Determining Source-Destination Connectivity in Uncertain Networks: Modeling and Solutions

Luoyi Fu[1], Xinzhe Fu[1], Zhiying Xu[2], Qianyang Peng[1], Xinbing Wang[1,2], and Songwu Lu[3]

[1]Dept. of Computer Science, Shanghai Jiao Tong University, China.
Email:{yiluofu,fxz0114,az950512,xwang8}@sjtu.edu.cn.
[2]Dept. of Electrical Engineering, Shanghai Jiao Tong University, China. Email:{xuzhiying}@sjtu.edu.cn
[3]Dept. of Computer Science, University of California, Los Angeles, USA. Email:{slu}@cs.ucla.edu

***Abstract*—Determination of source-destination connectivity in networks has long been a fundamental problem, where most existing works are based on deterministic graphs that overlook the inherent uncertainty in network links. To overcome such limitation, this paper models the network as an uncertain graph where each edge e exists independently with some probability $p(e)$. The problem examined is that of determining whether a given pair of nodes, a source s and a destination t, are connected by a path or separated by a cut. Assuming that during each determining process we are associated with an underlying graph, the existence of each edge can be unraveled through edge testing at a cost of $c(e)$. Our goal is to find an optimal strategy incurring the minimum expected testing cost with the expectation taken over all possible underlying graphs that form a product distribution.**

Formulating it into a combinatorial optimization problem, we first characterize the computational complexity of optimally determining source-destination connectivity in uncertain graphs. Specifically, through proving the NP-hardness of two closely related problems, we show that, contrary to its counterpart in deterministic graphs, this problem cannot be solved in polynomial time unless P=NP. Driven by the necessity of designing an exact algorithm, we then apply the Markov Decision Process framework to give a dynamic programming algorithm that derives the optimal strategies. As the exact algorithm may have prohibitive time complexity in practical situations, we further propose two more efficient approximation schemes compromising the optimality. The first one is a simple greedy approach with linear approximation ratio. Interestingly, we show that naive as it is, it enjoys significantly better performance guarantee than some other seemingly more sophisticated algorithms. Second, by harnessing the submodularity of the problem, we further design a more elaborate algorithm with better approximation ratio. The effectiveness of the proposed algorithms are justified through extensive simulations on three real network datasets, from which we demonstrate that the proposed algorithms yield strategies with smaller expected cost than conventional heuristics.

I. INTRODUCTION

Source and destination connectivity of networks has significant applications in real life. It concerns crucial issues such as reliability, routing, information diffusion [1], [2], etc. Hence, in the past few decades, a lot of research has been dedicated to this problem [3], [4], [5] and there have been many efficient algorithms proposed under various types of networks. A common feature shared by all those works is that the networks investigated are modeled as deterministic graphs [4], [5] with the source-destination connectivity problems transformed to the corresponding graph reachability problems.

However, as indeterminacy plagues in our life, deterministic graph often fails to serve as a suitable model for networks nowadays. Usually, we do not have certain knowledge of existence of network links. For instance, in social networks, due to the variability of social ties [6], the relations between network nodes may not be known in advance; in communication systems, established connections between nodes may frequently fail because of the unreliability of data links [7], [8]. It has also been pointed out that more than 90% of network links are observed to be unreliable [10]. Consequently, we may not obtain deterministic network configuration from the predesigned topology; sometimes we even have to intentionally blur the links for privacy reasons [11]. All those factors impose a need to incorporate uncertainty into the network, which can essentially be modeled as an uncertain graph [11], where, instead of appearing deterministically, each edge is associated with some prior existence probability. The existence probabilities not only are symbols of uncertainty, but also bear important attributes of network links. Take social network again for example. These probabilities may represent the confidence of link prediction [12], or the strength of the influence that a node has on the other [2]. In communication networks such as data center networks, these probabilities reflect the failure frequency of communication links [7].

When the graph is uncertain, traditional methods such as depth-first-traversal, breadth-first-traversal and graph labeling are no longer suitable for determining the source-destination connectivity of networks due to the lack of deterministic information on edges' existence. To hedge the uncertainty, we need to test the edges to determine whether they truly exist or not. However, such edge testing involves far more complicated procedures than simply identifying uncertain links and thus may turn out to be more costly. For example, in citation networks, we can establish probabilistic relationships between papers just by reference information. In contrast, to unravel the genuine relation between papers, we have to apply advanced data mining approaches which involves considerably more intensive computation. Consequently, it is extremely desirable to test the most cost-effective edges, i.e., to design a testing strategy that determines the source-destination connectivity of

The early version of this paper is to appear in the Proceedings of IEEE INFOCOM 2017 [41].

uncertain networks incurring minimum cost. Furthermore, to fully utilize the results of previous tests, the strategy should be adaptive, which means that we may determine the next edge to test based on the edge existence information we have already acquired through previous tests. And we defer a more detailed explanation of how the problem of interest can be applied to other realistic scenarios to the end of Section III-B.

In this paper, we are thus motivated to present a first look into the problem of determining source-destination connectivity in uncertain networks. Given a network modeled as an uncertain graph with each edge associated with an existence probability and a testing cost, together with two network nodes s, t designated as source and destination, we aim to derive efficient strategy specifying which edges to test so that we can verify whether s and t are connected by a path or separated by a cut with the minimum cost incurred. Note that the source and destination connectivity is also referred to as s-t connectivity. Comparing with s-t connectivity in deterministic graphs that can be easily solved by graph traversal methods in polynomial time, by proving the NP-hardness of the problem, we find that the s-t connectivity in uncertain graphs turns out to be far more complicated and highly non-trivial. Driven by the necessity of pursuing exact algorithms that can capture the features of the optimal solutions, we proceed by converting our problem into an equivalent Markov Decision Process (MDP) to give a dynamic programming algorithm that yields optimal strategies but has exponential running time. Considering that the prohibitive time complexity of such exact algorithm renders it unsuitable for practical applications, we therefore design approximation schemes to compromise the optimality of computed strategy for the efficiency of the algorithms. In doing so, we first put forward a simple greedy approach that computes near optimal solutions with linear approximation guarantee, which can be further improved by a second algorithm we propose through the exploration of submodularity in our problem.

Our key contributions are summarized as follows:

- **Theory:** We formally define the problem of determining s-t connectivity in uncertain networks. We prove computational complexity-theoretic results of the problem showing that it cannot be solved in polynomial time unless P=NP. The results provide useful insights to the inherent hardness and combinatoric nature of our problem.
- **Algorithm:** We derive an exact dynamic programming algorithm by converting our problem into an equivalent finite horizon Markov Decision Process. To further counter the problem, we design two approximation schemes. The first one is a simple greedy approach and we show that naive as it is, it can provide non-trivial performance guarantee. More surprisingly, its performance is far better than some other more complicated algorithms. Then, we further improve the approximation ratio of the greedy algorithm by utilizing the submodularity of the problem in the second algorithm.
- **Application:** We demonstrate the effectiveness of our algorithms on practical applications through extensive simulations with real network datasets. It is shown that our proposed algorithms are superior to the conventional heuristics as they achieve better tradeoff between the complexity of the algorithm and the optimality of the solutions.

The rest of the paper is organized as follows. we review related studies in Section II. In Section III, we formally introduce the definitions and notations related to our problem. In Section IV, we investigate the computational complexity of the problem. We present our exact dynamic programming algorithm based on Markov Decision Process framework in Section V. In Section VI, we present the two approximation algorithms and we evaluate our algorithms on real life data in Section VII. We give concluding remarks as well as future directions in Section VIII.

II. Related Work

1) Uncertain Networks: Uncertain network has been under intensive study for long. However, instead of verifying the existence of some structures in uncertain networks, most efforts have been devoted to calculating the existence probability of those structures. One of the fundamental problems in that regard is the network reliability problem, which asks the probability that uncertain networks are connected [1]. Following this, Jin et al. consider the distance-constrained reachability, i.e., the probability that two nodes are connected by a path shorter than a predefined threshold in an uncertain network [13]. The work in [14] focuses on discovering subgraphs with high reliability measure. In recent years, other types of study on uncertain networks (graphs) include reliable topology design [15], extracting representative subgraphs for the acceleration of various querying processes [16], performance analysis of unreliable wireless networks [8] as well as connectivity and resilience of secure sensor networks [9].

The modeling of uncertain networks in the present work is similar to random graph which was first introduced by Erdös and Renyi in [17]. Despite of this similarity, the problems investigated are quite different. Specifically, previous works on random graphs [17] [18] is dedicated to the analysis of model property in an asymptotic sense where the number of nodes goes to infinity. In contrast, our focus here lies in the combinatorial optimization problem formulated from the determination of source-destination connectivity in uncertain networks, with the model serving as a mean to characterize the uncertainty in networks and a parsimonious media for extracting the essence of the problem.

2) Sequential Testing: The nature of our problem is analogous to a class of sequential testing problems which involves diagnosing a system by determining the states of its components through a series of tests. The dependency of the whole system on its components' states can be given by explicit function or via an oracle. Existing results include optimal diagnosing strategies on series and parallel systems, double regular systems, etc. See [19] for a comprehensive review. A special class of sequential testing problems called Stochastic Boolean Function Evaluation (SBFE) have close connection to our problem. In SBFE, each component has two states and thus can be considered as a Boolean variable and the system is given by a Boolean function. The works in [20] and

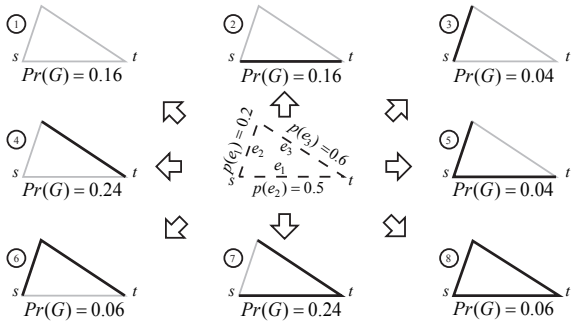


Fig. 1. An uncertain graph with three edges and its eight possible underlying graphs. The existence probability of each edge is labeled beside it. For clearance, we do not show the direction of each edge.

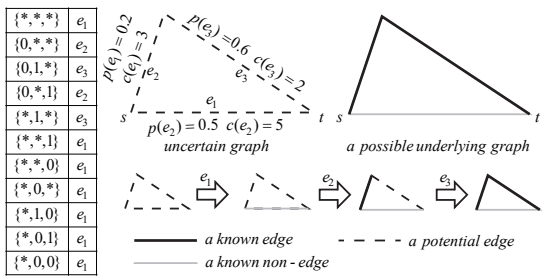


Fig. 2. The table in the left demonstrates an adaptive testing strategy with the action of terminating states omitted. The left entries denote the temporary states and the right entries represent the corresponding testing edges. The right part illustrates the transition of temporary states when the strategy is executed on the underlying graph in the figure. Following the strategy on the left, when the underlying graph is as shown on the right, the evolution of temporary state is {*,*,*},{0,*,*},{0,1,*},{0,1,1}. Note that some non-terminating states are not reachable during the testing process starting from the initial state, but we still show them in the figure in accordance to the definition of adaptive testing strategy, which is a mapping defined on the set of all temporary states. For clearance, we do not show the direction of each edge.

[21] propose approximate algorithms for evaluating DNF, CNF and CDNF formulas. Deshpande et al. [22] propose a general method called the Q-value approach to approximately solve SBFE problems based on the adaptive submodular framework proposed in [23].

We note that, there are no previous works that study the same problem as ours except the two from Kowshik [24] and Fu et al. [25] [26], respectively, but in more restrictive settings. Particularly, Kowshik derive the optimal solution for s-t connectivity problem in parallel-series and series-parallel uncertain graphs [24]. Fu et al. [25] [26] propose an efficient algorithm and prove its optimality in an ER graph, i.e, a complete graph where each edge has the same probability of existence and the cost of testing each edge is uniform. Our work is the first attempt to consider whether optimality exists in determination of s-t connectivity in a general uncertain graph.

