On determination of source-destination connectivity in independent random graphs

Abstract—**This paper investigates the problem of computing adaptive testing strategy to optimally determine the source-destination connectivity in random graphs. We consider a set of random graphs 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. Assuming that during each determining process, we are associated with an underlying graph. Testing each edge e incurs some cost c_e and the outcome is revealed according to the underlying graph. We aim to find an optimal strategy with the minimum expected cost with the expectation taken over all possible underlying graphs that forms a product distribution.**

We first characterize the computational complexity of this problem, showing that this problem is unlikely to be solved in polynomial time unless P=NP by two hardness results. Then, we apply the Markov Decision Process framework to give an exact algorithm in a dynamic programming approach with exponential time complexity. After that, we propose two approximation schemes, one simple greedy approach with linear approximation ratio and another algorithm based on adaptive submodularity framework that enjoys logarithmic guarantee when the number of $s-t$ cuts and paths are polynomial to the number of edges.Finally, we use extensive simulation to justify the effectiveness of our algorithms and demonstrate that the adaptive submodular algorithm has better approximation ratio than the greedy approach and the expected cost of the adaptive strategy it computes is only several percent larger than the optimal for real social network data trace.

I. INTRODUCTION

Many practical problems involve a common notion of "testing under uncertainty", that is, priorly we only have probabilistic knowledge about some object, and its true profile can be gradually revealed by tests. Each test will incur some cost and we aim to find a testing strategy with the minimum expected cost. This kind of sequential testing problem has been studied in many fields such as operations research, artificial intelligence and networking areas. Often most generally, the parts of the objects subjected to tests are described as variables and the true profile of the objects is given by a function on those variables. When the variables only take on boolean values and the function is a boolean function, then the problem naturally corresponds to the Stochastic Boolean Function Evaluation (SBFE) problem [13]. Many existing work consider simple boolean function as CNF or DNF formulas [14], parallel or series formulas and parallel-series formulas.

However, there are few works considering sequential testing problem on graphs [20] despite the fact that graph has been extensively used as models in a wide range of areas. In this paper, we focus on the problem of adaptive testing the source-destination connectivity of random graphs. Here, in a random graph, priorly, each edge e exists with some probability p_e and testing the edge incurs cost c_e, we aim to find an adaptive testing strategy to determine the connectivity of two given nodes in the graph consuming the minimum expected cost.

This problem has significant practical applications. For examples, in scholar networks, based on citations, institutions, conferences we can create paper maps or scholar maps. All these maps can be represented as graphs, with nodes corresponding to papers (or scholars) and edges corresponding to their relationships. However, we cannot discover the true relationship between papers just by information such as citations and conferences. These crude information can only provide us with a priori probability of whether two papers are truly related. To discover the existence of genuine connection between papers, we often need expensive procedures such as natural language processing techniques, or even human resources, and deciding on which two papers do we apply these expensive procedures will play an important row in effectively unraveling the true relations between papers and scholars. Another typical scenario is in communication networks (like wireless network, local area networks), each link is unreliable to some degree. We can capture this unreliability as a probability of its normal functioning. Furthermore, testing whether a link is normal is costly. We naturally want to examining whether two nodes can communicate normally, as corresponding to whether there exist a path such that all the edges in the path exist.

In this paper, we investigate the problem of determination of source-destination connectivity in a comprehensive way. First, through proving the complexity of two relevant problems, we derive the NP-hardness of the problem. This means that there is unlikely to be any polynomial time algorithm for the problem. Next, we apply the Markov Decision Process framework and design a dynamic programming algorithm that compute the optimal strategy in exponential time. Although the high time complexity prohibit the practical use of the algorithm, the exact algorithm can provide much insight to the problem. Then, to practically solve the problem, we turn to the design of good approximation scheme. We first show that a simple greedy approach can have a linear approximation guarantee. Then, to further improve the approximation ratio, we harness the power of adaptive submodularity and propose a more sophisticated algorithm whose approximation ratio is logarithmic to the number of $s-t$ cuts and $s-t$ paths in the graph. Since for most cases, the number of $s-t$ cuts and $s-t$ paths are polynomial to the number of edges, the algorithm has logarithmic performance guarantee for most graphs.

