Budgeted Reliability Maximization in Uncertain Graphs

Xiangyu Ke[1], Arijit Khan[1], Mohammad Al Hasan[2], Rojin Rezvansangsari[1]
Nanyang Technological University, Singapore[1]
Indiana University - Purdue University Indianapolis, USA[2]
{xiangyu001, arijit.khan, N1600122F}@ntu.edu.sg[1], alhasan@cs.iupui.edu[2]

ABSTRACT

Network reliability measures the probability that a target node is reachable from a source node in an uncertain graph, i.e., a graph where every edge is associated with a probability of existence. In this paper, we investigate the novel and fundamental problem of adding a small number of edges in the uncertain network for maximizing the reliability between a given pair of nodes. We study the $\mathbf{NP}$-hardness and the approximation hardness of our problem, and design effective, scalable solutions. Furthermore, we consider extended versions of our problem (e.g., multiple source and target nodes can be provided as input) to support and demonstrate a wider family of queries and applications, including sensor network reliability maximization and social influence maximization. Experimental results validate the effectiveness and efficiency of the proposed algorithms.

PVLDB Reference Format:
X. Ke, A. Khan, M. Al Hasan R. Rezvansangsari. Budgeted Reliability Maximization in Uncertain Graphs. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/TBD

1. INTRODUCTION

Rich expressiveness of probabilistic graphs and their utility to model the inherent uncertainty in a wide range of applications have prompted a large number of research works on probabilistic graphs by the data management research communities. In recent years, researchers in this community have proposed efficient algorithms for solving several interesting problems, such as, finding k-nearest neighbors [50], answering reachability queries [23], designing networks [57], and computing most reliable paths [28]—all in an uncertain graph setting. Uncertainty in a graph arises due to many reasons, including noisy measurements of an edge metric [2], edge imputation using inference and prediction models [1, 34], and explicit manipulation of edges e.g., for privacy purposes [7].

In an uncertain graph setting, *Network Reliability* is a well-studied problem [52, 12, 51, 5, 56], which requires to measure the probability that a target node is reachable from a source node. Reliability has been widely studied in device networks, i.e., networks whose nodes are electronic devices and the (physical) links between such devices have a probability of failure [3]. More recently, the attention has been shifted to social, communication, transportation, genomic, and logistic networks [26, 21, 20, 22]. Applications of reliability estimation include computing the packet delivery probability from a source to a sink node in a wireless sensor network, measuring information diffusion probability from an early adopter to a target customer in a social influence network, predicting new interactions by finding all proteins that are evidently (i.e., with high probability) reachable from a core (source) set of proteins in a protein-interaction network, as well as estimating on-demand delivery probability via different routes from an inventory to warehouses or customers in a road network, among many others.

In this paper, we investigate the novel problem of adding a small number of edges in an uncertain network for maximizing the reliability between a given pair of nodes. We refer to such edges as *shortcut edges* and the problem of identifying the best set of k edges as the *budgeted reliability maximization* problem. Our problem falls under the broad category of uncertain networks design [57], optimization [49], and modification [45] problems, yet surprisingly this specific problem has not been studied in the past.

The budgeted reliability maximization problem is critical in the context of many physical networks, such as transportation and communication networks. In mobile ad-hoc networks, the connectivity between sensor nodes and devices is estimated using noisy measurements, thus leading to edges naturally associated with a probability of existence [20]. Road networks can be modeled as uncertain graphs because of unexpected traffic congestion [21]. In these networks, creating new connections between nodes (e.g., building new roads, flyovers, adding Ethernet cables) is limited by physical constraints and budget. One can introduce only k new edges where k is decided based on resource constraints. Thus, our goal is to intelligently add k new edges such that the reliability between a pair of important nodes is maximized [57]. Furthermore, in social networks, finding k best shortcut edges could maximize the information diffusion probability from an early adopter to a target customer [53, 10, 27], thus the network host can actively recommend these links to the respective users. In case of protein-interaction networks, interactions are established for a limited number of proteins through noisy and error-prone experiments—each edge is associated with a probability accounting for the existence of the interaction. Therefore, finding the top-k shortcut edges can assist in de-noising protein-interaction networks [33].

Case study. To demonstrate the effectiveness of our problem, we conduct a case study on the *Intel Lab Data* (`http://db.csail.mit.edu/labdata/labdata.html`). This dataset contains the sensor network information with 54 sensors deployed in the Intel Berkeley Research Lab (map given in Figures 1 and 2) between February 28th and April 5th, 2004. The probabilities on links denote the percentages of messages from a sender successfully reached to a receiver. The average link probability is 0.33 (ignoring edge probabilities which are lower than 0.1).

Proceedings of the VLDB Endowment, Vol. 12, No. xxx

DOI: https://doi.org/TBD

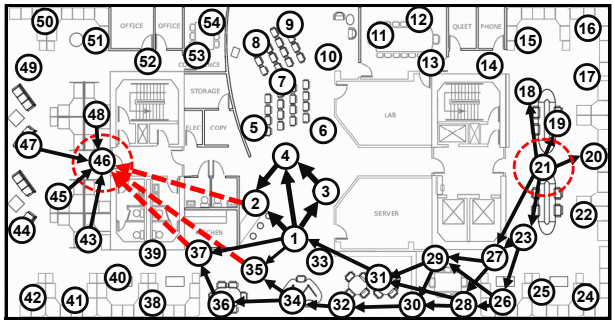


Figure 1: Improving the reliability from sensor 21 (right) to 46 (left) with 3 new links (marked by dotted lines).

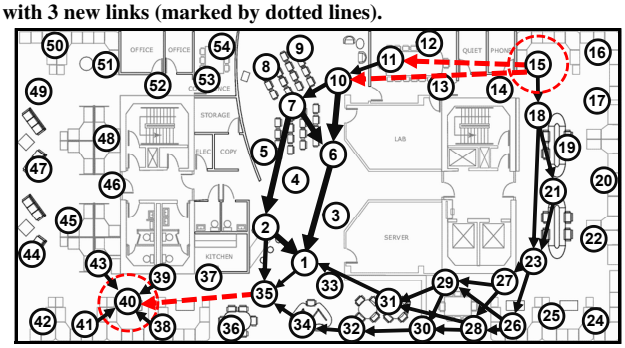


Figure 2: Improving the reliability from sensor 15 to 40 (on the diagonal) with 3 new links (marked by dotted lines).

Assume that our goal is to maximize the reliability from: (1) a sensor on the right hand side of the lab to a sensor on the left hand side (e.g., from sensor 21 to 46); (2) between two sensors on the diagonal of the lab (e.g., from sensor 15 to 40). The original reliability from sensor 21 to 46 is *0.40*, and that from sensor 15 to 40 is *0.28*. Due to budget constraints, only 3 new links are allowed for each case. We further assume that the probability of each new link would be the same as the average edge probability of the original dataset, that is, 0.33. We notice that if two sensors are more than 20 meters away, the original link probability between them is usually close to 0. Thus, we only allow establishing new links between a pair of sensors that are at most 15 meters away.

Figure 1 demonstrates the solution obtained by our algorithm for case (1). Only those links with probabilities higher than the average value (0.33) are shown in the figure, and the thickness represents link probabilities. Clearly, sensor 46 has very weak connections from outside, while sensor 21 is connected with the bottom part of the lab. A very dense network exists in the bottom part of the lab. Therefore, our solution for this case is to connect sensor 46 with sensors in the bottom part of the lab. By establishing three links 2 to 46, 35 to 46, and 37 to 46, we improve the reliability between 21 to 46 *from 0.40 to 0.88* (i.e., the new reliability is more than twice the original reliability from 21 to 46).

For case (2), notice in Figure 2 that the sensors in the center part of lab are well-connected, and the links in this region are thicker than those in the bottom part. Sensor 15 (the source node) has a few connections with sensors in the bottom part, but no connection with sensors in the center part. Sensor 40 (the destination node) has limited connections beyond its physical neighbors. Existing configuration offers a poor reliability of 0.28 for the connection between source and destination which we like to improve. The smart decision made by our algorithm is as follows: First, connect sensor 35 to 40, thus making sensor 35 a bridge between the center and the bottom region of the network; Second, enable connection from sensor 15 to the center part (by establishing link from 15 to 10, and from 15 to 11). This results in *0.58* overall reliability from sensor 15 to 40, which is more than double of the original reliability value. These results illustrate how our proposed solution for the budgeted reliability maximization problem can be useful in solving real-life problems. More applications can be found in §8.4.

Challenges and contributions. Unfortunately, budgeted reliability maximization problem is non-trivial. In fact, a simpler problem, which is to compute the exact reliability over uncertain graphs is $\#\mathbf{P}$-complete [56, 5]. Our thorough investigation of the budgeted reliability maximization problem have yielded the following theoretical results: First, we prove that, even assuming polynomial-time sampling methods to estimate reliability (such as, Monte Carlo sampling [16], or more sophisticated recursive stratified sampling [36]), our problem of computing a set of k shortcut edges that maximizes the reliability between two nodes remains $\mathbf{NP}$-hard; Second, the budgeted reliability maximization problem is not easy to approximate, as **(i)** it does not admit any $\mathbf{PTAS}$, and **(ii)** the underlying objective function is neither submodular, nor supermodular. The above pessimistic results are useful to comprehend the computation challenges associated with finding even an approximate solution to this problem, let alone an optimal solution. For instance, lack of submodular (or supermodular) property prevents us from using an iterative hill-climbing based greedy algorithm that maximizes the marginal gain at every iteration to obtain a solution with approximation guarantees. Moreover, a hill-climbing algorithm would be quite inefficient due to repeated computation of marginal gains for *all* candidate edges (which are, in fact, missing edges in the input graph, and can be $\mathcal{O}(n^2)$ in numbers for a sparse graph) at every iteration.

By considering the computation challenges as we have discussed above, in this paper we propose a practical algorithm for budgeted reliability maximization problem. Our proposed solution systematically minimizes the search space by only considering missing edges between nodes that have reasonably high reliability from the source node and to the target node. Next, we extract several highly-reliable paths between source and target nodes, after including those limited number of candidate edges in the input graph. This is motivated by the observation that what really matters in computing the reliability between two nodes is the set of paths connecting source to target, not the individual edges in the graph [29, 11, 31, 32]. Our algorithm then iteratively selects these paths so as to achieve maximum improvement in reliability while satisfying the constraint on the number of new edges (k) to be added.

We also consider a *restricted* version of our problem, which approximates the reliability by considering *only* the most reliable path between the source and the target node [31, 11, 32]. We prove that improving the probability of the most reliable path can be solved exactly in polynomial time, which yields an efficient algorithm for the restricted version of our problem. Finally, after studying the budgeted reliability maximization problem for a single source-target pair, we focus on *generalizations* where multiple source and target nodes can be provided as input, thus opening the stage to a wider family of queries and applications, e.g., network modification for targeted influence maximization [53, 10, 27, 25].

The main contributions of this paper are as follows.

- We study the novel and fundamental problem of maximizing the reliability between a given pair of nodes by adding a small number of edges in an uncertain graph. Our problem is $\mathbf{NP}$-hard, and is also hard to approximate, even when polynomial-time reliability estimation is employed (§2).
- We design effective and efficient solutions for our problem. The proposed algorithms first apply reliability-based search space elimination, then fill the remaining graph with missing edges, and finally select the top-k edges to add based on most reliable paths (§5).

- We further consider a restricted and one extended version of our problem to support a wider family of queries. In the restricted version, the reliability is estimated only by the most reliable path, thus it can be solved exactly in polynomial-time. In the extended version, multiple sources and targets can be provided as input. The proposed algorithms are generalized to multiple-source-target case. (§4 and §6).
- We conduct a thorough experimental evaluation with several real-world graphs (both social and device networks) to demonstrate the effectiveness, efficiency, and scalability of our algorithms, and illustrate the usefulness of our problem in critical applications such as sensor network reliability maximization and influence maximization in social networks (§8).

2. PRELIMINARIES

2.1 Problem Formulation

An uncertain graph $\mathcal{G}$ is a triple (V, E, p), where V is a set of n nodes, $E \subseteq V \times V$ is a set of m directed edges, and $p(e) \in [0, 1]$ is the probability that the edge $e \in E$ exists. Following bulk of the literature on uncertain graphs [56, 5, 23, 29, 50, 30], we assume that edge probabilities are independent of each other. Therefore, we employ the well-established *possible world* semantics: The uncertain graph $\mathcal{G}$ yields 2^m deterministic graphs $G \sqsubseteq \mathcal{G}$. Each possible world $G = (V, E_G)$ is a certain instance of the uncertain graph $\mathcal{G}$, where $E_G \subseteq E$ and is obtained by independent sampling of the edges. Its probability of being observed is given as:

$$Pr(G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)) \quad (1)$$

Given a source node $s \in V$, and a target node $t \in V$, the *reliability* $R(s, t, \mathcal{G})$, also known as the s-t reliability, is defined as the probability that t is reachable from s in $\mathcal{G}$. Formally, for a possible graph $G \sqsubseteq \mathcal{G}$, let $I_G(s, t)$ be an indicator function taking the value 1 if there exists a path from s to t in G, and 0 otherwise. $R(s, t, \mathcal{G})$ is computed as follows.

$$R(s, t, \mathcal{G}) = R(s, t, (V, E, p)) = \sum_{G \sqsubseteq \mathcal{G}} [I_G(s, t) \times Pr(G)] \quad (2)$$

The problem that we study in this work is stated below.