III. Models and Problem Formulation

A. Uncertain Graph Model

We denote an uncertain directed graph by $\mathcal{G} = (V, E, p, c)$, where V is the set of vertices, E is the set of edges, $p : E \mapsto (0, 1]$ is a function that assigns each edge e its corresponding existence probability, and $c : E \mapsto \mathbb{R}^+$ represents the testing cost of each edge.

Following the state of art [13], we assume the existence probability of each edge to be independent. And we interpret $\mathcal{G}$ as a distribution on the set $\{G = (V, E_G), E_G \subseteq E\}$ of $2^{|E|}$ possible underlying deterministic graphs, where $|\cdot|$ denotes the cardinality of a set. The probability of a deterministic graph $G(V, E_G)$ being the underlying graph is:

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

We also use $G \in \mathcal{G}$ to represent that G is a possible underlying graph for $\mathcal{G}$. We define $\mathcal{G}$ to be s-t connected if there exists an s-t path in the underlying graph of $\mathcal{G}$. Figure 1 demonstrates an example of a three-edge uncertain graph with its possible underlying graphs.

B. Problem Formulation

Definition 1. *(Temporary State) A temporary state* $\mathbf{s}$ *of an uncertain graph* $\mathcal{G}(V, E, p, c)$ *is an* $|E|$*-dimension vector with element "0", "1" and "*". And we define* $\mathcal{S} = \{0, 1, *\}^{|E|}$ *to be the set of temporary states associated with* $\mathcal{G}$*.*

Each temporary state $\mathbf{s} \in \mathcal{S}$ represents a set of outcomes during the testing process, where "0" means the corresponding edge has been tested and found not existing, "1" means the corresponding edge has been tested and found existing and "*" means the corresponding edge has not been tested. Additionally, we denote the condition of edge e in state $\mathbf{s}$ as $\mathbf{s}_e$. As our goal is to determine the s-t connectivity of the underlying graph for $\mathcal{G}$, for a temporary state $\mathbf{s}$, we define it to be a *terminating state* if either the edge set $\{e \mid \mathbf{s}_e = 1\}$ forms a superset of an s-t path in $\mathcal{G}$ or edge set $\{e \mid \mathbf{s}_e = 0\}$ forms a superset of an s-t cut[1] in $\mathcal{G}$. We successfully determine the s-t connectivity by reaching a terminating state.

Definition 2. *(Adaptive Testing Strategy) An adaptive testing strategy is a deterministic mapping* $\pi : \mathcal{S} \mapsto E \cup \{\perp\}$*. Initially starting from the all-*$*$ *state, an adaptive testing strategy specifies which edge to test (or terminate as denoted by* $\perp$*) based on the previous testing outcomes.*

In the present work, we restrict our consideration to reasonable strategies where all the terminating states are mapped to $\perp$ and no state is mapped to any edge that has already been tested in that state. Also note that some states may never be reached but we still include them in the strategy for consistency.

During each determining process, we are associated with an underlying graph. The outcome of tests are dictated by the underlying graph and after each test the current temporary state will evolve into a new state. Therefore, an adaptive testing strategy may test different sets of edges before termination when executed on different underlying graphs of $\mathcal{G}$. For a specific underlying graph G, we denote $E_\pi(G)$ as the set of edges strategy π tests on it. Note that as G is deterministic, $E_\pi(G)$ is also deterministic. It follows that the expected cost

[1]All the cuts in this paper are graph s-t cut, i.e., the minimal cut sets that partition s and t into different subsets.

of π is given by:

$$Cost(\pi) = \sum_{G \in \mathcal{G}} [Pr(G) \sum_{e \in E_\pi(G)} c(e)],$$

where $\sum_{e \in E_\pi(G)} c(e)$ equals to the cost incurred by π when the underlying graph is G, and the expected cost is the weighted sum of the costs incurred on all the possible underlying graphs. An example of an adaptive testing strategy on an uncertain graph is illustrated in Figure 2.

Based on all the conditions above, now we give a formal definition of our problem stated as follows.

Definition 3. *(The Connectivity Determination Problem) Given an uncertain directed graph*[2] $\mathcal{G}(V, E, p, c)$ *with two nodes* $s, t \in V$ *designated as source and destination, respectively, the goal is to find an adaptive testing strategy* π *that incurs the minimum expected cost.*

Remark: Apart from deriving the strategy's action in all temporary states at once, an algorithm for the Connectivity Determination problem can instead compute the strategy sequentially, only deciding the next edge to test based on the current state. In algorithmic point of view, we consider the time complexity of an algorithm for Connectivity Determination problem as the maximum time it takes to compute all the relevant actions of a determining process. Therefore, finding the optimal strategy in a sequential fashion, on the surface, may appear to simplify the problem compared to computing it holistically. However, we show in next section that the problem is NP-hard regardless of in which way we compute the optimal strategy. Table I summarizes the notations that will be used throughout the paper.

Applicability of the model: We illustrate how to project our model into real situations using several examples. Let us firstly take communication networks for example, where the edge existence probability corresponds to the reliability of network links. The testing cost represents the probing costs of the links. The connectivity determination problem asks for a minimum cost probing strategy determining the source-destination connectivity of the networks, allowing for the prevalent existence of edge uncertainty. Other than communication networks, such uncertain graph can also be projected into the structure of a priced information, with each edge representing a piece of information (data) and its testing cost corresponds to the price of the data. Under such circumstance, the optimal strategy allows us to successfully query the information paying minimum price [30]. Another application is a social network graph where an edge can be treated as a first cousin relationship, and the object of interest is to determine whether two individuals from a large population are distant cousins. Obviously, determining whether an edge exists between two individuals is usually very expensive, involving costly genetic testing that outweighs any computational cost. An edge could of course represent a variety of other relationships that are expensive to check, for instance due to confidentiality or physical restrictions. In those scenarios where users have no prior knowledge of the whole network structure, it is reasonable to render the link existence between nodes as a preassigned probability. And the algorithm for the problem thus helps us more efficiently discover the relationship between nodes, which manifests its significant use in link prediction [31]. Moreover, the structure of the social networks can also be revealed [32], [33] by applying the algorithm to different node pairs. Last but not least, more potential applications of the illustrated scenario also include sensor networks [28], P2P networks [29] and Data center networks [7].

[2]Without loss of generality, we assume the graph with vertex set V and edge set E is s-t connected, i.e., $\mathcal{G}$ is s-t connected if all its edges exist.

IV. Computational Complexity

In this section, we investigate the computational complexity of the Connectivity Determination problem. By demonstrating the hardness of two closely related problems, we show both computing the testing strategy with the minimum expected cost holistically and sequentially are NP-hard. More specifically, we first convert our problem into its corresponding *decision version* that asks for the existence of an adaptive testing strategy with expected cost less than some value l for a given uncertain graph. Then, we consider the problem of deciding which edge to test first in the optimal strategy. The inherent tension of the Connectivity Determination problem is therefore disclosed through demonstrating the NP-hardness of these two problems, as stated in Theorems 1 and 2, respectively.

Theorem 1. *The decision version of Connectivity Determination Problem is NP-hard.*

Proof: Inspired by [27], we prove the theorem by reduction from the s-t reliability problem [1]: Given a directed graph G and two nodes s and t. The s-t reliability is to compute the probability of s being connected to t assuming the edges in G exist independently with probability $\frac{1}{2}$. As s-t reliability problem is #P-hard [1][3], its decision version that quests whether the probability of s being connected to t is larger than some predefined value r_0 is NP-hard.

The reduction works as follows. For a graph $G(V, E)$, we transform it to an uncertain graph $\mathcal{G}(V, E', p, c)$ by adding an edge M between s, t and set the rest of $\mathcal{G}$ is just the same as G. Define n as the number of edges in G. We set the cost of M as $c(M) = n2^{n+1}$ and the cost of testing other edges as 1. Then we assign the existence probability of all edges in $\mathcal{G}$ as $\frac{1}{2}$. Finally, we designate s, t in $\mathcal{G}$ as the source and the destination in the constructed instance.

Let r be the s-t reliability in G and l be the expected cost incurred by the optimal testing strategy on $\mathcal{G}$. We define a generic G' as a subgraph resulted from an underlying graph of G with edge M removed. We will show that if we know l, then we can efficiently compute r.

First, from the definitions, we have $r = \frac{k}{2^n}$ for some integer k, and l must obey the following two constraints: $l \geq (1-r)c(M)$ and $l \leq rn + (1-r)c(M)$. Here, the first inequality follows from the fact that we have to test M whenever we find out that s and t is not connected in G'. The second inequality holds since the expected cost of the

[3]#P is a complexity class for counting problems. #P-hard is at least as hard as NP-hard [1].

TABLE I
NOTATIONS AND DEFINITIONS

Notation	Definition
$\mathcal{G}$	uncertain directed graph
V	vertex set of the uncertain graph
E	edge set of the uncertain graph
p	probability function indicating the existence probability of edges in the uncertain graph
c	cost function indicating the testing cost of edges in the uncertain graph
G	underlying deterministic graph
s, t	source and destination
$\mathcal{S}$	set of temporary states
$\mathbf{s}, \mathbf{s}', \mathbf{a}, \mathbf{b}$	temporary states
$\mathbf{s}_e$	the element corresponding to edge e in state $\mathbf{s}$
π	adaptive testing strategy
$E_\pi(G)$	set of edges π tests before termination when the underlying graph is G
$Cost(\pi)$	the expected cost of strategy π
u	utility function in the Markov Decision Process
g	utility function in the Q-value approach
$\mathcal{P}$	the collection of s-t paths in $\mathcal{G}$
$\mathcal{C}$	the collection of s-t cuts in $\mathcal{G}$
$\mathcal{P}_e$	the subfamily of s-t paths in $\mathcal{G}$ that edge e lies on
$\mathcal{C}_e$	the subfamily of s-t cuts in $\mathcal{G}$ that edge e lies on
Q	the goal value in the Q-value approach

optimal strategy is certainly no greater than that of a simple strategy that first test all the edges in E and test M if no s-t path is found. Combining the two inequalities, we have $2^n \frac{c(M)-l}{c(M)} \leq k \leq 2^n \frac{c(M)-l}{c(M)-n}$. Consequently, $k = \lfloor 2^n \frac{c(M)-l}{c(M)-n} \rfloor$. Therefore, if we have a polynomial time algorithm that solves the decision version of Connectivity Determination problem, then we can efficiently solve the decision version of s-t reliability problem. Since the latter is NP-hard, we conclude that the decision version of Connectivity Determination problem is also NP-hard. ■

Theorem 2. *Deciding the optimal first edge to test (the edge tested by the optimal strategy in the intial state) is NP-hard.*

Proof: We only present a proof sketch here and refer the details to Appendix A. The proof is done by reduction from set cover problem. Given a universe of elements, a family of subsets of the universe and a predefined integer k, a cover is a subfamily of sets whose union equals to the universe. The set cover problem asks whether there exists a cover of cardinality less than k. For a set cover instance, we construct a corresponding uncertain graph as follows. We first create a set vertex for each subset in the family and an element vertex for each element in the universe. Next, we add three special vertices: source s, destination t and a special set vertex s_M. Then, we add edges from s to each set vertex, from each element vertex to t and from each set vertex to the element vertices it contains in the original instance. Specially, we add edges from s_M to all the element vertices. By carefully assigning the cost and probability of each edge, we prove that the optimal first edge to test is the edge M from s to s_M if and only if there does not exist a cover of size smaller than k in the original set cover instance. Figure 3 presents the uncertain graph constructed for a set cover instance. ■

Remark: The two theorems characterize the complexity of the Connectivity Determination problem from two aspects. Theorem 1 establishes the NP-hardness of the decision version of our problem, which implies the NP-hardness of computing the optimal strategy in a holistic fashion. Theorem 2 shows that even computing the optimal testing strategy sequentially cannot be completed in polynomial time unless P=NP.