Our contributions can be summarized as follows:

- We are the first to formally define the problem of determining the source-destination connectivity in random graphs and prove its computational hardness, which provides useful insights to the problem.
- We use the Markov Decision Process framework to give an optimal dynamic programming algorithm. This algorithm can present some useful properties on the optimal edge to test in the problem.
- We design a simple greedy algorithm and a more sophisticated polynomial approximation scheme based on adaptive submodularity framework. We analyze the approximation ratio of the two algorithms.
- We evaluate our proposed algorithms on real network data. From the simulation results, we demonstrate that our algorithms have excellent approximation guarantee and is efficient to implement.

The rest of the paper is organized as follows. Section II formally introduce the definitions of notations relating to our problem. In Section III, we investigate the computational complexity of the problem. We present our exact dynamic programming problem based on Markov Decision Process framework in Section IV. In Section V, we present the two approximation algorithms and we evaluate our algorithms on real life data in Section VI. We conclude the paper in Section VII.

II. Problem Formulation

We consider that given an independent directed random graph $\mathcal{G}(V, E)$, a probability vector $\mathbf{p} = (p_1, p_2, ..., p_{|E|})$ indicating the prior existence probability of each edge, a cost vector $\mathbf{c} = (c_1, c_2, ..., c_{|E|})$ denoting the cost incurred by testing each edge and two nodes in V designated as s, t. We aim at finding an adaptive testing strategy with the minimum expected cost. The random graph $\mathcal{G}$ can be considered as a distribution on a series of underlying graphs $G(V_G, E_G)$ where $V_G = V$ and $E_G \subseteq E$. Since each edge exists independently with some probability, the distribution of underlying graphs is a product distribution. And when we perform the testing strategy on a certain underlying graph, the outcomes of testings are given by the existence of the tested edges in the underlying graph.

To formally define the adaptive testing strategy, we first introduce the notion of temporary states of a random graph. For a random graph $\mathcal{G}(V, E)$, we define a set of temporary states S associated with it as $S = \{0, 1, *\}^{|E|}$. A temporary state can be interpreted as our current knowledge of $\mathcal{G}$ during the testing process. Each temporary state is an $|E|$-dimensional vector with element "0", "1" and "$*$", where "0" means that the corresponding edge has been tested and found not existing, "1" means that the corresponding edge has been tested and found existing and "$*$" means that the corresponding edge has not been tested yet. Note that the set of temporary states S is not part of the input of our problem, and still the input size of our problem is polynomial to the number of edges $|E|$ in $\mathcal{G}$.

Based on the notion of temporary states, we formally define an adaptive testing strategy T as a mapping from S to $E \cup \{\perp\}$, i.e., a strategy specify which edge to test based on the test results so far or terminates (the terminating action is denoted as $\perp$). We call a strategy valid if it does not test any edge that has been tested and terminates as soon as it verifies the existence of all the edges in an $s-t$ path in $\mathcal{G}$ or the non-existence of all the edges in an $s-t$ cut of $\mathcal{G}$. In the sequel, we only consider valid strategies. From the above, we can see that when performed on different underlying graphs, the cost incurred by an adaptive testing strategy can be different. Hence, naturally we aim to find the adaptive testing strategy with the minimum expected cost with the expectation taken over all the possible underlying graphs. Furthermore, we actually do not need to explicitly compute the whole mapping of a strategy, what we only need to do is to determine the optimal edge to test next based on the past outcomes, that is, starting from the "all-$*$" state, we need to sequentially deciding the next edge to test and the outcomes of testings and the transitions of temporary states depend on the underlying graph.

III. Computational Complexity

In this section, we investigate the computational complexity of the problem. The computational complexity is captured by the hardness of two closely related problem. We state our results as the following two theorems.

Theorem III.1. *Computing the expected cost of the optimal policy is #P-hard.*

Proof. Inspired by [3], we prove the theorem by reduction from the $s-t$ reliability problem [4]: Give an 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 $\frac{1}{2}$.