PROBLEM 1. [Single-source-target budgeted reliability maximization] *Given an uncertain graph $\mathcal{G} = (V, E, p)$, a source node $s \in V$, a target node $t \in V$, a probability threshold $\zeta \in (0, 1]$, and a small positive integer k, find the top-k edges to add in $\mathcal{G}$, each with probability $p(e) = \zeta$, such that the reliability from s to t is maximized.*

$$E^* = \underset{E_1 \subseteq V \times V \setminus E}{\arg\max} R(s, t, (V, E \cup E_1, p))$$

$$\text{s. t.} \quad |E_1| = k; \quad \text{and} \quad p(e) = \zeta \quad \forall e \in E_1 \quad (3)$$

For simplicity, we adopt a fixed probability threshold ζ on new edges. The intuition is that when establishing a new edge, generally we consider a connection with the best/average possible reliability, e.g., an Ethernet cable with the highest reliability in case of LAN, or the average link reliability of sensor edges as we have used in the discussed case study. However, note that, if the user provides probability values for the missing edges as part of input, our proposed algorithm (§5) will work smoothly as it can simply use those values instead of ζ when finding the most reliable paths (see our experimental evaluation in Table 8.2, §8.2).

Remarks. Due to various physical and resource constraints, in practice, it might not be possible to consider all missing edges in the input graph as candidate edges for our problem. In a social network it is often realistic to recommend new connections between users who are within 2-3 hops. In a communication network, a new edge can be added only if the two nodes are within a certain geographical distance. While we analyze the complexity of our problem and develop algorithms for the generalized case, that is, all missing edges can potentially be candidate edges, in our solution as well as in experiments we provision for a threshold distance h: Two nodes can be added by a new edge only if they are within h-hops away. Note that **(1)** when h is the diameter of the graph (i.e., maximum shortest-path distance between any pair of nodes), this is essentially equivalent to the generalized case. **(2)** Smaller values of h reduces search space, thereby improving efficiency. In our experiments, we analyze scalability of our methods for different values of h.

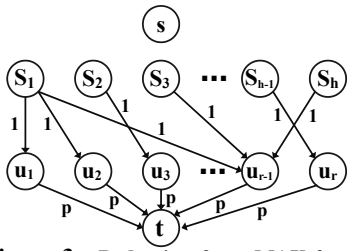


Figure 3: Reduction from MAX k-COVER to single s-t budgeted reliability maximization problem

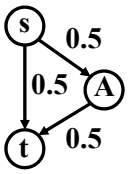


Figure 4: Example for non-submodularity and non-supermodularity

2.2 Hardness of the Problem

Problem 1 depends on reliability computation in uncertain graphs, which is $\#\mathbf{P}$-complete [56, 5]. Thus, single-source-target budgeted reliability maximization problem is hard as well. However, as reliability can be *estimated* in polynomial time via Monte Carlo (MC) sampling [16], or more sophisticated recursive stratified sampling [36], the key question is whether Problem 1 remains hard even if polynomial-time reliability estimation methods are employed. Due to combinatorial nature of our problem, and assuming $\mathcal{O}(n^2)$ missing edges in a sparse graph, one can design an *exact* solution that compares the s-t reliability gain for $\binom{n^2}{k}$ possible ways of adding k new edges, and then reports the best one. However, this is clearly infeasible for large networks. We, in fact, prove that our problem is $\mathbf{NP}$-hard, and it does not admit any $\mathbf{PTAS}$. Moreover, Problem 1 is neither submodular, nor supermodular with respect to the inclusion of edges.

THEOREM 1. *Problem 1 is $\mathbf{NP}$-hard in the number of newly added edges, k.*

PROOF. We prove $\mathbf{NP}$-hardness by a reduction from the MAX k-COVER problem, which is $\mathbf{NP}$-hard. In MAX k-COVER, there is a collection of subsets $S = \{S_1, S_2, ..., S_h\}$ of a ground set $U = \{u_1, u_2, ..., u_r\}$, where $S_i \subseteq U$ for all $i \in [1...h]$. The objective is to find a subset $S^* \subset S$ of size k such that maximum number of elements in U can be covered by S^*, i.e., so as to maximize $|\bigcup_{S_i \in S^*} S_i|$. For an instance of MAX k-COVER, we construct an instance of our Problem 1 in polynomial time as follows (Figure 3).

We create an uncertain graph $\mathcal{G}$ with a source node s and a target node t. For each element u_i in U, we add a node in $\mathcal{G}$. Each u_i is connected to t by an edge with probability p, such that $0 < p < 1$. Then, we also add each S_i in S as a node of $\mathcal{G}$. Node S_i is connected to node u_j with probability 1 if and only if $u_j \in S_i$. All other edges in $\mathcal{G}$ except those from s to all S_i have probability 0.

Therefore, the candidate set of edges to add in $\mathcal{G}$ for maximizing reliability from s to t are those edges from s to all S_i. Without them, there is no path from s to t with non-zero probability. Let $\zeta = 1$, after k of these edges are selected, q out of r elements in U are now reachable from s, then the s-t reliability $= 1 - (1 - p)^q$, which monotonically increases with larger q. This implies that

Table 1: Reliability gains of three possible solutions for the example in Figure 5 under different probability setting.

α	ζ	Reliability		
		$\{sA, sB\}$	$\{sA, Bt\}$	$\{sB, Bt\}$
0.5	0.7	0.403	0.473	**0.543**
0.5	0.3	**0.203**	0.173	0.143
0.9	0.7	**0.800**	0.674	0.660

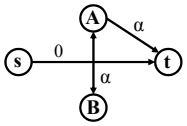


Figure 5: Example for the problem characterization

Problem 1 and MAX k-COVER are equivalent here. If there exists a polynomial time solution for Problem 1, the MAX k-COVER can be solved in polynomial time too. The theorem follows. □

Moreover, Problem 1 is also hard to approximate.

THEOREM 2. *Problem 1 does not admit any* **PTAS**, *unless* **P** =**NP**.

PROOF. See Appendix. □

We further show that neither submodularity nor supermodularity holds for the objective function of Problem 1, and demonstrate with the following counter example. Therefore, standard greedy hill-climbing algorithms do not directly come with approximation guarantees for Problem 1.

LEMMA 1. *The objective function of Problem 1 is neither submodular, nor supermodular w.r.t inclusion of edges.*

For any set $X \subseteq Y$ and all elements $x \notin Y$, a set function f is submodular if $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$. For supermodularity, the inequality is reversed.

Let us consider the example in Figure 4: s is the source node and t is the target nodes. Assume the node set $V = \{s, A, t\}$. Let $X = \{st\}, Y = \{st, sA\}$ be two edge sets. We have $R(s,t,(V,X,p)) = R(s,t,(V,Y,p)) = 0.5$. We find that $R(s,t,(V,X \cup \{At\},p)) = 0.5, R(s,t,(V,Y \cup \{At\},p)) = 1-(1-0.5)[1-0.5^2] = 0.625$. Clearly, submodularity does not hold in this example.

Next, considering $X' = \{sA\}$ and $Y' = \{sA, st\}$, we have $R(s,t,(V,X',p)) = 0$ and $R(s,t,(V,Y',p)) = 0.5$. Then, $R(s,t,(V,X' \cup \{At\},p)) = 0.25, R(s,t,\ (V,Y' \cup \{At\},p)) = 0.625$. Therefore, supermodularity also does not hold.

2.3 Characterization of the Problem

We next show in Observations 1∼3 that the optimal solution to our problem varies based on most input parameters, even if the other set of input parameters remains the same, thereby making it non-trivial to utilize pre-existing solutions of past queries, as well as indexing-based or incremental methods.

OBSERVATION 1. *The optimal solution for Problem 1 may vary with different input probability threshold ζ.*

OBSERVATION 2. *The optimal solution for Problem 1 may vary when the edge probabilities in the original graph change.*

OBSERVATION 3. *When $k_1 < k_2$, the optimal solution for Problem 1 with k_1 may not be a subset of that with k_2.*

All these three observation can be demonstrated with the example given in Figure 5, as follows.

EXAMPLE 1. *In Figure 5, there are edges AB and At, both with probability α ($0 < \alpha < 1$), in this graph. And the edge directly connecting s and t can not exist, e.g., no direct flight can be established between two airports if they are too far away. Clearly, original reliability between s and t is 0. $\{sA, sB, Bt\}$ is the candidate set of edges to add for improving the s-t reliability.*

If budget $k = 1$, $\{sA\}$ is always the optimal solution. Its reliability is $\alpha\zeta$, which is larger than both $\alpha^2\zeta$ for solution $\{sB\}$ and 0 for solution $\{Bt\}$.

If budget $k = 2$, the s-t reliability of 3 possible solutions, $\{sA, sB\}$, $\{sA, Bt\}$, and $\{sB, Bt\}$. The reliability between s and t after adding them can be calculated as follows:

$$R(s,t,(V,E \cup \{sA,sB\},p)) = [1-(1-\zeta)\cdot(1-\alpha\zeta)]\cdot\alpha$$
$$R(s,t,(V,E \cup \{sA,Bt\},p)) = \zeta\cdot[1-(1-\alpha)(1-\alpha\cdot\zeta)]$$
$$R(s,t,(V,E \cup \{sB,Bt\},p)) = \zeta\cdot[1-(1-\zeta)(1-\alpha^2)]$$

Table 1 presents the reliability of these solutions with different α and ζ. Clearly, rows 1 and 2 have same α and different ζ, and their optimal solutions are different: $\{sA, sB\}$ and $\{sB, Bt\}$, respectively. This confirms our Observation 1. Similarly, we have same ζ but different α in rows 1 and 3, and obtain different optimal solutions. Therefore, we draw Observation 2. Moreover, $\{sA\}$, the optimal solution when $k = 1$, is not a subset of the optimal solution $\{sB, Bt\}$ when $k = 2$ if $\alpha = 0.5$, $\zeta = 0.7$, which implies our Observation 3.

Finally, we conclude this section with an interesting observation below: The direct edge st, if missing in the input graph, will always be in the top-k optimal solution. In other words, when the direct st edge is missing and if it can be added, for the top-1 solution, adding the direct st edge is the best solution.

OBSERVATION 4. *If the direct edge from s to t, st, is missing in the input graph, and is allowed to be added, st will always be included in the top-k optimal solution.*

PROOF. Let G be a possible world (i.e., deterministic graph) of the original uncertain graph $\mathcal{G}$. Following Equation 2, the s-t reliability is calculated as: $\sum_{G \sqsubseteq \mathcal{G}}[I_G(s,t) \times Pr(G)]$. After adding k missing edges, G will partition into 2^k new possible worlds: $\{G_1, G_2, \ldots, G_{2^k}\}$. $Pr(G) = \sum_{i=0}^{2^k} Pr(G_i)$. Clearly, when t is reachable from s in G, it will still be reachable from s in each of $\{G_1, G_2, ..., G_{2^k}\}$, thus $I_{G_i}(s,t)$ will continue to be 1. Therefore, we only investigate those G containing no path from s to t, and thus reachability in a deterministic graph can be can be improved by adding new edges.

Suppose $\{e_1, e_2, ..., e_k\}$ is an optimal solution without having st. For any G_i in $\{G_1, G_2, ..., G_{2^k}\}$ obtained from some G (such that t was originally not reachable from s in G), we consider another solution by replacing e_j ($1 \leq j \leq k$) with st. e_j can have two possible status in G_i: **(1)** e_j exists in G_i. Here, I_{G_i} may or may not be 1. However, when replacing e_j with st, I_{G_i} will always return 1, and induces reachability improvement; **(2)** e_j is absent in G_i. Then, the value of I_{G_i} depends only on other edges in the solution set. Replacing e_j with st will not impact the reachability. Therefore, replacing e_j with st will result in a new solution which has reliability gain at least as large as the earlier one. This implies that st, if allowed, can always be added in the optimal solution. □

3. BASELINE METHODS

In this section, we first present several baseline methods, that are straightforward, and demonstrate how they suffer from both effectiveness and efficiency issues. The discussions will be instrumental in developing a more accurate and scalable solution in § 4 and 5.

3.1 Individual Top-k Method

In the most straightforward approach, we consider every candidate edge one by one, check the reliability gain due to its addition in the input graph with edge probability ζ, and select the top-k edges with highest individual reliability gains.

Time complexity. The reliability can be estimated in polynomial time via Monte Carlo (MC) sampling. It samples a set of Z deterministic graphs from the input uncertain graph, and estimates the reliability of an s-t pair as ratio of samples (i.e., a possible world) in

which the target is reachable from the source. The reachability in a deterministic graph can be evaluated via breadth first search (BFS) in time $\mathcal{O}(n+m)$, where n and m denote the number of nodes and edges in the input graph, respectively. Thus, the time complexity of MC sampling for each newly added edge is $\mathcal{O}(Z(n+m))$. Since real-world networks are generally sparse, the number of candidate edges is nearly $\mathcal{O}(n^2)$. Therefore, the overall complexity of individual top-k baseline is: $\mathcal{O}(n^2Z(n+m)+n^2\log k)$, where the last term is due to top-k search.

Shortcomings. (1) To achieve reasonable accuracy, MC sampling requires around thousands of samples [23, 26]. Performing this for $\mathcal{O}(n^2)$ times is not scalable for large graphs. **(2)** Once an edge is added into the input graph, the reliability gain of adding other candidate edges may change. Hence, selecting the top-k edges based on individual reliability gains results in low-quality solution.

3.2 Hill Climbing Method

A better-quality solution would be the hill climbing algorithm: It greedily adds the edge that provides the maximum marginal gain to the s-t reliability at the current round, until total k new edges have been selected. In particular, consider that a set $E_1 \subseteq V \times V \setminus E$ of new edges have been already included, in the next iteration the hill climbing baseline selects a new edge $e \in V \times V \setminus (E \cup E_1)$, with $p(e) = \zeta$, such that:

$$e^* = \underset{e \in V \times V \setminus (E \cup E_1)}{\arg\max} [R(s,t,(V,E \cup E_1 \cup \{e\},p)) - R(s,t,(V,E \cup E_1,p))] \quad (4)$$

Since Problem 1 is neither submodular nor supermodular, this approach does not provide approximation guarantees.

