Network Resilience Assessment via QoS Degradation Metrics: An Algorithmic Approach

LAN N. NGUYEN, University of Florida, USA
MY T. THAI*, University of Florida, USA

This paper focuses on network resilience to perturbation of edge weight. Other than connectivity, many network applications nowadays rely upon some measure of network distance between a pair of connected nodes. In these systems, a metric related to network functionality is associated to each edge. A pair of nodes only being functional if the weighted, shortest-path distance between the pair is below a given threshold T. Consequently, a natural question is on which degree the change of edge weights can damage the network functionality? With this motivation, we study a new problem, *Quality of Service Degradation*: given a set of pairs, find a minimum budget to increase the edge weights which ensures the distance between each pair exceeds T. We introduce four algorithms with theoretical performance guarantees for this problem. Each of them has its own strength in trade-off between effectiveness and running time, which are illustrated both in theory and comprehensive experimental evaluation.

ACM Reference Format:
Lan N. Nguyen and My T. Thai. 2019. Network Resilience Assessment via QoS Degradation Metrics: An Algorithmic Approach. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 1, Article 1 (March 2019), 32 pages. https://doi.org/10.1145/3311072

1 INTRODUCTION

Graph connectivity is considered as an important metric on measuring the functionality of a network. Typically, the connectivity-related problems usually ask for the minimum-size set of components (nodes or edges) whose removal disconnects the target set of nodes. This consideration has led to the investigation of many forms of cutting problems in a network: *e.g* the minimum cut problem, the minimum multicut problem, the sparest cut problem [43] and the most recent work, the Length-Bounded Multicut (`LB-MULTICUT`) problem [29]. In addition, various measures based on connectivity have formed the framework for assessment of network resilience to external attacks [19–22, 25, 35–37, 40–42].

However, many network applications now consider other factors when determining a network functionality in addition to connectivity. For example, in Bitcoin network, to guarantee synchronization, not only the network connectivity is required but a network is also configured in order to ensure the broadcasting time of transaction messages under several seconds [11]. As another example, consider a time-sensitive delivery on a road network, where edge weights represents the travel time between destinations. Connectivity between a source and a destination is insufficient when a guarantee on the delivery time is required.

*My T. Thai is the corresponding author.

Authors' addresses: Lan N. Nguyen, University of Florida, USA, lan.nguyen@ufl.edu; My T. Thai, University of Florida, USA, mythai@cise.ufl.edu.

2476-1249/2019/3-ART1 $15.00
https://doi.org/10.1145/3311072

Therefore, a natural question is whether a tech-savvy attacker can damage the network functionality without impacting the connectivity? Under various forms, this kind of attacks actually is common, yet stealthy. For example, in the I-SIG system, real-time vehicle trajectory data transmitted using the CV technology are used to intelligently control the duration and sequence of traffic signals [4–6, 14, 15, 28, 34]. An adversary, therefore, can compromise multiple vehicles and send malicious messages with false data (e.g., speed and location) to the I-SIG system to impact the traffic control decisions. As reported by previous works, it has been shown that even one single attack vehicle can manipulate the intelligent traffic control algorithm in the I-SIG system and cause severe traffic jams [4, 15]. To understand the severity of such attack, it is necessary to study on which roads the attackers can target to and what is the minimum number of vehicles the attackers have to compromise to cause large-scale congestions, e.g. traveling from two certain locations takes several hours longer than usual. Such attack can be for political or financial purposes, e.g. blocking traffics of business competitors [4].

As another example, in Bitcoin network or any Blockchain-based applications, an attacker can target to damage the consensus between copies of public ledger of major miners by delaying block propagation between them. Recent works [11] have shown that after receiving request for a block information from another node, a Bitcoin node can have up to 20 minutes to respond. An attacker, therefore, can flood the Bitcoin nodes with too many requests or "dust" messages to handle, thus delay their block delivery. By flooding multiple nodes, the attacker can disrupt miners to reach consensus on a certain state of Blockchain. The impact of this attack varies relying upon the victims. If the victim is a merchant, it is vulnerable to double spending attacks [1]. If the victim is a miner, the attack wastes its computational power [38]. If the victim is a regular node, it will have an outdated view of the Blockchain, and thus more vulnerable to the temporal attacks which exploit the lagging in Blockchain synchronization [18, 38]. Therefore, it is necessary to study which nodes are critical and how the attacker should attack such nodes (e.g. how much bandwidth consumption) to impact the Bitcoin network functionality, e.g. causing major miners several hours to reach consensus.

With this motivation, we consider the *Quality of Service Degradation* (`QoSD`) problem. Given a directed graph G representing a network, threshold T and set of pairs S in G, the objective is to identify a minimum budget to increase the edge (or node) weights to ensure the weighted, shortest-path distance between each pair in S is no smaller than T. Intuitively, the goal of this problem is to assess how robust the network is; the greater budget to increase edge weights found, the more resilient the network is to the perturbation in terms of edge weights. In addition, the budget to increase weight of a edge in the solution provides an indication of the importance of this edge to the desired functionality.

In the context of network reliability, Kuhnle et al. [29] have recently studied a special case of our problem under the name `LB-MULTICUT`. Different to our problem, the objective of this problem is to identify a minimum set of edges whose **removal** ensures the distance between each pair of nodes is no smaller than T. Directly adopting the `LB-MULTICUT` solutions to our `QoSD` problem is not feasible since most of those solutions exploited a trait that their problems can be formulated by Integer Programming and exhibit submodular behaviors. `QoSD` problem, on the other hand, is shown to be neither submodular nor supermodular, making `QoSD` more challenging to devise an efficient algorithm. Also, modern networked systems are increasingly massive in scale, often with size of millions of vertices and edges. The need for a scalable algorithm on large-scale networks poses another challenge for our problem. Motivated by these observations, the main contribution of this work are as follows.

- We provide three highly scalable algorithms for our problem: Two iterative algorithms, `IG` and `AT`, with approximation ratio $O(\gamma^{-1}(\ln \mathsf{T} + \mathsf{h} \ln n))$ and $O(\ln \mathsf{T} + \mathsf{h} \ln n)$ respectively, where γ is

a metric measuring the concave property of edge weight functions w.r.t a budget to increase edge weights, h is the maximum number of edges of a path connecting between a pair in S, and n is the number of nodes in G; and SA, a probabilistic approximation algorithm returning $O(\frac{\ln \mathsf{T}+\mathsf{h}\ln \mathsf{d}}{\gamma(1-e^{-\gamma})(1-\epsilon)})$ approximation result with high probability, where d is the maximum degree of G.
- When the edge weight functions are linear w.r.t the cost to increase edge weight, we propose LR, a randomized rounding algorithm based on LP relaxation of the problem. LR provides $O(\mathsf{h}\ln n)$ approximation guarantee.
- We extensively evaluate our algorithms on both synthetic networks and large-scale, real-world networks. All of our four algorithms are demonstrated to scale to networks with millions of nodes and edges in under a few hours and return nearly optimal solutions. Also, the experiments show the trade-off between our proposed algorithms in terms of runtime and quality of solution.

Organization. The rest of this paper is organized as follows. Section 2 reviews literatures related to our problem. In Section 3, we formally define the problem and discuss its challenges. The four solutions, IG, AT, SA and LR, are presented in Section 4, 5, 6 and 7, respectively. In Section 8, we evaluate our algorithms, comparing to heuristic methods for the general case and to algorithms in [29] for the special case. Finally, Section 9 concludes the paper.

2 RELATED WORKS

Relationship with Kuhnle et al. [29] Kuhnle et al. has studied the Length-Bounded Multicut Problem (LB-MULTICUT). The objective of this problem is to identify a minimum set of edges whose **removal** ensures the distance between each pair of nodes of a given set S is no smaller than T. LB-MULTICUT is a special case of QoSD where we restrict to two conditions: 1) the only way to increase an edge weight is making the weight greater than T and 2) the cost of doing so is uniform among edges.

Our QoSD problem is more general and realistic than LB-MULTICUT, as briefly discussed earlier. In the adversarial perspective, it is impractical to remove edges out of a network structure. Taking the I-SIG system as an example, the attacker can only damage the network functionality by compromising multiple vehicles, causing severe traffic jams on road network rather than physically damaging road lines. Furthermore, on the Bitcoin-based applications, the Bitcoin protocol only allows a maximum delay of 20 minutes for any packet delivery. For any damage of a P2P connection, the protocol creates another connection to guarantee the connectivity of Bitcoin network. Thus, the LB-MULTICUT cannot be applied on those two applications.

Other than the special case, LB-MULTICUT and QoSD are fundamentally different, thus solutions to LB-MULTICUT are not readily applied to QoSD. More specifically, Kuhnle *et al.* proposed three approximation algorithms for LB-MULTICUT, which are MIA, TAG, SAP [29]. We are going to discuss the limits of these algorithms w.r.t solving QoSD.

The general idea of MIA is to find the multicut of sub-graphs of the input network such that each optimal multicut is a lower bound of the optimal solution of LB-MULTICUT instance. In this solution, the authors exploit the similarity between LB-MULTICUT and the multicut problem where cutting an edge in a single path is sufficient to disconnect this path. With the multicut solution, MIA utilizes the $O(n^{11/23})$ approximation algorithm proposed by Agarwal et al. [10]. Thus MIA's performance guarantee is bounded by $O(Mn^{11/23})$ where M is the number of considered subgraphs. Our problem does not require edge removals, so there is not clear connection with multicut. Therefore, we find it infeasible to apply MIA, even with modification, to solve our problem.

The next algorithm of LB-MULTICUT is TAG. In general, TAG is a dynamic algorithm, which uses a primal-dual solution to bound the worst-case performance under incremental graph changes and improves the solution in practice by periodic pruning. TAG utilizes the trait that cutting all edges, which are in the maximal set of disjoint paths connecting target pairs of nodes, is sufficient to disconnect those pairs. However, this solution may not be practical in our problem. Increasing weights of those edges to maximum does not guarantee the shortest paths, which connect target pairs of nodes, no smaller than the threshold T.

The SAP algorithm is a greedy, sampling-based solution with an $O(\mathsf{h}\log n)$ approximation guarantee (h is the maximum number of edges of a single path connecting a pair in S), which holds with the probability of at least $1-1/m$. Our algorithm SA is inspired by SAP in that we also use a greedy approach based on path samples, generated by using probabilistic hints based upon shortest path computations to guide the sampling. However, since our objective function is non-submodular, we prove that an approximation guarantee of SA depends on γ, where γ measures the concave property of edge weight functions. Moreover, we boost the process of obtaining a feasible solution by allowing a finite budget of at most q to be added on each step of sampling, where q can be any number. We prove that q does not impact the performance guarantee of SA.

Optimization on Integer Lattice. As there is a finite budget to increase the edge weight, we model our problem in a form of minimization problem on Integer Lattice: given a set of functions $\{f_i | f_i : (\mathbb{Z}^+ \cup \{0\})^n \rightarrow \mathbb{R}^+\}$ on the Integer Lattice, the objective is to minimize the cardinality of $\mathbf{x}$ that $f_i(\mathbf{x}) \geq \theta_i$ for all i. The optimization on the Integer Lattice has received much attention recently. However, most of those works focus on the maximization version, which asks for maximizing $f(\mathbf{x})$ under a cardinality constraint $||\mathbf{x}|| \leq k$. When f is non-submodular, those works exploits either the submodularity ratio γ_s [17], generalized curvature α [12] or the diminishing-return ratio γ_d [30, 32] to devise approximation solutions with performance guarantee in terms of those parameters. However, the fact that those parameters can be small and computationally hard to obtain on several real-world objectives raises a concern on those theoretical approximation ratios. For example, Kuhnle et al. [30] proposed a fast maximization of Non-Submodular, Monotonic Functions on the Integer Lattice with approximation ratio $(1-e^{-\gamma_d\gamma_s}-\eta)$ for any $\eta > 0$. If γ_d or γ_s is 0, this ratio will be smaller than 0. In our work, we utilize the concave property of edge weight functions to introduce the *concave ratio* γ, which we use to prove the theoretical guarantee of IG and SA, and bound the sampling size of SA. γ can be found easily from the derivative of edge weight functions or scanning through all edge weight functions with $O(m)$ time complexity. γ can be small in some cases, so we devise the AT solution from an improved IG algorithm, which discards the dependence on γ value to obtain better theoretical performance guarantee but a worse runtime in trade-off.

Classical Multicut Problem. The Multicut problem asks for the minimum number of edges (or nodes) whose removal ensures each pair in S is topologically disconnected. For the edge version in an undirected graph, an $O(\log k)$ approximation was developed by Garg et al. [24] by considering multicommodity flow. In directed graphs, Gupta [26] developed an $O(\sqrt{n})$ approximation algorithm, which was later improved to $O(n^{11/23})$ by Agarwal et al. [10]. These solutions were based on the optimal solution of the linear relaxation modeling the problem instance. Our LR algorithm was inspired by this approach but we have to deal with the challenge that a LP-optimal value of each edge could be larger than 1. Therefore, any discretization technique of the Multicut problem cannot be directly applied to our problem. We have devised a randomized rounding technique on which we can obtain a feasible solution with high probability while ensuring an $O(\mathsf{h}\log n)$ performance ratio.

3 PROBLEM FORMULATION

In this section, we formally define the *Quality of Service Degradation* (QoSD) problem in the format of cardinality minimization on the Integer Lattice and present challenges on solving QoSD.

We abstract the network using a weighted directed graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ directed edges. Each edge e is associated with a function $f_e : \mathbb{Z}^{\geq} \longrightarrow \mathbb{Z}^{+}$ which indicates the weight of e w.r.t a budget to increase weight of e. In another word, if we spend x on edge e, the weight of edge e will be $f_e(x)$. f_e is monotonically increasing.

Let b_e be the maximum possible budget to increase the weight of edge e. Denote $\mathbf{x} = \{x_1, ...x_m\}$ is a vector where x_i is the budget to increase weight of the i^{th} edge and similarly $\mathsf{b} = \{b_1, ...b_m\}$, we have $x_i \leq b_i \; \forall i \in [1, m]$. b is called *the box*. The overall budget to increase weight of all edges is denoted by $||\mathbf{x}|| = \sum_e x_e$. Let $f = \{f_1, f_2, ...f_m\}$ be a set of edge weight functions. Note that, for simplicity, the notation e is used to present an edge in E and also the index of this edge, i.e. if we write x_e, we mean the budget to increase the weight of edge e (to $f_e(x)$) and also the element in $\mathbf{x}$ that is corresponding to e. The same rule is applied with b_e, f_e. Also, if we write $e - 1$ (or $e + 1$), we indicate the edge right next to e on the left (right) in $\mathbf{x}$.