The reduction work as follows: For a graph $G(V, E)$, we transform it to a random graph $\mathcal{G}(V, E')$ by adding an edge M between s and t with existence probability $\frac{1}{2}$ and the rest of G' is just the same as G. We set the cost of M as $|E|2^{|E|+1}$ (with a little bit abuse of notation, we also refer to the cost of M as M) and the cost of testing other edges as 1. So formally, the constructed instance of our problem consists of a random graph $\mathcal{G}(V, E')$ where $E' = E \cup (s, t)$, a probability vector $\mathbf{p} = (\frac{1}{2}, ..., \frac{1}{2})$ with $|E| + 1$ elements, a cost vector $\mathbf{c} = (1, ..., 1, |E|2^{|E|+1})$ with $|E| + 1$ elements and two nodes designated as s, t as in the network reliability instance. Let p be the $s-t$ reliability in G and l be the expected cost incurred by the optimal testing strategy on $\mathcal{G}$, we will show that if we know l, then we can efficiently compute p.

First, from the definitions, we have $p = \frac{k}{2^{|E|}}$ for some integer k and l must obey the following constraints:

$$l \geq (1-p)M \tag{1}$$

$$l \leq p|E| + (1-p)M. \tag{2}$$

Here, inequality (1) follows from the fact that we have to test M whenever we find out that s and t is not connected in the underlying graph of $\mathcal{G}$. Inequality (2) holds since a simple strategy that first test all the edges corresponding to E in $\mathcal{G}(V, E')$ and test M if s and t are not connected in the subgraph corresponding to G in $\mathcal{G}$. Combining the two inequalities, we have $2^{|E|}\frac{M-l}{M} \leq k \leq 2^{|E|}\frac{M-l}{M-|E|}$. Consequently, $k = \lfloor 2^{|E|}\frac{M-l}{M-|E|} \rfloor$. □

The above theorem indicates the complexity of computing the expected cost of an adaptive testing strategy. This implicates that our problem may not be an NP optimization problem. Indeed, up till now, we have found no way to succinctly describe the whole adaptive strategy, and the above theorem demonstrate that there is no way to compute the value of a strategy in polynomial time. However, the above theorem only is not a strong enough indicator to the inherent complexity of the problem, because we may not need to compute the whole testing strategy. Actually, we only need to based on the current knowledge and choose the next edge to test. Therefore, we present another result relating to this issue and state it as the following theorem.

Theorem III.2. *Deciding the optimal action of the initial state in our problem is NP-hard.*

Proof. Due to the space limit, we only present a proof sketch here and defer the detail in the appendix. The proof is done by reduction from set cover problem. For a set cover instance, we construct a graph as illustrated in **figure**. By carefully assigning the cost and probability of each edge, we prove that the optimal first edge to test is M if and only if there exists a set cover of size smaller than k in the original set cover instance. □

IV. An Exact Algorithm

We apply the Markov Decision Process (MDP) framework to give an exact algorithm for our problem. We adopt the notations in [1] and first describe how the elements in our problem can be fit into a Markov Decision Process. Specifically, in the following we will show that the counterpart of a MDP's state set, action set, decision epochs, transition probability, reward, decision policy and optimality criterion in our problem.

The state set naturally corresponds to the set of temporary states S in our problem and for each state $s \in S$, we define its associated action set A_s as the subset of edges that have not been tested in s and for terminating states, their action set also contains the action $\perp$. So the action set $A = \bigcup_{s\in S} A_s = E \cup \{\perp\}$. The decision epochs is every time we decide the next edge to test based on previous testing outcomes and to determine the $s-t$ connectivity we need to test at most $|E|$ edges so our MDP is of finite horizon. We may also partition the state S into $|E|$ disjoint subsets based on the number of edges having been tested in the states as $S = S_0 \cup S_1 \cup ... \cup S_{|E|}$ and in decision epoch i, only states in S_i are valid. Further, under state s, the transition probability of action e (testing edge e) is given by the existence probability of edge e. Denote by $s \cdot e$ the temporary state of setting the element corresponding to e in s as 1 and by $s\backslash e$ the temporary state of setting the element corresponding to e in s as 0. Formally, the transition probability function is given by:

$$p(s'|s,e) = \begin{cases} p_e & \text{if } s' = s \cdot e, \\ 1 - p_e & \text{if } s' = s\backslash e, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$p(s'|s,\perp) = \begin{cases} 1 & \text{if } s' = s \\ 0 & \text{otherwise.} \end{cases}$$

Then it follows that the reward function is $c(s,e) = -c_e$ and $c(s,\perp) = 0$, and the decision policy is equivalent to an adaptive testing strategy. Note that the reward function is negative, corresponding cost and the transition probability and reward function are independent with regard to decision epoch or previous state, which demonstrate the Markov property of our problem. Finally, since we aim to find the adaptive testing strategy with minimum expected cost, the optimality criterion is expected total reward criterion.