Time complexity. Assuming the number of missing edges to be $\mathcal{O}(n^2)$, coupled with MC sampling, the time complexity of each iteration of the hill climbing approach is $\mathcal{O}(n^2Z(n+m))$. For total k iterations, overall complexity is $\mathcal{O}(n^2kZ(n+m))$.

Shortcomings. Hill climbing also suffers from efficiency and accuracy issues. **(1)** This is more inefficient compared to individual top-k baseline. **(2)** In terms of accuracy, hill climbing still suffers from the *cold start* problem: At initial rounds, there would be several new edges with marginal reliability gain zero (or, quite small), resulting in random selections, which in turn produces sub-optimal solutions at later stages.

3.3 Centrality-based Method

Another intuitive approach is to find highly central nodes in the input graph, and connect them by new edges if they are not already connected, until the budget k on new edges is exhausted. In particular, we consider degree-based centrality, that is, nodes having higher aggregated edge probabilities considering all incoming and outgoing edges. Such nodes are also known as the *hub nodes*: Connecting these hub nodes help in reducing network distances (as well as improving reliability over uncertain graphs).

Time complexity. The algorithm requires going through all nodes and checking their in/out going edges, which costs $\mathcal{O}(m+n)$ time. Then, it ranks the nodes based on their aggregated edge probabilities, which consumes $\mathcal{O}(n\log n)$ time. Finally, we connect the top-k pairs of nodes which are not already connected. Thus, total complexity: $\mathcal{O}(m+n\log n)$.

Shortcomings. Although the method is efficient, and in general improves the s-t reliability, it is not customized for a specific s-t pair. This often results in low-quality solution.

3.4 Eigenvalue-based Method

Wang et al. [58] studied the importance of the largest eigenvalue of graph topology in the dissemination process over real networks. To model the virus propagation in a network, they assumed a fixed infection rate β for an infected node to pass the virus to its neighbor, and another fixed curing rate δ for an infected node. Then, they proved that if $\frac{\beta}{\delta} < \frac{1}{\lambda}$, where λ is the largest eigenvalue of the adjacency matrix of this network, the virus will die out in this network. Therefore, one can optimize the leading eigenvalue to control the virus dissemination in a network, e.g., with smaller λ, smaller curing rate δ is required for the same infecting rate β. Recently, Chen et al. studied the problem of maximizing the largest eigenvalue of a network by edge-addition [10], which is discussed in the Appendix.

Time complexity. The overall time complexity of Eigenvalue-based optimization in [10] is $\mathcal{O}(m+nt+kt^2)$, where $t = max(k, d_{in}, d_{out})$. We discuss it in details in the Appendix.

Shortcomings. (1) This method is not customized for a specific s-t pair, and may report low-quality solutions. **(2)** To the best of our knowledge, there is no equivalent transformation from virus propagation threshold $\frac{\beta}{\delta}$ to the s-t reliability. Therefore, maximizing the leading eigenvalue (to improve virus propagation) may not be equivalent to maximizing the s-t reliability.

4. A SIMPLIFIED PROBLEM: IMPROVE THE MOST RELIABLE PATH

Due to limitations of baseline approaches as discussed in § 3, we now explore an orthogonal direction following the notion of the *most reliable path*. The idea that we shall develop in this section will be the basis of our ultimate solution (to be introduced in § 5) for the budgeted reliability maximization problem. A path between a source s and a target node t in an uncertain graph $\mathcal{G}$ is called the most reliable path $MRP(s,t,\mathcal{G})$ if the probability of that path (i.e., product of edge probabilities on that path) is maximum in comparison with all other paths between these two nodes.

$$MRP(s,t,\mathcal{G}) = \underset{P \in \mathcal{P}(s,t,\mathcal{G})}{\arg\max} \prod_{e \in P} p(e) \quad (5)$$

$\mathcal{P}(s,t,\mathcal{G})$ denotes the set of all paths from s to t in $\mathcal{G}$. The problem that we investigate here is a simplified version of our original problem (i.e., Problem 1) as stated next.

PROBLEM 2. [Single-source-target most reliable path improvement] *Given an uncertain graph $\mathcal{G} = (V,E,p)$, a source $s \in V$, a target $t \in V$, a probability threshold $\zeta \in (0,1]$, and a small positive integer k, find the top-k edges to add in $\mathcal{G}$, each new edge e having probability $p(e) = \zeta$, such that the probability of the most reliable path from s to t in the updated graph is maximized.*

$$E^* = \underset{E_1 \subseteq V \times V \setminus E}{\arg\max} \prod_{e \in MRP(s,t,(V,E \cup E_1,p))} p(e)$$
$$s.t. \quad |E_1| = k; \quad and \quad p(e) = \zeta \quad \forall e \in E_1 \quad (6)$$

Notice that the probability of the most reliable path from s to t cannot be larger than the s-t reliability. Thus, Problem 2 might be considered as a *simplified* version of the budgeted reliability maximization problem. Nevertheless, as reported in earlier studies [32, 31, 11], the most reliable path often provides a good approximation to the reliability between a pair of nodes. Thus, our intuition is simple: If the solution of Problem 2 is more efficient and results in higher-quality top-k edges (compared to baselines for the original budgeted reliability maximization problem), then we can augment this idea (e.g., instead of the most reliable path, one may consider multiple highly-reliable paths from s to t) to develop even better-quality solution for the budged reliability maximization problem.

Fortunately, Problem 2 can be solved exactly in polynomial time. We shall provide a constructive proof, which can also be used as an algorithm for Problem 2.

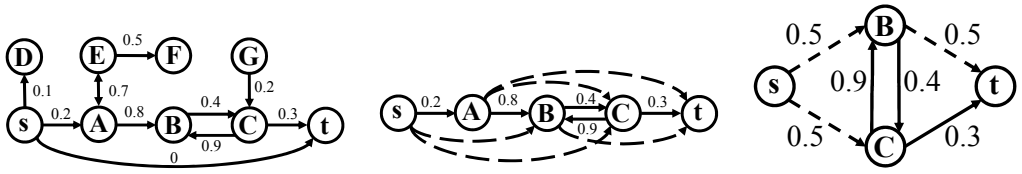
THEOREM 3. *Problem 2 can be solved exactly in polynomial time.*

Table 2: Reliability gain and running time comparison without search space elimination. $k = 10$, $\zeta = 0.5$, *lastFM*

Method	Reliability Gain	Running Time (sec)
Individual Top-k	0.27	39184
Hill Climbing	**0.32**	**406512**
Centrality-based	0.03	19
Eigenvalue-based	0.09	213
Most Reliable Path	**0.26**	**467**
Individual Path Inclusion (proposed method)	0.29	332
Batch-edge Selection (proposed method)	0.31	421

Table 3: Reliability gain and running time comparison after search space elimination. $k = 10$, $\zeta = 0.5$, $l = 30$, $r = 100$, *lastFM*. Additional time for search space elimination: 16 sec.

Method	Reliability Gain	Running Time (sec)
Individual Top-k	0.27	136
Hill Climbing	0.31	1256
Centrality-based	0.13	5
Eigenvalue-based	0.20	19
Most Reliable Path	0.25	20
Individual Path Inclusion (proposed method)	**0.30**	**16**
Batch-edge Selection (proposed method)	**0.33**	**22**



(a) Input graph (b) Adding relevant candidate edges (c) Select top-l reliable paths

Figure 6: Run-through example for the proposed algorithm

PROOF. First, we color all existing edges in the input graph $\mathcal{G}$ as blue. Then, we add all missing edges to the graph, each having edge probability ζ (thus, resulting in a complete graph), and color these new edges as red. Name this new graph as $\overline{\mathcal{G}}$. The goal of Problem 2 is to find the most reliable path from s to t containing at most k red edges (can have zero or more blue edges), if any. Notice that we can convert the uncertain graph $\overline{\mathcal{G}}$ into an weighted graph G_0, which has same set of edges and nodes as $\overline{\mathcal{G}}$, and the weight of each edge e in G_0 is: $w(e) = -\log p(e)$. Equivalently, we aim at finding the shortest path from s to t in G_0 containing at most k red edges (can have zero or more blue edges), if any.

To find such paths, we make k identical copies of G_0. So, we have graphs $G_0, G_1, G_2, \ldots, G_k$; and we update them as follows.

1. Remove all red edges from G_k.
2. For every $0 \leq i \leq k-1$
   For every red edge $e_j = (v_a, v_b)$ in G_i
   Remove e_j from G_i
   Draw a new edge from v_a in G_i to v_b in G_{i+1}

Now, we employ the Dijkstra's algorithm to find the shortest paths from s in G_0 to every t in G_i ($0 \leq i \leq k$). Each shortest path from s in G_0 to t in G_i corresponds to a path in the original graph $\overline{\mathcal{G}}$ with at most i red edges . We refer to these paths (if they exist) as $P_0, P_1, \ldots, P_k$, respectively.

Consider a function W that gets as input a path, and returns the aggregate weight of edges on that path. If for every $1 \leq i \leq k$, we have $W(P_0) \leq W(P_i)$, then this means that adding no $k' \leq k$ edges to $\mathcal{G}$ can improve the probability of the most reliable path from s to t. Otherwise, we find $P = \arg\min_{1 \leq i \leq k} W(P_i)$, and consider all red edges in P. Adding these edges to $\mathcal{G}$ will result in the maximum probability of the most reliable path from s to t.

The time required for the above method is due to running the Dijkstra's algorithm for $k+1$ times over a graph with $(k+1)n$ nodes and $(k+1)n^2$ edges. Hence, the overall time complexity of our method is $\mathcal{O}(k^2n^2 + k^2 n \log(kn))$, which is polynomial in input size. The theorem follows. □

Comparison with baselines. As shown in Table 2, solving the simplified most reliable path problem (Problem 2) is much faster than both baselines: Individual Top-k and Hill Climbing for the original problem (Problem 1). As expected, the improvement in s-t reliability via most reliable path-based solution is 0.26, which is lower but comparable to that of Hill Climbing: 0.32. However, the most reliable path approach terminates in 467 seconds, while Hill Climbing consumes about 4.7 days. For other two baselines, Centrality-based and Eigenvalue-based, most reliable path technique significantly outperforms them in reliability gain.

5. PROPOSED SOLUTION: BUDGETED RELIABILITY MAXIMIZATION

In this section, we present the method that we ultimately design for an effective and efficient solution to the single-source-target budgeted reliability maximization (Problem 1). Due to the success of the most reliable path technique as detailed in § 4, our final solution is developed based on a similar notion by employing multiple reliable paths, and further improved in two ways: **(1)** Reduction of search space by identifying only the most relevant candidate edges for a given s-t pair, and **(2)** improving solution quality by considering *multiple* highly reliable paths from s to t. In the following, we discuss various steps of our framework, and demonstrate accuracy and efficiency improvements compared to previous baselines.

5.1 Search Space Elimination

5.1.1 Reliability-based Search Space Elimination

In a sparse input graph $\mathcal{G}$, one can have as many as $\mathcal{O}(n^2)$ candidate edges. However, given a specific s-t query, all candidate edges may not be equally relevant. In particular, let us consider two nodes u and v: Both have low reliability either from source s, or to target t; then adding an edge between u and v will not improve the s-t reliability significantly. Therefore, we select "relevant" candidate edges as follows. **(1)** We find the top-r nodes with the highest reliability from s. Similarly, we compute the top-r nodes having the highest reliability to t. Let us refer to these sets as $C(s)$ and $C(t)$, respectively. Notice that $s \in C(s)$ and $t \in C(t)$. **(2)** For two distinct nodes u, v, such that $u \in C(s)$, $v \in C(t)$, and u, v are not connected in the input graph $\mathcal{G}$, then we consider the new edge (u, v), with edge probability $p(u, v) = \zeta$, as a candidate edge. We denote by E^+ the set of relevant candidate edges. Thus, we reduce the number of candidate edges from $\mathcal{O}(n^2)$ to only $\mathcal{O}(r^2)$.

The time complexity of this step is $\mathcal{O}(Z(n+m) + n \log r + r^2)$. The first term is due to MC sampling to compute the reliability of all nodes from s and to t, and the second term is due to sorting all nodes based on these reliability values.

5.1.2 Top-l Most-Reliable Paths Selection

Given the success of the most reliable path-based approach (§ 4), we further improve it by considering multiple highly reliable paths from s to t. Recent research has shown that what really matters

in computing the reliability between two nodes is the set of highly reliable paths connecting them [29, 11, 31, 32].

On adding the relevant candidate edges E^+, we refer to the updated graph as $\mathcal{G}^+ = (V, E \cup E^+, p)$. Next, by applying the Eppstein's algorithm [15, 29], we find the top-l most reliable paths from s to t, with running time $\mathcal{O}(m + n \log n + l)$. If a new edge does not appear in any of these top-l paths, it can be removed from E^+. This further reduces the search space.

EXAMPLE 2. *Let us demonstrate "search space elimination" with Figure 6. Suppose we set $r = 3$, $l = 3$, and $\zeta = 0.5$. First, we select the top-3 nodes with highest reliability from source s. Clearly, $\{s, A, B\}$ will be selected. Similarly, $\{B, C, t\}$ are the top-3 nodes with highest reliability to target t. Node D, E, F, and G will be eliminated, and we obtain a graph presented in Figure 6(b). Then, we select top-3 most reliable paths between s and t after adding all missing edges (dotted lines) with given probability $\zeta = 0.5$ in Figure 6(b). They will be $\{sBt, sCBt, sCt\}$ (in decreasing order). Node A does not appear in any of these paths, and will be eliminated. Finally, we have a simplified graph shown in Figure 6(c).*

Benefits of search space elimination. As shown in Table 3, our search space elimination methods can save about 99% of running time for the baselines: Individual Top-k and Hill Climbing without accuracy loss. For Centrality-based and Eigenvalue-based baselines, both efficiency and accuracy get improved, because these baselines are now applied over a smaller and more relevant (to a specific s-t pair) subgraph. After including the time cost for conducting search space elimination: 16 seconds, the overall running time for most reliable path method and our proposed algorithms can be reduced by over 70% without accuracy loss.