A path $p = p_0, p_1, ...p_l \in G$ is a sequence of vertices such that $(p_{i-1}, p_i) \in E$ for $i = 1, .., l$. A path can also be understood as the sequence of edges $\{(p_0, p_1), (p_1, p_2), ...(p_{k-1}, p_k)\}$. In this work, a path is used interchangeably as a sequence of edges or a sequence of nodes. A *single path* is a path containing no cycles (i.e repeated vertices). Under a budget vector $\mathbf{x}$, the length of a path p is defined as $\sum_{e \in p} f_e(x_e)$. We now formally define QoSD as follows:

Definition 1. Quality of Service Degradation (QoSD). *Given a directed graph $G = (V, E)$, a set $f = \{f_e : \mathbb{Z}^{\geq} \rightarrow \mathbb{Z}^{+}\}$ of edge weight functions, a box b and a target set $S = \{(s_1, t_1), ...(s_k, t_k)\}$, determine a minimum budget $||\mathbf{x}||$ such that under $\mathbf{x}$, the weighted, shortest-path between each pair in S exceeds a threshold T. A problem instance may be represented by the tuple $(G, f, \mathsf{b}, S, \mathsf{T})$*

For each edge $e \in E$, let $w_e = f_e(0)$ denote the initial weight of e. In this work, we assume $w_e > 0$ for all $e \in E$, which can be justified by the fact that most networks have positive costs associated with their edges, even when there is no interference from external sources (i.e., propagation delay in communication networks, processing delay in Blockchains).

Let $\mathcal{P}_i$ denote a set of simple paths connecting the pair $(s_i, t_i) \in S$ and $\sum_{e \in p} w_e < \mathsf{T}$ for all $p \in \mathcal{P}_i$. Let $\mathcal{F} = \cup_{i=1}^{k} \mathcal{P}_i$, we call a path $p \in \mathcal{F}$ a *feasible path* and $\mathcal{F}$ is a set of all feasible paths in G. Let $\mathsf{w} = \min_e w_e$, it is trivial that the number of edges of a feasible path is upper-bounded by $\lceil \frac{\mathsf{T}}{\mathsf{w}} \rceil$. Denote $\mathsf{h} = \lceil \frac{\mathsf{T}}{\mathsf{w}} \rceil$.

Under $\mathbf{x}$, given a pair of nodes (s, t), if there exists no single path p from s to t which satisfies $\sum_{e \in p} f_e(x_e) < \mathsf{T}$, we call s is separated from t or the pair (s, t) is *separated* by $\mathbf{x}$. Also, given a feasible path $p \in \mathcal{F}$, if $\sum_{e \in p} f_e(x_e) \geq \mathsf{T}$, we call p is *blocked* by $\mathbf{x}$ or $\mathbf{x}$ blocks p.

The QoSD problem can be formulated as the follows:

$$\min \quad ||\mathbf{x}|| \tag{1}$$

$$\text{s.t.} \sum_{e \in p} f_e(x_e) \geq \mathsf{T} \qquad \forall p \in \mathcal{F} \tag{2}$$

$$x_e \leq b_e \qquad \forall e \in E \tag{3}$$

$$x_e \in \mathbb{Z}^{+} \cup \{0\} \qquad \forall e \in E \tag{4}$$

Note that even $x_e \in \mathbb{Z}^{+} \cup \{0\}$, this is not an Integer Program because $f_e(x)$ may not be a linear function.

We can see this formulation as the cardinality minimization on the Integer lattice to satisfy multiple constraints. Before going further, we will look at several notations, mathematical operators on Integer lattice, which will be used along the theoretical proofs of our algorithms. Given $\mathbf{x} = \{x_1, ...x_m\}, \mathbf{y} = \{y_1, ...y_m\} \in \mathbb{Z}^m$, we have:

$$
\begin{aligned}
\mathbf{x}+\mathbf{y} &= \{x_1 + y_1, ...x_m + y_m\} \\
\mathbf{x}-\mathbf{y} &= \{x_1 - y_1, ...x_m - y_m\} \\
\mathbf{x}\wedge\mathbf{y} &= \{\min(x_1, y_1), \dots \min(x_m, y_m)\} \\
\mathbf{x}\vee\mathbf{y} &= \{\max(x_1, y_1), \dots \max(x_m, y_m)\} \\
\mathbf{x}/\mathbf{y} &= \{\max(x_1 - y_1, 0), \dots \max(x_n - y_n, 0)\} \\
c\mathbf{x} &= \{cx_1, ...cx_m\} \quad \forall c \in \mathbb{Z}
\end{aligned}
$$

Moreover, we say $\mathbf{x} \le \mathbf{y}$ if $x_i \le y_i$ for all $i \in [1, m]$, the similar rule is applied to $<, \ge, >$.

Let $\mathbf{s}_i$ be a unit vector with the same dimension with $\mathbf{x}$, $\mathbf{s}_i$ has value 1 in the i^{th} element and 0 elsewhere. Therefore, we could also write $\mathbf{x} = \sum_{i=1}^{m} x_i \mathbf{s}_i$. Table 1 summarizes all the notations we have so far.

Discussion. Given an instance of QoSD $(G, f, \mathsf{b}, S, \mathsf{T})$, the optimal solution can be obtained by formulating the problem as the following Integer Programming (IP):

$$\min \quad \sum_{e}\sum_{i=0}^{b_e} i \cdot y_{e,i} \tag{5}$$

$$\text{s.t.} \quad \sum_{i=0}^{b_e} y_{e,i} = 1 \qquad \forall e \in E \tag{6}$$

$$\sum_{e\in p}\sum_{i=0}^{b_e} f_e(i) \cdot y_{e,i} \ge \mathsf{T} \qquad \forall p \in \mathcal{F} \tag{7}$$

$$y_{e,i} \in \{0, 1\} \qquad \forall e \in E, i \in [0, b_e] \tag{8}$$

where $y_{e,i}$ is an indicator variable which is 1 if $x_e = i$ and 0 otherwise. The first constraint (Eq. 6) is to guarantee the budget to increase weight of edge e is a value in range $[0, b_e]$ and the second constraint (Eq. 7) is to ensure the length of each feasible path is at least T. However, solving this IP is extremely expensive. Not only because solving IP is NP-hard (the performance is strongly dependent on which solver is used) but also listing all the paths for the second constraint is very expensive in practice since it requires $O(m^{\mathsf{h}})$ in the worst case. Our algorithms are designed to be efficient even when G is large and hence do not require a listing of $\mathcal{F}$ or an optimal solution of the linear relaxation of this IP formulation.

Hardness and Inapproximability. Since LB-MULTICUT is a special case of QoSD, QoSD is NP-hard. Furthermore, any inapproximability result of LB-MULTICUT or the Multicut problem is also the inapproximability of QoSD. We summarize those results as follows:

- Kuhnle et al. [29] Let $\mathsf{T} \ge 16$. Unless $NP \subseteq BPP$, there is no polynomial-time algorithm to approximate QoSD within a factor of $\lfloor \frac{\mathsf{T}}{6} \rfloor - 1 - \epsilon$ for any $\epsilon > 0$.
- Lee et al. [31]: When T is fixed and initial edge weights are uniform, QoSD is inapproximable within a factor of $\Omega(\sqrt{T})$ assuming the Unique Games Conjecture.
- Chawla et al. [13]: There exist no $O(\log\log n)$-approximation algorithm for QoSD unless $P = NP$.

Node version of the problem. The node version of the QoSD problem asks for the minimum budget to increase node weights rather than edge weights in the problem definition above. All our four

Table 1. Notation

Notation	Definition
$G = (V, E)$	Input directed graph
V, E	Vertex and edge sets of G, respectively
n, m	Number of vertices, edges in G, respectively
d	The maximum degree of G
S	The set of target pairs of nodes
k	The number of pairs in target set S
T	The threshold on the path length
$f_e(x)$	The weight function of edge e w.r.t a budget x
f	The set of all weight functions of edges in G
$\mathcal{F}$	The set of all feasible paths
h	The maximum number of edges of a path in $\mathcal{F}$
q	The maximum added cost in each iteration of SA
$\mathbf{x} = \{x_1, ..x_m\}$	The budget vector, x_i is the budget on edge i
$\mathbf{s}_i$	Unit vector, 1 in the i^{th} element and 0 elsewhere
γ	The concave ratio of the function set f
α	Bias parameter in the sampling of SA
x^*	Optimal solution to the problem instance
$\mathsf{OPT} = \|\|\mathbf{x}^*\|\|$	Size of optimal solution

algorithms can be easily adapted for the node version and keep the same theoretical performance guarantees.

4 ITERATIVE SOLUTION

There are two challenging tasks to solve the QoSD problem. The first one is the number of feasible paths could be extremely large, thus we need to avoid listing all the feasible paths as discussed earlier. The second challenge is that the objective function of QoSD can be non-submodular, depending on the edge weight functions. We handle the challenges via two different algorithms: *Iterative Greedy* (IG) and *Adaptive Trading* (AT). After the discussion of IG and AT, we provide the theoretical analysis and approximation guarantee of both algorithms.

To tackle the first challenge, instead of listing all feasible paths of the network, we build a set $\mathcal{P}$ of candidate paths which is a subset of $\mathcal{F}$ but blocking all paths in $\mathcal{P}$ is sufficient to separate all pairs in S. $\mathcal{P}$ is built incrementally and iteratively. For each iteration, we find a budget vector $\mathbf{x} = \{x_1, ..x_m\}$ to block all paths in $\mathcal{P}$. Then, we set the length of an edge e to be $f_e(x_e)$. Next, we check whether $\mathbf{x}$ is sufficient to separate all pairs in S by checking whether there exists the shortest path of a certain pair in S whose length is smaller than T. If yes, then blocking all paths in $\mathcal{P}$ is not sufficient to separate all pairs in S; we add all the shortest paths of pairs whose length has not exceeded T into $\mathcal{P}$ and continue to the next iteration. If no, then $\mathbf{x}$ is sufficient to separate all pairs in S; we terminate the algorithm and return $\mathbf{x}$. The full algorithms is represented by Alg. 1.

Since the maximum number of edges of a feasible path could reach up to $\mathsf{h} = \lfloor \frac{\mathsf{T}}{\mathsf{w}} \rfloor$, the number of feasible paths of the network G is upper bounded by $O(n^{\mathsf{h}})$. Because we guarantee there should be at least a feasible path is added into $\mathcal{P}$ in each iteration (line 3 Alg. 1), the number of iterations in Alg. 1 is at most $O(n^{\mathsf{h}})$. This is a large number and comparable to the case if we tried to enumerate

Algorithm 1 Iterative Solution

```
Input $G, f$, b, T, $S$
Output QoS adjustment vector $\mathbf{x}$
1: $\mathcal{P} = \emptyset$
2: while There exists path $p \in \mathcal{F}$ whose length < T do
3:     $\mathcal{P} \leftarrow \mathcal{P} \cup$ potentialPaths($G, S, \mathbf{x}$).
4:     $\mathbf{x} = \{0\}^m$
5:     Find $\mathbf{x}$ to block all paths in $\mathcal{P}$
Return $\mathbf{x}$
```

all feasible paths. However in experiment, we found that the number of iterations is much smaller even on large and highly dense networks.

Algorithm 2 potentialPaths($G, S, \mathbf{x}$)

```
Input $G, S, \mathbf{x}$
Output Set $\mathcal{P}$ of paths whose lengths is smaller than T
1: Assign edge $e$ length is $f_e(x_e)$ $\forall$ $e \in E$
2: for each pair $(s, t) \in S$ do
3:     $p \leftarrow$ shortest path between $s$ and $t$
4:     if length of $p$ is smaller than T then
5:         $\mathcal{P} = \mathcal{P} \cup p$
Return $\mathcal{P}$
```

Lemma 4.1. *The approximation guarantee of Alg. 1 equals to the approximation guarantee of the algorithm that finds* $\mathbf{x}$ *to block all paths in* $\mathcal{P}$

Proof. Since $\mathcal{P}$ is a subset of all feasible paths in G, the optimal solution to block all feasible paths is also a feasible solution to block all paths in $\mathcal{P}$. Therefore, the optimal solution to block all paths in $\mathcal{P}$ is at most the size of the optimal solution of QoSD. Denote $\mathbf{x}^o$ and $\mathbf{x}^*$ as the optimal solutions to block paths in $\mathcal{P}$ and $\mathcal{F}$ respectively. Assume the algorithm in line 5 of Alg. 1 returns α-approximation result. We have $||\mathbf{x}|| \leq \alpha \cdot ||\mathbf{x}^o|| \leq \alpha \cdot ||\mathbf{x}^*||$. And since finally $\mathbf{x}$ is a feasible solution to our problem, then the output $\mathbf{x}$ of Alg. 1 is within α factor to optimal solution $\mathbf{x}^*$. □

Now let us discuss the the second challenge: how to block all paths in $\mathcal{P}$, line 5 of Alg. 1. To address this, we propose two algorithms, *Greedy* and *Adaptive Trading*. Before delving into the details of each algorithm, we introduce the parameter γ, which is used to measure the concave property of weight functions. γ would be utilized on performance analysis for our algorithms.

4.1 Concave property of weight functions

The concave ratio of a set of functions is defined as follows:

Definition 2. (Concave ratio) *The concave ratio of a set F of non-negative functions is the largest scalar $\gamma \in [0, 1]$ such that:*

$$f(x+1) - f(x) \geq \gamma \cdot \big(f(y+1) - f(y)\big) \tag{9}$$

For all $f \in F$ and $0 \leq x \leq y$

In our problem, the set of non-negative functions contains all weight functions of edges in G. Therefore, for simplicity, we denote γ as the *concave ratio* of these set of weight functions. Now, we will utilize γ to get several useful exploration for our solutions. First, given a path p and a vector $\mathbf{x}$, define:

$$\mathsf{r}(p, \mathbf{x}) = \min(\mathsf{T}, \sum_{e \in p} f_e(x_e)) \tag{10}$$

Let $g(\mathcal{P}, \mathbf{x})$ be an arbitrary linear combination of $\mathsf{r}(p, \mathbf{x})$ for all $p \in \mathcal{P}$. $g(\mathcal{P}, \mathbf{x})$ could be presented as follows:

$$g(\mathcal{P}, \mathbf{x}) = \sum_{p \in \mathcal{P}} \beta_p \mathsf{r}(p, \mathbf{x}) \qquad \beta_p \in \mathbb{R}^+ \ \forall\, p \in \mathcal{P} \tag{11}$$

Given a vector $\mathbf{z}$, define:

$$\Delta_{\mathbf{z}} g(\mathcal{P}, \mathbf{x}) = g(\mathcal{P}, \mathbf{x} + \mathbf{z}) - g(\mathcal{P}, \mathbf{x}) \tag{12}$$

We have the following lemma.

Lemma 4.2. *Given two budget vectors* $\mathbf{x}, \mathbf{y}$ *where* $\mathbf{x} \leq \mathbf{y}$ *and a unit vector* $\mathbf{s}$*, we have:*

$$\Delta_{\mathbf{s}} g(\mathcal{P}, \mathbf{x}) \geq \gamma \Delta_{\mathbf{s}} g(\mathcal{P}, \mathbf{y})$$

Proof overview. Without lost of generality, we assume $\mathbf{s} = \mathbf{s}_i$, a unit vector which has value 1 at the i^{th} element and 0 elsewhere. We prove that: given a feasible path p, the marginal gain of $\mathsf{r}(p, \mathbf{x})$ by $\mathbf{s}_i$ is at least γ times the marginal gain of $\mathsf{r}(p, \mathbf{y})$ by $\mathbf{s}_i$.