Now we illustrate how to apply the framework to solve our problem. First, we define the cost function u of temporary states as the expected cost incurred by the optimal adaptive testing strategy starting at that state and for a terminating state s, $u(s) = 0$. This cost function can be interpreted as the expected cost to "go" from a state to terminating states. Then, based on results in [1] and particularly the Bellman principle, we have the principle of optimality in our problem.

Theorem IV.1. *For any state $s \in S$, the cost function satisfies* $u(s) = \max_{a\in A_s}\{c(s,a) + \sum_{s'\in S} p(s' \mid s,e)u(s')\}$. *Particularly, if s is a non-terminating state, then* $u(s) = \max_{e\in A_s}\{-c_e + p_e u(s\cdot e) + (1-p_e)u(s\backslash e)\}$ *and for any terminating state, its cost function is 0.*

The above theorem is rather straightforward so we omit the proof here. Strict and detailed proof can be constructed analogously as in [1].

Based on the theorem, we design an algorithm to compute the optimal testing strategy π following the dynamic programming paradigm as shown in the following figure.

```
Input: Random graph $\mathcal{G}(V,E)$, probability vector $\mathbf{p}$,
       cost vector $\mathbf{c}$, node $s$ and $t$
Output: The optimal testing strategy $\pi$
Initialize: $u(s) = 0$, for all $s \in S_{|E|}$
for $i = |E|$ to 0 do
    for All $s$ in $S_i$ do
        if $s$ is a terminating state then
            $u(s) = 0$, $\pi(s) = \perp$.
        end
        else
            $e^* = \arg\max_{e\in A_s}\{-c_e + p_e u(s\cdot e) + (1 - p_e)u(s\backslash e)\}$,
            $u(s) = -c_{e^*} + p_{e^*}, u(s\cdot e^*) + (1 - s_{e^*})u(s\backslash e^*)$,
            $\pi(s) = e^*$.
        end
    end
end
```
Algorithm 1: The Exact Algorithm

Mind the min and max, cost to go function of the optimal strategy

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

Theorem IV.2. *The exact algorithm yields an optimal adaptive testing strategy.*

Proof. Denote an optimal testing strategy as π^*, the the strategy given by the exact algorithm as π and the cost function incurred by the π^* as u^*. We prove that the cost function u incurred by π is no less than $u*$ on every state, which implies that π is an optimal strategy. The proof is done by standard backward induction.

First, for all $s \in S_{|E|}$, obviously $u(s) = u^*(s) = 0$.

Suppose for all states $s \in S_i, i \leq k$, $u(s) \geq u^*(s)$, then we prove that for all states $s \in S_{k-1}, u(s) \geq u^*(s)$. Indeed, by the selecting criterion of the algorithm, for a state $s \in S_{k-1}$ that is non-terminating,

$$\begin{aligned} u(s) =& \max_{e \in A_s}\{-c_e + p_e u(s \cdot e) + (1-p_e) u(s \backslash e)\} \\ \geq& c(s, \pi^*(s)) + p_{\pi^*(s)} u(s \cdot \pi^*(s)) + (1-p_{\pi^*(s)}) u(s \backslash \pi^*(s)) \\ \geq& c(s, \pi^*(s)) + p_{\pi^*(s)} u^*(s \cdot \pi^*(s)) + (1-p_{\pi^*(s)}) u^*(s \backslash \pi^*(s)) \\ =& u^*(s). \end{aligned}$$

And if s is a terminating state, then also $u(s) = u^*(s) = 0$. Hence, we prove that under every state s, following π is optimal, and particularly from the initial all$-*$ state, π incurs minimum expected cost. □

V. Approximation Algorithm

Due to the inherent computational complexity of the problem, instead of pursuing polynomial-time exact algorithms, we turn to designing efficient approximation algorithms with good approximation guarantee.