5.2 Top-k Edges Selection

After reducing the number of candidate edges, our next objective is to find the top-k edges from the reduced set E^+ of candidate edges, so to maximize the s-t reliability within the given budget. We formulate the problem as follows.

PROBLEM 3. [Budgeted Path Selection] *Given the set $\mathcal{P}$ of the top-l most reliable paths from s to t in the updated graph $\mathcal{G}^+$, find a path set $\mathcal{P}^* \subseteq \mathcal{P}$ such that:*

$$\mathcal{P}^* = \arg\max_{\mathcal{P}_1 \subseteq \mathcal{P}} R(s, t, \mathcal{P}_1)$$
$$\text{s. t.} \qquad |\{e : e \in E^+ \cap \mathcal{P}_1\}| \leq k \tag{7}$$

In the budgeted path selection problem, $R(s, t, \mathcal{P}_1)$ denotes the s-t reliability on the subgraph induced by the path set $\mathcal{P}_1$. In other words, we find a path set $\mathcal{P}^* \subseteq \mathcal{P}$ that maximize the s-t reliability, while also satisfying the constraint on k, the number of newly-added edges. Unfortunately, this problem is **NP**-hard as well, which can be proved by a reduction from the MAX k-COVER. Since the proof is analogous to the one in Theorem 1, we omit this for brevity. Instead, we design two practical and effective solutions for the budgeted path selection problem as given below.

5.2.1 Individual Path-based Edge Selection

First, we combine all paths from $\mathcal{P}$ that do not have any candidate edges from E^+. We refer to these paths as $\mathcal{P}_1$, and the subgraph induced by $\mathcal{P}_1$ as G^*. Then, in each successive round, we iteratively include a remaining path P^* from $\mathcal{P} \setminus \mathcal{P}_1$ into G^* which maximally increases the reliability (estimated via MC-sampling) from s to t in G^*, while still maintaining the budget k on the number of included candidate edges in G^*. It can be formulated as:

$$P^* = \arg\max_{P \in \mathcal{P} \setminus \mathcal{P}_1} R(s, t, \mathcal{P}_1 \cup \{P\}) \tag{8}$$

During this procedure, we ensure that the number of included candidate edges from E^+ in $\mathcal{P}_1$ does not exceed k. The included candidate edges in G^* are reported as our solution.

Let us denote by n' and m' the number of nodes and edges, respectively, in the subgraph induced by the top-l most-reliable path set $\mathcal{P}$, and T the number of MC samples required in each iteration. At most, we need k iterations, which implies that overall time complexity is $\mathcal{O}(kZ|P|(n' + m'))$.

5.2.2 Path Batches-based Edge Selection

The effectiveness of individual path selection can be improved by considering the relationships between paths in $\mathcal{P}$. The intuitions are: (1) different paths can share same set of candidate edges; (2) the candidate edge set of a path can be a subset of that for another path; and (3) different paths may have different number of candidate edges to be included.

Therefore, we design a path batch-based (instead of individual path-based) edge selection algorithm. First, we go through all paths in $\mathcal{P}$. If two paths share same set of candidate edges, they shall be put into the same "path batch". Each path batch is labeled by its candidate edge set, and in our algorithm we include a path batch in every round. When evaluating the marginal gain of a path batch, all other path batches whose candidate edge set is a subset of it, shall also be included in G^* in the current round. The marginal gain of this batch is normalized by the size of its candidate edge set. The detailed procedure is shown in the following example.

EXAMPLE 3. *Consider Example 2 and Figure 6, we are now selecting top-2 edges from 3 candidate edges $\{sB, sC, Bt\}$. If selecting paths individually, path sBt has the highest marginal gain 0.25 and will be selected in the first round. As budget $k = 2$ is exhausted, the solution set is $\{sB, Bt\}$. However, path sCt has reliability gain 0.15, and only adds 1 new edge. Its marginal gain per new edge is higher than that of sBt. Further, by considering it in batch path selection manner, including path $sCBt$ will also activate path sBt The reliability gain of adding them in batch is 0.3075, and the marginal gain per new edge is 0.1538, which is the winner of this round, and we find the optimal solution $\{sC, Bt\}$ in this example. The reliability gains for the other 2 possible solution are 0.28 for $\{sB, Bt\}$, 0.18 for $\{sB, sC\}$. This demonstrates the effectiveness of path batch selection procedure.*

Benefits of path batches-based edge selection. As shown in Table 3, path batch selection and Hill Climbing have similar reliability gain, while path batch selection significantly outperforms Hill Climbing in running time. Comparing with individual path inclusion, path batch selection has some improvement in reliability gain, with comparable running time.

5.3 Improvement via Advanced Sampling

Recently, several advanced sampling methods have been proposed for estimating s-t reliability, including lazy propagation [37], recursive sampling [23], recursive stratified sampling (RSS) [36], and probabilistic tree [41]. While our problem and the proposed solution are orthogonal to the specific sampling method used, its efficiency can further be improved by employing more sophisticated sampling strategies. In particular, instead of MC sampling, we shall consider RSS in the experiments, both for our proposed method and for the baselines. The detailed description and the analysis of RSS can be found in the Appendix.

6. MULTIPLE-SOURCE-TARGET RELIABILITY MAXIMIZATION

In practice, queries may consist of multiple source and/or target nodes, rather than a single s-t pair. For example, in targeted marketing [31, 38, 25, 35] via social networks, the campaigner wants to maximize the information diffusion from a group of early adopters

to a set of target customers. For such real-world applications, we extend our problem to adapt to multiple source/target nodes. In particular, we focus on maximizing an aggregate function (e.g., average, maximum, minimum) over reliability of *all* source-target pairs.

PROBLEM 4. [Multiple-source-target budgeted reliability maximization] *Given an uncertain graph $\mathcal{G} = (V, E, p)$, a set of source nodes $S \subset V$, a set of target nodes $T \subset V$, a probability threshold $\zeta \in (0, 1]$, and a small positive integer k, find the top-k edges to add in $\mathcal{G}$, with each new edge having probability $p(e) = \zeta$, such that an aggregate function F over reliability of all s-t pairs ($s \in S$, $t \in T$) is maximized.*

$$E^* = \underset{E_1 \subseteq V \times V \setminus E}{\arg\max} \mathop{\mathbb{F}}_{s,t \in S \times T} (R(s, t, (V, E \cup E_1, p)))$$
$$\text{s. t.} \quad |E_1| = k; \quad \text{and} \quad p(e) = \zeta \quad \forall e \in E_1 \tag{9}$$

Due to **NP**-hardness of Problem 1, its generalization, Problem 4 is also **NP**-hard. In the following sections, we consider three widely-used aggregate functions: average, minimum, maximum; and design efficient solutions.

6.1 Maximizing the Average Reliability

Our objective is:

$$\underset{E_1 \subseteq V \times V \setminus E}{\arg\max} \frac{1}{|S||T|} \sum_{s,t \in S \times T} R(s, t, (V, E \cup E_1, p)) \tag{10}$$

Note that this is equivalent to maximizing the sum of reliability of all s-t pairs. From the perspective of targeted marketing in social networks, a campaigner would like to maximize the spread of information to the entire target group; and therefore, she would prefer to maximize the average reliability.

Similar to the single-source-target budgeted reliability maximization problem, we first compute the reliable sets from source and target nodes, that is, $C(s)$ for all $s \in S$, and $C(t)$ for all $t \in T$. Next, for each pair of distinct nodes u, v, such that $u \in C(s), \forall s \in S$, $v \in C(t), \forall t \in T$, and u, v are not connected in the input graph $\mathcal{G}$, we consider a new edge (u, v), having edge probability $p(u, v) = \zeta$, as a relevant candidate edge. We denote by E^+ the set of relevant candidate edges, and after adding them to $\mathcal{G}$, we refer to the updated graph as $\mathcal{G}^+ = (V, E \cup E^+, p)$.

Now, for each s-t pair, we identify the top-l most reliable paths in $\mathcal{G}^+$. Then we have total $|S||T|l$ paths in this set, and the path set might contain more than k new edges. Therefore, we employ the path batches-based edge selection method (§ 5.2.2): The algorithm iteratively includes path batches that maximize the marginal gain considering our current objective function (Equation 10), while maintaining the budget k on the number of newly inserted edges.

Time complexity. For simplicity, let $\mathcal{O}(P_1)$ denote the time complexity of reliability-based search space elimination, $\mathcal{O}(P_2)$ denote that of top-l most-reliable paths selection, and $\mathcal{O}(P_3)$ denote that of path batches-based edge selection, for the single-source-target problem. The time complexity of the proposed algorithm for average multiple-source-target budgeted reliability maximization problem is $\mathcal{O}((|S| + |T|)P_1 + |S||T|(P_2 + P_3))$. We need to evaluate all nodes' reliability from/to each source/target, which results in the first term. The second term is due to applying top-l path selection algorithm $|S||T|$ times for each s-t pair, and the path set size will be $|S||T|$ times of that for single-source-target problem.

6.2 Maximizing the Minimum Reliability

Our objective is:

$$\underset{E_1 \subseteq V \times V \setminus E}{\arg\max} \min_{s,t \in S \times T} R(s, t, (V, E \cup E_1, p)) \tag{11}$$

In other words, we aim at including k new edges such that the reliability of the s-t pair having the lowest reliability (after the addition of k edges) is maximized. In the targeted marketing setting, this can happen during complementary influence maximization [40], where multiple products are being campaigned simultaneously, and they are complementary in nature: Buying a product could boost the probability of buying another. Now, consider that each source node (e.g., an early adopter) is campaigning a different, but complementary product. In this situation, the campaigner would prefer to maximize the minimum spread of her campaign from any of the early adopters to any of her target users, because only a small percentage of the users who have heard about a campaign will buy the corresponding product.

To solve this problem, we first estimate the s-t reliability for each pair in $S \times T$ over the input graph $\mathcal{G}$. We sort these s-t pairs in ascending order in a priority queue based on their current reliability. Next, in successive rounds, we keep improving the reliability of the pair having the smallest current reliability, until the budget on k new edges can be exhausted. In particular, at any point in our algorithm, we know which source-target pair has the minimum reliability. We extract this pair from the top of the priority queue, and improve its reliability with the addition of a batch of suitable, new edges. For this purpose, we employ our algorithm for the single-source-target pair (discussed in § 5). The batch size can be set as $k_1 << k$. Note that the addition of new edges not only updates the reliability of the current pair, instead this will also increase the reliability of other s-t pairs.

Thus, after adding a batch of k_1 new edges, we re-compute the reliability of all s-t pairs and re-organize them in the priority queue. Once again, we extract the pair from the top of the priority queue, and improve its reliability with the addition of k_1 suitable, new edges. We repeat the above steps. Ultimately, we terminate the algorithm when we exhaust our budget of adding total k new edges.

Time complexity. The time complexity of our algorithm is $\mathcal{O}((|S| + |T|)P_1 + \frac{k}{k_1}AP_1 + \frac{k}{k_1}P_2 + P_3)$, $A = min(|S|, |T|)$. Similar to maximizing the average reliability, we need evaluating all nodes' reliability from/to each source/target. All s-t pair's original reliability can also be known through this process. However, after improving the reliability of the currently selected s-t pair by adding k_1 edges, we need to update the reliability of all s-t pairs. This will happen $\frac{k}{k_1}$ times, and the cost is $\frac{k}{k_1}AP_1$, $A = min(|S|, |T|)$. The top-l paths selection will be operated $\frac{k}{k_1}$ times. The complexity of executing top-k_1 edge selection is $\mathcal{O}(\frac{k_1}{k}P_3)$, and it will happen $\frac{k}{k_1}$ times. Thus, the total time cost for edge selection remains $\mathcal{O}(P_3)$.

6.3 Maximizing the Maximum Reliability

Our objective function is:

$$\underset{E_1 \subseteq V \times V \setminus E}{\arg\max} \max_{s,t \in S \times T} R(s, t, (V, E \cup E_1, p)) \tag{12}$$

In the targeted marketing scenario, let us again consider complementary influence maximization [40], where each source user (e.g., an early adopter) is campaigning a different, but complementary product. However, each target user is now a celebrity in Twitter. Hence, the campaigner wants at least one target user to be influenced by one of her products. In other words, the campaigner would be willing to maximize the spread of information from at least one early adopter to at least one target customer.

Note that if $S \cap T \neq \phi$, the problem is trivial, as the maximum reliability is already one. Therefore, below we consider the case when $S \cap T = \phi$. A straightforward solution to our problem would be to separately consider each s-t pair from $S \times T$, improve its reliability by adding k new, suitable edges. Then, we pick the pair which achieves the maximum final reliability, and report those k new edges that were selected for this s-t pair. However, the time

complexity of this approach is $\mathcal{O}(|S||T|)$ times to that of a single $s-t$ pair.

Next, we develop a more efficient algorithm without significantly affecting the quality. Our approach is similar to that of maximizing the minimum reliability (discussed in § 6.2). In each round, we maximize the reliability (by adding $k_1 << k$ new edges) of the pair having the current maximum reliability. After this, we re-compute the reliability of all pairs, and again pick the one which has the current maximum reliability. We terminate the algorithm when we exhaust our budget of adding total k new edges.

Time complexity. The time complexity will be the same as that of maximizing the minimal reliability, which is $\mathcal{O}((|S|+|T|)P_1 + \frac{k}{k_1}AP_1 + \frac{k}{k_1}P_2 + P_3)$, $A = min(|S|, |T|)$.

7. RELATED WORK

Network design problems. Network design, optimization, and modification are widely studied research topics, where one modifies the network structure or attributes, targeting at some objective metrics or functions.