By definition, the $\mathsf{r}(p, \cdot)$ value of any budget vector cannot exceed T. Also, $\mathsf{r}(p, \mathbf{u}) \leq \mathsf{r}(p, \mathbf{v})$ if $\mathbf{u} \leq \mathbf{v}$. Therefore, we consider three different cases: (1) $\mathsf{r}(p, \mathbf{x}) < \mathsf{r}(p, \mathbf{x} + \mathbf{s}_i) < \mathsf{T}$; (2) $\mathsf{r}(p, \mathbf{x}) < \mathsf{r}(p, \mathbf{x} + \mathbf{s}_i) = \mathsf{T}$; and (3) $\mathsf{r}(p, \mathbf{x}) = \mathsf{r}(p, \mathbf{x} + \mathbf{s}_i) = \mathsf{T}$. All three cases guarantee $\mathsf{r}(p, \mathbf{x} + \mathbf{s}_i) - \mathsf{r}(p, \mathbf{x}) \geq \gamma\big(\mathsf{r}(p, \mathbf{y} + \mathbf{s}_i) - \mathsf{r}(p, \mathbf{y})\big)$. Since $g(\mathcal{P}, \mathbf{x})$ is a linear combination of $\mathsf{r}(p, \mathbf{x})$, the lemma follows. □

Lemma 4.3. *Given three budget vectors* $\mathbf{x}, \mathbf{y}, \mathbf{z}$ *where* $\mathbf{x} \leq \mathbf{y}$ *we have:*

$$\Delta_{\mathbf{z}} g(\mathcal{P}, \mathbf{x}) \geq \gamma \Delta_{\mathbf{z}} g(\mathcal{P}, \mathbf{y})$$

Proof. Let $\mathbf{z} = \sum_{i=1}^{||\mathbf{z}||} \mathbf{s}_i$ where $\mathbf{s}_i$ is a unit vector, we have:

$$\Delta_{\mathbf{z}} g(\mathcal{P}, \mathbf{y}) = \sum_{j=1}^{||\mathbf{z}||} \Delta_{\mathbf{s}_j} g(\mathcal{P}, \mathbf{y} + \mathbf{s}_1 + \ldots + \mathbf{s}_{j-1}) \leq \frac{1}{\gamma}\Big(\sum_{j=1}^{||\mathbf{z}||} \Delta_{\mathbf{s}_j} g(\mathcal{P}, \mathbf{x} + \mathbf{s}_1 + \ldots + \mathbf{s}_{j-1})\Big) \leq \frac{1}{\gamma} \Delta_{\mathbf{z}} g(\mathcal{P}, \mathbf{x})$$

which completes the proof. □

Note that $\mathsf{r}(p, \mathbf{x}) \leq \mathsf{T}$. A budget vector $\mathbf{x}$ is sufficient to block all paths in $\mathcal{P}$ iff $\mathsf{r}(p, \mathbf{x}) = \mathsf{T}$ for all $p \in \mathcal{P}$. Therefore, to block all paths in $\mathcal{P}$, we find the minimum $||\mathbf{x}||$ such that:

$$\mathsf{D}(\mathcal{P}, \mathbf{x}) = \sum_{p \in \mathcal{P}} \mathsf{r}(p, \mathbf{x}) = |\mathcal{P}| \cdot \mathsf{T} \tag{13}$$

In the next subsections, we devise two approximation algorithms to find such $\mathbf{x}$ and provide their performance guarantees.

4.2 Iterative Greedy algorithm

The first algorithm to block all paths in $\mathcal{P}$ is the *iterative greedy* algorithm (IG). The general idea is that: we iteratively add a unit vector $\mathbf{s}$ into $\mathbf{x}$, which maximizes the marginal gain $\Delta_{\mathbf{s}}\mathrm{D}(\mathcal{P}, \mathbf{x})$, until $\mathbf{x}$ is sufficient to block all paths in $\mathcal{P}$. Hence, the final overall budget ($||\mathbf{x}||$) is equal to the number of iterations of the algorithm. IG is fully presented by Alg. 3.

However, the objective function $\mathrm{D}(\mathcal{P}, \mathbf{x})$ is neither submodular nor supermodular w.r.t $\mathbf{x}$. If each edge weight function is concave, $\mathrm{D}(\mathcal{P}, \cdot)$ exhibits a *submodular behavior.* On the other hand, if each weight function is convex, then $\mathrm{D}(\mathcal{P}, \mathbf{x})$ can be much more than the sum of $\mathrm{D}(\mathcal{P}, \mathbf{s})$ values of unit vectors $\mathbf{s}$ constituting $\mathbf{x}$, which is a *supermodular behavior.* The non-submodularity of $\mathrm{D}(\mathcal{P}, \cdot)$ means that the $\mathbf{x}$ returned by IG may not have an $O(\log n)$ approximation ratio. Actually the concave ratio γ plays an important role on the performance guarantee of IG, which is proved theoretically by Theorem 4.4 and would be further illustrated in the experimental evaluation.

Algorithm 3 Greedy blocking paths (IG)

```
Input $G, f, \mathrm{b}, \mathrm{T}, \mathcal{P}$
Output a cost vector $\mathbf{x}$
1: $\mathbf{x} = \{0\}^m$
2: while $\mathrm{D}(\mathcal{P}, \mathbf{x}) \leq |\mathcal{P}|\mathrm{T}$ do
3:     for each unit vector $\mathbf{s}$ do
4:         $\Delta_{\mathbf{s}}\mathrm{D}(\mathcal{P}, \mathbf{x}) = \mathrm{D}(\mathcal{P}, \mathbf{x} + \mathbf{s}) - \mathrm{D}(\mathcal{P}, \mathbf{x})$
5:     $\mathbf{x} = \mathbf{x} + \mathrm{argmax}_{\mathbf{s}}\Delta_{\mathbf{s}}\mathrm{D}(\mathcal{P}, \mathbf{x})$
Return $\mathbf{x}$
```

Theorem 4.4. IG *returns a solution within* $O(\gamma^{-1}(\mathrm{h} \ln n + \ln \mathrm{T}))$ *factor of the optimal solution for blocking all paths in* $\mathcal{P}$.

Proof overview. Denote $\mathbf{x}^*$ as an optimal solution to the QoSD instance ($||\mathbf{x}^*|| = \mathrm{OPT}$). Denote $\mathbf{x}_i$ as our obtained solution before the i^{th} iteration in Alg. 3. The key of our proof is that: the gap between $|\mathcal{P}|\mathrm{T}$ and $\mathrm{D}(\mathcal{P}, \mathbf{x})$ will be reduced after each iteration by a factor at least $1 - \frac{\gamma}{\mathrm{OPT}}$. To be specific:

$$|\mathcal{P}|\mathrm{T} - \mathrm{D}(\mathcal{P}, \mathbf{x}_{i+1}) \leq (1 - \frac{\gamma}{\mathrm{OPT}})(|\mathcal{P}|\mathrm{T} - \mathrm{D}(\mathcal{P}, \mathbf{x}_i))$$

This was proved by using the property of concave ratio from lemma 4.2 and the greedy selection.

Furthermore, since there should exist at least a feasible path $p \in \mathcal{P}$ such that $\mathrm{r}(p, \mathbf{x}) \leq \mathrm{T} - 1$ before the final iteration of the algorithm, we prove that the number of iterations is upper bounded by $O(\frac{\ln |\mathcal{P}|\mathrm{T}}{\gamma/\mathrm{OPT}})$. The theorem follows as the number of iterations is equal to $||\mathbf{x}||$. □

4.3 Adaptive Trading algorithm

The concave ratio of the edge weight functions could be very small if the weight functions are convex, which makes the approximation guarantee of IG undesirable. Therefore, in this section, we propose a solution whose performance guarantee does not depend on the concave ratio γ. We name this algorithm *Adaptive Trading* (AT).

The algorithm still works in the iterative manner and terminates only when the desired $\mathbf{x}$ is found, but different from IG on how the solution $\mathbf{x}$ is improved in each iteration. To be specific, in each iteration, the algorithm finds an amount of additional budget to increase the weight of an edge such that maximize the ratio between the increasing amount of $\mathrm{D}(\mathcal{P}, \mathbf{x})$ and the additional budget.

Therefore, in each iteration, the additional budget could be bigger than 1. To find such amount, the simplest way is to scan through all possible amounts of additional budget of each edge. Note that the maximum budget which can be added to increase weight of edge e is upper bounded by b_e. Therefore, the computation complexity in each iteration of AT is upper bounded by $O(||\mathsf{b}||)$. Denote $\mathbf{u}(e,i) \in \mathbb{R}^m$ as a vector where the element corresponding to edge e has value i and other elements are 0. AT is fully presented in Alg. 4 and its approximation guarantee is provided by Theorem 4.5.

Algorithm 4 Adaptive Trading solution (AT)

```
Input $G, f, \mathsf{b}, \mathsf{T}, \mathcal{P}$
Output QoS adjustment vector $\mathbf{x}$
1: $\mathcal{P} = \emptyset$
2: while $\mathsf{D}(\mathcal{P}, \mathbf{x}) \le |\mathcal{P}|\mathsf{T}$ do
3:     for each edge $e \in E$ do
4:         $z_e = \text{argmax}_z \frac{\Delta_{\mathbf{u}(e,z)}\mathsf{D}(\mathcal{P},\mathbf{x})}{z}$
5:     $\mathbf{x} = \mathbf{x} + \text{argmax}_{\mathbf{u}(e,z_e)} \frac{\Delta_{\mathbf{u}(e,z_e)}\mathsf{D}(\mathcal{P},\mathbf{x})}{z_e}$
Return $\mathbf{x}$
```

Theorem 4.5. AT *returns a solution within* $O(\mathsf{h} \ln n + \ln \mathsf{T})$ *factor of the optimal solution for blocking all paths in* $\mathcal{P}$.

Proof overview. Denote $\mathbf{x}_i = \{x_1, ...x_m\}$ as our obtained solution before the i^{th} iteration in Alg. 4. Let $\mathbf{x}_i^o = \{x_1^o, ...x_m^o\}$ be an optimal solution which is in addition to $\mathbf{x}_i$ to block all paths in $\mathcal{P}$. Denote $\mathbf{v}(e) = \{x_1, ..x_{e-1}, x_e + x_e^o, ...x_m + x_m^o\}$. Trivially, $\mathbf{v}(1) = \mathbf{x}_i + \mathbf{x}^o$ and $\mathbf{v}(m+1) = \mathbf{x}_i$. Let $\mathbf{u}(e_i, j_i)$ be a vector we add into solution $\mathbf{x}_i$ in the i^{th} iteration. The key of our proof is that the following inequality is always guaranteed after each iteration.

$$\frac{\Delta_{\mathbf{u}_i(e_i,j_i)}\mathsf{D}(\mathcal{P},\mathbf{x}_i)}{j_i} \ge \frac{\mathsf{D}(\mathcal{P},\mathbf{v}(e)) - \mathsf{D}(\mathcal{P},\mathbf{v}(e+1))}{x_e^o} \tag{14}$$

for any $e \in E$. This is proved by utilizing the monotonicity of $\mathsf{r}(p,\mathbf{x})$ w.r.t $\mathbf{x}$ and the trait that the selection of our algorithm ensures $\Delta_{\mathbf{u}(e,q)}\mathsf{D}(\mathcal{P},\mathbf{x}_i) \le \frac{q}{j_i}\Delta_{\mathbf{u}(e_i,j_i)}\mathsf{D}(\mathcal{P},\mathbf{x}_i)$ for any $e \in E$ and $q \in \mathbb{Z}^+$.

Furthermore, the Eq. 14 helps us to prove that: the gap between $|\mathcal{P}|\mathsf{T}$ and $\mathsf{D}(\mathcal{P},\mathbf{x})$ will be reduced after each iteration by a factor at least $1 - \frac{j_i}{\mathsf{OPT}}$. To be specific:

$$|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P},\mathbf{x}_{i+1}) \le (1 - \frac{j_i}{\mathsf{OPT}})\big(|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P},\mathbf{x}_i)\big)$$

since there should exist at least a feasible path $p \in \mathcal{P}$ such that $\mathsf{r}(p,\mathbf{x}) \le \mathsf{T} - 1$ before the final iteration of the algorithm, utilizing Cauchy theorem [3], we bound the budget $||\mathbf{x}||$ by $\mathsf{OPT}\cdot O(\ln |\mathcal{P}|\mathsf{T})$. Since $|\mathcal{P}| \le n^{\mathsf{h}}$, the theorem follows. □

5 SAMPLING APPROACH

In this section, we introduce a sampling solution SA to QoSD which has $O(\frac{\ln \mathsf{T} + \mathsf{h} \ln d}{\gamma(1-e^{-}\gamma)(1-\epsilon)})$ approximation guarantee with probability at least $1-\delta$ where $\epsilon, \delta > 0$ are arbitrarily small numbers. SA runs in polynomial time when the parameter T is fixed.

We define a *blocking metric* of a budget vector $\mathbf{x}$ as follows

$$B(\mathbf{x}) = \sum_{p \in \mathcal{F}} \mathsf{r}(p,\mathbf{x})$$

It is trivial that $\mathbf{x}$ blocks all pairs in $\mathcal{F}$ iff $B(\mathbf{x}) = |\mathcal{F}| \cdot \mathsf{T}$.

In essence, SA attempts to minimize $||\mathbf{x}||$ while ensuring $B(\mathbf{x}) = |\mathcal{F}| \cdot \mathsf{T}$. To do so, SA works in the greedy manner as follows: in each iteration, SA finds a budget vector $\mathbf{v} = \{v_1, ...v_m\}$, $||\mathbf{v}|| \leq q$, to add into $\mathbf{x}$ which maximizes $B(\mathbf{x} + \mathbf{v})$. Rather than an expensive listing of $\mathcal{F}$, an estimator is employed by path sampling procedure to find the vector $\mathbf{v}$. This process is repeated until the budget vector $\mathbf{x}$ is sufficient to block all paths in $\mathcal{F}$. SA is fully presented in Alg. 5.

Algorithm 5 Sampling Algorithm (SA)

```
Input $G, S, \mathsf{T}, f, \mathsf{b}$ and $q, \epsilon, \delta$
Output cost vector $\mathbf{x}$
1: Initiate $\mathbf{x} = \{0\}^m$
2: while There exists a path $p \in \mathcal{F}$ whose length $< \mathsf{T}$ do
3:     Generate $\mathcal{P} = \mathcal{N}(q, \epsilon, \delta/||\mathsf{b}||)$ sample paths $p_1, p_2, ...$
4:     Greedily select $\mathbf{v}$ ($||\mathbf{v}|| \leq q$) that maximizes $\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v})$
5:     $\mathbf{x} = \mathbf{x} + \mathbf{v}$
Return $\mathbf{x}$
```

Since we will not list $\mathcal{F}$, the questions now are (1) how to estimate $B(\cdot)$; and (2) how many sample paths should be generated to bound the error between the estimator of $B(\cdot)$ and its actual value. In sub-section 5.1, we define the estimator $\hat{B}(\cdot)$ employed in each iteration of Alg. 5. We provide the approximation guarantee of greedily selection on sub-section 5.2. Sub-section 5.3 provides the lower bound on the number of sampling paths to bound the error. We then put all the results together to obtain the performance guarantee of SA.