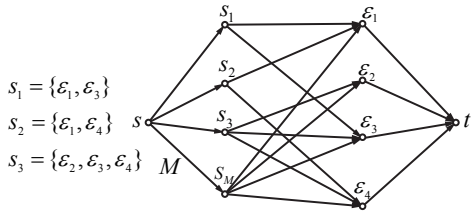


Fig. 3. The uncertain graph (right) constructed for the set cover instance (left). We omit the probability and cost of edges in the uncertain graph and refer them to the Appendix A.

V. MDP-BASED EXACT ALGORITHM

The NP-hardness analysis in the previous section implies that solving the problem exactly may lead to a prohibitively large cost. However, it is still essential to design an exact algorithm to capture the features of the optimal solutions and gain insights of our Connectivity Determination problem. The main idea of seeking for an exact algorithm is through converting our problem into an equivalent Markov Decision Process (MDP). Adopting the notations in [34], in the sequel, we will first show how the elements in our problem can be naturally mapped to the components in a finite horizon MDP.

A. Mapping the Problem into MDP

As a mathematical framework for planning or navigating uncertain systems, MDP models the way of a decision maker's choosing actions so that the system can perform optimally with regard to some predefined criterion. The key components of an MDP include decision epochs, state space, action sets, transition probabilities, rewards, decision policy and optimality criterion. Regarding this, now we show the mapping between these components and the elements in our problem one by one. The correspondence is also summarized in Table II.

TABLE II
THE CORRELATION BETWEEN CONNECTIVITY DETERMINATION PROBLEM AND MARKOV DECISION PROCESS

Markov Decision Process	Connectivity Determination Problem
state space	set of temporary states $\mathcal{S}$
state	a temporary state $\mathbf{s}$
action set	testing or termination, $E \cup \{\perp\}$
transition probability function P	probability function p of edges
reward function r	cost function c of edges
decision policy	adaptive testing strategy π

- **Decision Epochs:** In an MDP, decisions are made at points of time called decision epochs. In our problem the decision epochs are the times we need to decide which edge to test next or terminate. Since we at most need to test $|E|$ edges where $|E|$ is the number of possible edges in the uncertain graph, our corresponding MDP is of finite horizon.
- **State Space:** The state space of an MDP represents the possible states that a system can be in. It naturally

corresponds to the set of temporary states $\mathcal{S}$ in our problem. We may also partition the state space $\mathcal{S}$ into $|E|$ disjoint subsets based on the number of edges having been tested in the states as $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_{|E|}$. In decision epoch i, the system can only be in a state in $\mathcal{S}_i$.

- **Action Sets:** For each state $\mathbf{s} \in \mathcal{S}$, there is a set of actions that can be performed under it. We define the associated action set $A_{\mathbf{s}}$ of state $\mathbf{s}$ as the set of edges that have not been tested in $\mathbf{s}$. Additionally, for terminating states, their action set also contains the terminating action $\perp$. As a result, the whole action set $A = \bigcup_{\mathbf{s}\in\mathcal{S}} A_{\mathbf{s}} = E \cup \{\perp\}$.
- **Transition Probabilities and Rewards:** The transition probability function and reward function characterize the result of choosing some action at some state. Generally speaking, at each state, choosing an action will gain some reward and the system will evolve into other states probabilistically at the next decision epoch. Projecting into our problem, the transition probability of action e (testing edge e) is given by the existence probability of edge e. Denote by $\mathbf{s} \cdot e$ the temporary state evolved from $\mathbf{s}$ by setting $\mathbf{s}_e$ as 1 and by $\mathbf{s}\backslash e$ the temporary state evolved from $\mathbf{s}$ by setting $\mathbf{s}_e$ as 0. Formally, the transition probability function is given by:
$$P(\mathbf{s}'|\mathbf{s}, e) = \begin{cases} p(e) & \text{if } \mathbf{s}' = \mathbf{s} \cdot e, \\ 1 - p(e) & \text{if } \mathbf{s}' = \mathbf{s}\backslash e, \\ 0 & \text{otherwise,} \end{cases}$$
and
$$P(\mathbf{s}'|\mathbf{s}, \perp) = \begin{cases} 1 & \text{if } \mathbf{s}' = \mathbf{s}, \\ 0 & \text{otherwise.} \end{cases}$$
Then it follows that the reward function is $r(\mathbf{s}, e) = -c(e)$ and $r(\mathbf{s}, \perp) = 0$. Note that the reward function is negative, corresponds to the cost and the transition probability and reward function are independent with regard to decision epochs or previous state, which demonstrates the Markov property of our problem.
- **Decision Policy:** A decision policy is a mapping from state space to action set. Therefore, in our problem, it is equivalent to an adaptive testing strategy.
- **Optimality Criterion:** Obviously, in our case, the optimality criterion is the expected total reward criterion, i.e., the decision policy with the maximum expected total reward of the constructed MDP corresponds to the optimal adaptive testing strategy.

B. Exact Dynamic Programming Algorithm

From the equivalence between our problem and an MDP, it follows that our problem also satisfies the "Principle of Optimality" in MDP [34], i.e., starting at any states, the optimal adaptive testing strategy incurs the minimum expected cost among all strategies. This enables us to define the *optimal utility function* u of states assigning each temporary state the expected reward (negative cost) of the optimal strategy starting at that state. Similarly, we define a utility function u_π associated with strategy π as the reward gained by π starting from each state. By the Bellman equation [34], we have the following lemma.

Lemma 1. *For any state $\mathbf{s} \in \mathcal{S}$, the optimal utility function satisfies*
$u(\mathbf{s}) = \max_{a\in A_{\mathbf{s}}}\{r(\mathbf{s}, a) + \sum_{\mathbf{s}'\in\mathcal{S}} P(\mathbf{s}' \mid \mathbf{s}, e)u(\mathbf{s}')\}$.
Particularly, if $\mathbf{s}$ is a non-terminating state, then
$u(\mathbf{s}) = \max_{e\in A_{\mathbf{s}}}\{-c(e) + p(e)u(\mathbf{s}\cdot e) + (1-p(e))u(\mathbf{s}\backslash e)\}$
and for any terminating state, its utility is 0.

Based on the Lemma 1, we design an algorithm that computes the optimal testing strategy π following the standard dynamic programming paradigm, as shown in **Algorithm 1**.

Algorithm 1 The MDP-based Exact Algorithm

Input: Uncertain graph $\mathcal{G}(V, E, p, c)$, source s, destination t
Output: The optimal testing strategy π

Initialize: $u_\pi(\mathbf{s}) = 0$, for all $\mathbf{s} \in \mathcal{S}_{|E|}$
for $i = |E|$ to 0 **do**
  for All $\mathbf{s}$ in $\mathcal{S}_i$ **do**
    if $\mathbf{s}$ is a terminating state **then**
      $u_\pi(\mathbf{s}) := 0$, $\pi(\mathbf{s}) := \perp$.
    else
      $e^* := \arg\max_{e\in A_{\mathbf{s}}}\{-c(e) + p(e)u_\pi(\mathbf{s}\cdot e) + (1-p(e))u_\pi(\mathbf{s}\backslash e)\}$,
      $u_\pi(\mathbf{s}) := -c(e^*) + p(e^*)u_\pi(\mathbf{s}\cdot e^*) + (1-p(e^*))u_\pi(\mathbf{s}\backslash e^*)$,
      $\pi(\mathbf{s}) := e^*$.
return π

We prove the correctness of the dynamic programming algorithm in the following theorem.

Theorem 3. *For an uncertain graph $\mathcal{G}$, Algorithm 1 yields an optimal adaptive testing strategy and has a complexity of $O((|V| + |E|)3^{|E|})$, where $|V|$ denotes the number of nodes and $|E|$ denotes the number of edges in $\mathcal{G}$.*

Proof: Denote an optimal testing strategy as π^*, the strategy given by Algorithm 1 as π. By backward induction, we prove that the utility function u_π of π is no less than the optimal utility function $u_{\pi^*} = u$ on every state, which implies that π is an optimal strategy.

First, for all $\mathbf{s} \in \mathcal{S}_{|E|}$, obviously $u_\pi(\mathbf{s}) = u_{\pi^*}(\mathbf{s}) = 0$. Suppose for all states $\mathbf{s} \in \mathcal{S}_i, i \geq k$, $u_\pi(\mathbf{s}) \geq u_{\pi^*}(\mathbf{s})$, then we prove that for all states $\mathbf{s} \in \mathcal{S}_{k-1}, u_\pi(\mathbf{s}) \geq u_{\pi^*}(\mathbf{s})$. Indeed, by the selection criterion of the algorithm, for a state $\mathbf{s} \in \mathcal{S}_{k-1}$ that is non-terminating, we have

$$\begin{aligned} u_\pi(\mathbf{s}) =& \max_{e\in A_{\mathbf{s}}}\{-c(e) + p(e)u_\pi(\mathbf{s}\cdot e) + (1-p(e))u_\pi(\mathbf{s}\backslash e)\} \\ \geq& -c(\pi^*(\mathbf{s})) + p(\pi^*(\mathbf{s}))u_\pi(\mathbf{s}\cdot\pi^*(\mathbf{s})) \\ &+ (1-p(\pi^*(\mathbf{s})))u_\pi(\mathbf{s}\backslash\pi^*(\mathbf{s})) \\ \geq& -c(\pi^*(\mathbf{s})) + p(\pi^*(\mathbf{s}))u_{\pi^*}(\mathbf{s}\cdot\pi^*(\mathbf{s})) \\ &+ (1-p(\pi^*(\mathbf{s})))u_{\pi^*}(\mathbf{s}\backslash\pi^*(\mathbf{s})) \qquad (1) \\ =& u_{\pi^*}(\mathbf{s}), \end{aligned}$$

where Inequality (1) follows from the induction hypothesis. And if $\mathbf{s}$ is a terminating state, then also $u_\pi(\mathbf{s}) = u_{\pi^*}(\mathbf{s}) = 0$. Hence, we prove that under every state $\mathbf{s}$, following π is optimal, and particularly from the initial all-$*$ state, π returns the maximum expected reward, or equivalently, incurs the minimum expected cost.

The time complexity of the algorithm can be justified as follows. There are in total $3^{|E|}$ temporary states associated with the uncertain graph. Qualifying whether a state $\mathbf{s}$ is a terminating state can be realized by querying the s-t connectivity on two deterministic graphs $G^1_{\mathbf{s}}(V, E_1)$ and $G^2_{\mathbf{s}}(V, E_2)$, where $E_1 = \{e \mid \mathbf{s}_e = 1\}$ and $E_2 = \{e \mid \mathbf{s}_e = 1 \text{ or } \mathbf{s}_e = *\}$. This is implementable in $O(|V| + |E|)$ time by depth-first traversal, and selecting the optimal action for each state requires $O(|E|)$ time. Hence, the algorithm terminates and finds the optimal solution in $O((|V| + |E|)3^{|E|})$ time. ■

Remark: Note that the exact algorithm computes the whole testing strategy in one run, meaning that we can obtain the corresponding action for all temporary states by invoking the algorithm once. Therefore, on average, the exact algorithm takes $O(|E|)$ time to determine the optimal action for each temporary state. However, as the time complexity we consider here is the total time for an algorithm to determine all the actions of a testing process and we have to execute the whole algorithm at the beginning of each testing process, the exact algorithm still has exponential time complexity. On the other hand, the approximation algorithms presented in the following section exploits the sequential way of computing the testing strategy. Based on each temporary state, they use polynomial time to determine the corresponding action, thus can be categorized as polynomial time approximation algorithms.