There exist many different metrics to characterize the "goodness" of the network, including average shortest paths [4, 6, 44], ratio of connected nodes [55], relative size of the largest connected component and average size of other components [4], network flow and delay [42, 43], the diameter of a network [14], centrality [24, 48] or eccentricity [14], average path length [46], and spectral measures [13, 8, 10]. The last group are derived from the adjacency and the Laplacian matrices of a graph. For example, [10] optimized the leading eigenvalue of a network by edge addition/deletion, due to the finding that the leading eigenvalue of the underlying graph is the key metric in determining the so-called "epidemic threshold" for a variety of dissemination models [58]. However, such global metric is not query-specific. In real-world, users may tend to optimize the network in a way that is relevant *only* to themselves, e.g., a campaigner would like to improve the influence of her product to her target customers, but not that of all similar products (from other competitors), and neither to other users who are not her targets. Moreover, many network metrics studied in the past cannot be easily generalized to probabilistic scenarios (e.g., connected component size). Our objective, *reliability*, is a fundamental metric to capture the probability that a given target node is reachable from a specific source node in an uncertain graph. Furthermore, we show that it is possible to generalize our objective to multiple-source-target cases in order characterize a larger region in the network.

The major network manipulation operations include node addition/deletion [4, 6], edge addition/deletion [10, 44, 14], edge rewiring [8, 6], and updating edge weights [43]. Our goal is to improve the reliability between s-t pairs in a network: In our application scenarios, adding new edges is usually more practical. For example, it is often not realistic to set up a new airport only to improve the reliability of connections between two existing airports, rather establishing some new flights is much easier. In this paper, we study the problem of maximizing the reliability between a given pair of nodes by adding a small number of new edges. Altering the existing edge probabilities is not investigated here, and can be an interesting future research direction on this problem.

Reliability in uncertain networks. Due to the **#P**-hardness of s-t reliability estimation problem, various efficient sampling approaches have been proposed in the literature. Monte Carlo (MC) sampling [16] is a fundamental approach, which samples Z possible worlds from the input uncertain graph, and approximates the s-t reliability with the ratio of possible world in which t is reachable from s. One may combine MC sampling with BFS from the source node to further improve its efficiency [23, 28]. [37] proposed Lazy Propagation, which utilizes geometric distribution to avoid frequent probing of edges. BFSSharing improves the efficiency with offline indexes. Recursive sampling [23] and recursive stratified sampling [36] reduces the estimator variance by recursively partitioning the search space. Less samples are required for them to achieve the same variance as previous methods, thereby improving the efficiency. More recently, ProbTree index [41] was designed to support faster s-t reliability queries over uncertain graphs. Our problem and the proposed solution are orthogonal to the specific sampling method used, we have demonstrated in § 5 and the Appendix that its efficiency can be improved by employing recursive stratified sampling.

Some orthogonal directions to our problem include finding one "good" possible world [47, 54], adaptive edge testing [19, 17, 18] and crowdsourcing [39] for reducing uncertainty. In this work, we focus on improving the reliability of a source-target pair by adding a limited number of new edges.

Table 4: Properties of datasets

Dataset	#Nodes	#Edges	Edge Prob: Mean, SD, Quartiles	Type
Intel Lab Data	54	969	0.33 ± 0.19, $\{0.16, 0.27, 0.44\}$	Device
LastFM	6 899	23 696	0.29 ± 0.25, $\{0.13, 0.20, 0.33\}$	Social
AS_Topology	45 535	172 294	0.23 ± 0.20, $\{0.08, 0.21, 0.31\}$	Device
DBLP	1 291 298	7 123 632	0.11 ± 0.09, $\{0.05, 0.10, 0.14\}$	Social
Twitter	6 294 565	11 063 034	0.14 ± 0.10, $\{0.10, 0.10, 0.19\}$	Social

8. EXPERIMENTAL RESULTS

We perform experiments to demonstrate effectiveness (i.e., reliability improvement), efficiency, scalability, and memory usage of our algorithms. We report sensitivity analysis by varying all input parameters in this section, and provide more sensitivity tests in the Appendix. The code is implemented in C++, executed on a single core, 40GB, 2.40GHz Xeon server.

8.1 Experimental Setup

Datasets. We use five real-world graphs, consisting of three social and one device networks (Table 4). **(1)***Intel Lab Data* (`http://db.csail.mit.edu/labdata/labdata.html`). It is a collection of sensor communication data with 54 sensors deployed in the Intel Berkeley Research Lab between February 28th and April 5th, 2004. **(2)** *LastFM* (`www.last.fm`). It is a musical social network, where users listen to their favorite musics, and share with their friends. We crawl the music-listening history record, and form a bi-directed graph, where nodes represent users, and an edge exists if two users communicate at least once. **(3)** *AS_Topology* (`http://data.caida.org/datasets/topology/ark/ipv4/`). An autonomous system (AS) is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators on behalf of a single administrative entity, e.g., a university. The AS connections are established with BGP protocol. It may fail due to various reasons, e.g., failure of physical links when one AS updates its connection configuration to ensure stricter security setting, while some of its peers can no longer satisfy it, or some connections are cancelled manually by the AS administrator. We downloaded one network snapshot per month, from January 2008 to December 2017. **(4)** *DBLP* (`https://dblp.uni-trier.de/xml/`). It is a well-known collaboration network. We downloaded it on March 31, 2017. Each node is an author and edges denote their co-author relations. **(5)** *Twitter* (`http://snap.stanford.edu/data/`). This is a widely used social network: Nodes are users and edges are re-tweets.

Edge probability models. Our problems and solutions are *orthogonal to the specific way of assigning edge probabilities*. We adopt some widely-used models for generating edge probabilities in our evaluation. **(1)** *Intel Lab Data* and **(3)** *AS_Topology*. The edge probabilities in these two datasets are real probabilities. For *Intel Lab Data*, the probabilities on edges denote the percentages of

Table 5: Single-source single-target budgeted reliability maximization on different datasets. $k = 10$, $\zeta = 0.5$, $r = 100$, $l = 30$.

Dataset	Reliability Gain				Running Time (sec)				Memory Usage (GB)			
	HC	MRP	IP	BE	HC	MRP	IP	BE	HC	MRP	IP	BE
lastFM	0.31	0.25	0.30	**0.33**	717	24	**14**	25	0.06	**0.04**	**0.04**	**0.04**
AS_Topology	**0.42**	0.40	0.41	**0.42**	785	30	**26**	32	0.30	**0.28**	**0.28**	0.29
DBLP	**0.24**	0.19	0.22	**0.24**	1105	125	**118**	129	6.9	**6.2**	6.5	6.5
Twitter	0.13	0.11	0.15	**0.19**	1053	140	**127**	141	11.0	**9.4**	9.8	9.8

messages from a sender successfully reached to a receiver. For *AS_Topology*, once an AS connection (i.e., an edge) is observed for the first time, we calculate the ratio of snapshots containing this connection within all follow-up snapshots as the probability of existence for this edge. **(2)** *LastFM*. The probability on any edge corresponds to the inverse of the out-degree of the node from which that edge is outgoing. **(4)** *DBLP* and **(5)** *Twitter*. We assign the edge probability following $1 - e^{-t/\mu}$, which is an exponential cdf of mean μ to a count t [23]. In *DBLP*, t denotes the count of the collaborations between two authors. In *Twitter*, t is the count of re-tweet actions. We set $\mu = 20$.

Queries. For each dataset and single-source-target queries, we select 100 different s-t pairs. In practice, if two nodes are too close to each other, their original reliability will be naturally high; thus, it might be unnecessary to improve their reliability further. Therefore, we first select a source node uniformly at random, and find all its neighbors within 3-5 hops. A target node is chosen from those neighbors uniformly at random. We also demonstrate experimental results by varying the distance between s-t pairs in the Appendix.

For multiple-source-target queries, we first generate a single-source-target query s-t. Then, for that s, we find all its neighbors that are within 5-hops away, and randomly select q of them into the source set S (i.e., source set size=q). Similarly, we pick q of the within 5-hop neighbors of t as the target set T, uniformly at random. We ensure that the source set and the target set do not overlap. Finally, 100 different source-target sets are generated.

Parameters setup. **(1)** Budget on #new edges (k). For single s-t queries, we vary k from 5 to 50, and use 10 as default. For multiple-source-target queries, we vary k from 10 to 500, use 100 as default. **(2)** Probability on new edges (ζ). We vary ζ from 0.3 to 0.7, and use 0.5 as default. **(3)** Number of candidate nodes (r). We vary r from 20 to 300, and use 100 as default. **(4)** Number of most-reliable paths (l). We vary l from 10 to 50, and use 30 as default. **(5)** #Sources and #targets. For multiple-source-target queries, we vary source and target set sizes from 3 to 500. **(6)** The ratio of $\frac{k_1}{k}$. For multiple-source-target case with Max and Min aggregate functions, we further have a parameter k_1 as the budget for the current selected pair. We vary k_1 from 5% to 30% of k, and use 10% as default. **(7)** Distance constraint for new edges. In practice, some missing edges cannot be candidate edges due to physical constraints. As an example, in our case study in §1, only short distance connections ($\leq$ 15 meters) are allowed to be established. In social networks, if two users have no common friends, it is unlikely that they will start communicating. In current experiments, we assume that a missing edge can be added only if its two endpoints are $\leq h$-hops in the input graph. We vary h from 2 to 5, and use 3 as default. For real-world applications, one can easily set this constraint based on her requirements.

Competing methods. For single-source-target query, we compare our ultimate method: path batches-based edge selection (BE) with individual path-based edge selection (IP), most reliable path (MRP), and our best baseline: Hill Climbing (HC). For multiple-source-target case, we employ Hill Climbing (HC) and Eigenvalue-based Optimization (EO) [10] as competitors. Moreover, on the smallest dataset, *Intel Lab Data*, we have the exact solution (ES) as a competitor, which enumerates all possible combinations of k missing edges, and find the best one with highest reliability gain. All the methods are coupled with our search space elimination strategy (§ 5.1.1) and an advanced sampling method: RSS (§ 5.3).

Table 6: Comparison with the exact solution (ES), $k = 3$, $\zeta = 0.33$, $r = 54$, $l = 30$, *Intel Lab Data*.

Method	Reliability Gain	Running Time (sec)
ES	0.252	19189
IP	0.222	8
BE	**0.237**	**12**

Performance Metrics. **(1)** Reliability gain. We compute reliability gain due to k new edges for each pair of source and target nodes, and report the average reliability gain over 100 distinct s-t pairs. **(2)** Running time. We report the end-to-end running time, averaged over 100 queries. **(3)** Memory usage. We report the average memory usage of running each query.

8.2 Single-source-target results

Comparison with the exact solution. The exact solution (ES) enumerates all possible combinations of k missing edges, and finds the best one with the highest reliability gain. The number of missing edges can reach $\mathcal{O}(n^2)$ in sparse graph, and results in $\binom{n^2}{k}$ possible choices. Thus, the exact solution (ES) is extremely inefficient in larger graphs. However, due to the small size of *Intel Lab Data*, we can apply such exhaustive search, and empirically compare the performance of our proposed solution, BE.

Here, we follow the setting used in our case study in §1, that is, only 3 new short distance ($\leq$ 15 meters) links are allowed to be established, each have the average probability 0.33. 30 distinct pairs of sensors, which are remote and with lower original reliabilities, are selected as queries.

As shown in Table 6, our proposed solution, BE, exhibits very close performance against the exact solution (ES) in reliability gain. Particularly, BE returns same set of edges as ES, in 25 out of 30 queries. However, the running time of BE are at least three orders of magnitude faster than ES. This demonstrates the both the effectiveness and the efficiency of our methods.

Comparison of all competing methods on different datasets with default parameters. In Table 5, we present the reliability gain obtained by four methods, and the corresponding running time and memory usage, on various datasets with default parameters. Clearly, our ultimate method, path batches-based edge selection (BE) outperforms others. For reliability gain, it wins on all datasets. On *Twitter*, the advantage of BE is more prominent. The reason is that *Twitter* is a sparser graph compared to other datasets, and the highly reliable paths connecting source to target are more likely to contain more than one missing edges — this fact enhances the impact of path batches. Individual path-based edge selection (IP) always has lower reliability gain compared to BE. The polynomial-time solution, MRP for the restricted version of our problem has the lowest reliability gain among these methods, as expected.

Considering the running time, IP is the best one. However, BE is only about 10-20 seconds slower than IP across all the datasets. Both of them are about an order of magnitude faster than the baseline HC. The memory usages of IP and BE are similar, while MRP costs slightly less memory.

Table 7: Reliability gain and running time comparison with varying budget on #new edges k. $\zeta = 0.5$, $r = 100$, $l = 30$, *LastFM*.

k	Reliability Gain				Running Time (sec)			
	HC	MRP	IP	BE	HC	MRP	IP	BE
3	0.24	**0.25**	0.21	**0.25**	257	11	**10**	13
5	**0.27**	0.25	0.26	**0.27**	386	17	**12**	17
8	**0.29**	0.25	0.27	0.28	525	20	**13**	19
10	0.31	0.25	0.30	**0.33**	717	24	**14**	25
15	0.34	0.25	0.32	**0.35**	953	26	**17**	28
20	0.35	0.25	0.35	**0.37**	1378	30	**20**	32
30	0.38	0.25	**0.39**	**0.39**	2010	37	**26**	36
50	0.40	0.25	0.40	**0.41**	4005	48	**34**	44

Table 8: Reliability gain and running time comparison with varying budget on #new edges k. $\zeta = 0.5$, $r = 100$, $l = 30$, *DBLP*.

k	Reliability Gain				Running Time (sec)			
	HC	MRP	IP	BE	HC	MRP	IP	BE
3	0.19	0.19	0.19	**0.20**	373	96	**95**	100
5	**0.21**	0.19	0.20	**0.21**	576	103	**97**	106
8	0.22	0.19	0.21	**0.23**	923	111	**107**	110
10	**0.24**	0.19	0.22	**0.24**	1097	117	**112**	121
15	0.24	0.19	0.23	**0.25**	1504	127	**119**	126
20	**0.26**	0.19	0.25	**0.26**	1974	136	**125**	131
30	**0.26**	0.19	**0.26**	**0.26**	3091	148	**131**	136
50	0.27	0.19	**0.28**	**0.28**	5102	162	**139**	142

Table 9: Reliability gain, running time comparison with varying probability ζ on new edges. $k = 10$, $r = 100$, $l = 30$, *AS_Topology*.