5.1 Estimator

Let an instance $(G, f, \mathsf{b}, \mathsf{T})$ of QoSD be given. Denote $\mathcal{J}$ as a set of all single paths in G. For each $p \in \mathcal{J}$, define:

$$\mathsf{R}(p, \mathbf{x}) = \begin{cases} \mathsf{r}(p, \mathbf{x}) & \text{if } p \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases}$$

It is trivial that $\sum_{p \in \mathcal{J}} \mathsf{R}(p, \mathbf{x}) = \sum_{p \in \mathcal{F}} \mathsf{r}(p, \mathbf{x})$. Inspired by the estimation on the number of paths in a graph [39], we define the estimator of $B(\mathbf{x})$ in the following way: Given a probability distribution ρ on $\mathcal{J}$ such that $\rho(p) > 0$ for all $p \in \mathcal{F}$. Let $\mathcal{P} = \{p_1, p_2, ...p_l\}$ be a set of l paths samples from ρ, $B(\mathbf{x})$ could be estimated by

$$\hat{B}(\mathcal{P}, \mathbf{x}) = \frac{1}{l} \sum_{i=1}^{l} \frac{\mathsf{R}(p_i, \mathbf{x})}{\rho(p_i)}$$

Lemma 5.1. *$\hat{B}(\mathcal{P}, \mathbf{x})$ is an unbiased estimator of $B(\mathbf{x})$*

Proof.

$$\mathbb{E}[\hat{B}(\mathcal{P}, \mathbf{x})] = \mathbb{E}\Big[\frac{\mathsf{R}(p, \mathbf{x})}{\rho(p)}\Big] = \sum_{p \in \mathcal{J}} \frac{\mathsf{R}(p, \mathbf{x})}{\rho(p)} \cdot \rho(p) = \sum_{p \in \mathcal{J}} \mathsf{R}(p, \mathbf{x}) = \sum_{p \in \mathcal{F}} \mathsf{r}(p, \mathbf{x})$$

□

Algorithm 6 Sampling path

```
Input $G, S, \mathsf{T}, \mathbf{x}$
Output Sample path $p$
1: $(s,t) \longleftarrow$ randomly select a transaction.
2: $p \longleftarrow \{s\}$
3: do
4:     $u = \text{tail}(p)$
5:     Let $N(u)$ be the set of outcoming neighbors of $u$
6:     $p \longleftarrow p \cup \text{NeighborSelection}(N(u), p, (s,t))$
7: while $u \neq t$ and $\sum_{e \in p} f_e(x_e) < \mathsf{T}$
Return $p$
```

To sampling paths, we utilize the following biased, self-avoiding random walk sampling technique, which was once proposed by Kuhnle [29]. First, we randomly select a pair (s,t) from S and put s into the sample path p. Considering in a certain moment, $p = \{s, ..u\}$ (u is called a tail node of p at this time). The NeighborSelection procedure would select a node among the out-going neighbors of u to add into p. The selection is as follows: Let $\mathcal{T}$ be the shortest-path tree directed towards t. Let v be the parent of u in $\mathcal{T}$. If $N(u)/p = \{v\}$, then the next node we add into p is v. If $v \in N(u)/p$, we select v with probability α and the other nodes in $N(u)/p$ with probability of $\frac{1-\alpha}{|N(u)/p|-1}$. If $v \notin N(u)/p$, we select the next node uniform randomly among $N(u)/p$. The sampling procedure ends when we meet the node t or the length of p exceeds T. With the path-sampling procedure defined, given a path p, we could easily find $\rho(p)$. Also, $\rho(p) > 0$ for all $p \in \mathcal{F}$. The sampling technique is fully presented in Alg. 6.

5.2 Greedy selection on the estimator

Having defined the estimator $\hat{B}(\mathcal{P}, \mathbf{x})$ and the path sampling procedure, we now find the budget vector $\mathbf{v}$, $||\mathbf{v}|| \leq q$, to maximize $\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v})$. $\mathbf{v}$ is found in the greedy manner as follows: we run in q iterations and in each iteration, selecting the unit vector that maximizes the marginal gain of $\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v})$. Since it is trivial, we will not write down the pseudo-code on how we find $\mathbf{v}$.

The question now is what approximation guarantee $\mathbf{v}$ can provide? Note that $\hat{B}(\mathcal{P}, \mathbf{x})$ is a finite combination of functions $\mathsf{r}(p, \mathbf{x})$ with $p \in \mathcal{P}$. Hence, $\hat{B}(\mathcal{P}, \mathbf{x})$ is submodular if all weight functions are concave and supermodular if they are convex. So maximizing $\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v})$ using greedy algorithm may not return $1 - 1/e$ approximation result. Therefore, similar to `IG`, we use the concave ratio γ to obtain the performance guarantee of the greedy selection to maximize $\hat{B}(\mathcal{P}, \mathbf{x})$.

Denote $\mathbf{v}^o = \sum_{i=1}^{q} \mathbf{u}_i$ as an optimal solution that maximizes $\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v})$, where $\mathbf{u}_i$ is a unit vector ($i \in [0, m]$). Lemma 5.2 provides approximation guarantee of the greedy selection.

Lemma 5.2.

$$\Delta_{\mathbf{v}} \hat{B}(\mathcal{P}, \mathbf{x}) \geq (1 - e^{-\gamma}) \Delta_{\mathbf{v}^o} \hat{B}(\mathcal{P}, \mathbf{x}^o)$$

Proof overview. Denote $\mathbf{v}_i$ as the budget vector $\mathbf{v}$ after greedily selecting first i unit vectors. The key of the proof comes from the following inequality:

$$\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_{i+1}) \leq (1 - \frac{\gamma}{q})\Big(\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i)\Big)$$

This inequality is proved by using the property of γ from lemma 4.2 and the trait that $\hat{B}(\mathcal{P},\mathbf{x})$ is monotone w.r.t $\mathbf{x}$. Using this inequality, we prove that

$$\Delta_{\mathbf{v}}\hat{B}(\mathcal{P},\mathbf{x}) \geq (1-(1-\frac{\gamma}{q})^q)\Delta_{\mathbf{v}^o}\hat{B}(\mathcal{P},\mathbf{x}) \geq (1-e^{-\gamma})\Delta_{\mathbf{v}^o}\hat{B}(\mathcal{P},\mathbf{x})$$

in which the lemma follows. □

5.3 Sample size and Performance guarantee

We have proved the performance guarantee of the additional budget vector $\mathbf{v}$ to maximize $\hat{B}(\mathcal{P},\mathbf{x}+\mathbf{v})$. The question now is: what is the size of $\mathcal{P}$ to bound the error between $\Delta_{\mathbf{v}}\hat{B}(\mathcal{P},\mathbf{x})$ and $\Delta_{\mathbf{v}}B(\mathbf{x})$? In this part, we will answer this question. Then, putting together with the performance guarantee of selecting $\mathbf{v}$ on $\mathcal{P}$, we provide the performance guarantee of SA.

To find the minimum number of samples, we utilize the following Chernoff Bound theory.

Theorem 5.3. (Chernoff Bound theorem [27]) *Let $X_1, X_2, ...X_n$ be random variables such that $a \leq X_i \leq b$ for all i. Let $X = \sum_{i=1}^n X_i$ and set $\mu = \mathbb{E}(X)$. Then for all $\epsilon > 0$, we have:*

$$\Pr[X \geq (1+\epsilon)\mu] \leq \exp(-\frac{2\epsilon^2\mu^2}{n(b-a)^2}) \tag{15}$$

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp(-\frac{\epsilon^2\mu^2}{n(b-a)^2}) \tag{16}$$

Considering a path $p \in \mathcal{F}$, we have:

$$\rho(p) \geq \frac{1}{|S|}(\frac{1-\alpha}{\mathsf{d}-1})^{\mathsf{h}} = \Omega(\mathsf{d}^{-\mathsf{h}}|S|^{-1})$$

where d is the maximum out-going degree of a node in G. Therefore, for any single path p, $0 \leq \frac{\mathsf{R}(p,\mathbf{x})}{\rho(p)} \leq O(\mathsf{T}|S|\mathsf{d}^{\mathsf{h}})$

Denote $\mathbf{v}^*$ as an optimal solution that maximizes $\Delta_{\mathbf{v}}B(\mathbf{x})$.

Lemma 5.4. *Given $0 < \epsilon_1, \delta_1 < 1$, with the number of sampling paths satisfies*

$$|\mathcal{P}| \geq \frac{\ln(1/\delta_1)\mathsf{T}^2|S|^2\mathsf{d}^{2\mathsf{h}}}{\epsilon_1^2\Delta_{\mathbf{v}^*}^2 B(\mathbf{x})} \tag{17}$$

the following condition is guaranteed:

$$\Pr[\Delta_{\mathbf{v}^*}\hat{B}(\mathcal{P},\mathbf{x}) \geq (1-\epsilon_1)\Delta_{\mathbf{v}^*}B(\mathbf{x})] \geq 1-\delta_1 \tag{18}$$

This lemma is trivially derived from Eq. 16.

Lemma 5.5. *Given $0 < \epsilon_2, \delta_2 < 1$, with the number of sampling paths satisfies*

$$|\mathcal{P}| \geq \frac{\ln(\binom{n+q}{q}/\delta_2)\mathsf{T}^2|S|^2\mathsf{d}^{2\mathsf{h}}}{2(1-e^{-\gamma})^2\epsilon_2^2\Delta_{\mathbf{v}^*}^2 B(\mathbf{x})}$$

we have $\Delta_{\mathbf{v}_q}\hat{D}(x) \leq \Delta_{\mathbf{v}_q}D(\mathbf{x}) + (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{v}^*}D(\mathbf{x})$ for all budget vectors $\mathbf{v}_k$, which satisfy $||\mathbf{v}_k|| = q$, with probability at least $1-\delta_2$

Proof. Let us consider an arbitrary budget vector $\mathbf{v}_q$, $||\mathbf{v}_q|| = q$

$$\begin{aligned}
&\Pr\Big[\Delta_{\mathbf{v}_q}\hat{B}(\mathcal{P},\mathbf{x}) \geq \Delta_{\mathbf{v}_q}B(\mathbf{x}) + (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{v}^*}B(\mathbf{x})\Big] \\
&= \Pr\Big[\Delta_{\mathbf{v}_q}\hat{B}(\mathcal{P},x) \geq \Delta_{\mathbf{v}_q}B(\mathbf{x})\Big(1 + (1-e^{-\gamma})\epsilon_2\frac{\Delta_{\mathbf{v}^*}B(\mathbf{x})}{\Delta_{\mathbf{v}_q}B(\mathbf{x})}\Big)\Big] \\
&\leq \exp\Big(\frac{2(1-e^{-\gamma})^2\epsilon_2^2|\mathcal{P}|\Delta^2_{\mathbf{v}^*}B(\mathbf{x})}{\mathsf{T}^2\mathsf{d}^{2\mathsf{h}}}\Big)
\end{aligned}$$

Using the union bound theory, to let $\Delta_{\mathbf{v}_q}\hat{B}(\mathcal{P},x) \leq \Delta_{\mathbf{v}_q}B(\mathbf{x}) + (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{x}^*}B(\mathbf{x})$ satisfy for any budget vector $\mathbf{v}_q$, $||\mathbf{v}_q|| = k$, we have

$$\Pr\Big[\Delta_{\mathbf{v}_q}\hat{B}(\mathcal{P},\mathbf{x}) \geq \Delta_{\mathbf{v}_q}B(\mathbf{x}) + (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{v}^*}B(\mathbf{x})\Big] \leq \binom{n+q}{q}\exp\Big(\frac{2(1-e^{-\gamma})^2\epsilon_2^2|\mathcal{P}|\Delta^2_{\mathbf{v}^*}B(\mathbf{x})}{\mathsf{T}^2|S|^2\mathsf{d}^{2\mathsf{h}}}\Big)$$

The lemma follows by letting $\binom{n+q}{q}\exp\Big(\frac{2(1-e^{-\gamma})^2\epsilon_2^2|\mathcal{P}|\Delta^2_{\mathbf{v}^*}B(\mathbf{x})}{\mathsf{T}^2|S|^2\mathsf{d}^{2\mathsf{h}}}\Big) \leq \delta_2$ □

Lemma 5.6. *Given $0 \leq \epsilon_1, \epsilon_2, \delta_1, \delta_2 \leq 1$, let $\epsilon \geq \epsilon_1 + \epsilon_2$ and $\delta \geq \delta_1 + \delta_2$. If the number of sampling paths is at least*

$$\frac{\mathsf{T}^2|S|^2\mathsf{d}^{2\mathsf{h}}}{\Delta^2_{\mathbf{v}^*}B(\mathbf{x})}\max\Big(\frac{\ln(1/\delta_1)}{\epsilon_1^2}, \frac{\ln(\binom{n+q}{q}/\delta_2)}{2(1-e^{-\gamma})^2\epsilon_2^2}\Big) \tag{19}$$

the greedy algorithm on $\mathcal{P}$ returns a budget vector $\mathbf{v}$ *that guarantees*

$$\Pr[\Delta_{\mathbf{v}}B(\mathbf{x}) \geq (1-e^{-\gamma})(1-\epsilon)\Delta_{\mathbf{v}^*}B(\mathbf{x})] \geq 1-\delta$$

Proof. For the given number of sample paths, we have

$$\begin{aligned}
\Delta_{\mathbf{v}}B(\mathbf{x}) &\geq \Delta_{\mathbf{v}}\hat{B}(\mathcal{P},\mathbf{x}) - (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{v}^*}B(\mathbf{x}) && (20)\\
&\geq (1-e^{-\gamma})\Delta_{\mathbf{v}^*}\hat{B}(\mathcal{P},\mathbf{x}) - (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{x}^*}B(\mathbf{x}) && (21)\\
&\geq (1-e^{-\gamma})(1-\epsilon_1)\Delta_{\mathbf{v}^*}B(\mathbf{x}) - (1-e^{-\gamma})\epsilon_2\Delta_{\mathbf{v}^*}B(\mathbf{x}) && (22)\\
&\geq (1-e^{-\gamma})(1-\epsilon)\Delta_{\mathbf{v}^*}B(\mathbf{x}) && (23)
\end{aligned}$$

