)$.

With the above result, a greedy algorithm only needs to know the values of ζ for each vertex. The main bottleneck of computing ζ involves shortest path computation between all pairs of vertices. We address this complexity by a different sampling scheme. We estimate ζ 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 3. *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 6. *Given a sample* $P, |P| = p = O(\frac{logn}{\epsilon^2})$*, if* v_g *and* v_a *are the vertices chosen by GR and PCS respectively, then*

$$Pr[|RR(v_g|S) - RR(v_a|S)| < \epsilon] > 1 - \frac{1}{n^2}.$$

Thm. 6 shows that the error of PCS w.r.t. GR is bounded by ϵ with probability $1 - \frac{1}{n^2}$ at a single step. The number of samples needed by PCS is $O(\frac{log(n)}{\epsilon})$; 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 $O(\frac{diam^2}{l_{min}^2})$ less than that in general networks.

Algorithm PCS computes TS based on the estimates of number of shortest paths through each vertex. The overall complexity is $O(kp(m+n))$. Details of the algorithm and its running time are available in the extended version [14].

IV. Experimental Results

A. Datasets

The real-world datasets[3] for evaluation are listed in Table I. 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 *Traffic* data is from the highway network of Los Angeles, CA [21], 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). For the Twitter dataset, we disregard the

[3]The code is available: http://www.cs.ucsb.edu/~dbl/software.php

direction of edges. Node delays in this network represent the average inter-arrival time between posts on a given topic. We experiment with different topics described in [22]. 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.

name	value	$\|V\|$	$\|E\|$
Traffic	inverse speed	2K	6K
Twitter-Celeb	posting delay	28K	240K
Twitter-Politics	posting delay	100K	7.4M
Twitter-Science	posting delay	100K	3.3M
DBLP	random	1.1M	5M

TABLE I: Dataset description and statistics.

B. Comparison to baselines

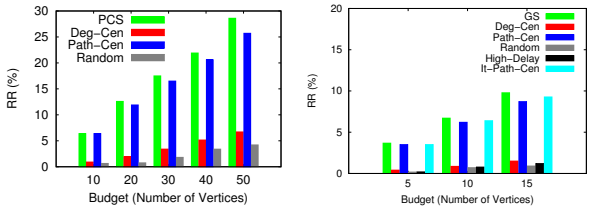
We evaluate the performance of our algorithms in comparison to alternatives. Some baselines select TS vertices based on local properties: degree (Deg-Cen) or delay (High-Delay); while others—based on the product of global path centrality and delay (Path-Cen and It-Path-Cen [9]). It-Path-Cen updates the number of shortest paths through a vertex after each selection of a target vertex.

Fig. 3a presents the RR of competing techniques on the Traffic network with uniform delays using $50log(n)$ samples for PCS. 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. 3b). Using $10log(n)$ 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. II. As expected based on their theoretical complexity, Path-Cen and It-Path-Cen [9] do not scale well for large datasets. Our algorithms complete in at most 36 min, while the alternatives take close to or more than 5h on the same input (DNF stands for "does not finish in 5 hours"). Twitter-50K in this experiment is a subgraph of the Twitter-Politics network involving 50K nodes, while in the uniform-delay setting we evaluate PCS on a subgraph of DBLP of 100K nodes (DBLP-100K).



(a) Traffic-Uniform (b) Traffic-Delay

Fig. 3: Comparison of baselines on Traffic: (a) PCS in the *Uniform Model*; and (b) GS in the *General Model*.

Data	PCS	GS	Path-Cen	It-Path-Cen
DBLP-100K (unif.)	2 m	–	4.5 h	DNF
Twitter-50K (gen.)	–	36 m	DNF	DNF

TABLE II: Running time comparison of our algorithms and those proposed in [9] (budget = 5).