ζ	Reliability Gain				Running Time (sec)			
	HC	MRP	IP	BE	HC	MRP	IP	BE
0.3	0.26	0.23	0.25	**0.27**	780	28	**25**	30
0.4	**0.34**	0.31	0.32	0.33	774	**27**	28	29
0.5	**0.42**	0.40	0.41	**0.42**	785	30	**26**	32
0.6	**0.51**	0.48	0.50	**0.51**	801	**32**	**32**	37
0.7	0.59	0.57	0.58	**0.60**	810	37	**35**	40

Table 10: Reliability gain, running time comparison with varying probability ζ on new edges. $k = 10$, $r = 100$, $l = 30$, *Twitter*.

ζ	Reliability Gain				Running Time (sec)			
	HC	MRP	IP	BE	HC	MRP	IP	BE
0.3	0.10	0.07	0.10	**0.12**	1003	139	**134**	137
0.4	0.12	0.09	0.12	**0.15**	1025	138	**134**	140
0.5	0.13	0.11	0.15	**0.19**	1053	140	**136**	141
0.6	0.17	0.15	0.19	**0.24**	1136	142	**137**	142
0.7	0.22	0.17	0.27	**0.29**	1175	143	**137**	143

Table 11: Analysis with different probabilities on new edges. $k = 10$, $r = 100$, $l = 30$, *Twitter*.

New edge probabilities	Reliability Gain				Running Time (sec)				Memory (GB)			
	HC	MRP	IP	BE	HC	MRP	IP	BE	HC	MRP	IP	BE
rand$(0, 1)$	0.17	0.13	0.16	**0.18**	1049	139	**134**	141	11.0	**9.4**	9.8	9.8
rand$(0.2, 0.6)$	0.13	0.10	0.11	**0.14**	1019	132	**127**	134	10.9	**9.4**	9.8	9.8
rand$(0.4, 0.8)$	**0.21**	0.17	0.20	**0.21**	1052	140	**134**	143	11.0	**9.4**	9.8	9.8
$N(0.5, 0.038)$	0.19	0.14	0.16	**0.20**	1036	135	**133**	136	11.0	**9.4**	9.8	9.8

Table 12: Scalability analysis of BE. $k = 10$, $\zeta = 0.5$, $r = 100$, $l = 30$, *Twitter*.

# Nodes	Reliability Gain	Running Time (sec)	Memory Usage (GB)
1M	0.15	101	6.8
2M	0.17	109	5.7
3M	0.18	115	6.8
4M	0.19	122	7.9
5M	0.20	130	8.8
6M	0.19	141	9.8

Varying the budget k on #new edges. We present the results on *LastFM* and *DBLP* datasets in Tables 7 and 8, respectively. Similar trends can be observed that the reliability gain increases with larger k. Such growth is more significant when k is small, for example the reliability gain increases from 0.27 to 0.33 when k increases from 5 to 10, while only 0.02 increase can be obtained when permitting k from 20 to 30, on *LastFM*. The reliability gain nearly saturates at k=20 on *DBLP*. On both datasets, BE outperforms other methods in reliability gain, no matter how large is k. The reliability gain of MRP converges at the beginning. This is because in the restricted version of our problem, we only consider the most reliable path. A path containing larger number of new edges will have longer length, and tends to have lower probability. Such paths are unlikely to be the most reliable path.

For running time, HC is $\approx$100$\times$ slower than others. MRP, IP, and BE are comparable in running time. All of them finish within 200 seconds with the largest $k = 50$. The running time of MRP increases faster than IP and BE with larger k, since it requires k copies of the original graph to find the most reliable paths with exactly 0 to k missing edges, although this does not help improve the solution quality in practice.

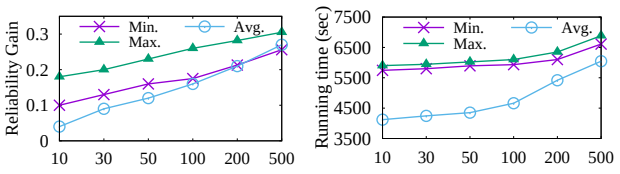
Varying probability ζ on new edges. The experimental results on *AS_Topology* and *Twitter* are provided in Tables 9 and 10, respectively. The reliability grows almost linearly with the probability threshold ζ. Sometimes, the growth rate may be even higher (e.g., on *Twitter*). The reason is that the optimal solution set of edges may change with different ζ (Observation 1), and a sharp increase may happen when shifting from a set of edges to another (Example 1). The running times of all the methods are not sensitive to different ζ. However, with larger ζ, the running time slightly increases.

Table 8.2 provides additional analysis about the probabilities on new edges. Here, instead of a fixed threshold ζ, we allow different probabilities on different new edges. Probabilities on new edges are generated uniformly at random in different range, or generated following normal distribution $N(0.5, 0.038)$ (99% of value generated are in range $(0, 1)$). It can be viewed that the results are very similar to all our previous study with fixed threshold ζ. *This confirms that our proposed algorithm, BE works well even when different probabilities for the missing edges are provided as input.*

Scalability analysis. We conduct scalability analysis of our method, BE by varying the graph size on the largest dataset, *Twitter*. We select 1M, 2M, 3M, 4M, 5M, and 6M nodes uniformly at random to generate 6 subgraphs, and apply our algorithm on them. Table 12 shows that the running time and the memory usage are both linear to the graph size, which confirms good scalability of BE.

8.3 Multiple-source-target results

Varying #source-target nodes. We conduct experiments to evaluate reliability gain and running time of our methods to maximize an aggregate function of reliability for multiple source-target pairs. The results on the largest dataset, *Twitter*, is provided in Tables 14, 13, and 15, for the aggregate functions: *Minimum*, *Maximum*, and *Average*, respectively. Our purposed method, BE (coupled with the framework suggested in § 6), significantly outperforms the baselines, Hill Climbing (HC) and Eigenvalue-based optimization (EO) in reliability gain, and runs at least 40$\times$ and 2$\times$ faster than HC, with *Average* and *Minimum/Maximum* aggregate functions, respectively. The running times of BE and EO are comparable for *Minimum/Maximum*, while EO outperforms BE when using *Average* aggregate function. EO is not query-specific, and its edge selection cost remains the same as the single s-t case. In general, the



(a) Reliability gain, *Twitter* (b) Running time, *Twitter*

Figure 7: Reliability gain and running time comparison with varying budget on #new edges k. $\zeta = 0.5$, $r = 100$ $l = 30$, #Sources=#Targets=100, *Twitter*.

Table 13: Reliability gain and running time comparison for multiple-source-target pairs. $k = 100$, $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *Twitter*, aggregate function=*Min*.

#Source:#Target	Reliability Gain			Running Time (sec)		
	HC	EO	BE	HC	EO	BE
3:3	0.34	0.29	**0.38**	1662	384	**358**
10:10	0.27	0.19	**0.36**	2140	**979**	1007
50:50	0.19	0.12	**0.28**	4051	**2910**	3049
100:100	0.15	0.06	**0.19**	7144	5766	**5708**
200:200	0.12	0.02	**0.16**	13615	**8711**	8981
500:500	0.12	0.04	**0.15**	30111	**17198**	18082

Table 14: Reliability gain and running time comparison for multiple-source-target pairs. $k = 100$, $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *Twitter*, aggregate function=*Max*.

#Source:#Target	Reliability Gain			Running Time (sec)		
	HC	EO	BE	HC	EO	BE
3:3	**0.27**	0.20	**0.27**	1682	404	**377**
10:10	0.24	0.17	**0.25**	2381	**982**	1071
50:50	0.24	0.16	**0.28**	4051	**2955**	3366
100:100	0.23	0.16	**0.26**	7144	**5822**	6101
200:200	0.19	0.14	**0.23**	13615	**8801**	9114
500:500	0.20	0.12	**0.22**	30111	**17699**	18988

Table 15: Reliability gain and running time comparison for multiple-source-target pairs. $k = 100$, $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *Twitter*, aggregate function=*Avg*.

#Source:#Target	Reliability Gain			Running Time (sec)		
	HC	EO	BE	HC	EO	BE
3:3	0.21	**0.24**	**0.24**	4221	269	**239**
10:10	0.17	0.19	**0.21**	28948	**642**	787
50:50	0.14	**0.17**	**0.17**	131487	**1766**	2321
100:100	0.12	0.13	**0.15**	194449	**2978**	4662
200:200	-	0.09	**0.12**	-	**5014**	7812
500:500	-	0.06	**0.10**	-	**9221**	13908

running time of our method, BE is almost linear to the number of source/target nodes.

Furthermore, our method results in higher reliability gain with *all* 3 aggregate functions, especially for *Minimum* and *Maximum*. This is because EO is not query-specific. EO optimizes the leading eigenvalue of a graph, which is a global metric and may have little to do with the query pair having *Minimum* or *Maximum* reliability.

Varying the budget k on #new edges. Similar to single-source-target case, we vary k, now in a larger scale: 10 to 500, and present the result in Figure 7. The reliability gains for all three aggregate functions increase with larger k. The running time of BE with *Minimum/Maximum* aggregate function is less sensitive to a larger k, since the complexity of their top-k edge selection part remains the same as single-source-target case, while the search space elimination part scales up. On the contrary, the running time of BE with *Average* is almost linear to k. However, *Average* is still less time consuming than *Minimum/Maximum* with large k.

8.4 Application in influence maximization

We present an application of budgeted reliability maximization in the classic influence maximization problem [26] using *DBLP*.

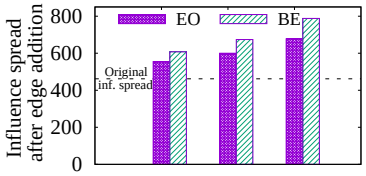


Figure 8: Influence spread comparison in *Databases* area. $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *DBLP*.

In social influence maximization following the widely-used independent cascade model [26], when some node u first becomes active at step t, it gets a single chance to activate each of its currently inactive out-neighbors v at step $t+1$, with probability $p(u, v)$. Initially, only the source nodes are active, and the activation continues in discrete steps. When no more nodes can be activated, the number of active nodes in target set is referred to as the influence spread. With possible world notation, the influence spread from source set S to target set T can be formulated as:

$$Inf(S,T) = \sum_{G \sqsubseteq \mathcal{G}} \left[Pr(G) \sum_{t \in T} I_G(S,t) \right] \tag{13}$$

As discussed in § 6.1, the *average reliability* from S to T is:

$$R_{avg}(S,T) = \frac{1}{|S||T|} \sum_{G \sqsubseteq \mathcal{G}} \left[Pr(G) \sum_{s,t \in S \times T} I_G(s,t) \right] \tag{14}$$

Clearly, in each possible world, if we only check whether there is *at least one* path to t from any $s \in S$, instead of counting the *exact number* of $s \in S$ which has a path to t, our problem becomes equivalent to the (targeted) influence maximization problem. Adding a new edge in this network implies recommending and/or establishing collaboration with an author in the real-world [9].

We select a set of junior researchers in *Databases* area, containing 1000 authors randomly selected from all the authors with 1-3 papers in top-tier venues [SIGMOD, VLDB, ICDE]. In a similar way, we choose 50 senior researchers with more than 10 papers in [SIGMOD, VLDB, ICDE], uniformly at random. The expected influence spread from the senior to the junior group is around *462*, using the IC model. Next, we aim at maximally improving the influence spread from the senior group to the junior group, by adding up to 100 new edges. As shown in Figure 8, our method outperforms Eigenvalue-based optimization (EO) [10], and results in about 326 more influenced authors within the junior set.

9. CONCLUSIONS

In this paper, we introduced and investigated the novel and fundamental problem of maximizing the reliability between a given pair of nodes in an uncertain graph by adding a small number of edges. We proved that this problem is **NP**-hard and also hard to approximate. Several interesting observations are presented to characterize our problem. Our purposed solution first eliminates the search space based on original reliability, and then selects the top-k edges following an iterative most-reliable path-batches inclusion algorithm. We further studied one restricted and several extended versions of the problem, to support a wider family of queries. The experimental results validated the effectiveness, efficiency, and scalability of our method, and rich real-world case studies demonstrated the usefulness of our budgeted reliability maximization problem. In future study, a total reliability budget on new edges, instead of a fixed/ individual budget on each new edge, shall be considered. This will add more complexity about selecting proper candidate edges and allocating reliability budget to them.

10. REFERENCES

[1] E. Adar and C. Re. Managing Uncertainty in Social Networks. *IEEE Data Eng. Bull.*, 2007.

[2] C. Aggarwal. *Managing and Mining Uncertain Data*. Springer, 2009.

[3] K. K. Aggarwal, K. B. Misra, and J. S. Gupta. Reliability Evaluation: A Comparative Study of Different Techniques. *Micro. Rel.*, 14(1), 1975.

[4] R. Albert, H. Jeong, and A.-L. Barabasi. Error and Attack Tolerance of Complex Networks. *Nature*, 406:378–382, 2000.

[5] M. O. Ball. Computational Complexity of Network Reliability Analysis: An Overview. *IEEE Tran. on Reliability*, 1986.

[6] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish. Improving Network Robustness by Edge Modification. *Physica A: Stat. Mech. and its Appl.*, 357:593–612, 2005.