VI. Approximation Algorithms

As stated previously, it is unrealistic to pursue efficient exact algorithm on general uncertain graphs due to the inherent tension of the Connectivity Determination problem. Therefore, in this section, we propose approximation schemes that have both polynomial time complexity and good approximation guarantee. Specifically, we focus on analyzing the approximation ratios of the two proposed algorithms. The readers may refer to Appendix IV for more details of their time complexity.

A. A Simple Greedy Approach

An intuitive greedy algorithm is to test the edge with the minimum cost (breaking ties arbitrarily). Surprisingly, we show that this greedy algorithm has a non-trivial approximation ratio of $O(|E|)$.

Theorem 4. *Given an instance of our problem with uncertain graph $\mathcal{G}(V, E, p, c)$ and two nodes s, t as source and destination, let π be a strategy that tests the edges in $\mathcal{G}$ according to their costs sorted in an increasing order. Then, $Cost(\pi) \leq |E| \cdot Cost(\pi^*)$, where π^* is the optimal strategy.*

Proof: Suppose that we know the underlying graph G of $\mathcal{G}$ in advance. Denote $Cert(G)$ as the certifier of G's s-t connectivity with the minimum cost. If s and t are connected in G, then a certifier consists of an s-t path in $\mathcal{G}$ whose edges exist in G. If s and t are disconnected in G, then a certifier consists of an s-t cut in $\mathcal{G}$ whose edges do not exist in G. Since π tests the edges from cheap to expensive, we must have $\sum_{e \in E_\pi(G)} c(e) \leq |E| \cdot Cert(G)$ for any G. And due to the fact that even the optimal strategy has no prior knowledge of the underlying graph G, clearly $\sum_{e \in E_{\pi^*}(G)} c(e) \geq Cert(G)$. Therefore, $Cost(\pi) = \sum_{G \in \mathcal{G}} [Pr(G) \sum_{e \in E_\pi(G)} c(e)] \leq |E| \cdot \sum_{G \in \mathcal{G}} [Pr(G) Cert(G)] \leq |E| \cdot Cost(\pi^*)$. ■

Strictly speaking, the greedy algorithm yields non-adaptive strategies, i.e., the strategies it derives do not make decisions based on previous results but tests the edges following a predefined order. We can modify it into an adaptive version, which omits the edges that do not effect the s-t connectivity, e.g. the edges that do not lie on any of the s-t paths in the current state. Although the adaptive version of the greedy algorithm has better performance, Theorem 4 holds for both the adaptive and non-adaptive version.

There are three further notes regarding the properties of the greedy algorithm: (i) Although the proof of Theorem 4 is completed by comparing the performance of the greedy algorithm to the minimum certifier cost, the resulting bound is actually tight, (ii) The approximation ratio of an alternative greedy algorithm based on the existence probability of edges is far worse than $O(|E|)$ and (iii) The greedy algorithm only considers the testing cost of edges. However, it has significantly better performance guarantee than some seemingly more sophisticated algorithms that take into account the existence probabilities of edges. The soundness of the above three arguments is supported in our Appendices C.

B. Adaptive Submodular Algorithm

To further improve the approximation ratio, we adopt the Q-value approach in Stochastic Boolean Function Evaluation Problem (SBFE) proposed by [22]. Based on that, we utilize the adaptive submodularity [23] in our problem and propose the Adaptive Submodular Algorithm, of which the approximation ratio is logarithmic to the number of edges for most uncertain graphs.

1) Preliminaries: The Q-value approach is proposed for the SBFE problem, which has close connection with our problem. The SBFE problem is that given a Boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ on an unknown input $\mathbf{x}$. Each bit x_i of $\mathbf{x}$ can only be determined by paying a cost c_i. The prior probability of the value of each bit being 1 is specified by a probability vector $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{x}$ is drawn from the corresponding product distribution. The goal is to derive an evaluation strategy minimizing the expected cost. The relation between our problem and SBFE can be established by considering each edge as a Boolean variable and see the function as implicitly given by the s-t connectivity of the uncertain graph. However, as constructing such a function from a graph may have exponential time complexity, we cannot directly use the algorithms proposed for SBFE problems.

We then introduce some useful definitions for the Q-value approach adapted for our problem. For two temporary states $\mathbf{a}, \mathbf{b} \in \mathcal{S}$, $\mathbf{a}$ is an extension of $\mathbf{b}$, written as $\mathbf{a} \sim \mathbf{b}$, if $\mathbf{a}_i = \mathbf{b}_i$ for all $\mathbf{b}_i \neq *$. A function $g : \mathcal{S} \mapsto \mathbb{N}$ is said to be monotone if for $\mathbf{s} \in \mathcal{S}$ and all $\mathbf{s}' \sim \mathbf{s}$, $g(\mathbf{s}') - g(\mathbf{s}) \geq 0$. g is submodular if $g(\mathbf{s} \cdot e) - g(\mathbf{s}) \geq g(\mathbf{s}' \cdot e) - g(\mathbf{s}')$ and $g(\mathbf{s} \backslash e) - g(\mathbf{s}) \geq g(\mathbf{s}' \backslash e) - g(\mathbf{s}')$ whenever $\mathbf{s}' \sim \mathbf{s}$ and $\mathbf{s}_e = \mathbf{s}'_e = *$. For a state $\mathbf{s} \in \mathcal{S}$ and edge e with $\mathbf{s}_e = *$, the expected marginal gain of the edge to the current state with respect to g is given by $p(e)g(\mathbf{s} \cdot e) + (1 - p(e))g(\mathbf{s} \backslash e) - g(\mathbf{s})$.

The Q-value approach: The Q-value approach [22] states that if we have a utility function $g : \mathcal{S} \mapsto \mathbb{N}$ that satisfies: (1) g is monotone and submodular, (2) $g(*, *, \ldots, *) = 0$, and (3) for any temporary state $\mathbf{s} \in \mathcal{S}$, $g(\mathbf{s}) = Q$ iff $\mathbf{s}$ is a terminating state, then g is *assignment feasible* with goal value Q for our problem. And by using the adaptive greedy algorithm proposed in the Adaptive Submodular framework in [23] that suggests testing the edge with the maximum ratio between its expected marginal gain and cost each time, we yield a solution that is within a factor of $(\ln Q + 1)$ of the optimum. For completeness, we state the relevant result in the following lemma.

Lemma 2. *[23] Given a utility feasible function g, set of states $\mathcal{S}$ and goal value Q, we are to select a certain action on current states, and the state transition is governed by the action. Our goal is to make the state evolve to some terminating state $\mathbf{s}$ such that $g(\mathbf{s}) = Q$. Then, the strategy that maximize the ratio between the gain with respect to g and the cost of the action is a $(\ln Q+1)$-approximation of the minimum cost strategy.*

2) Applying the Q-value Approach: To harness the Q-value approach, we need to choose appropriate utility function g. And we present in the following the utility function that we design for our algorithm.

Given an uncertain graph $\mathcal{G}(V, E, p, c)$, we denote by $\mathcal{P}$ the collection of s-t paths in $\mathcal{G}$, and by $\mathcal{C}$ the collection of s-t cuts in $\mathcal{G}$. For an edge e, we define $\mathcal{P}_e$ as the set of s-t paths it lies on in $\mathcal{G}$ and $\mathcal{C}_e$ as the set of minimal s-t cuts[4] it lies on in $\mathcal{G}$. Note that the above definitions interpret $\mathcal{G}$ as a deterministic graph with vertex set V and edge set E. Then, for each temporary state s, we define two auxiliary functions g_p and g_c as:

$$g_p(\mathbf{s}) = |\bigcup_{e:\mathbf{s}_e=0} \mathcal{P}_e|, \quad g_c(\mathbf{s}) = |\bigcup_{e:\mathbf{s}_e=1} \mathcal{C}_e|.$$

Our utility function $g : \mathcal{S} \mapsto \mathbb{N}$ is given by:

$$g(\mathbf{s}) = |\mathcal{P}||\mathcal{C}| - (|\mathcal{P}| - g_p(\mathbf{s}))(|\mathcal{C}| - g_c(\mathbf{s})).$$

The intuitive explanation for the functions g_p, g_c and g is that if we view the determining process as a covering process, then the non-existence of an edge can be regarded as covering the paths it lies on and the existence of an edge is equivalent to covering the cuts it lies on. If all the paths in $\mathcal{G}$ have been covered in some state $\mathbf{s}$, then we have $g_p(\mathbf{s}) = |\mathcal{P}|$ and conclude that s and t in the underlying graph of $\mathcal{G}$ must be disconnected and the converse is also true. The dual case holds similarly. Therefore, when $\mathbf{s}$ is a terminating state, we must have $g(\mathbf{s}) = Q$. Theorem 5 demonstrates the validity of the utility function that we construct.

Theorem 5. *The utility function g is assignment feasible.*

Proof: We prove the theorem by showing that g satisfies the three conditions mentioned above. First, obviously both g_p and g_c are monotone, it follows that g is also monotone. And since g_p and g_c are easily verified to be submodular, we have

$$g_p(\mathbf{s} \cdot e) - g_p(\mathbf{s}) \geq g_p(\mathbf{s}' \cdot e) - g_p(\mathbf{s}'),$$
$$g_p(\mathbf{s} \backslash e) - g_p(\mathbf{s}) \geq g_p(\mathbf{s}' \backslash e) - g_p(\mathbf{s}'),$$
$$g_c(\mathbf{s} \cdot e) - g_c(\mathbf{s}) \geq g_c(\mathbf{s}' \cdot e) - g_c(\mathbf{s}'),$$
$$g_c(\mathbf{s} \backslash e) - g_c(\mathbf{s}) \geq g_c(\mathbf{s}' \backslash e) - g_c(\mathbf{s}'),$$

whenever $\mathbf{s}' \sim \mathbf{s}$ and $\mathbf{s}'_e = \mathbf{s}_e = *$. Note that actually, $g_p(\mathbf{s} \cdot e) - g_p(\mathbf{s}) = g_c(\mathbf{s} \backslash e) - g_c(\mathbf{s}) = 0$ for all $\mathbf{s}$ and e such that $\mathbf{s}_e = *$. Then, combining the fact that $g(\mathbf{s} \cdot e) - g(\mathbf{s}) = (|\mathcal{P}| - g_p(\mathbf{s} \cdot e))(g_c(\mathbf{s} \cdot e) - g_c(\mathbf{s}))$ and $g(\mathbf{s} \backslash e) - g(\mathbf{s}) = (|\mathcal{C}| - g_c(\mathbf{s} \backslash e))(g_p(\mathbf{s} \backslash e) - g_p(\mathbf{s}))$, we have g is submodular. Second, since $g_p(*, *, \ldots, *) = g_c(*, *, \ldots, *) = 0$, $g(*, *, \ldots, *)$ is also zero. The third condition is explained above. Hence, g is an assignment feasible utility function. ■

3) The Adaptive Submodular Algorithm: By applying the Q-value approach [22] with our utility function, we have our Adaptive Submodular algorithm shown in **Algorithm 2**. Note that the algorithm computes the testing strategy sequentially, i.e., in one iteration, it only determines the next edge to test based on the current temporary state.

4) Performance Guarantee: By Lemma 2, the Adaptive Submodular algorithm yields an approximation of $O(\ln Q) = O(\ln(|\mathcal{P}||\mathcal{C}|))$. Since for most graphs, the number of paths and the number of cuts are polynomial to the number of edges, Algorithm 2 has logarithmic approximation ratio in most cases. However, in some extreme cases, the number of paths or cuts can be exponentially large, making the worst case approximation ratio still turn out to be $O(|E|)$. Also note that in the algorithm we do not specify how to implement the selection rule in line 3 of Algorithm 2, hence the algorithm can be viewed as a framework that can embody any valid greedy selection algorithm. Standard implementation of the selection rule involves counting the number of paths and cuts in graph $\mathcal{G}$, which is #P-hard [1] in general. So, we may use polynomial time approximate counting schemes [35], [36] that can preserve an approximation ratio of $O(\ln(\alpha|\mathcal{P}||\mathcal{C}|))$, if the scheme can guarantee that the expected marginal gain of the selected edge is within a factor α of the optimal gain. Furthermore, when the number of paths and the number of cuts are polynomial to the number of edges, we can also use efficient enumerating schemes [37] to select the next edge to test following the criterion of our Adaptive Submodular algorithm.