The inequality (20) happens with probability $1-\delta_1$ while the inequality (22) happens with probability $1-\delta_1$. Overall $\Delta_{\mathbf{v}}B(\mathbf{x}) \geq (1-e^{-\gamma})(1-\epsilon)\Delta_{\mathbf{v}^*}B(\mathbf{x})$ with probability at least $(1-\delta_1)(1-\delta_2) \geq 1-\delta$. □

There is a drawback of the threshold (19): it depends on $\Delta_{\mathbf{v}^*}B(\mathbf{x})$, which is untraceable. However, we can use the simple lower bound of $\Delta_{\mathbf{v}^*}B(\mathbf{x})$ as follows: As long as the algorithm has not terminated, there should be at least a path $p \in \mathcal{F}$ such that the length of p is at most $\mathsf{T}-1$. So the marginal gain of the optimal solution should be at least 1. Therefore, we have the following threshold, which is the sufficient number of sample paths to bound the error between approximation ratio of $\mathbf{v}$ on $\Delta_{\mathbf{v}}\hat{B}(\mathcal{P},\mathbf{x})$ and $\Delta_{\mathbf{v}}B(\mathbf{x})$.

$$\mathcal{N}(q,\epsilon,\delta) = \min_{\epsilon_1;\delta_1}\Bigg(\mathsf{T}^2|S|^2\mathsf{d}^{2\mathsf{h}}\max\Big(\frac{\ln(1/\delta_1)}{\epsilon_1^2}, \frac{\ln(\binom{n+q}{q}/(\delta-\delta_1))}{2(1-e^{-\gamma})^2(\epsilon-\epsilon_1)^2}\Big)\Bigg)$$

Theorem 5.7. *Given $0 < \epsilon, \delta < 1$, by generating $\mathcal{N}(k,\epsilon,\frac{\delta}{||\mathsf{b}||})$ of sample paths in each sampling iteration,* SA *returns a solution within $O(\frac{\mathsf{h}\ln\mathsf{d}+\ln\mathsf{T}}{\gamma(1-e^{-\gamma})(1-\epsilon)})$ factor of optimum to the* QoSD *instance with probability at least $1-\delta$.*

Proof overview. Denote $\mathbf{x}^o = \sum_i^s \mathbf{u}_i$ as an optimal solution, which is in addition to $\mathbf{x}$ to block all paths in $\mathcal{F}$ ($\mathbf{u}_i$ is a unit vector). Let $\mathbf{v}$ be a budget vector we get from greedy selection on the sample set $\mathcal{P}$. The key of our proof is that

$$\Delta_{\mathbf{v}} B(\mathbf{x}) \geq \frac{\gamma q}{||\mathbf{x}^o||}(1 - e^{-\gamma})(1-\epsilon)\Delta_{\mathbf{x}^o} B(\mathbf{x})$$

This is proved by the finding that there exists a budget vector $\mathbf{w}$ such that $||\mathbf{w}|| \leq q$ and $\Delta_{\mathbf{w}} B(\mathbf{x}) \geq \frac{\gamma q}{||\mathbf{x}^o||}\Delta_{\mathbf{x}^o} B(\mathbf{x})$.

Therefore, we observe that: after each sampling iteration, the gap between $|\mathcal{F}|\mathsf{T}$ and $B(\mathbf{x})$ shrinks by a factor at least $(1 - \frac{k\gamma}{\mathsf{OPT}}(1-e^{-\gamma})(1-\epsilon))$ with probability at least $1 - \frac{\delta}{||\mathsf{b}||}$.

Furthermore, since there should exist at least a feasible path $p \in \mathcal{P}$ such that $\mathsf{r}(p, \mathbf{x}) \leq \mathsf{T} - 1$ before the final sampling iteration, we prove that the number of iterations is upper bounded by $O(\frac{\ln \mathsf{T} + \mathsf{h} \ln \mathsf{d}}{q\gamma(1-e^{-\gamma})(1-\epsilon)})\mathsf{OPT}$. Since in each iteration, a budget vector $\mathbf{v}$, $||\mathbf{v}|| \leq q$, is added into solution, out final solution guarantees $O(\frac{\ln \mathsf{T} + \mathsf{h} \ln \mathsf{d}}{\gamma(1-e^{-\gamma})(1-\epsilon)})$ approximation ratio with probability at least $1 - \delta$. □

Interestingly, the approximation ratio of SA does not depend on q. So whatever the value of q is, the result of SA always has the same upper bound, which means a large value of q could reduce the number of sampling iterations but the number of sample paths in each iteration would increase as the trade-off.

6 LINEAR WEIGHT FUNCTIONS

Having considered approximation algorithms to QoSD, we now propose a solution, called *Linear Rounding* (LR), for the case where the edge weight functions are linear. LR obtains $O(\mathsf{h} \log n)$ approximation guarantee, which is the best ratio compared among all the proposed solutions.

For each $e \in E$, the weight function of e is represented as $f_e(x) = \beta_e x + \alpha_e$, where $\beta_e, \alpha_e \in \mathbb{Z}^+$. Denote $\beta = \max_e \beta_e$. The QoSD instance can be solved by the following Integer Programming.

$$\min \quad \sum_{e \in E} x_e \tag{24}$$

$$\text{s.t.} \quad \sum_{e \in p} (\beta_e x_e + \alpha_e) \geq \mathsf{T} \qquad \forall p \in \mathcal{F} \tag{25}$$

$$x_e \leq b_e \qquad \forall e \in E \tag{26}$$

$$x_e \in \mathbb{Z}^+ \cup \{0\} \qquad \forall e \in E \tag{27}$$

This IP has a simple linear relaxation by replacing constraint (27) with:

$$x_e \in \mathbb{R}^+ \cup \{0\} \qquad \forall e \in E \tag{28}$$

Although constructing this relaxation maybe intractable due to the extremely large size of $\mathcal{F}$, this LP still can be solved in polynomial time using *ellipsoid method* with a simple separation oracle similar to Multicut problem [43].

Denote the vector $\mathbf{x}' = \{x'_1, ...x'_m\}$ as the optimal solution to the LP relaxation, x'_e can be a real number. The problem now is how to obtain a discrete solution $\mathbf{x}$ from $\mathbf{x}'$ and what approximation guarantee $\mathbf{x}$ provides? To do so, we applied the randomized rounding technique as follows: Given an edge e, if x'_e is an integer, let $x_e = x'_e$. Otherwise, denote $\rho_e = x'_e - \lfloor x'_e \rfloor$ and given η, which would be defined later, then:

- If $\eta\rho_e \geq 1$, $x_e = \lceil x'_e \rceil$. Let $y_e = x_e$
- If $\eta\rho_e < 1$, $x_e = \lceil x'_e \rceil$ with probability $\eta\rho_e$ and $\lfloor x'_e \rfloor$ otherwise. Let $y_e = \lfloor x'_e \rfloor$

Algorithm 7 Linear Rounding algorithm (LR)

```
Input $G, S, \mathsf{T}, \mathsf{b}, f, \delta$
Output cost vector $\mathbf{x} = \{x_1, ..x_m\}$
1: $\mathbf{x}' \leftarrow$ optimal solution of LP-relaxation.
2: $\beta = \max_e \beta_e$; $\eta = \frac{\beta}{1-\exp(-\beta)}(\ln \frac{n^{\mathsf{h}}}{\delta} + 1)$
3: for each $e \in E$ do
4:     if $x'_e$ is a integer then
5:         $x_e = x'_e$
6:     else
7:         $\rho_e = x'_e - \lfloor x'_e \rfloor$
8:         $x_e = \lceil x'_e \rceil$ with probability $\eta\rho_e$; $\lfloor x'_e \rfloor$ otherwise.
Return x
```

LR is fully presented in Alg. 7.

Consider a path $p \in \mathcal{F}$, it is trivial that $\mathbf{x}$ will block p if $\sum_{e \in p} f_e(y_e) \geq \mathsf{T}$. The question is whether $\mathbf{x}$ can block p if $\sum_{e \in p} f_e(y_e) < \mathsf{T}$? Denote:

$$T_p = \mathsf{T} - \sum_{e \in p} f_e(y_e)$$
$$\mathcal{E}_p = \{e \in p; y_e < x_e\}$$

So:

$$T_p \leq \sum_{e \in \mathcal{E}_p} \beta_e(x'_e - y_e) = \sum_{e \in \mathcal{E}_p} \beta_e \rho_e$$

Then the probability that $\mathbf{x}$ does not block p is given as follows:

$$\Pr[p \text{ is not blocked by } \mathbf{x}] = \Pr\Big[\sum_{e \in p}(\beta_e x_e + \alpha_e) < \mathsf{T}\Big] = \Pr\Big[\sum_{e \in \mathcal{E}_p} \beta_e(x_e - \lfloor x'_e \rfloor) < T_p\Big] \tag{29}$$

$$= \Pr\Big[\exp\Big(-\sum_{e \in \mathcal{E}_p} \beta_e(x_e - \lfloor x'_e \rfloor)\Big) > \exp(-T_p)\Big] \tag{30}$$

$$\leq \exp(T_p) \cdot \mathbb{E}\Big[\exp\Big(-\sum_{e \in \mathcal{E}_p} \beta_e(x_e - \lfloor x'_e \rfloor)\Big)\Big] \tag{31}$$

$$= \exp(T_p) \cdot \prod_{e \in \mathcal{E}_p} \Big(\exp(-\beta_e) \cdot \eta\rho_e + (1 - \eta\rho_e)\Big) \tag{32}$$

$$\leq \exp(T_p) \cdot \left(1 - \frac{\sum_{e \in \mathcal{E}_p} \eta\rho_e(1 - \exp(-\beta_e))}{|\mathcal{E}_p|}\right)^{|\mathcal{E}_p|} \tag{33}$$

$$\leq \exp(T_p) \cdot \left(1 - \frac{1 - \exp(-\beta)}{\beta} \cdot \frac{\eta T_p}{|\mathcal{E}_p|}\right)^{|\mathcal{E}_p|} \tag{34}$$

$$\leq \exp(T_p) \cdot \exp\left(-\eta T_p \frac{1 - \exp(-\beta)}{\beta}\right) \leq \exp\left(-\left(\eta \frac{1 - \exp(-\beta)}{\beta} - 1\right)\right) \tag{35}$$

Eq. 31 comes from Markov inequality [8] while Eq. 33 is from Cauchy Theorem [3]. Since there are at most n^{h} feasible paths in $\mathcal{F}$, using Union Bound theory [2], the probability that $\mathbf{x}$ cannot block all paths in $\mathcal{F}$ is at most

$$n^{\mathsf{h}} \cdot \exp\left(-\left(\eta\frac{1-\exp(-\beta)}{\beta}-1\right)\right) \tag{36}$$

Theorem 6.1. *Given fixed* $0 < \delta < 1$ *and* $\eta = \frac{\beta}{1-\exp(-\beta)}(\ln\frac{n^{\mathsf{h}}}{\delta}+1)$, LR *returns a solution within* $O(\mathsf{h}\ln n)$ *factor of optimum to the* QoSD *instance with probability at least* $1-\delta$.

Proof. From Eq. 36 and the given η, the probability that $\mathbf{x}$ blocks all paths in $\mathcal{F}$ is at least $1-\delta$. Also

$$\begin{aligned}\mathbb{E}[||\mathbf{x}||] &= \sum_e \left(\lceil x'_e\rceil\eta\rho_e + \lfloor x'_e\rfloor(1-\eta\rho_e)\right)\\ &\le \sum_e \left(\lfloor x'_e\rfloor + \eta\rho_e\right) \le \sum_e \eta x'_e\\ &= \eta\cdot||\mathbf{x}'|| \le \eta\cdot||\mathbf{x}^*||\end{aligned}$$

which completes the proof. □

7 DISCUSSION

In this section, we discuss the trade-off between the performance guarantee and the runtime complexity of the four proposed algorithms, summarized in Table. 2.

First, we consider the performance guarantee of the IG and AT algorithm. The approximation ratio of IG and AT are $O(\frac{1}{\gamma}(\mathsf{h}\ln n+\ln\mathsf{T}))$ and $O(\mathsf{h}\ln n+\ln\mathsf{T})$ respectively, where γ is the *concave ratio* of edge weight functions. γ plays an important role in the differences between IG and AT solutions. The smaller γ is - which signifies a more convex of edge weight functions - the worse IG performs. But if all edge weight functions are concave - or at least linear - γ equals to 1, then IG and AT obtain the same approximation guarantee. Not only achieve the same ratio, the two algorithms also return the same solution because in AT, $\frac{\Delta_{u(e,x)}\mathsf{D}(\mathcal{P},\mathbf{x})}{x}$ reaches maximum at $x=1$. So in each iteration, the budget increases at most by 1, and it is also the selection of IG. Overall, AT theoretically returns better solutions than IG.

However, in trade-off, AT has higher computational complexity than IG. Both algorithms use the same framework as in Alg. 1. The maximum number of iterations in this framework (line 2 of Alg. 1) is upper bounded by n^{h}, which is theoretically a large number. However, from our experiments on both random graphs and real-but-dense networks, the number of iterations never reach this amount. Considering the strategy of blocking paths, the number of computation in each inner iteration (line 2 of Alg. 4) of AT is $O(||\mathsf{b}||)$, while this number (line 2 of Alg. 3) in IG is $O(m)$. In the worst-case scenario, the number of inner iterations of both IG and AT can reach up to $O(||\mathsf{b}||)$. Therefore, the worst-case runtime complexity of IG and AT is $O(n^{\mathsf{h}}m||\mathsf{b}||)$ and $O(n^{\mathsf{h}}||\mathsf{b}||^2)$ respectively.

With SA, to obtain the $O(\frac{\ln\mathsf{T}+\mathsf{h}\ln d}{\gamma(1-e^{-\gamma})(1-\epsilon)})$ ratio, we have to generate $\mathcal{N}(k,\epsilon,\delta/||\mathsf{b}||)$ paths with $O(m)$ time complexity for each path in each sampling steps. Also, after sampling, a budget vector $\mathbf{v}$ ($||\mathbf{v}||\le q$) is added into $\mathbf{x}$, which makes the number of sampling steps at most $O(\frac{||\mathsf{b}||}{q})$. Moreover, the greedy selection on a sample set costs $O(qm)$ runtime complexity. Therefore, the worse-case runtime complexity of SA is bounded by $O(\mathcal{N}(k,\epsilon,\frac{\delta}{||\mathsf{b}||})\cdot||\mathsf{b}||\cdot m^2)$. However, if h is large, the number of samples required by SA becomes large and its sampling procedure dominates its runtime; this is ameliorated by trivially parallelizing the sampling process, which is possible since each sample is independent. In practice, the parameter α greatly reduces the required number of samples;

Table 2. Algorithm performance ratio and time complexity

Algorithm	Approximation Ratio	Worst-case Runtime
IG	$O(\gamma^{-1}(\ln \mathsf{T} + \mathsf{h} \ln n))$	$O(n^{\mathsf{h}} m \|\|\mathsf{b}\|\|)$
AT	$O(\ln \mathsf{T} + \mathsf{h} \ln n)$	$O(n^{\mathsf{h}} \|\|\mathsf{b}\|\|^2)$
SA	$O(\frac{\ln \mathsf{T} + \mathsf{h} \ln \mathsf{d}}{\gamma(1-e^{-\gamma})(1-\epsilon)})$	$\mathcal{N}(q, \epsilon, \frac{\delta}{\|\|\mathsf{b}\|\|}) \cdot O(m^2 \|\|\mathsf{b}\|\|)$
LR	$O(\mathsf{h} \ln n)$	LP-solver() + m

with $\alpha = 0.8$, we found that $O(|S|)$ samples were sufficient to provide feasible solutions within reasonable runtime.