Since the methods by Dilkina et al. [9] do not scale for large graphs, we compare the quality of our sampling schemes with that of Deg-Cen and High-Delay on the full large-graph datasets (High-Delay is replaced by Random in the uniform model experiments as delays in this setting are equal). To enable even higher scalability for GS, we use multi-threading with 4 threads to compute the shortest paths (steps $4-7$ in Alg. 2). For the rest of the experiments, we use GS(4T).

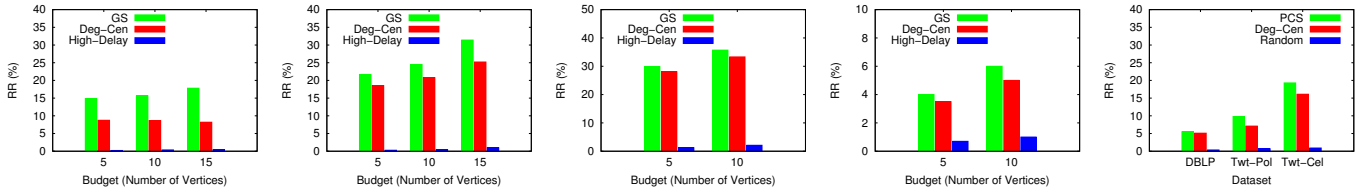
Tab. III presents the running times of our algorithms in both the uniform and general delay settings together with the number of sampled pairs of each run over the full networks. The number of samples $clog(n)$ depends on both the size of the network and the constant c (which we set to values not exceeding 20). In the uniform scenario (datasets denoted *unif.*), we assume delay 1 associated with nodes. PCS completes in the order of minutes in uniform-delay networks and GS within 62 minutes on the largest DBLP dataset.

Figs. 2a-2d show the quality of GS in Twitter and DBLP. In all cases GS performs better than alternatives for increasing budget, since the alternatives fail to capture the dependency between upgraded nodes and are limited to local node properties. We get higher quality in Twitter-Celeb as we use relatively higher number of samples. The RR in DBLP is relatively low due to the large network size and disproportionately small budgets (5 and 10 out of $1.1M$ nodes). Fig. 2e presents an analogous comparison for uniform delay. Our technique PCS outperforms alternative in Twitter (budget $k = 5$). In DBLP, Deg-Cen has similar quality to that of PCS since authors of high degree tend to be central.

Other Experiments: The number of samples 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, experiments with Airlines data and other experiments are available in [14].

V. Previous Work

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. [23] generalized this model assuming varying costs for vertex/edge upgrades. Lin et al. [3] also proposed a delay minimization problem with weights associated with undirected edges. The problems considered



(a) Twitter-Celeb (b) Twitter-Politics (c) Twitter-Science (d) DBLP-Random (e) Uniform

Fig. 2: (a-d) General Model: Quality of GS and baselines on Twitter-Celeb, Twitter-Politics, Twitter-Science and DBLP. (e) Uniform Model: Quality of PCS and baselines for budget=5 on DBLP, Twitter-Politics and Twitter-Celeb.

Data	Algo.	#Sample	Time(min)
Twitter-Celeb (unif.)	PCS	148	2.5
Twitter-Politics (unif.)	PCS	166	3.5
DBLP (unif.)	PCS	200	9
Twitter-Celeb (gen.)	GS(4T)	148	1
Twitter-Politics (gen.)	GS(4T)	64	19
Twitter-Science (gen.)	GS(4T)	64	12
DBLP (gen.)	GS(4T)	40	62

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

in Dilkina et al. [9] are closer to our setting, in that they correspond to a general version of DMP. Delay minimization and other global objectives (vertex eccentricity, diameter, all-pairs shortest paths etc.) have been previously addressed by *edge addition* [10], [24], [25], [26], [27]. All the above problems, however, are based on adding new edges i.e., structural modification, and hence are complementary to our setting. Other related problems involve efficient computation of betweenness centrality [28], [20].

VI. 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 equal node delays. We proved approximation guarantees for a restricted formulation via randomized schemes based on VC dimension theory. 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 two 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 four hours.