Algorithm 2 The Adaptive Submodular Algorithm

Input: Uncertain graph $\mathcal{G}(V, E, p, c)$, source and destination nodes s, t.

Output: An approximate adaptive testing strategy

1: **Initialize:** Current state $\mathbf{s} := (*, *, \ldots, *)$, The set of tested edges E_π as an empty set.
2: **Repeat** until s becomes a terminating state.
3: $e^* := \arg\max_{e \in E \backslash E_\pi} \{\frac{p(e)g(\mathbf{s} \cdot e) + (1-p(e))g(\mathbf{s} \backslash e) - g(\mathbf{s})}{c(e)}\}$.
4: $E_\pi := E_\pi \cup \{e^*\}$, *test* e^* and observe the outcome.
5: **if** edge e^* exists **then**
6: $\quad \mathbf{s}_{e^*} := 1$
7: **else**
8: $\quad \mathbf{s}_{e^*} := 0$

[4] An s-t cut is minimal if and only if no proper subset of it is an s-t cut.

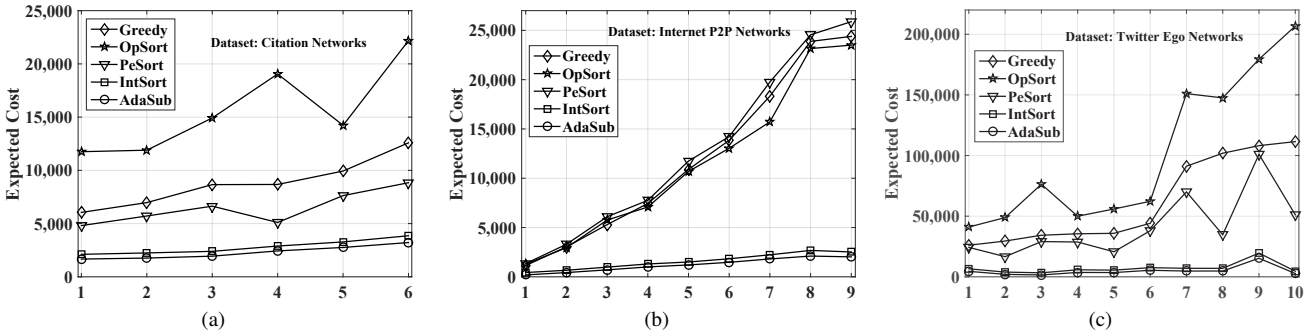


Fig. 4. The expected cost of the adaptive testing strategies yielded by different algorithms. The x-coordinates of the figure follow the increasing order of the size of the uncertain graphs.

VII. Simulations

In this section, we present our simulations on the performance of the proposed algorithms on various datasets. We first introduce our simulation environment in the following and show the detailed results in subsequent sections.

A. Simulation Settings

1) Simulation Datasets: We adopt three real life datasets in our simulations. The basic descriptions and statistics are listed as follows:

- **Citation Networks (from Microsoft Academic Graph [40]):** We extract six citation networks of different subfields in Microsoft Academic Graph ranging from 749 nodes (1429 edges) to 273751 nodes (993025 edges) to generate six uncertain graphs.
- **Internet Peer to Peer Networks [38]:** This dataset contains a snapshot of the Gnutella peer-to-peer file sharing network in August 2002. We extract nine subnetworks ranging from 1000 nodes (1700 edges) to 5000 nodes (16469 edges) to form nine uncertain graphs.
- **Twitter Ego Networks [39]:** This dataset consists of ego networks in Twitter. We select 10 ego networks ranging from 95 nodes (1376 edges) to 213 nodes (17930 edges) to create 10 uncertain graphs. Note that the ego networks we use have high density.

For each uncertain graph generated above, we use Jaccard's coefficient [12], which is an established metric for link prediction in social networks, to assign the existence probabilities of the edges. Specifically, for an edge $e = (x, y)$ in uncertain graph $\mathcal{G}$, the existence probability of e is given as $p(e) = |\Gamma(x) \cap \Gamma(y)|/|\Gamma(x) \cup \Gamma(y)|$, where $\Gamma(\cdot)$ denotes the set of neighbors of a node in the graph. We construct the cost function of the uncertain graphs by assigning the cost of each edge from a Gaussian distribution with mean 50 and standard deviation 10. The negative part of the distribution is truncated.

2) Calculation of the Performance Metric: The performance metric in the simulations is the expected cost of the strategies derived by the algorithms. However, to calculate the exact expected cost of a strategy requires testing it on all the possible underlying graphs of an uncertain graph, of which the number is extremely large. Therefore, instead, we first generate 1000 underlying graphs by sampling from the distribution given by the uncertain graph and then use the average cost of a strategy incurs on the 1000 underlying graphs to approximate the expected cost of the strategy. To further eliminate the random noise in data, we designate 10 pairs of source and destination in each uncertain graph, and the final results shown in the figures are the average costs incurred by strategies among all pairs of source and destination.

3) Algorithms Involved in Performance Comparisons: To evaluate the performance of our proposed algorithms, we include three additional heuristics adapted from [19]. We briefly introduce the algorithms as follows:

- **Greedy Algorithm (Greedy):** The greedy algorithm that tests the edges from low cost to high cost proposed in Section VI.
- **Adaptive Submodular Algorithm (AdaSub):** The Adaptive Submodular algorithm based on the Q-value approach proposed in Section VI.
- **MDP-based Algorithm (MDP):** The exact dynamic programming algorithm applying the Markov Decision Process framework proposed in Section 1.
- **Optimistic Sort Algorithm [19] (OpSort):** The algorithm yields a strategy that tests the edges following the increasing order of c/p. OpSort is optimal when the uncertain graph is a parallel graph [19].
- **Pessimistic Sort Algorithm [19] (PeSort):** The algorithm yields a strategy that tests the edges following the increasing order of $c/(1-p)$. PeSort is optimal when the uncertain graph is a serial graph [19].
- **Intersection Sort Algorithm [19] (IntSort):** The algorithm that tests the edge with the minimum cost that lies on the intersection of a shortest s-t path and a minimum s-t cut in the uncertain graph under the current state.

Due to the prohibitive time complexity of the MDP-based algorithm, we only apply it to a sequence of subnetworks with 20 edges that are extracted from the citation networks.

B. Evaluation of Proposed Algorithms

We plot the expected costs of the strategies derived by different algorithms on various uncertain graphs in Figures 4 and 5.

From Figure 5, we can see that the Adaptive Submodular algorithm indeed yields near optimal testing strategies, of which the expected cost is at most 8% higher than the minimum expected cost achieved by the MDP-based algorithm. On the other hand, although the Greedy algorithm is relatively simple,

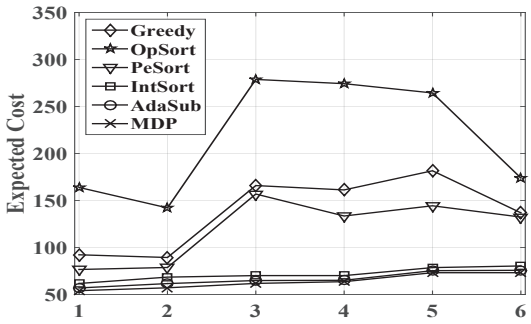


Fig. 5. Comparisons with the exact MDP-based algorithm on six small subgraphs of the citation networks.

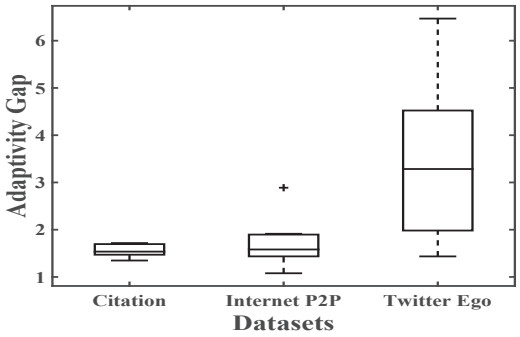


Fig. 6. The adaptivity gap of the Greedy algorithm on different datasets.

the expected costs of the transmission schemes it derives are within a factor of 2.7 times the optimal ones.

As demonstrated in Figure 4, the Adaptive Submodular algorithm derives the strategies with the minimum expected cost among the five compared algorithms in all three data sets. However, the gap between it and the Intersection Sort heuristic is small. This can be attributed to the fact that in many cases, the intersection of a shortest s-t path and a minimum s-t cut is identical to the edge selected by the rule in the Adaptive Submodular algorithm. However, as shown in Appendix C-D, the Intersection Sort heuristic does not possess such theoretical guarantee as the Adaptive Submodular algorithm.

For the other three compared algorithms, although it seems that the Optimistic Sort and Pessimistic Sort utilize more information than the Greedy algorithm, as they take into account the existence probabilities of edges. There exists no significant gap between the Greedy algorithm and the other two. An explanation to this is that the real life networks are mixed with parallel and serial structures. So often the sole selection criterion in OpSort or PeSort does not match the optimum. Also, as demonstrated in Appendix C-C, surprisingly, the worst case approximation ratios of both OpSort and PeSort are far *worse* than the Greedy algorithm. Therefore, the Greedy algorithm indeed achieves a desirable compromise between OpSort and PeSort.

Finally, an important observation from our simulation results is that: **larger size is not equivalent to higher cost**. Despite that the largest uncertain graph generated by the citation networks has about one million edges, the cost of determining s-t connectivity in it is still considerably lower than in Twitter networks. This phenomenon results from the fact that the Twitter networks are far denser than the other two datasets, which means that there are significantly larger number of edges that lie on the paths from source and destination, influencing the s-t connectivity. Therefore, instead of the total number of edges, it is the number of relevant edges that determines the scale of the expected testing cost.

C. Characterization of the Adaptivity Gap

We investigate the significance of the adaptivity of testing strategies by capturing the gap between the adaptive and non-adaptive versions of the Greedy algorithm on different datasets. Specifically, we define the *adaptivity gap* of the Greedy algorithm as the ratio between the expected cost of the strategies derived by the non-adaptive version and the adaptive version. We calculate the adaptivity gap for each uncertain graph in the three datasets and show the results in Figure 6. It turns out that the adaptivity gap in Citation and Internet Peer to Peer networks can be as large as two. For the dense Twitter Ego network, the gap may even reach up to six. Hence, harnessing the results of previous tests and making the strategy adaptive bring significant gain in the testing costs.

VIII. Conclusion and Future Work

In this paper, we modeled the network as an uncertain graph where each edge e exists independently with some probability $p(e)$ and examined the problem of determining whether a given pair of source node and destination node are connected by a path or separated by a cut. Assuming that during each determining process we are associated with an underlying graph, the existence of each edge can be unraveled through edge testing at a cost of $c(e)$. We aimed to find an optimal strategy incurring the minimum expected cost with the expectation taken over all possible underlying graphs. We have formulated it into a combinatorial optimization problem and first investigated its computational complexity. Specifically, through proving the NP-hardness of two closely related problems, we have shown that this problem cannot be solved in polynomial time unless P=NP. Then, we have applied the Markov Decision Process framework to give an exact dynamic programming algorithm with exponential time complexity. Moreover, we have proposed two efficient approximation schemes: a simple greedy approach with linear approximation ratio and a second Adaptive Submodular algorithm with logarithmic approximation ratio for most uncertain graphs. Finally, we have justified the effectiveness and superiority of our proposed algorithms through theoretical analysis and extensive simulations on real network datasets.