Next, consider the LR solution, which is only used if all the weight functions are linear. The runtime of LR strongly depends on the linear programming solver (LP-solver()). In the experimental evaluation, we observe that in most cases, the number of edges - whose x'_e is real - is inconsiderably small. Therefore, after randomized rounding, the size of the discrete solution $\mathbf{x}$ has a diminutive gap comparing with $\mathbf{x}'$'s. Hence, although IG and AT perform fairly well in general cases, LR usually returns the best solution if the edge weight functions are linear.

8 EXPERIMENTAL EVALUATION

In this section, we evaluate our proposed approximation algorithms by 1) comparing their performance to an intuitive heuristic as there is no other solution to QoSD, in a general case; and 2) comparing our algorithms to [29] as a special case of QoSD. The experiments were conducted on a Linux machine with 2.3Ghz Xeon 18 core processor and 256GB of RAM. The programming language we used is C++. Several steps in our algorithm are parallelized by using OpenMP with 64 threads. The reported running time is real-world time, not CPU time. The source code is available at [9].

8.1 Experiment Settings

We evaluated the following algorithms; the source code of all of our implementation is written in C++.

- AT: In this solution, to find the shortest paths between a pair of nodes, we utilized the Dijkstra algorithm and computed each path separately. The reason for this implementation is that by doing so, we can parallelize the process by dividing it into independent tasks. Therefore, even the theoretical time complexity of the Dijkstra algorithm for all-pair shortest paths is worse than Floyd-Warshall methods, the parallelization helps to boost the performance of the Dijkstra algorithm while it is impossible to do so with Floyd-Warshall.
- IG: this algorithm used the same settings as AT.
- SA: We set the bias ratio $\alpha = 0.8$ and the number of sample paths is $O(|S|)$ for all experiments. We found this value of α and the number of samples are sufficient to obtain feasible solutions within reasonable runtime in most cases.
- LR: We used CPLEX [16] to solve the linear programming. Implementing the *ellipsoid method* could result in impractical performance. So we used the same concept of IG and AT to solve the LP relaxation as follows: rather than listing all feasible paths in $\mathcal{F}$, we iteratively listed the shortest LP-weighted paths as constraints until the length of the shortest paths between each pair exceeded T.
- Centrality Cutting (CC) heuristic: Centrality has been commonly used as a metric to identify critical components of a network in the literature. CC works in iterative manner as follows:

Table 3. Statistics of datasets

Data	Type	Nodes	Edges	Diameter
Gnutella	Directed	10.9 K	40.0 K	9
RoadCA	Undirected	2.0 M	2.8 M	786
Skitter	Directed	1.7 M	11.1 M	25

First, we set $\mathbf{x} = \{0\}^m$. In each iteration, we found the shortest paths between a pair of nodes under the current budget vector $\mathbf{x}$ and computed the number of appearances of each edge in those paths. The algorithm then raised the weight of the edge that appears the most to maximum. All those steps are repeated until there were no shortest paths whose length was smaller than T. When finding shortest paths of each pair, we also used parallelization to boost CC performance.

- SAP, MIA, TAG [29]: These algorithms were only implemented in comparison on the special case of QoSD (the LB-MULTICUT problem). The source code of those algorithms was taken from [7] and it was only available for undirected networks.

To obtain S, we sampled uniformly random sets of pairs of nodes on each network. All results were averaged over 5 independent repetitions of each experiment. The weight function of each edge was selected from following functions

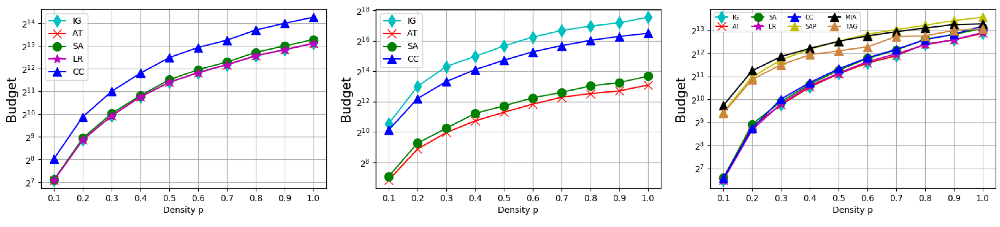
- A linear function $f_e(x) = \Theta(x)$.
- A convex function $f_e(x) = \Theta(x^2)$. This function was inspired by the average delay calculation on computer networks w.r.t packet arrival rate.
- A concave function $f_e(x) = \Theta(\ln x)$. This function was inspired by the additive metric on IoT network w.r.t packet error rate.
- A cutting function: f_e only received two values, $f_e(0) = 1$ and $f_e(1) = \mathtt{T}$. This function was used when we compared our solution with the algorithms of the LB-MULTICUT problem.

Each function was set such that the initial weight $f_e(0) = 1$ and the maximum weight was T. Since there exists heterogeneous coupling delays in modern networks, in our experiment, the weight function of each edge was randomly selected from the linear, convex or concave functions as mentioned above. In the experiments with the presence of LR, all weight functions were linear. On the other hand, all weight functions were cutting function if compared with the algorithms of LB-MULTICUT.

The algorithms were implemented on both synthesized networks and real-world networks. The synthesized networks we used were the Erdos-Renyi (ER) [23] graphs with 240 nodes and varied the edge density parameter ρ. For the real-world networks, we used the datasets from Stanford Network Analysis Project [33], including Gnutella, Skitter and Roadnet. Skitter is highly dense IPv4 Internet topology graph, which were collected by traceroutes run daily in 2005; Gnutella is the snapshots of peer-to-peer file sharing; and RoadCA is a road network of California where intersections and endpoints are represented by nodes, and the roads connecting these intersection or endpoints are represented by undirected edges. Information of real-world datasets are summarized in Table 3.

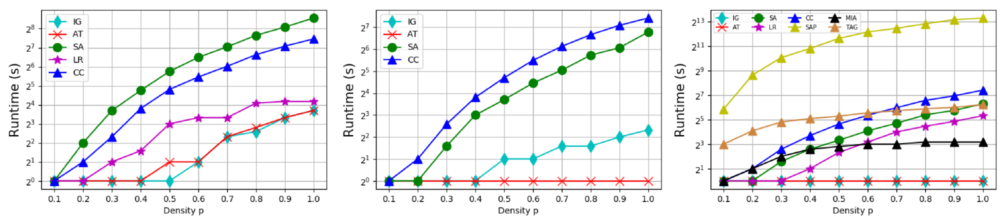
8.2 Performance comparison

8.2.1 Small size random graph. In these experiments, we compared our algorithms with the CC solution on directed ER networks with $n = 240$ and we varied the edges density ρ. The threshold T was set to be 3 and the size of S was 10.



(a) Linear (b) Heterogeneous (c) Cutting

Fig. 1. Solution quality of algorithms on Random graph

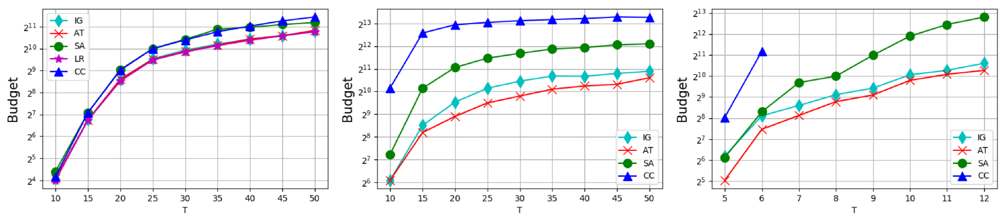


(a) Linear (b) Heterogeneous (c) Cutting

Fig. 2. Runtime of algorithms on Random graph

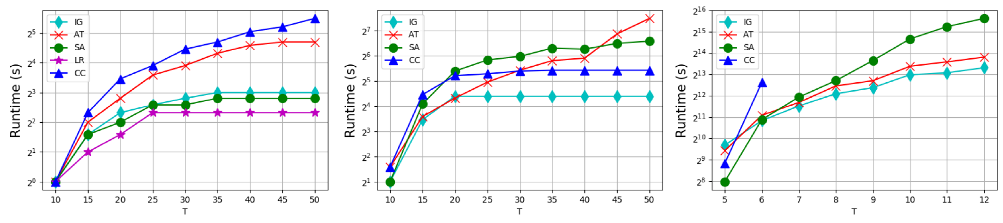
Fig. (1a) and Fig. (2a) show the results and runtimes of the algorithms when edge weight functions are linear. We notice that our four algorithms performed almost similarly in terms of quality of solution and very close to the optimal solution of LP relaxation. Meanwhile, CC was far from being optimal when its solutions were always at least double to the solution of other algorithms. In terms of runtime, the ranking from best to worst was IG, AT, LR, CC and SA. SA performed worse especially when the edge density increased and approached to 1. This can be explained by the following: with high value of the bias parameter α, most paths of the sample set $\mathcal{P}$ were the shortest paths of pairs in S. But because the edge density is high, there would be multiple paths between a pair of nodes whose length is smaller than T, which makes the number of sampling iterations on SA increases. Therefore, SA had a high runtime on finding shortest paths and then sampling, which was the main factor degrading its runtime. We observe that at the smallest edge density (0.1), all algorithms performed the best on both quality of solution and runtime, which is promising since most of real-world networks are sparse [29].

Next, we compared our algorithms in the scenario with heterogeneous weight functions. LR was no longer applicable, which explains why we did not plot LR in Fig. (1b) and Fig. (2b). Although having the same quality of solution in linear delay, IG performed much worse than AT when its sizes of solutions were always at least 20 times of AT's. The concave ratio γ was 0 in this scenario because there existed a weight function whose value did not change by adding several cost units. This experiment clearly illustrated the impact of concave ratio γ on the performance guarantee of IG. Moreover, γ also impacted on IG's runtime because the IG runtime is proportional to the solution size. From density 0.4, the gap between IG and AT's runtime became distinguishable.



(a) Gnutella - Linear (b) Gnutella - Heterogeneous (c) Skitter - Heterogeneous

Fig. 3. Solution quality of algorithms on Gnutella and Skitter



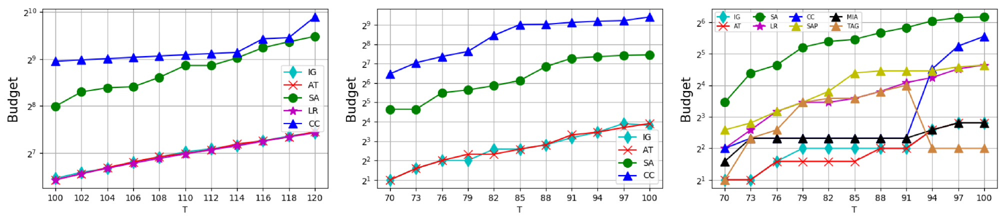
(a) Gnutella - Linear (b) Gnutella - Heterogeneous (c) Skitter - Heterogeneous

Fig. 4. Runtime of algorithms on Gnutella and Skitter

Finally, we compared our algorithms with three methods proposed by Kuhnle et al. [29] for the LB-MULTICUT problem. The random graph was undirected and had 240 nodes. The size of solution and runtimes are reported in Fig. (1c) and Fig. (2c). All of our four algorithms returned the best results while SAP, MIA and TAG were even worse than CC in term of quality of solution. The gap between the final budgets of those three algorithms and our algorithms was significant with small ρ and became smaller when ρ increased. In terms of runtime, SAP and TAG performed the worst while MIA bypassed SA after $\rho = 0.3$ and LR after $\rho = 0.5$. IG and AT were the fastest by far. It took less than one second for these two algorithms to finish no matter the edge density.

8.2.2 Results on real networks. In this subsection, we evaluate our algorithms on the real-world networks. We mainly examined the effect of varying the threshold T on the algorithm performances. The number of pairs is set to be 100. We limited the runtime by a day (24 hours); any experiments, which ran longer than a day, were terminated.

First, we discuss the results on the smallest network, Gnutella, in which we let T vary from 10 to 50. The result and the runtime of each algorithm are shown in Fig. (3a), (3b), (4a) and (4b). When the weight functions were all linear, we observed the same pattern as in the random graph, where the quality of solution of IG, AT and LR almost overlapped. Actually, LR always returned the best solution but the gap between LR and AT and IG was insignificant. Meanwhile, the sizes of the SA's solutions were always within 1.3 factor from LR. However, in terms of runtime, AT performed the worst among our proposed algorithms while LR again was the best. The next best algorithm in terms of runtime was SA, which stayed within 1.4 factor from LR. Starting from T = 35, the runtime of LR, IG and SA almost stayed the same while the runtime of AT and CC kept increasing.



(a) Linear (b) Heterogeneous (c) Cutting

Fig. 5. Solution quality of algorithms on RoadnetCA

However, it was a different matter in the experiments with heterogeneous weight functions, shown in Fig. (3b) and Fig. (4b). In terms of the quality of solution, the ranking from best to worst was AT, IG, SA and CC. These algorithms were now more virtually distinguishable in solution quality. The sizes of solutions of IG could be up to 1.25 factor from AT's while this number of SA was 4. In terms of runtime, CC was no longer the worst algorithm. Starting from T = 45, AT ran slower than SA and CC. IG was by far the fastest among the algorithms.

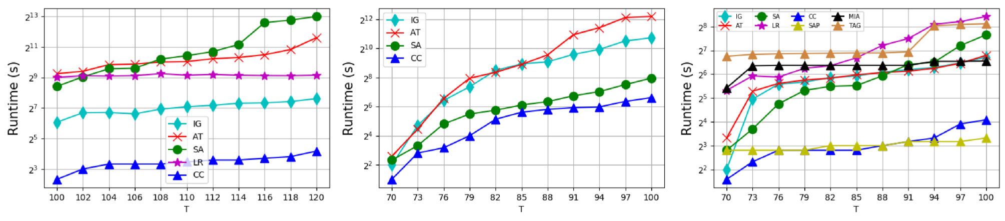
Next, we experimented our algorithms on large scale networks. The Roadnet network contains 2 millions of nodes but only 2.8 millions of edges, which made it the sparest network among the datasets we used for experiment. First, we varied the value of T from 100 to 120 and plotted the results as in Fig. (5a) and Fig. (6a). SA performed much worse than LR, AT and IG. Its sizes of solutions were always at least 3 times greater than the others' and roughly near CC when T increased. Although CC always returned the worst solution, it was by far the fastest. The second best in terms of runtime was IG but it was always at least 15 times slower than CC. This number in LR and AT were 30 and 60 respectively. With T = 100, SA was slightly faster than LR and AT, therefore, we reduced the experimental range of T to [70, 100] on next experiment to observe the behaviors of SA. Interestingly, SA performed much more faster than AT and IG in this range.