[7] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa. Injecting Uncertainty in Graphs for Identity Obfuscation. In *VLDB*, 2012.

[8] H. Chan and L. Akoglu. Optimizing Network Robustness by Edge Rewiring: a General Framework. *Data Min. Knowl. Disc.*, 30:1395–1425, 2016.

[9] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt. Recommendations to Boost Content Spread in Social Networks. In *WWW*, 2012.

[10] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Eigen-Optimization on Large Graphs by Edge Manipulation. *ACM Trans. Knowl. Discov. Data*, 10(4):49:1–49:30, 2016.

[11] W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *KDD*, 2010.

[12] C. J. Colbourn. Edge-packing of Graphs and Network Reliability. *Discrete Mathematics*, 72:49 – 61, 1988.

[13] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of Complex Networks: a Survey of Measurements. *Adv. Phys.*, 56:167–242, 2007.

[14] E. D. Demaine and M. Zadimoghaddam. Minimizing the Diameter of a Network Using Shortcut Edge. *SWAT. ser.Lec. Notes in Comput. Sci.*, pages 420–431, 2010.

[15] D. Eppstein. Finding the k Shortest Paths. *SIAM J. Comput*, 28(2):652–673, 1998.

[16] G. S. Fishman. A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness. *IEEE Tran. Rel.*, 1986.

[17] L. Fu, X. Fu, Z. Xu, Q. Peng, X. Wang, and S. Lu. Determining Source-Destination Connectivity in Uncertain Networks: Modeling and Solutions. *IEEE/ACM Trans. Netw.*, 25(6):3237–3252, 2017.

[18] L. Fu, X. Wang, and P. R. Kumar. Optimal Determination of Source-destination Connectivity in Random Graphs. In *MobiHoc*, 2014.

[19] X. Fu, Z. Xu, Q. Peng, L. Fu, and X. Wang. Complexity vs. optimality: Unraveling Source-Destination Connection in Uncertain Graphs. In *INFOCOM*, 2017.

[20] J. Ghosh, H. Q. Ngo, S. Yoon, and C. Qiao. On a Routing Problem Within Probabilistic Graphs and its Application to Intermittently Connected Networks. In *INFOCOM*, 2007.

[21] M. Hua and J. Pei. Probabilistic Path Queries in Road Networks: Traffic Uncertainty aware Path Selection. In *EDBT*, 2010.

[22] R. Jin, L. Liu, and C. Aggarwal. Discovering Highly Reliable Subgraphs in Uncertain Graphs. In *KDD*, 2011.

[23] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-Constraint Reachability Computation in Uncertain Graphs. In *VLDB*, 2011.

[24] U. Kang, S. Papadimitriou, J. Sun, and H. Tong. Centralities in Large Networks: Algorithms and Observations. In *SDM*, 2011.

[25] X. Ke, A. Khan, and G. Cong. Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks. In *SIGMOD*, 2018.

[26] D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the Spread of Influence through a Social Network. In *KDD*, 2003.

[27] E. B. Khalil, B. Dilkina, and L. Song. Scalable Diffusion-aware Optimization of Network Topology. In *KDD*, 2014.

[28] A. Khan, F. Bonchi, A. Gionis, and F. Gullo. Fast Reliability Search in Uncertain Graphs. In *EDBT*, 2014.

[29] A. Khan, F. Bonchi, F. Gullo, and A. Nufer. Conditional Reliability in Uncertain Graphs. *TKDE*, 2018.

[30] A. Khan, Y. Ye, and L. Chen. *On Uncertain Graphs*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

[31] A. Khan, B. Zehnder, and D. Kossmann. Revenue Maximization by Viral Marketing: A Social Network Host's Perspective. In *ICDE*, 2016.

[32] M. Kimura and K. Saito. Tractable Models for Information Diffusion in Social Networks. In *PKDD*, 2006.

[33] O. Kuchaiev, M. Rasajski, D. J. Higham, and N. Przulj. Geometric De-noising of Protein-Protein Interaction Networks. *PLOS Computational Biology*, 5(8):1–10, 08 2009.

[34] D. L.-Nowell and J. M. Kleinberg. The Link Prediction Problem for Social Network. In *CIKM*, 2003.

[35] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding Effectors in Social Networks. In *KDD*, 2010.

[36] R. Li, J. X. Yu, R. Mao, and T. Jin. Recursive Stratified Sampling: A New Framework for Query Evaluation on Uncertain Graphs. *IEEE Trans. Knowl. Data Eng.*, 28(2):468–482, 2016.

[37] Y. Li, J. Fan, D. Zhang, and K.-L. Tan. Discovering Your Selling Points: Personalized Social Influential Tags Exploration. In *SIGMOD*, 2017.

[38] Y. Li, D. Zhang, and K. Tan. Real-time Targeted Influence Maximization for Online Advertisements. *PVLDB*, Volume 8 Issue 10:1070–1081, 2015.

[39] X. Lin, Y. Peng, B. Choi, and J. Xu. Human-Powered Data Cleaning for Probabilistic Reachability Queries on Uncertain Graphs. *IEEE TKDE*, 29(7):1452–1465, 2017.

[40] W. Lu, W. Chen, and L. V. S. Lakshmanan. From Competition to Complementarity: Comparative Influence Diffusion and Maximization. *PVLDB*, 9(2):60–71, 2015.

[41] S. Maniu, R. Cheng, and P. Senellart. An Indexing Framework for Queries on Probabilistic Graphs. *ACM Trans. Database Syst.*, 42(2):13:1–13:34, 2017.

[42] T. Matisziw and A. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *J. Comput. Oper. Res.*, 36:16–26, 2009.

[43] S. Medya, J. Vachery, S. Ranu, and A. Singh. Noticeable

Network Delay Minimization via Node Upgrades. *PVLDB*, Volume 11 Issue 9:988–1001, 2018.

[44] A. Meyerson and B. Tagiku. Minimizing Average Shortest Path Distances via Shortcut Edge Addition. In *Approx., Rand., and Comb. Opt. Algo. and Tech.*, pages 272–285, 2009.

[45] A. Natanzon, R. Shamir, and R. Sharan. Complexity Classification of Some Edge Modification Problems. In *WG*, volume 1665 of *Springer Lecture Notes in Computer Science*, 1999.

[46] M. Papagelis. Refining Social Graph Connectivity via Shortcut Edge Addition. *TKDD*, 10, 2015.

[47] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi. The Pursuit of a Good Possible World: Extracting Representative Instances of Uncertain Graphs. In *SIGMOD*, 2014.

[48] N. Parotsdis, E. Pitoura, and P. Tsaparas. Centrality-Aware Link Recommendations. In *WSDM*, 2016.

[49] J. Peng, B. Zhang, and S. Li. Towards Uncertain Network Optimization. *J. Uncertainty Analysis and Applications*, 3(1), 2015.

[50] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-Nearest Neighbors in Uncertain Graphs. *PVLDB*, 2010.

[51] J. S. Provan and M. O. Ball. Computing Network Reliability in Time Polynomial in the Number of Cuts. *Operations Research*, 1984.

[52] G. Rubino. *Network Reliability Evaluation*. 1999.

[53] D. Sheldon, B. Dilkina, A. N. Elmachtoub, R. Finseth, A. Sabharwal, J. Conrad, C. Gomes, D. Shmoys, W. Allen, O. Amundsen, and W. Vaughan. Maximizing the Spread of Cascades Using Network Design. In *UAI*, 2010.

[54] S. Song, Z. Zou, and K. Liu. Triangle-Based Representative Possible Worlds of Uncertain Graphs. In *DASFAA*, 2016.

[55] F. Sun and M. Shayman. On pairwise connectivity of wireless multihop networks. *Int. J. Network Security*, 2:37–49, 2007.

[56] L. G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. on Computing*, 1979.

[57] Y. Wang, J. X. Cao, R. D. Wang, and X. X. Li. Research on Uncertain Network Design Problem. In *Advances in Transportation*, volume 505 of *Applied Mechanics and Materials*, pages 613–618. Trans Tech Publications, 2014.

[58] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint. In *SRDS*, 2003.

APPENDIX

A. PROOF OF THEOREM 2

PROOF. A problem is said to admit a *Polynomial Time Approximation Scheme* ($\mathbf{PTAS}$) if the problem admits a polynomial-time constant-factor approximation algorithm for every constant $\beta \in (0, 1)$. We prove the theorem by showing that one can find at least one value of β such that, if a β-approximation algorithm for Problem 1 exists, then we can solve the well-known SET COVER problem in polynomial time. Since SET COVER is $\mathbf{NP}$-hard, this can happen only if $\mathbf{P} = \mathbf{NP}$.

In SET COVER, there is a collection of subsets $S = \{S_1, S_2, \ldots, S_h\}$ of a ground set $U = \{u_1, u_2, ..., u_r\}$, where $S_i \subseteq U$ for all $i \in [1...h]$. The decision version of SET COVER asks if there is a subset $S^* \subset S$ of size k such that *all* elements in U can be covered by S^*.

Given an instance of SET COVER, we construct an instance of our problem in polynomial time by following the same method as in $\mathbf{NP}$-hardness proof (Figure 3). In the SET COVER instance, if there is a solution with k subsets, then the optimal solution OPT of our problem will add k edges such that the s-t reliability after edge addition is: $1 - (1 - p)^r$. This is because $|U| = r$. In contrast, if no solution with k subsets exists for SET COVER, then OPT will produce s-t reliability at most: $1-(1-p)^{r-1}$ (because at least one of u_j would not be covered).

Let there be a polynomial-time β-approximation algorithm, Approx for Problem 1, such that $0 < \beta < 1$. According to the definition of approximation ratio, Approx will produce s-t reliability at least β times to that produced by OPT. Now, let us consider the inequality: $1 - (1 - p)^{r-1} < \beta[1 - (1 - p)^r]$. If this inequality has a solution for some values of β and p, then by simply running Approx on our instance of Problem 1, and checking the s-t reliability of the solution returned by Approx, one can answer SET COVER in polynomial time: a solution to SET COVER exists iff the solution given by Approx has s-t reliability $\geq \beta[1 - (1 - p)^r]$. Thus, to prove the theorem, we require to show that a solution to that inequality exists.

Our inequality has a solution iff $\beta > \frac{1-(1-p)^{r-1}}{1-(1-p)^r}$. One can verify that $\frac{1-(1-p)^{r-1}}{1-(1-p)^r} < 1$, for all $r \geq 1$ and $p > 0$. This implies that there will always be a value of $\beta \in (0, 1)$ and p for which $\beta > \frac{1-(1-p)^{r-1}}{1-(1-p)^r}$ is satisfied, regardless of r. Hence, there exists at least one value of β such that the inequality $[1 - (1 - p)^{r-1}] < \beta[1 - (1 - p)^r]$ has a solution, and, based on the above argument, such that no β-approximation algorithm for Problem 1 can exist. The theorem follows. □

B. EIGENVALUE-BASED OPTIMIZATION

Recently, Chen et al. studied the problem of maximizing the largest eigenvalue of a network by edge-addition [10]. They proved that the eigenvalue gain of adding a set of k new edges E_1 can be approximated by $\sum_{e_x \in E_1} \mathbf{u}(i_x)\mathbf{v}(j_x)$, where $\mathbf{u}$ and $\mathbf{v}$ are the corresponding left and right eigenvectors with the leading eigenvalue of the original adjacency matrix (i_x and j_x are the two end points of the new edge e_x). They also proved that each e_x in optimal E_1 has left endpoint from the subset of $(k+d_{in})$ nodes with the highest left eigen-score $\mathbf{u}(i_x)$, and right endpoint from the subset of $(k+d_{out})$ nodes with the highest right eigen-score $\mathbf{v}(j_x)$, where d_{in} and d_{out} are the maximum in-degree and out-degree in the original graph, respectively. Therefore, one can find the optimal k new edges to increase the eigenvalue of the input graph by the following steps: First, calculate the largest eigenvalue of the input graph, and the corresponding left and right eigenvectors. Then, compute the maximum in-degree and out-degree of this graph, and find the subset of nodes I with top-$(k + d_{in})$ left eigen-scores and the subset of nodes J with top-$(k + d_{out})$ right eigen-scores. Finally, connect the nodes from I to J (if no such edge exists in the original graph), and select the top-k pairs with largest eigen-scores $\mathbf{u}(i_x)\mathbf{v}(j_x)$.

Time complexity. The first step can be solved with power iteration method in $\mathcal{O}(n)$ time. Finding maximum in/out degrees takes $\mathcal{O}(n+m)$ time, and finding subset I and J requires $\mathcal{O}(n(d_{in}+k))$ and $\mathcal{O}(n(d_{out} + k))$ times, respectively, which can be written as $\mathcal{O}(nt)$, $t = max(k, d_{in}, d_{out})$. The final step consumes $\mathcal{O}(kt^2)$ time. Therefore, the overall time complexity is $\mathcal{O}(m + nt + kt^2)$.

C. RECURSIVE STRATIFIED SAMPLING

The recursive stratified sampling [36] partitions the probability space Ω into $r+1$ non-overlapping subspaces (Ω_0,...,Ω_r) via selecting r edges. In stratum i, we set the status of edge i to 1, the status of those edges before it as 0, and all other edges as undetermined.

Table 16: Running time comparison for reliability-based search space elimination, $r = 100$. We also report the number of samples (Z) required by *MC* and *RSS*.

Dataset	MC sampling		RSS sampling	
	Z	Time (sec)	Z	Time (sec)
lastFM	1000	16	250	9
AS_Topology	500	166	250	12
DBLP	750	498	250	98
Twitter	1000	439	500	114

Table 17: Running time comparison for top-k edges selection, $r = 100$. We also report the number of samples (Z) required by *MC* and *RSS*.