There remains a lot of future directions that can be explored. For example, it is desirable to design an algorithm with better approximation ratio and scalability, so that we can solve the connectivity determination problem more efficiently and more aptly apply it to large scale networks. Another interesting work is to derive the theoretical bound of the adaptivity gap of the Greedy algorithm and the approximation ratio of the Intersection Sort Algorithm. Finally, it is also worthwhile to investigate the approximation hardness of the Connectivity Determination Problem.

ACKNOWLEDGEMENTS

This work was supported by NSF China (No. 61532012, 61325012, 61521062, 61602303 and 91438115).

REFERENCES

[1] M. O. Ball, "Computational Complexity of Network Reliability Analysis: An Overview", in *IEEE Trans. on Reliability*, Vol. 35, No. 3, pp. 230-239, 1986.
[2] D. Kempe, J. Kleinberg and E. Tardos, "Maximizing the Spread of Influence through A Social Network", in *Proc. ACM SIGKDD*, 2003.
[3] L. Roditty and U. Zwick, "A Fully Dynamic Reachability Algorithm for Directed Graphs with an Almost Linear Update Time", in *SIAM J. Comput.*, Vol. 45, No. 3, pp. 712-733, 2016.
[4] A. R. Khakpour and A. X. Liu, "Quantifying and Querying Network Reachability", in *Proc. IEEE ICDCS*, June, 2010.
[5] A. D. Zhu, W. Lin, S. Wang and X. Xiao, "Reachability Queries on Large Dynamic Graphs: A Total Order Approach", in *Proc. ACM SIGMOD*, June, 2014.
[6] A. Sadilek, H. Kautz and J. P. Bigham, "Modeling The Interplay of People's Location, Interactions, and Social Ties", in *Proc. ACM IJCAI*, Aug., 2013.
[7] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications", in *Proc. ACM SIGCOMM*, Vol. 41, No. 4, pp. 350-361, 2011.
[8] S. Zhao and X. Wang, "Node Density and Delay in Large-scale Wireless Networks with Unreliable Links", in *IEEE/ACM Trans. on Networking*, Vol. 22, No. 4, pp. 1150-1163, 2014.
[9] J. Zhao, "Probabilistic Key Predistribution in Mobile Networks Resilient to Node-Capture Attacks", to appear *in IEEE Transactions on Information Theory*, 2017.
[10] S. Ji, R. Beyah and Z. Cai, "Snapshot and Continuous Data Collection in Probabilistic Wireless Sensor Networks", in *IEEE Trans. on Mobile Computing*, Vol. 13, No. 3, pp. 626-637, 2014.
[11] P. Parchas, F. Gullo, D. Papadias and F. Bonchi, "Uncertain Graph Processing through Representative Instances", in *ACM Trans. on Database Systems (TODS)*, Vol. 40, No. 3, 2015.
[12] D. L. Nowell and J. Kleinberg, "The Link Prediction Problem for Social Networks", in *J. of the American Society for Information Science and Technology*, Vol. 58, No. 7, pp. 1019-1031, 2007.
[13] R. Jin, L. Liu, B. Ding and H. Wang, "Distance-constraint Reachability Computation in Uncertain Graphs", in *Proc. the VLDB Endowment*, Vol. 4, No. 9, pp. 551-562, 2011.
[14] R. Jin, L. Liu and C. C. Aggarwal, "Discovering Highly Reliable Subgraphs in Uncertain Graphs", in *Proc. ACM SIGKDD*, Aug., 2011.
[15] M. Johnston, H. Lee and E. Modiano, "A Robust Optimization Approach to Backup Network Design with Random Failures", in *IEEE/ACM Trans. on Networking*, Vol. 23, No. 4, pp. 1216-1228, 2015.
[16] P. Parchas, F. Gullo, D. Papadias, F. Bonchi, "The Pursuit of a Good Possible World: Extracting Representative Instances of Uncertain Graphs", in *Proc. ACM SIGMOD*, June 2014.
[17] P. Erdos, A. Renyi, "On Random Graphs. I", in *Publicationes Mathematicae*, Vol. 6, pp. 290-297, 1959.
[18] F. Chung and L. Lu, "Connected components in random graphs with given expected degree sequences", in *Annals of combinatorics*, Vol. 6, No. 2, pp. 125-145, 2002.
[19] T. Unluyurt, "Sequential Testing of Complex Systems: A Review", in *Discrete Applied Mathematics*, Vol. 142, No. 1, pp. 189-205, 2004.
[20] H. Kaplan, E. Kushilevitz and Y. Mansour, "Learning with Attribute Costs", in *Proc. ACM STOC*, May, 2005.
[21] S. R. Allen, L. Hellerstein, D. Kletenik and T. Unluyurt, "Evaluation of DNF formulas", arXiv preprint arXiv:1310.3673, 2013.
[22] A. Deshpande, L. Hellerstein and D. Kletenik, "Approximation Algorithms for Stochastic Boolean Function Evaluation and Stochastic Submodular Set Cover", in *Proc. ACM SODA*, Jan., 2014.
[23] D. Golovin and A. Krause, "Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization", in *J. of Artificial Intelligence Research*, Vol. 42, No. 1, pp. 427-486, 2011.
[24] H. Kowshik, "Information Aggregation in Sensor Networks", PhD Thesis in University of Illinois at Urbana-Champaign, 2011.
[25] L. Fu, X. Wang and P. R. Kumar, "Optimal Determination of Source-destination Connectivity in Random Graphs", in *Proc. ACM MobiHoc*, Aug., 2014.
[26] L. Fu, X. Wang, P. R. Kumar, "Are we connected? Optimal Determination of Source-destination Connectivity in Random Networks," in *IEEE/ACM Trans. on Networking*, 2016.
[27] C. H. Papadimitriou and M. Yannakakis, "Shortest Paths without a Map", in *Theoretical Computer Science*, Vol. 84, No. 1, pp. 127-150, 1991.
[28] A. Y. Teymorian et al., "3D underwater sensor network localization," in *IEEE Trans. Mobile Comput.*, Vol. 8, No. 12, pp. 1610-1621, 2009.
[29] L. Xiao, Y. Liu, and L. M. Ni, "Improving unstructured peer-to-peer systems by adaptive connection establishment," in *IEEE Trans. Comput.*, vol. 54, no. 9, pp. 1091-1103, 2005.
[30] Charikar, Moses, et al. "Query Strategies for Priced Information", in *Proc. of ACM STOC*, 2000.
[31] J. Tang, T. Lou, J. Kleingerg and S. Wu, "Transfer Link Prediction across Heterogeneous Social Networks", in *ACM Trans. on Embedded Computing Systems*, Vol. 9, No. 4, Article 39, 2010.
[32] F. Buccafurri, G. Lax, A. Nocera and D. Ursino." Discovering links among social networks", in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, 2012.
[33] M. E. J. Newman, "The Structure of Scientific Collaboration Networks", in *Proc. of the National Academy of Sciences*, Vol. 98, No. 2, pp. 404-409, 2001.
[34] M. L. Puterman, "Markov Decision Processes: Discrete Stochastic Dynamic Programming", in *John Wiley & Sons*, 2014.
[35] D. Karger, "A Randomized Fully Polynomial Time Approximation Scheme for the All-terminal Network Reliability Problem", in *SIAM Rev.*, Vol. 43, No. 3, pp. 499-522, 2001.
[36] R. Karp, M. Luby and N. Madras, "Monte-Carlo Approximation Algorithms for Enumeration Problems", in *J. of Algorithms*, Vol. 10, No. 3, pp. 429-448, 1989.
[37] V. Vazirani and M. Yannakakis, "Suboptimal Cuts: Their Enumeration, Wight and Number", in *Automata, Languages and Programming*, Vol. 623, pp. 366-377, 1992.
[38] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design" arXiv preprint cs/0209028 (2002).
[39] J. McAuley and J. Leskovec. "Learning to Discover Social Circles in Ego Networks", in *NIPS*, 2012.
[40] Microsoft Academic Graph, https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/.
[41] X. Fu, Z. Xu, Q. Peng, L. Fu and X. Wang, "Complexity vs. Optimality: Unraveling Source-Destination Connection in Uncertain Graphs", to appear in *Proc. of IEEE INFOCOM*, 2017.

APPENDIX A
PROOF OF THEOREM 2

The proof is done by reduction from the set cover problem, which is a classic NP-complete problem. A set cover problem instance consists of a universe $\mathcal{U}$, a collection $\mathcal{S}$ of subsets of $\mathcal{U}$ and an integer k, the question is whether there exists a subfamily $\mathcal{C} \subseteq \mathcal{S}$ such that $\bigcup_{C \in \mathcal{C}} C = \mathcal{U}$ and $|\mathcal{C}| \leq k$.

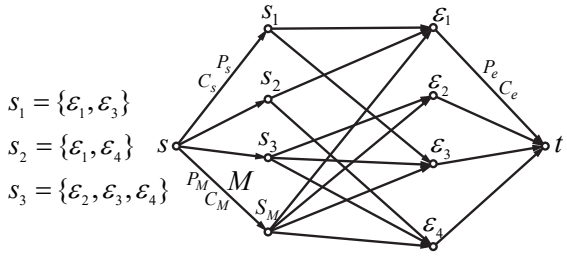


Fig. 7. The uncertain graph constructed for the set cover instance.

A. The Reduction Process

Given a set cover problem instance, we construct an instance of our Connectivity Determination problem, specifically the uncertain graph $\mathcal{G}(V, E, p, c)$ as follows. For each subset $S \in \mathcal{S}$, we create a node for it and call the nodes created for all $S \in \mathcal{S}$ as set nodes. For each element $u \in \mathcal{U}$, we also create a node for it and refer to the nodes created for all $u \in \mathcal{U}$ as element nodes. Then, we add two new nodes s and t as source

and destination, respectively. After that, we add an edge from each set node to all the nodes corresponding to the elements contained in the set. These edges have existence probability 1, which means that they do not need to be tested so their costs are irrelevant. Next, We add an edge from s to each set node and denote these as set edges. Similarly we add an edge from each element node to t and denote these as element edges. Finally, we create a special set node s_M and add a special edge M from s to s_M, which is also referred to as a set edge. We then add an edge of existence probability 1 from s_M to each element node. We finish the construction of $\mathcal{G}$ by assigning proper probabilities and costs to edges. Each set edge except M is assigned with the same probability P_s and cost C_s; Each element edge is assigned with the same probability P_e and cost C_e; The existence probability and cost of edge M are denoted as P_M and C_M. Set $m = |\mathcal{S}|$ and $n = |\mathcal{U}|$. The values of P_s, P_e, P_M, C_s, C_e and C_M are given as follows:

$$
\begin{aligned}
P_s =& 1 - \frac{1}{15mk},\\
P_e =& \frac{1}{2n}\min\{\frac{(1-P_s)^k}{2}, \frac{\left({P_s}^{2k+1}-1\right)(m+k+1)+\frac{1}{2}}{m+k+1}\},\\
P_M =& \frac{1}{2},\\
C_s =& 1,\\
C_e =& \frac{m+C_M}{\frac{1}{2}(1-P_s)^m+(1-P_e)^n-1},\\
C_M =& \frac{1}{2}\left(\frac{1}{(1-P_e)^n P_s^k}-P_s^{k+1}\right)m+\frac{1}{2}\left(1-\frac{1}{2{P_s}^k}\right)k\\
&+\frac{1}{2}\left((1-P_e)^n-\frac{1}{2}\right){P_s}^{k+1}(k+1).
\end{aligned}
$$

Note that all the probabilities and costs are rational numbers and they can be represented in size polynomial to $m+n$. Hence, the reduction process is polynomial to the size of the instance. An example of the reduction process is demonstrated in Figure 7.