In the experiment with heterogeneous weight functions, as can be seen in Fig. (5b) and Fig. (6b), SA ran up to 8 and 16 times faster than IG and AT respectively. However, SA's solutions were still worse than those two algorithms. From our observation, we found it hard to predict the behaviors of SA especially when the set $\mathcal{F}$ becomes larger. SA can perform well when this set is small and is very stable in a certain range of this set's size. But when it exceeds this range, SA's runtime increases at a higher rate than any other algorithms we have considered.

Finally, we evaluated the algorithms on the cutting scenario and reported the results in Fig. (5c) and Fig. (6c). Up to T = 91, IG and AT were the best in term of quality of solution but then were bypassed by TAG. In term of runtime, in most cases, our algorithms were 30 times slower than the fastest one, SAP.

The last network we did experiments on was Skitter, which is a dense graph where the average degree of a node is 6.5. In this experiment, we varied T in the range from 5 to 12. Fig. (3c) and Fig. (4c) show the performance of our algorithms. IG, AT and SA could finish within the limited runtime while CC was unable to run even at T = 7, which is why we did not show CC's results from T = 7 in those figures. Also, this experiment clearly shows the trade-off between IG and AT. The solution of IG was up to 1.25 times of AT while running faster with almost the same factor.

8.2.3 *Summary of results.* The experimental results can be summarized as follows.



(a) Linear (b) Heterogeneous (c) Cutting

Fig. 6. Runtime of algorithms on RoadnetCA

- In case of linear weight functions, LR always returned the best solution. The solution quality of IG and AT were worse than LR but usually by only a small factor. In addition, IG usually ran the fastest while the runtime of AT was more impacted by the varying of T than the other two.
- In general cases where LR is no more applicable, AT was always the algorithm that returned the best quality of solution. IG was competitive to AT only if the weight functions tent to be more concave. In trade-off, IG performed much more faster than AT in most experiments.
- In most experiments, SA was the worst among our algorithms in term of both the quality of solution and runtime. However, in several cases when the set of feasible paths was small or the input network was sparse, SA outperformed our other algorithms in runtime and the intuitive heuristics in solution quality.

9 CONCLUSION

In this work, we have introduced a new QoSD problem together with four solutions IG, AT, SA and LR, each of which scales to networks with millions of edges and nodes in under several hours and has a proven performance guarantee. Future work would include lowering the number of samples required by SA, making it more scalable. In addition, bounding the size of a set of candidate paths on IG and AT is necessary to reduce the burden on memory and waste of works when the candidate set is undesirable, and considering the correlation in increasing the edges' weights. Following that, we will investigate more on QoS degradation assessment on interdependent networks where networks are intertwined and interdependent, making the task of devising efficient algorithms much more challenging.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. We would also like to show our gratitude to Dr. Figueiredo (UFRJ) for shepherding our paper. This work is supported in part by NSF EFRI-1441231, NSF CNS-1814614, and DTRA HDTRA1-14-1-0055.

REFERENCES

[1] 2018. Blockchain: how a 51 percent attack works (double spend attack). https://medium.com/coinmonks/what-is-a-51-attack-or-double-spend-attack-aa108db63474. (2018). Accessed: 2019-01-03.
[2] 2018. Boole's inequality. https://en.wikipedia.org/wiki/Boole%27s_inequality. (2018). Accessed: 2018-10-26.
[3] 2018. Cauchy-Schwarz inequality. https://en.wikipedia.org/wiki/Cauchy%E2%80%93Schwarz_inequality. (2018). Accessed: 2018-10-26.

[4] 2018. Congestion Attack on CV-based Traffic Signal Control. (2018). https://sites.google.com/view/cav-sec/congestion-attack Accessed: 2018-10-26.

[5] 2018. Connected Vehicle Pilot Deployment Program. https://www.its.dot.gov/pilots/. (2018). Accessed: 2018-10-26.

[6] 2018. CV Pilot Deployment Program. https://www.its.dot.gov/pilots/cv_pilot_apps.htm. (2018). Accessed: 2018-10-26.

[7] 2018. LB-MULTICUT source code. (2018). Retrieved October 4, 2018 from https://gitlab.com/kuhnle/multi-pcut Accessed: 2018-10-26.

[8] 2018. Markov's inequality. https://en.wikipedia.org/wiki/Markov%27s_inequality. (2018). Accessed: 2018-10-26.

[9] 2019. Source code. https://github.com/lannn2410/qosd. (2019). Accessed: 2019-01-10.

[10] Amit Agarwal, Noga Alon, and Moses S Charikar. 2007. Improved approximation for directed cut problems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM, 671–680.

[11] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. 2017. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 375–392.

[12] Andrew An Bian, Joachim M Buhmann, Andreas Krause, and Sebastian Tschiatschek. 2017. Guarantees for greedy maximization of non-submodular functions with applications. *arXiv preprint arXiv:1703.02100* (2017).

[13] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D Sivakumar. 2006. On the hardness of approximating multicut and sparsest-cut. *computational complexity* 15, 2 (2006), 94–114.

[14] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, and others. 2011. Comprehensive experimental analyses of automotive attack surfaces.. In *USENIX Security Symposium*. San Francisco, 77–92.

[15] Qi Alfred Chen, Yucheng Yin, Yiheng Feng, Z Morley Mao, and Henry X Liu. 2018. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *Network and Distributed Systems Security (NDSS) Symposium 2018*.

[16] IBM ILOG CPLEX. 2009. V12. 1: User's Manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.

[17] Abhimanyu Das and David Kempe. 2011. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975* (2011).

[18] Richard Dennis, Gareth Owenson, and Benjamin Aziz. 2016. A temporal blockchain: a formal analysis. In *Collaboration Technologies and Systems (CTS), 2016 International Conference on*. IEEE, 430–437.

[19] Thang N Dinh and My T Thai. 2015. Assessing attack vulnerability in networks with uncertainty. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2380–2388.

[20] Thang N Dinh and My T Thai. 2015. Network under joint node and link attacks: Vulnerability assessment methods and analysis. *IEEE/ACM Transactions on Networking* 23, 3 (2015), 1001–1011.

[21] Thang N Dinh, My T Thai, and Hien T Nguyen. 2014. Bound and exact methods for assessing link vulnerability in complex networks. *Journal of Combinatorial Optimization* 28, 1 (2014), 3–24.

[22] Thang N Dinh, Ying Xuan, My T Thai, EK Park, and Taieb Znati. 2010. On Approximation of New Optimization Methods for Assessing Network Vulnerability.. In *INFOCOM*, Vol. 2010. 1–9.

[23] Paul Erdos and Alfréd Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.

[24] Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. 1996. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM J. Comput.* 25, 2 (1996), 235–251.

[25] Tony H Grubesic, Timothy C Matisziw, Alan T Murray, and Diane Snediker. 2008. Comparative approaches for assessing network vulnerability. *International Regional Science Review* 31, 1 (2008), 88–112.

[26] Anupam Gupta. 2003. Improved results for directed multicut. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Citeseer, 454–455.

[27] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.

[28] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and others. 2010. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 447–462.

[29] Alan Kuhnle, Victoria G Crawford, and My T Thai. 2018. Network Resilience and the Length-Bounded Multicut Problem: Reaching the Dynamic Billion-Scale with Guarantees. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 1 (2018), 4.

[30] Alan Kuhnle, J David Smith, Victoria G Crawford, and My T Thai. 2018. Fast Maximization of Non-Submodular, Monotonic Functions on the Integer Lattice. *arXiv preprint arXiv:1805.06990* (2018).

[31] Euiwoong Lee. 2016. Improved hardness for cut, interdiction, and firefighter problems. *arXiv preprint arXiv:1607.05133* (2016).

[32] Benny Lehmann, Daniel Lehmann, and Noam Nisan. 2006. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior* 55, 2 (2006), 270–296.

[33] Jure Leskovec and Andrej Krevl. 2018. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (2018). Accessed: 2018-10-26.

[34] Sahar Mazloom, Mohammad Rezaeirad, Aaron Hunter, and Damon McCoy. 2016. A Security Analysis of an In-Vehicle Infotainment and App Platform.. In *WOOT*.

[35] Subhankar Mishra, Xiang Li, My T Thai, and Jungtaek Seo. 2014. Cascading Critical Nodes Detection with Load Redistribution in Complex Systems. In *International Conference on Combinatorial Optimization and Applications*. Springer, 379–394.

[36] Dung T Nguyen, Yilin Shen, My T Thai, and others. 2013. Detecting Critical Nodes in Interdependent Power Networks for Vulnerability Assessment. *IEEE Trans. Smart Grid* 4, 1 (2013), 151–159.

[37] Tianyi Pan, Alan Kuhnle, Xiang Li, and My Thai. 2018. Vulnerability of Interdependent Networks with Heterogeneous Cascade Models and Timescales. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 290–299.

[38] Carlos Pinzón and Camilo Rocha. 2016. Double-spend attack models with time advantange for bitcoin. *Electronic Notes in Theoretical Computer Science* 329 (2016), 79–103.

[39] Ben Roberts and Dirk P Kroese. 2007. Estimating the Number of st Paths in a Graph. *J. Graph Algorithms Appl.* 11, 1 (2007), 195–214.

[40] Arunabha Sen, Sudheendra Murthy, and Sujogya Banerjee. 2009. Region-based connectivity-a new paradigm for design of fault-tolerant networks. In *High Performance Switching and Routing, 2009. HPSR 2009. International Conference on*. IEEE, 1–7.

[41] Yilin Shen, Thang N Dinh, and My T Thai. 2012. Adaptive algorithms for detecting critical links and nodes in dynamic networks.. In *MILCOM*. Citeseer, 1–6.

[42] Yilin Shen, Nam P Nguyen, Ying Xuan, and My T Thai. 2013. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Transactions on Networking* 21, 3 (2013), 963–973.

[43] Vijay V Vazirani. 2013. *Approximation algorithms*. Springer Science & Business Media.

APPENDIX

PROOF OF LEMMA 4.2

Let $\mathbf{s} = \{s_1, ..s_m\}$, since $\mathbf{s}$ is a unit vector, there is only one value among $s_1, ..s_m$ is 1 and the others are all 0. By extending the Equ. 12, we have

$$\Delta_{\mathbf{s}} g(\mathcal{P}, \mathbf{x}) = \sum_{p \in \mathcal{P}} \beta_p \Big(\min(\mathsf{T}, \sum_{e \in p} f_e(x_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(x_e)) \Big)$$

For each path p, we will prove that:

$$\begin{aligned} &\min(\mathsf{T}, \sum_{e \in p} f_e(x_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(x_e)) \\ &\geq \gamma \cdot (\min(\mathsf{T}, \sum_{e \in p} f_e(y_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(y_e))) \end{aligned}$$

We consider three cases:

- $\mathsf{T} \geq \sum_{e \in p} f_e(x_e + s_e) \geq \sum_{e \in p} f_e(x_e)$. Then we have

$$\begin{aligned} &\min(\mathsf{T}, \sum_{e \in p} f_e(x_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(x_e)) \\ &= \sum_{e \in p} f_e(x_e + s_e) - \sum_{e \in p} f_e(x_e) \\ &\geq \gamma \cdot \Big(\sum_{e \in p} f_e(y_e + s_e) - \sum_{e \in p} f_e(y_e) \Big) \\ &\geq \gamma \cdot \Big(\min(\mathsf{T}, \sum_{e \in p} f_e(y_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(y_e)) \Big) \end{aligned}$$

- $\sum_{e \in p} f_e(x_e + s_e) \geq \mathsf{T} \geq \sum_{e \in p} f_e(x_e)$. In this case,

$$\sum_{e \in p} f_e(y_e + s_e) \geq \sum_{e \in p} f_e(x_e + s_e) \geq \mathsf{T}$$

  also

$$min(\mathsf{T}, \sum_{e \in p} f_e(y_e)) \in [\sum_{e \in p} f_e(x_e), \mathsf{T}]$$

  Therefore:

$$\begin{aligned} &\min(\mathsf{T}, \sum_{e \in p} f_e(y_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(y_e)) \\ &= \mathsf{T} - \min(\mathsf{T}, \sum_{e \in p} f_e(y_e)) \\ &\leq \mathsf{T} - \sum_{e \in p} f_e(x_e) \\ &= \min(\mathsf{T}, \sum_{e \in p} f_e(x_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(x_e)) \end{aligned}$$

- $\sum_{e \in p} f_e(x_e + s_e) \geq \sum_{e \in p} f_e(x_e) \geq \mathsf{T}$. This case is trivial because both $\min(\mathsf{T}, \sum_{e \in p} f_e(y_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(y_e))$ and $\min(\mathsf{T}, \sum_{e \in p} f_e(x_e + s_e)) - \min(\mathsf{T}, \sum_{e \in p} f_e(x_e))$ are 0.

Hence, $\Delta_{\mathbf{s}} g(\mathcal{P}, \mathbf{x}) \geq \gamma \Delta_{\mathbf{s}} g(\mathcal{P}, \mathbf{y})$, which completes the proof.