VII. Acknowledgments

Research was sponsored by the Army Research Laboratory and accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. We also would like to thank Arlei Silva for helpful discussions.

References

[1] A. Gupta and J. Könemann, "Approximation algorithms for network design: A survey," *Surveys in Operations Research and Management Science*, 2011.

[2] M. E. O'Kelly and H. J. Miller, "The hub network design problem: a review and synthesis," *Journal of Transport Geography*, 1994.

[3] Y. Lin and K. Mouratidis, "Best upgrade plans for single and multiple source-destination pairs," *GeoInformatica*, 2015.

[4] Q. K. Zhu, *Power distribution network design for VLSI*. John Wiley & Sons, 2004.

[5] E. B. Khalil, B. Dilkina, and L. Song, "Scalable diffusion-aware optimization of network topology," in *KDD*, 2014.

[6] S. Saha, A. Adiga, B. A. Prakash, and A. K. S. Vullikanti, "Approximation algorithms for reducing the spectral radius to control epidemic spread," *arXiv preprint arXiv:1501.06614*, 2015.

[7] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt, "Recommendations to boost content spread in social networks," in *WWW*, 2012.

[8] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, "Gelling, and melting, large graphs by edge manipulation," in *CIKM*, 2012.

[9] B. Dilkina, K. J. Lai, and C. P. Gomes, "Upgrading shortest paths in networks," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 2011.

[10] A. Meyerson and B. Tagiku, "Minimizing avergae shortest path distances via shortcut edge addition," in *APPROX-RANDOM, I. Dinur, K.Janson, J.Noar and J. D. P. Rolim Eds, Vol. 5687. Springer*, 2009.

[11] D. Paik and S. Sahni, "Network upgrading problems," *Networks*, 1995.

[12] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *WWW*, 2009.

[13] S. AhmadBeygi, A. Cohn, Y. Guan, and P. Belobaba, "Analysis of the potential for delay propagation in passenger airline networks," *Journal of Air Transport Management*, 2008.

[14] S. Medya, P. Bogdanov, and A. Singh, "Towards scalable network delay minimization," http://arxiv.org/abs/1609.08228.

[15] B. Liu, G. Cong, D. Xu, and Y. Zeng, "Time constrained influence maximization in social networks," in *ICDM*. IEEE, 2012.

[16] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.

[17] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," in *PKDD*, 2006.

[18] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *KDD*, 2010.

[19] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, 1963.

[20] Y. Yoshida, "Almost linear-time algorithms for adaptive betweenness centrality using hypergraph sketches," in *KDD*, 2014.

[21] A. Silva, P. Bogdanov, and A. Singh, "Hierarchical in-network attribute compression via importance sampling," in *ICDE*, 2015.

[22] P. Bogdanov, M. Busch, J. Moehlis, A. K. Singh, and B. K. Szymanski, "The social media genome: Modeling individual topic-specific behavior in social media," in *ASONAM*, 2013.

[23] S. Krumke, M. Marathe, H. Noltemeier, R. Ravi, and S. Ravi, "Approximation algorithms for certain network improvement problems," *Journal of Combinatorial Optimization*, 1998.

[24] M. Papagelis, F. Bonchi, and A. Gionis, "Suggesting ghost edges for a smaller world," in *CIKM*, 2011.

[25] N. Parotisidis, E. Pitoura, and P. Tsaparas, "Selecting shortcuts for a smaller world," in *SDM*, 2015.

[26] E. D. Demaine and M. Zadimoghaddam, "Minimizing the diameter of a network using shortcut edges," *in SWAT, ser.Lecture Notes in Computer Science, H. Kaplan,Ed.*, 2010.

[27] S. Perumal, P. Basu, and Z. Guan, "Minimizing eccentricity in composite networks via constrained edge additions," in *MILCOM*, 2013.

[28] M. Riondato and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling," in *WSDM*, 2014.