B. Justification of the Reduction

In this section, we show the validity of the reduction, i.e. we prove that the optimal edge to test initially is edge M if and only if there does not exist a cover of size less than k in the original set cover instance, thus implying the NP-hardness of the problem of deciding the first edge to test in our problem. The idea of the proof is as follows. We begin with defining certain sequences of tests as trials. Then, based on trials, we classify all the testing strategies into three categories. After that, we show that the optimal strategy must come from one category. Finally, we demonstrate that the optimal strategy in the aforementioned category start with testing M if and only if there does not exist a cover of size less than k in the set cover instance.

First, we define a basic process of the testing strategy for $\mathcal{G}$. We define a process of testing with the following form as a "trial": a trial begins with testing some edge (a set edge or element edge) first. If the edge does not exist, then the trial ends; If the edge exists, then in the trial we test all the edges that lie on a same path with the edge. If one of these edges exists, then the entire determine process ends with verifying the s-t connectivity in the underlying graph of $\mathcal{G}$. And the trial also ends if none of these edges exists. Note that for a set edge, the edges that share some paths with it can only be element edges and vice versa.

We now present the lemma, which serves as the basis of classifying strategies with respeect to trials.

Lemma 3. *The optimal testing strategy must (only) consist of trials.*

Proof: The reason is as follows. If we intend to find out the non-existence of edges in an s-t cut to show s-t disconnectivity, after verifying the existence of the first tested edge, we need to test all the edges that lie on a same path with the first tested edge to show the non-existence of edges in an s-t cut. And if we intend to find a path to demonstrate s-t connectivity, we have

$$
\begin{aligned}
P_e > P_M P_e, \quad C_e < C_M + C_e,\\
P_e > P_s P_e, \quad C_e < C_s + C_e.
\end{aligned}
$$

This means that conditioned on the existence of the first edge, if we begin a trial with a set edge, the probabilities of the existence are higher and the total testing costs are lower for the paths that the first tested edge lie on than any other path. We also have

$$
\begin{aligned}
P_M > P_M P_e, \quad C_M < C_M + C_e,\\
P_M > P_s P_e, \quad C_M < C_s + C_e,\\
P_s > P_M P_e, \quad C_s < C_M + C_e,\\
P_s > P_s P_e, \quad C_s < C_s + C_e.
\end{aligned}
$$

Similarly, this means that, conditioned on the existence of the first tested edge, if we begin a trial with an element edge, the probabilities of the existence are higher and the total testing costs are lower for the paths that the first tested edge lies on than any other path. After testing all the paths that the first edge lies on, the resulting uncertain subgraph still bears the same structure as $\mathcal{G}$ and the same argument still holds. Therefore, the optimal strategy is to continue to do such trials until there is no set edge or element edge left untested, unless the determining process comes to an end during a trial. ■

Based on the above lemma, we classify the strategies that observe the above necessary condition for being optimal into the following three sets

1) The strategies that begin all trials with testing a set edge. We denote C_1 as the minimum expected cost of the strategies in this set.
2) The strategies that begin all trials testing an element edge. We denote C_2 as the minimum expected cost of the strategies in this set.
3) The strategies that begin some trials starting with testing a set edge and others starting with testing an element edge. We denote C_3 as the minimum expected cost of the strategies in this set.

Following the classification, we establish the superiority of the first set over the other two sets in the following lemma.

Lemma 4. *The optimal testing strategy is in the first set.*

Proof: We first show that the optimal strategy in the first set is superior than that in the second set. The first set of strategies test at most all the set edges when none of them exists and at most all edges in other cases. And the second set of strategies have to test all element edges when they are all non-existing. Thus, we have the following inequalities:

$$\begin{aligned} C_1 <& (1-P_s)^m (1-P_M)(C_M + mC_s), \\ &+ (1-(1-P_s)^m (1-P_M))(C_M + mC_s + nC_e) \\ C_2 >& (1-P_e)^n (nC_e). \end{aligned}$$

It follows that

$$\begin{aligned} C_2 - C_1 >& nC_e\left((1-P_s)^m (1-P_M) + (1-P_e)^n - 1\right) \\ &- mC_s - C_M \end{aligned}$$

After plugging in the values of P_s, C_s, P_e, C_n, we have that $C_2 - C_1 > 0$, which means that the optimal strategy in the first has smaller expected cost than the optimal strategy in the second set. Note that if we just reduce m, n or remove the term $1 - P_M$ and C_M in the inequalities, we still have $C_1 < C_2$. Hence, for any intermediate subgraph of $\mathcal{G}$ during the testing process, we still have that the optimal strategy in the first set has smaller expected cost than one in the second set.

Now, we proceed to show that the optimal strategy in the first set also incurs smaller cost than the optimal strategy in the third set. The testing strategies in the third set involve trials that begin with testing element edges and in these trials, each element edge and its corresponding untested set edges form a subgraph of $\mathcal{G}$. Then, from the above reasoning, it follows that in any of these subgraphs, substituting the trial that begins with testing an element edge with trials that begins with testing a set edge would lead to a strategy with smaller expected cost. Consequently, each strategy in the third set can be mapped to a corresponding strategy in the first set with smaller expected cost. It follows that $C_1 < C_3$. Therefore, the optimal strategy for $\mathcal{G}$ consists of trials that all begin with testing a set edge. ∎

Now the question remains that which is the optimal strategy in the first set. Specifically, we need to determine which set edge the first trial begins with in the optimal strategy. Again, we partition the strategies in the first sets into two subcases:

1) The strategies that begin the first trial with testing M. We denote C_{set1} as the expected total cost spent on set edges by the best strategy in this set and $C_{element1}$ as the expected total cost spent on element edges by the best strategy in this set.
2) The strategies that begin the first trial with testing some set edge other than M. We define C_{set2} and $C_{element2}$ here in a similar manner as in the first case.

The comparison between two sets of strategies is as follows. Since the existence probabilities and costs of all the element edges are the same, if we restrict our attention to the tests of element edges, the two sets of strategies only differ in the order of testing. Hence they lead to same expected cost on element edges, i.e., $C_{element1} = C_{element2}$. We also note that for the second set of strategies, the size of the minimum set cover for the "truncated" instance diminishes after the first trial if the set edge exists but none of its corresponding element edges exists. Under this circumstance, the next trial still begins with a set edge other than M. For general cases, for strategies in the second set, if some trial begins with a set edge other than M and results in the existence of the set edge but non-existence of its corresponding element edges, the next trial will still begin a trial with a set edge except M. Suppose the minimum set cover in the original instance has size l. Then we have

$$\begin{aligned} &C_M + l(1-P_M)(1-P_e)^n C_s + C_{element1} < C_{set1} < \\ &C_M + (1-P_M)\left(lP_s^l + \left(1-P_s^l\right)m\right)C_s + C_{element1}, \end{aligned} \tag{2}$$

$$\begin{aligned} &l(1-P_e)^n P_s^l C_s + C_{element2} < C_{set2} < \\ &l(1-P_e)^n P_s^l C_s + \left(1-(1-P_e)^n P_s^l\right)(m + C_M)C_s \\ &+ C_{element2}. \end{aligned} \tag{3}$$

Inequality (2) holds because if the first trial fails (i.e., edge M does not exist in the underlying graph) and none of the element edge exists in the underlying graph, the optimal strategy in the first subcase has to test at least the l set edges corresponding to the sets in the minimum set cover, the cost of which, together with $C_{element1}$, corresponds to the lower bound of C_{set1}. Also if the first trial fails, when all the l edges corresponding to the minimum set cover exist, the optimal strategy in the first subcase needs to test just these set edges. And it at most need to test all the set edges if some set edge in the cover does not exist in the underlying graph. The total cost of these together with $C_{element1}$ makes the upper bound of C_{set1} in inequality (2). Inequality (3) holds because if all set edges corresponding to the sets in the minimum set cover exist and none of the element edge exists, the optimal strategy in the second subcase has to test exactly these l set edges, which, together with $C_{element2}$, constitutes the lower bound of C_{set2}. Otherwise, the optimal strategy in the second subcase needs to test at most all the set edges, the total cost of which, combined with the total cost of the previous part and $C_{element2}$, forms the upper bound in Inequality (3).

Due to the special values we assigned for the parameters, it follows that if $l \leq k$, $C_{set1} < C_{set2}$, and otherwise if $l > k$, then $C_{set1} > C_{set2}$. This means that if the original instance permits a set cover of size less than or equal to k, then the optimal strategy in the second set is better than the optimal strategy in the first set and vice versa.

Therefore, if the optimal strategy is to test edge M first, then there does not exist a set cover of size less than or equal to k. On the other hand, if it first tests some edge other than M, then the original instance has a set cover of size k or less. As the set cover problem is NP-Complete, we conclude that deciding the optimal first edge to test in our problem is NP-hard.

APPENDIX B
TIME COMPLEXITY OF THE APPROXIMATION ALGORITHMS

In this section, we present analysis on the time complexity of the two approximation algorithms proposed in Section VI.

Recall that the time complexity of an algorithm for Connectivity Determination problem is defined as the maximum time it takes to compute all the relevant actions associated with a determining process.

A. Preprocessing Procedures

In both algorithms, before deciding the action of the current state, we need to determine whether the state is a terminating state. As mentioned before, for a temporary state $\mathbf{s}$ of uncertain graph $\mathcal{G}(V, E, p, c)$, this can be implemented by querying the s-t connectivity in two deterministic graphs $G^1_{\mathbf{s}}(V, E_1)$ and $G^2_{\mathbf{s}}(V, E_2)$, with $E_1 = \{e \mid \mathbf{s}_e = 1\}$ and $E_2 = \{e \mid \mathbf{s}_e = 1 \text{ or } \mathbf{s}_e = *\}$. If $G^1_{\mathbf{s}}$ is s-t connected, then we conclude that s and t are connected in the underlying graph of $\mathcal{G}$; If $G^2_{\mathbf{s}}$ is s-t disconnected, then we conclude that s and t are disconnected in the underlying graph. For the two cases above, we have that $\mathbf{s}$ is a terminating state. However, if $G^1_{\mathbf{s}}$ is s-t disconnected and $G^2_{\mathbf{s}}$ is s-t connected, then $\mathbf{s}$ is a non-terminating state.

Furthermore, we may also apply another preprocessing procedure to filter out irrelevant edges and only keep the edges that lie on some path from s to t for the current state. This can be completed through reachability queries.

The above two procedures both take a time of $O(|V|+|E|)$. Therefore, the time complexity of the preprocessing procedure is $O(|V| + |E|)$.

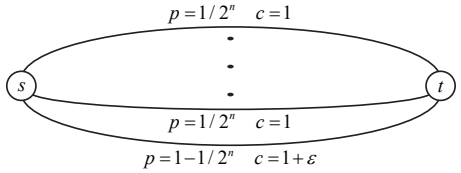


Fig. 8. A tight instance for the Greedy algorithm

B. The Greedy Algorithm

The non-adaptive version of the Greedy algorithm is implementable by sorting the edges according to their cost using the quick sort algorithm. Hence, the time complexity of the non-adaptive version is $O(|E| \log(|E|))$. On the other hand, in the adaptive version, we need to filter out irrelevant edges and sort the unfiltered edges for each current state. Therefore, the time complexity of the adaptive version of the Greedy algorithm is $O(|E|^2 \log(|E|))$.