PROOF OF THEOREM 4.4

Denote $\mathbf{x}^*$ is optimal solution to the QoSD instance. Define $\mathbf{x}_i$ as our obtained solution before the i^{th} iteration in Alg. 3. Denote $\mathbf{x}_i^o$ as an optimal solution that is in additional to $\mathbf{x}_i$ to block all paths in $\mathcal{P}$. We have:

$$||\mathbf{x}^*|| \geq ||\mathbf{x}^*/\mathbf{x}_i|| \geq ||\mathbf{x}_i^o|| \tag{37}$$

Assume $\mathbf{x}_i^o = \sum_{i=1}^{l} \mathbf{u}_i$ where $\mathbf{u}_i$ is a unit vector. We have:

$$\begin{aligned}
\mathsf{D}(\mathcal{P}, \mathbf{x}_i + \mathbf{x}_i^o) - \mathsf{D}(\mathcal{P}, \mathbf{x}_i) &= \sum_{j=1}^{l} \Delta_{\mathbf{u}_j} \mathsf{D}(\mathcal{P}, \mathbf{x}_i + \sum_{z=1}^{j-1} \mathbf{u}_z) && (38)\\
&\leq \frac{1}{\gamma} \sum_{j=1}^{l} \Delta_{\mathbf{u}_j} \mathsf{D}(\mathcal{P}, \mathbf{x}_i) \qquad \text{(Lemma 4.2)} && (39)\\
&\leq \frac{||\mathbf{x}_i^o||}{\gamma} \max_{\mathbf{s}} \Delta_{\mathbf{s}} \mathsf{D}(\mathcal{P}, \mathbf{x}_i) && (40)\\
&\leq \frac{\mathsf{OPT}}{\gamma} (\mathsf{D}(\mathcal{P}, \mathbf{x}_{i+1}) - \mathsf{D}(\mathcal{P}, \mathbf{x}_i)) && (41)\\
&= \frac{\mathsf{OPT}}{\gamma} (|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_i) - (|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_{i+1}))) && (42)
\end{aligned}$$

Equ. 41 follows by greedy selection. Since $\mathsf{D}(\mathcal{P}, \mathbf{x}_1 + \mathbf{x}_i^o) = |\mathcal{P}|\mathsf{T}$,

$$|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_{i+1}) \leq (1 - \frac{\gamma}{\mathsf{OPT}})(|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_i))$$

Note that the Alg. 3 will terminate after $||\mathbf{x}||$ iterations. Therefore:

$$\begin{aligned}
|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_{||\mathbf{x}||}) &\leq (1 - \frac{\gamma}{\mathsf{OPT}})(|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_{||\mathbf{x}||-1})) \leq \dots\\
&\leq (1 - \frac{\gamma}{\mathsf{OPT}})^{||\mathbf{x}||}(|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \{0\}^{|E|}))\\
&\leq (1 - \frac{\gamma}{\mathsf{OPT}})^{||\mathbf{x}||}|\mathcal{P}|\mathsf{T}
\end{aligned}$$

Since there should be at least a path $p \in \mathcal{P}$ whose overall delay is at most $\mathsf{T} - 1$ in final round, we have $|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_l) \geq 1$. Therefore:

$$||\mathbf{x}|| \leq \frac{\ln |\mathcal{P}|\mathsf{T}}{\ln \frac{1}{1 - \frac{\gamma}{\mathsf{OPT}}}} = \frac{\ln |\mathcal{P}|\mathsf{T}}{\ln(1 + \frac{\gamma/\mathsf{OPT}}{1-\gamma/\mathsf{OPT}})}$$

We have $\ln(1 + x) \geq x - \frac{x^2}{2}$ for $x \in (0, 1)$. So

$$||\mathbf{x}|| \leq \frac{\ln |\mathcal{P}|\mathsf{T}}{\frac{\gamma}{\mathsf{OPT}}(1 - \frac{\gamma}{2\mathsf{OPT}})} \leq \mathsf{OPT} \cdot O(\frac{\ln |\mathcal{P}|\mathsf{T}}{\gamma})$$

And since $|\mathcal{P}| \leq n^{\mathsf{h}}$, IG obtains $O(\frac{1}{\gamma}(\mathsf{h} \ln n + \ln \mathsf{T}))$ approximation guarantee, which completes the proof.

PROOF OF THEOREM 4.5

Denote $\mathbf{x}^* = \{x_1^*, ...x_m^*\}$ as optimal solution to the QoSD problem. Define $\mathbf{x}_i = \{x_1, ...x_m\}$ is our obtained solution before the i^{th} iteration in Alg. 4. Denote $\mathbf{x}_i^o = \{x_1^o, ...x_m^o\}$ as an optimal solution in additional to $\mathbf{x}_i$ to block all paths in $\mathcal{P}$. We have:

$$||\mathbf{x}^*|| \geq ||\mathbf{x}^*/\mathbf{x}_i|| \geq ||\mathbf{x}_i^o||$$

Denote $\mathbf{v}(e) = \{x_1, ..x_{e-1}, x_e + x_e^o, ...x_m + x_m^o\}$. Trivially, $\mathbf{v}(1) = \mathbf{x}_i + \mathbf{x}^o$ and $\mathbf{v}(m+1) = \mathbf{x}_i$. Assume $\mathbf{u}(e_i, j_i)$ is the vector we would add into solution $\mathbf{x}_i$ in iteration i^{th}. We have following lemma.

Lemma .1. *For all $e \in E$, we have:*

$$\frac{\Delta_{\mathbf{u}(e_i,j_i)}\mathsf{D}(\mathcal{P}, \mathbf{x}_i)}{j_i} \geq \frac{\mathsf{D}(\mathcal{P}, \mathbf{v}(e)) - \mathsf{D}(\mathcal{P}, \mathbf{v}(e+1))}{x_e^o}$$

Proof. Denote $\mathbf{w}(e) = \{x_1, ...x_{e-1}, x_e + x_e^o, x_{e+1}, ...x_m\}$. Consider a single path $p \in \mathcal{P}$, denote

$$h(p,s) = \sum_{e \in p \& e < s} f_e(x_e) + \sum_{e \in p \& e \geq s} f_e(x_e + x_e^o)$$
$$g(p,s) = \sum_{e \in p \& e \neq s} f_e(x_e) + f_e(x_s + x_s^o)$$

then we have:

$$\mathsf{r}(p, \mathbf{v}(s)) - \mathsf{r}(p, \mathbf{v}(s+1)) = \min(\mathsf{T}, h(p,s)) - \min(\mathsf{T}, h(p, s+1))$$
$$\mathsf{r}(p, \mathbf{w}(s)) - \mathsf{r}(p, \mathbf{x}_i) = \min(\mathsf{T}, g(p,s)) - \min(\mathsf{T}, \sum_{e \in p} f_e(x_e))$$

Trivially, we have that:

$$h(p,s) - h(p, s+1) = g(p,s) - \sum_{e \in p} f_e(x_e) = f_s(x_s + x_s^o) - f_s(x_s)$$

and due to monotonicity of $\mathsf{r}(p, \mathbf{x})$

$$h(p,s) \geq g(p,s)$$
$$h(p, s+1) \geq \sum_{e \in p} f_e(x_e)$$

Therefore, using the similar proof as lemma 4.2, we have:

$$\mathsf{D}(\mathcal{P}, \mathbf{v}(e)) - \mathsf{D}(\mathcal{P}, \mathbf{v}(e+1)) \leq \mathsf{D}(\mathcal{P}, \mathbf{w}(e)) - \mathsf{D}(\mathcal{P}, \mathbf{x}_i)$$
$$= \Delta_{\mathbf{u}(e, x_e^o)}\mathsf{D}(\mathcal{P}, \mathbf{x}_i)$$

Due to AT selection, we have that:

$$\frac{\Delta_{\mathbf{u}(e,x_e^o)}\mathsf{D}(\mathcal{P}, \mathbf{x}_i)}{x_e^o} \leq \frac{\Delta_{\mathbf{u}(e_i,j_i)}\mathsf{D}(\mathcal{P}, \mathbf{x}_i)}{j_i}$$

in which the lemma follows. □

Now, we will find the approximation guarantee of AT solution. We have:

$$\begin{aligned}\mathsf{D}(\mathcal{P}, \mathbf{x}_i + \mathbf{x}_i^o) - \mathsf{D}(\mathcal{P}, \mathbf{x}_i) &= \sum_e (\mathsf{D}(\mathcal{P}, \mathbf{v}(e)) - \mathsf{D}(\mathcal{P}, \mathbf{v}(e+1))) \\ &\leq \sum_e \frac{x_e^o}{j_i} \Delta_{\mathbf{u}(e_i, j_i)} \mathsf{D}(\mathcal{P}, \mathbf{x}_i) \\ &\leq \frac{\mathsf{OPT}}{j_i} (\mathsf{D}(\mathcal{P}, \mathbf{x}_{i+1}) - \mathsf{D}(\mathcal{P}, \mathbf{x}_i))\end{aligned}$$

Since $\mathsf{D}(\mathcal{P}, \mathbf{x}_i + \mathbf{x}_i^o) = |\mathcal{P}|\mathsf{T}$, we have

$$|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_{i+1}) \leq (1 - \frac{j_i}{\mathsf{OPT}})(|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_i))$$

Assume AT stops after l iterations, we have

$$|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \mathbf{x}_l) \leq \prod_{i=1}^{l} (1 - \frac{j_i}{\mathsf{OPT}})(|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \{0\})) \tag{43}$$

$$\leq \left(1 - \frac{\sum_{i=1}^{l} j_i}{l \cdot \mathsf{OPT}}\right)^l (|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \{0\})) \tag{44}$$

$$\leq e^{-\frac{||\mathbf{x}||}{\mathsf{OPT}}} (|\mathcal{P}|\mathsf{T} - \mathsf{D}(\mathcal{P}, \{0\})) \tag{45}$$

Equ. 44 comes from the following Cauchy theorem

Theorem .2. (Cauchy Theorem *[3]) Given n non-negative numbers* $x_1, ...x_n$, *we have*

$$\prod_{i=1}^{n} x_i \leq (\frac{\sum_{i=1}^{n} x_i}{n})^n$$

Equ. 45 comes from observation that $(1 - \frac{x}{n})^n \leq e^{-x}$.

Therefore, $||\mathbf{x}||_1 \leq \mathsf{OPT} \ln |\mathcal{P}|\mathsf{T}$. Since $|\mathcal{P}|$ is bounded by n^{h}, AT obtains $O(\mathsf{h} \log n + \log \mathsf{T})$ approximation guarantee.

PROOF OF LEMMA 5.2

Denote $\mathbf{v}_i$ as the budget vector $\mathbf{v}$ after greedily selecting first i unit vectors, then by monotonicity $\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) \leq \hat{D}(\mathcal{P}, \mathbf{v} + \mathbf{v}^o + \mathbf{v}_i)$. We have

$$\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) \leq \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o + \mathbf{v}_i) \tag{46}$$

$$= \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i) + \sum_{j=1}^{q} \Delta_{\mathbf{u}_j} \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i + \sum_{i=1}^{j-1} \mathbf{u}_i) \tag{47}$$

$$\leq \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i) + \frac{1}{\gamma} \sum_{j=1}^{q} \Delta_{\mathbf{u}_j} \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i) \tag{48}$$

$$\leq \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i) + \frac{q}{\gamma} (\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_{i+1}) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i)) \tag{49}$$

The inequality (49) is due to greedy selection. Therefore,

$$\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_{i+1}) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i) \geq \frac{\gamma}{q} (\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i))$$

Which also means

$$\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_{i+1}) \\ \leq (1 - \frac{\gamma}{q})(\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}_i))$$

Therefore

$$\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}) \leq (1 - \frac{\gamma}{q})^q (\hat{B}(\mathcal{P}, \mathbf{x} + \mathbf{v}^o) - \hat{B}(\mathcal{P}, \mathbf{x}))$$

So

$$\Delta_{\mathbf{v}} \hat{B}(\mathcal{P}, \mathbf{x}) \geq (1 - (1 - \frac{\gamma}{q})^q) \Delta_{\mathbf{v}^o} \hat{B}(\mathcal{P}, \mathbf{x}) \\ \geq (1 - e^{-\gamma}) \Delta_{\mathbf{v}^o} \hat{B}(\mathcal{P}, \mathbf{x})$$

which completes the proof.

PROOF OF THEOREM 5.7

First, considering the greedy selection $\mathbf{v}$ in each sampling iteration, from lemma 5.6, we have

$$\Delta_{\mathbf{v}} B(\mathbf{x}) \geq (1 - e^{-\gamma})(1 - \epsilon) \Delta_{\mathbf{v}^*} B(\mathbf{x})$$

with probability at least $1 - \delta/||\mathsf{b}||$.

Denote $\mathbf{x}^o = \sum_i^s \mathbf{u}_i$ as an optimal solution, which is additional to $\mathbf{x}$, can block all paths in $\mathcal{F}$ ($\mathbf{u}_i$ is a unit vector). Let split $\mathbf{x}^o$ into $l = \lceil \frac{||\mathbf{x}^o||}{q} \rceil$ parts $L_1, ...L_l$ where $L_i = \sum_{j=(i-1)q+1}^{iq} \mathbf{u}_j$, we have:

$$\Delta_{\mathbf{x}^o} B(\mathbf{x}) = \sum_{j=1}^{l} \Delta_{L_j} B(\mathbf{x} + L_1 + ... + L_{j-1}) \leq \frac{1}{\gamma} \sum_{j=1}^{l} \Delta_{L_j} B(\mathbf{x})$$

Therefore, there should be at least a value $\Delta_{L_j} B(\mathbf{x}) \geq \frac{\gamma}{l} \Delta_{\mathbf{x}^o} B(\mathbf{x})$. And since $||L_j|| \leq q$, we have:

$$\Delta_{\mathbf{v}} B(\mathbf{x}) \geq \frac{\gamma}{l}(1 - e^{-\gamma})(1 - \epsilon) \Delta_{\mathbf{x}^o} B(\mathbf{x})$$

which also means

$$|\mathcal{F}|\mathsf{T} - B(\mathbf{x} + \mathbf{v}) \leq (1 - \frac{q\gamma}{\mathsf{OPT}}(1 - e^{-\gamma})(1 - \epsilon))(|\mathcal{F}|\mathsf{T} - B(\mathbf{x}))$$

Now, denote $\mathbf{x}_i$ as our solution after the i^{th} iteration of Alg. 5. We have

$$|\mathcal{F}|\mathsf{T} - B(\mathbf{x}_{i+1}) \leq (1 - \frac{q\gamma}{\mathsf{OPT}}(1 - e^{-\gamma})(1 - \epsilon))(|\mathcal{F}|\mathsf{T} - B(\mathbf{x}_i))$$

Assume the algorithm terminates after g iterations, we have:

$$|\mathcal{F}|\mathsf{T} - B(\mathbf{x}_g) \leq (1 - \frac{q\gamma}{\mathsf{OPT}}(1 - e^{-\gamma})(1 - \epsilon))(|\mathcal{F}|\mathsf{T} - B(\mathbf{x}_{g-1})) \leq ... \\ \leq \left(1 - \frac{q\gamma}{\mathsf{OPT}}(1 - e^{-\gamma})(1 - \epsilon)\right)^g (|\mathcal{F}|\mathsf{T} - B(\{0\}^m))$$

Each inequality happens with probability at least $1 - \frac{\delta}{||\mathsf{b}||}$. So the probability such that $|\mathcal{F}|\mathsf{T} - B(\mathbf{x}_g) \leq \left(1 - \frac{q\gamma}{\mathsf{OPT}}(1 - e^{-\gamma})(1 - \epsilon)\right)^g (|\mathcal{F}|\mathsf{T} - B(\{0\}^m))$ is at least $1 - \frac{\delta g}{||\mathsf{b}||} \geq 1 - \delta$. Moreover, since in the g^{th} iteration, there should exist a path $p \in \mathcal{F}$, whose length smaller than T. So, the maximum length of p is $\mathsf{T} - 1$. Therefore

$$1 \leq (1 - \frac{q\gamma}{\mathsf{OPT}}(1 - e^{-\gamma})(1 - \epsilon))^g |\mathcal{F}| \mathsf{T}$$

So $g \leq O(\frac{\ln \mathsf{T} + h \ln d}{q\gamma(1-e^{-\gamma})(1-\epsilon)})\mathsf{OPT}$. Since in each iteration, a budget vector $\mathbf{v}$, $||\mathbf{v}|| \leq q$ is added into solution, out final solution guarantees $O(\frac{\ln \mathsf{T} + h \ln d}{\gamma(1-e^{-\gamma})(1-\epsilon)})$ approximation ratio with probability at least $1 - \delta$.

Received December 2018; revised January 2019; accepted February 2019