Dataset	MC sampling time (sec)				RSS sampling time (sec)			
	Z	HC	MRP	Batch-edge	Z	HC	MRP	Batch-edge
lastFM	500	1256	20	22	250	708	15	16
AS_Topology	500	1508	23	29	250	758	19	22
DBLP	500	1818	34	50	250	1007	27	31
Twitter	500	1677	38	44	250	939	26	27

Table 18: Reliability gain and running time comparison with varying #candidate nodes r. $k = 10$, $\zeta = 0.5$, $l = 30$, *lastFM*. Time 1 denotes the time cost for search space elimination, and Time 2 is the time cost for top-k edges selection.

r	Reliability Gain				Time 1 (sec)	Time 2 (sec)			
	HC	MRP	IP	BE		HC	MRP	IP	BE
20	**0.29**	0.25	0.27	**0.29**	6	223	5	**2**	9
50	0.30	0.25	0.29	**0.32**	8	399	8	**3**	10
80	0.31	0.25	0.30	**0.33**	8	587	11	**4**	13
100	0.31	0.25	0.30	**0.33**	9	708	15	**5**	16
150	0.31	0.25	0.30	**0.33**	13	816	21	**5**	17
200	0.31	0.25	0.30	**0.33**	19	849	26	**7**	17
300	0.31	0.25	0.30	**0.33**	29	898	35	**9**	19

Table 19: Reliability gain and running time comparison with varying #candidate nodes r. $k = 10$, $\zeta = 0.5$, $l = 30$, *DBLP*. Time 1 denotes the time cost for search space elimination, and Time 2 is the time cost for top-k edges selection.

r	Reliability Gain				Time 1 (sec)	Time 2 (sec)			
	HC	MRP	IP	BE		HC	MRP	IP	BE
20	0.19	0.18	0.19	**0.20**	58	488	**11**	13	14
50	**0.22**	0.19	0.20	0.21	71	650	19	**14**	20
80	0.23	0.19	0.23	**0.24**	80	822	23	**18**	25
100	**0.24**	0.19	0.22	**0.24**	98	1007	27	**22**	31
150	**0.24**	0.19	0.22	**0.24**	135	1339	37	**25**	35
200	**0.24**	0.19	0.22	**0.24**	190	1458	46	**28**	38
300	**0.24**	0.19	0.22	**0.24**	411	1519	59	**32**	43

The probability π_i of stratum i can be calculated as the product of the absent probability $1 - p(e)$ of all edges with 0 status, multiplied by the probability of edge i. The sample size of stratum i is set as $Z_i = \pi_i \cdot Z$, where Z is the total sample size. The algorithm recursively computes the sample size to each stratum and simplify the graph. It applies Monte Carlo sampling on the simplified graph when the sample size of a stratum is smaller than a given threshold. Reliability is then calculated by finding the sum of the reliabilities in all subspaces. The time complexity of recursive stratified sampling [36] is same as that of the MC sampling, i.e., $\mathcal{O}(Z(m+n))$, while the variance of the estimator is significantly reduced. Therefore, if we require same variance, recursive stratified sampling runs faster due to its smaller sample size Z.

Benefits of recursive stratified sampling. We compute the variance of an estimator by repeating experiments with different number of samples (Z), and we consider the ratio $\rho_Z = \frac{V_Z}{R_Z}$ to decide if the estimator has converged over a given dataset. Here, V_Z is the average variance of repeating 100 different s-t queries for 100 times, and R_Z is the mean of reliability of these queries. The ratio of variance to mean, also known as the index of dispersion, is a normalized measure of the dispersion of a dataset . If $\rho_Z < 0.001$, we conclude that the estimator has converged.

Table 16 and 17 report the number of samples (Z) required for convergence in each dataset, together with the running time comparison. Clearly, applying recursive stratified sampling (RSS) significantly reduces the running time of sampling-based methods. For reliability-based search space elimination, the sampling is conducted on the original graphs (which are large in size). RSS requires about half of the sample size, compared to that of MC sampling, and reduces the running time by 50%-90%. For top-k edges selection, the sampling is applied over a simplified (smaller) subgraph, however the benefit of RSS over MC sampling can still be up to 40%.

D. SENSITIVITY ANALYSIS

Varying #candidtae nodes (r). In reliability-based search space elimination, we only keep the top-r nodes $C(s)$ with the highest reliability from s, and the top-r nodes $C(t)$ with the highest reliability to t. $C(s)$ and $C(t)$ are candidate node sets, and only those missing edges from a node in $C(s)$ to a node in $C(t)$ will be considered as candidate edges. As demonstrated in Table 19, small r incurs low-quality result, due to the excessive elimination. The accuracy does not keep improving if r exceeds 80 and 100, respectively for *LastFM* and *DBLP*. We find out that $r = 100$ is sufficient for all the methods to work on all datasets in our experiments.

As shown in Table 18 and 19, the running time for search space elimination (Time 1), when varying r, increases sharply with larger r. Although the time cost of checking all nodes' reliability from/to a node is not relevant to r, we need to add at most $\mathcal{O}(r^2)$ missing edges after determining the candidate nodes. Since we shall also verify the distance constraint before adding a missing edge, the time cost of adding edges is non-trivial. When $r \leq 100$, the increasing rate for the running time of search space elimination is modest. Together with the previous finding that $r = 100$ can ensure a good accuracy, we set $r = 100$ as default in other experiments.

The running times for top-k edge selection (Time 2) for methods IP and BE increase little with larger r, since they estimate the reliability gain of missing edges only on the subgraph induced by a few most-reliable paths. Time 2 for MRP, on the other hand, increases linearly with r, since the size of each copy of graph is linear to r. The time cost of edge selection by HC is also linear to r at the beginning, and slows down with larger r. This is because although the time complexity of sampling is linear to the graph size theoretically; when coupling with BFS search, low-reliability nodes added later are less frequent to be explored during the sampling.

Varying #most reliable paths (l). Tables 20 and 21 demonstrate the sensitivity analysis of our IP and BE methods to the number of most reliable paths, l. The reliability gain increases with larger l, and saturates at around $l = 30$. The running time is linear to l. Thus, we set $l = 30$ as default in the rest of our experiments.

Varying distance (d) between query nodes s and t. We further select queries where each pair of s and t are exactly d-hops away in the input graph. The experimental result for varying d is given in Table 22. Clearly, the original reliability decreases with larger d. And the reliability gain at $d = 3$ and $d = 4$ is about the highest, for both HC and BE.

The running time is small either with too large or too small d. For small d, the reason is that the candidate node sets $C(s)$ and

Table 20: Reliability gain and running time comparison with varying #most-reliable paths l. $k = 10$, $\zeta = 0.5$, $r = 100$, *AS_Topology*.

l	Reliability Gain		Running Time (sec)	
	IP	BE	IP	BE
10	**0.29**	**0.29**	**18**	24
20	0.34	**0.37**	**23**	30
30	0.41	**0.42**	**26**	32
40	**0.42**	**0.42**	**29**	34
50	0.42	**0.43**	**31**	37

Table 21: Reliability gain and running time comparison with varying #most-reliable paths l. $k = 10$, $\zeta = 0.5$, $r = 100$, *Twitter*.

l	Reliability Gain		Running Time (sec)	
	IP	BE	IP	BE
10	0.11	**0.12**	**117**	124
20	0.15	**0.19**	**127**	133
30	0.15	**0.19**	**136**	141
40	0.15	**0.19**	**148**	150
50	0.15	**0.19**	**159**	160

Table 22: Reliability gain and running time comparison with varying distance d between query nodes. $k = 10$, $\zeta = 0.5$, $r = 100$, $l = 30$, *AS_Topology*.

d	Reliability Gain		Running Time (sec)	
	HC	BE	HC	*BE*
2	0.29	0.31	669	140
3	0.42	**0.43**	844	142
4	**0.39**	**0.39**	830	133
5	0.21	0.24	471	128
6	0.11	0.13	420	129

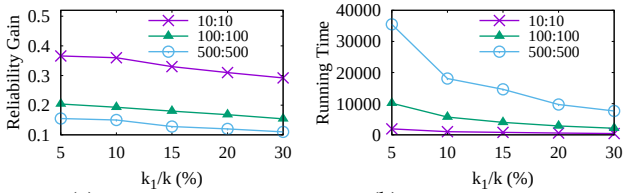
Table 23: Reliability gain and running time comparison with varying distance constraint h for newly added edges. $k = 10$, $\zeta = 0.5$, $r = 100$, $l = 30$, *Twitter*.

d	Reliability Gain		Running Time (sec)	
	HC	BE	HC	BE
2	0.11	**0.14**	661	130
3	0.13	**0.19**	1053	141
4	0.17	**0.21**	1615	166
5	0.19	**0.22**	1970	178

$C(t)$ are likely to have a large overlap, thus less missing edges are found. On the other hand, the distance between nodes in $C(s)$ and $C(t)$ tends to increase with larger d, thus the distance constraint may forbid many missing edges from being added into the graphs. The running time of HC is more sensitive to d, since it iterates over each new edge.

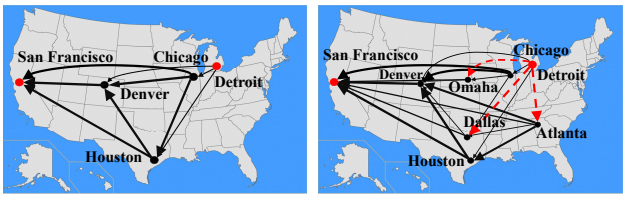
Varying distance constraint (h) for newly added edges. We constrain that a missing edge can only be added if the distance between its two endpoints in the original graph is at most h. Smaller h prevents more edges from being added. As shown in Table 23, with larger h, we can obtain more reliability improvement. However, this allows many remote links to be established, which may not be realistic in practice. Moreover, many candidate edges also increase the running time, both for HC and BE.

Varying the ratio $\frac{k_1}{k}$. For multiple-source-target case, we further have a parameter k_1 as the budget for current selected pair, when considering *Maximum/Minimum* aggregate functions. As shown in Figure 9, with larger $\frac{k_1}{k}$, the effectiveness of BE decreases. However, the running time also decreases linearly with larger $\frac{k_1}{k}$. Since the accuracy loss from 5% to 10% is much smaller than that of further increasing the ratio, $\frac{k_1}{k} = 10\%$ is selected as the default setting.



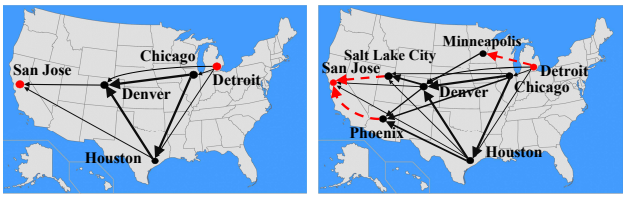
(a) Rel. Gain, *Twitter* (b) Running Time, *Twitter*

Figure 9: Reliability gain and running time comparison with varying ratio $\frac{k_1}{k}$. $k = 100$, $\zeta = 0.5$, $r = 100$ $l = 30$, #Sources=#Targets=100, *Twitter*, aggregate function=*Max*.



(a) Original routes (b) Original and new routes

Figure 10: Improving the reliability from Detroit to San Francisco with 3 new connections (marked by dotted edges).



(a) Original routes (b) Original and new routes

Figure 11: Improving the reliability from Detroit to San Jose with 3 new connections (marked by dotted edges).

E. MORE APPLICATION IN AIR-TRAFFIC RELIABILITY MAXIMIZATION

We conduct another case study about air-traffic reliability maximization on the *US Flight* dataset (`https://www.kaggle.com/usdot/flight-delays`). This dataset contains flight information (depart/arrival airports, departure/arrival times, etc.) between 322 airports in the USA in 2015. We construct a flight network, where airports are represented as nodes.

When booking a trip between two long-distance airports, customers may have various preferences. Some of them may even conflicted, e.g., traveling with short layover to reach the destination faster, or long gaps for a good rest. Clearly, having more candidate flights will make it easier for a customer to find a proper route following her own mind. Thus, we assign edge probability based on daily flight number per connection, and then improve the reliability by allowing new connections. If there are t direct flights from node u to v per day on average, an edge (u, v) exists with a probability: $1 - e^{-t/\mu}$ (an exponential cdf of mean μ to this count t [23]). We set $\mu = 5$.

Suppose our goal is to improve the air-traffic reliability from Detroit to two cities in California: (1) San Francisco, and (2) San Jose, respectively. Since passengers would not like to transfer flights multiple times when traveling within a country, we only investigate those routes within 3 hops (i.e., at most 2 layovers). Figures 10(a) and 11(a) present the original important routes between Detroit and the two destinations. The thickness of the edges represents the edge probability. The original reliability from Detroit to San Francisco is *0.40*, and that from Detroit to San Jose is *0.16*.

If we only allow three new short-distance (500-1500 miles, i.e., 1-3 hours of flight duration) connections with 5 flights per day on each connection (i.e., 0.63 edge probability), the optimal set we

found for Detroit to San Francisco will be: *Detroit to Dallas, Detroit to Atlanta, and Detroit to Omaha*. Original routes between Detroit and San Francisco are heavily relied on three hub cities: Denver, Chicago, and Houston. Compared to San Francisco, Detroit has much lower reliability to these hubs. Linking it to more hub cities will bring more reliable routes. The three new hubs suggested by our algorithm already have higher reliability to San Francisco and to three original hubs (shown in Figure 10(b)). By adding Detroit with them, the reliability from Detroit to San Francisco route is increased to *0.89 (49% increase)*.

For San Jose, its reliability from Detroit also depends on same hub cities: Denver, Houston, and Chicago. However, its links from these hubs are even weaker than Detroit. The three new flight connections found by our algorithm are: *Detroit to Minneapolis, Phoenix to San Jose, and Salt Lake City to San Jose*, which result in *0.39 (23% increase)* in overall reliability.