C. The Adaptive Submodular Algorithm

The time complexity of the Adaptive Submodular algorithm relies highly on the method used for the selection rule. If we use path and cut enumeration schemes (e.g. [37]), the next edge to test specified by the algorithm can be determined in time $O(|E|(|\mathcal{P}| + |\mathcal{C}|))$. And the total time complexity of the algorithm would be $O(|E|^2(|\mathcal{P}|+|\mathcal{C}|))$. This is still polynomial when the number of s-t paths and s-t cuts are polynomial to the number of edges. When the number of paths and cuts in $\mathcal{G}$ is super-polynomial, to guarantee that the algorithm still has polynomial running time, we may use previously proposed approximate counting schemes [35], [36] to select edges with approximately maximum ratio between the expected marginal gain and cost, while preserving the approximation ratio of the whole algorithm. Note that if some approximate counting scheme is guaranteed to select edges with α-optimal marginal gain (as is the case of [35]), the approximation ratio of the whole algorithm is preserved as $O(\alpha \ln(|\mathcal{P}||\mathcal{C}|))$.

APPENDIX C
ADVERSE INSTANCES FOR THE ALGORITHMS

In this section, we present the tight or adverse instances for some algorithms mentioned in the paper. The analysis also justify the argument in Section VI-A that the performance guarantee of the Greedy Algorithm is far better than some other more sophisticated algorithm.

A. Tight Instance for the Greedy Algorithm

An instance that matches the $O(|E|)$ bound of the approximation ratio of the Greedy algorithm is illustrated in Figure 8. The instance is a parallel uncertain graph with two nodes (designated as source and destination) and n edges, of which one has existence probability $(1 - 1/2^n)$ and cost $(1 + \epsilon)$ and the other $(n - 1)$ edges have probability $1/2^n$ and cost 1.

In this case, the Greedy algorithm yields a strategy π that first tests the edges with cost 1 one by one. If such an edge exists, the strategy terminates with verifying the s-t connectivity. If none of the edges with cost 1 exists, then the strategy tests the edge with cost $1+\epsilon$. However, obviously the optimal strategy π^* is to first test the edge with cost $(1 + \epsilon)$ and then test the rest edges if the first one does not exist in the underlying graph. Hence, for sufficiently large n, we have

$$\begin{aligned} Cost(\pi) &= (1 - \frac{1}{2^n})^{n-1}(1 + \epsilon) + \sum_{i=0}^{n-2}(1 - \frac{1}{2^n})^i \\ &\geq \sum_{i=0}^{n-2}(1 - \frac{1}{2^n})^i \\ &\geq \sum_{i=0}^{n-2}(1 - \frac{i}{2^n}) \\ &= n - 1 - \frac{(n-1)(n-2)}{2^{n+1}}. \end{aligned}$$

On the other hand, we also have

$$\begin{aligned} Cost(\pi^*) &= (1 + \epsilon) + \frac{1}{2^n}\sum_{i=0}^{n-2}(1 - \frac{1}{2^n})^i \\ &\leq 1 + \epsilon + \frac{n}{2^n}. \end{aligned}$$

If follows that $\frac{Cost(\pi)}{Cost(\pi*)} \geq \frac{n-1-[(n-1)(n-2)]/2^{n+1}}{1+\epsilon+n/2^n} = O(n)$. Thus, there exists some instance where the gap between the Greedy algorithm and the optimal reaches up to n (the number of edges in the uncertain graph). Therefore, the bound in Theorem 4 is actually tight.

B. Adverse Instance for the Alternative Greedy Algorithm

Now we proceed to present an instance where the approximation ratio of the alternative greedy algorithm which yields strategies that test the edges following the decreasing order of there existence probability is *exponential* to the number of edges.

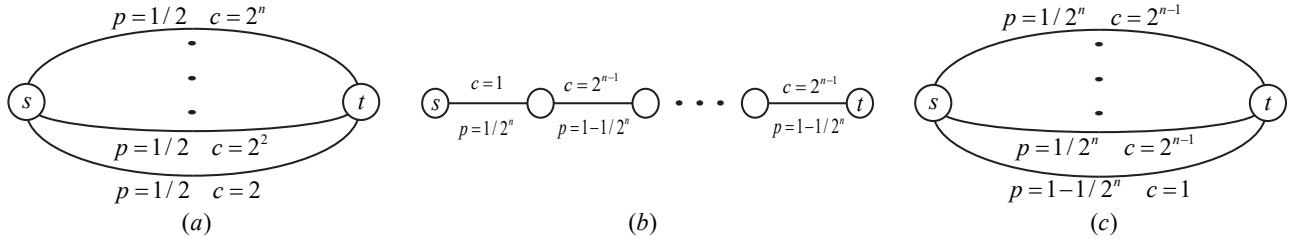


Fig. 9. Adverse instances for the other heuristics: (a) Alternative Greedy Algorithm, (b) Optimistic Sort, (c) Pessimistic Sort.

The instance is illustrated in Figure 9(a), which is a simple parallel uncertain graph that only consists of two nodes (designated as source and destination respectively) and n parallel edges (labeled as $e_1, e_2, \ldots, e_n$) between them. The cost of edge e_k is 2^k for all $1 \leq k \leq n$ and the existence probabilities of all edges are $1/2$. The strategy π^* that tests according to the increasing order of edge cost (which is the optimal in this case) has expected cost $2+2^2(1-1/2)+\ldots+2^n(1-1/2)^{n-1} = 2n$. On the other hand, the alternative greedy algorithm that tests according to the decreasing order of existence probability may generate a testing strategy π that follows the decreasing order of edge cost since the existence probability of all the edges are the same. It follows that the expected cost of π equals to $2^n + 2^{n-1}(1-1/2) + \ldots + 2(1-1/2)^{n-1} > 2^n$. Hence we have $\frac{Cost(\pi)}{Cost(\pi^*)} \geq 2^{n-1}/n$, which is significantly larger than $O(n)$.

C. Adverse Instances for Optimistic Sort and Pessimistic Sort

Although superficially, the Intersection Sort and Pessimistic Sort are more sophisticated than the Greedy algorithm, we now show by two hard instances that, on the contrary, their theoretical performance guarantees are much worse than that of the Greedy Algorithm.

The adverse instance for the Optimistic Sort is demonstrated in Figure 9(b), which illustrates a serial graph with n edges. In such graph, one edge has an existence probability of $1/2^n$ and a testing cost of 1, while the other $(n-1)$ edges have existence probability $1 - 1/2^n$ and cost 2^{n-1}. As $\frac{2^{n-1}}{1-1/2^n} < \frac{1}{1/2^n}$ for sufficiently large n, in this case the Optimistic Sort yields a strategy π that first tests the edges with cost 2^{n-1} and tests the edge with cost 1 in the end if needed. Therefore, its expected cost is given as

$$\begin{aligned} Cost(\pi) &= (1-\frac{1}{2^n})^{n-1} + \sum_{i=0}^{n-2} 2^{n-1}(1-\frac{1}{2^n})^i \\ &\geq \sum_{i=0}^{n-2} 2^{n-1}(1-\frac{1}{2^n})^i \\ &\geq 2^{n-1}\sum_{i=0}^{n-2}(1-\frac{i}{2^n}) \\ &= 2^{n-1}[n-1-\frac{(n-1)(n-2)}{2^{n+1}}]. \end{aligned}$$

On the other hand, the optimal testing strategy π^* firsts test the edge with cost 1. Its expected cost equals to

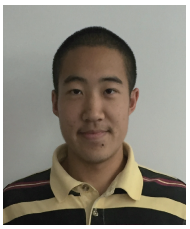
$$\begin{aligned} Cost(\pi^*) &= 1 + \frac{1}{2^n}\sum_{i=0}^{n-2} 2^{n-1}(1-\frac{1}{2^n})^i \\ &\leq 1 + \frac{n}{2}. \end{aligned}$$

Thus, we have $\frac{Cost(\pi)}{Cost(\pi^*)} \geq \frac{2^{n-1}[n-1-(n-1)(n-2)/2^{n+1}]}{1+n/2} \geq 2^{n-1}$ for sufficiently large n. Hence, the approximation ratio of the Optimistic Sort is at least *exponential* to the number of edges. Similarly, for the parallel uncertain graph shown in Figure 9(c), the approximation ratio of the Pessimistic Sort is also no less than an exponential value of the number of edges. Therefore, theoretically, the Greedy algorithm enjoys significantly better performance guarantee than the Pessimistic Sort and the Optimistic Sort.

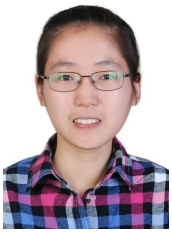
D. Adverse Instance for the Intersection Sort

Finally, we show that the approximation guarantee of the Intersection Sort heuristic is worse than our proposed Adaptive Submodular algorithm. The constructed instance is the same as the one in Figure 8. Here, the strategy generated by the Intersection Sort first tests the edges with cost 1 one by one. If an edge exists, the strategy terminates by verifying the s-t connectivity. If none of the edges with cost 1 exists, then the strategy tests the edge with cost $1+\epsilon$. From the analysis in Section C-A, we have that the expected cost of the strategy is approximately n times the optimal one. However, there are only n s-t paths and one s-t cut in the graph. Therefore, the approximation ratio of the Intersection Sort is worse than $O(\ln(|\mathcal{P}||\mathcal{C}|))$, i.e., the performance guarantee of our Adaptive Submodular algorithm is better than the Intersection Sort. However, as the Intersection Sort algorithm is more intuitive and simpler than Adaptive Submodular algorithm, it would be interesting to investigate its approximation ratio. This presents an interesting future direction of our work.

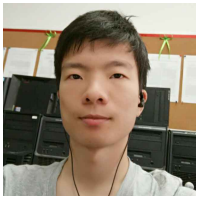
Luoyi Fu received her B. E. degree in Electronic Engineering from Shanghai Jiao Tong University, China, in 2009 and Ph.D. degree in Computer Science and Engineering in the same university in 2015. She is currently an Assistant Professor in Department of Computer Science and Engineering in Shanghai Jiao Tong University. Her research of interests are in the area of social networking and big data, scaling laws analysis in wireless networks, connectivity analysis and random graphs.



Xinzhe Fu received his B. E. degree in Department of Computer Science and Engineering at Shanghai Jiao Tong University, China, 2017. During his undergraduate study, he was working as an research intern supervised by Prof. Xinbing Wang. His research interests include combinatorial optimizationasymptotic analysis and privacy protection in social networks. He will pursue Ph. D. degree in the Massachusetts Institute of Technology (MIT), Massachusetts, USA, 2017.



Zhiying Xu is an undergraduate student in Department of Electronic Engineering at Shanghai Jiao Tong University, China. She is currently working as an research intern supervised by Prof. Xinbing Wang. Her research interests include network topology and asymptotic analysis in social networks.



Qianyang Peng received his B. E. degree in Computer Science and Engineering from Shanghai Jiao Tong University, China, in 2017, and will pursue master degree in the University of Illinois at UrbanaCChampaign, Illinois (UIUC), USA in 2017.

Xinbing Wang received the B.S. degree (with hons.) from the Department of Automation, Shanghai Jiaotong University, Shanghai, China, in 1998, and the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2001. He received the Ph.D. degree, major in the Department of electrical and Computer Engineering, minor in the Department of Mathematics, North Carolina State University, Raleigh, in 2006. Currently, he is a professor in the Department of Electronic Engineering, Shanghai Jiaotong University, Shanghai, China. Dr. Wang has been an associate editor for IEEE/ACM Transactions on Networking and IEEE Transactions on Mobile Computing, and the member of the Technical Program Committees of several conferences including ACM MobiCom 2012, ACM MobiHoc 2012-2014, IEEE INFOCOM 2009-2017.

Songwu Lu is currently a professor of Computer Science Department at University of California, Los Angeles (UCLA), USA. He received his M.S. and Ph.D. degrees from Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign, in 1996 and 1999, respectively. His research interests include wireless networking, wireless network security, mobile systems and computer networks. He received NSF CAREER award in 2001 and has been serving on the TPC or organizing committees of premier networking conferences including ACM MOBICOM, MOBIHOC, MOBISYS, IEEE INFOCOM, ICNP, IPSN, etc. He is leading the WiNG research group at UCLA.