Our problem belongs to a broader class of stochastic boolean function evaluation setting [13]. For stochastic boolean function evaluation, we need to adaptively test a set of boolean variables at a certain cost and the prior distribution of variables is given as a product distribution. The goal is to evaluate the value of a given boolean function f while incurring minimum cost. Existing work on stochastic boolean function evaluation includes the approximation framework in [13], the evaluation scheme for DNF formula in [14] and approximation scheme for some special instances in [15]. In our problem, we can consider the edges as variables and that the target boolean function f is implicitly given as the connectivity of the two nodes in the graph. Since transforming this connectivity function into a DNF or CNF formula requires exponential time, the algorithms in [15], [14] do not apply to our problem.

Our algorithm adopts the framework for adaptive submodular function optimization [12]. In the next section, we will present the details and prove that the algorithm yields $O(\ln |E|)$ solution for graphs whose number of cuts and paths are polynomial to the number of edges $|E|$. But first, we present some basic knowledge about adaptive submodular function optimization as preliminaries.

Consider a set of elements E and a set function $f : 2^E \mapsto \mathbb{R}^+$. The function is called monotone if for any $A \subseteq B \subseteq E$, $f(A) \leq f(B)$. It is submodular, if for any $A \subseteq B \subseteq E$ and element $e \in E \backslash B$, it holds that $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. $f(A \cup \{e\}) - f(A)$ is called the marginal gain of element e with respect to set A. In many optimization settings, we need to pick a set of elements with minimum cost to satisfy a desired utility (e.g. set cover). When the utility function and the set function are monotone and submodular, it has been proven that a simple greedy algorithm that pick the element with the largest marginal gain regarding the current set can achieve a $O(\ln |E|)$ approximation ratio, which is also the best possible unless P=NP.

However, in many cases, we only have a prior distribution of the true states of the elements and the state of each element can only be determined when we select it. To tackle this [12], we formulate the set of true states as O and define $\phi : E \mapsto O$ as a realization of a problem instance. We then generalize the utility function to $f : 2^E \times O^E \mapsto \mathbb{R}^+$. For a set $A \subseteq E$ and a realization ϕ, $f(A, \phi)$ represents the utility of selecting subset A when the true realization is ϕ. Further, we define a partial realization $\psi \subseteq E \times O$, representing the item-observation pairs over a subset of E. In particular, we formally define the domain of ψ as $D(\psi) := \{e \in E | \exists o \in O : (e, o) \in \psi\}$. We also write $\psi(e) = o$, if $(e, o) \in \psi$, and call ψ consistent with realization ϕ (denoted by $\phi\ \psi$), if $\psi(e) = \phi(e)$, for all e in $D(\psi)$.

Then, we define the expected marginal gain of an element $e \in E$ under partial realization ψ as

$$\Delta(e \mid \psi) := \mathbb{E}[f(D(\psi) \cup \{e\}, \Phi) - f(D(\psi), \Phi) | \Phi\ \psi].$$

Now, we are ready to define the adaptive monotone and adaptive submodular properties analogously to their non-adaptive counterparts [12]:

- f is called adaptive monotone, if $\Delta(e|\psi) \leq 0$, for all $e \in E$, and all feasible ψ.
- f is called adaptive submodular, if $\Delta(e|\psi') \leq \Delta(e|\psi)$, for all $e \in E \backslash D(\psi')$, for all $\psi \subseteq \psi'$.

[12] shows that if f is adaptive monotone submodular, then the greedy policy that pick the element with the largest expected marginal gain under current realization yields a $O(\ln |E|)$ approximation in terms of the optimal policy with the least expected cost.

A. Our Algorithm

In this section, we introduce our approximation algorithm based on the previously presented adaptive submodular optimization framework.

First, from the result in [15], we show that a naive greedy algorithm that just test the edge with the cheapest cost has a linear approximation ratio.

Theorem V.1. *Let $\mathcal{G}(V, E)$ be a random graph and π be a policy that test the edges in G according to their costs (from small to large). Then, $cost(\pi) \leq |E| cost(\pi*)$, where*

$\pi*$ is the optimal policy for $\mathcal{G}$, i.e., π provides an $O(|E|)$-approximation.

Proof. The proof of the theorem is due to [15]. We put it here for completeness. Actually, we prove a stronger result, that $cost(\pi, G) \leq |E| cert(G)$ for any realization G of $\mathcal{G}$. Suppose after k tests, π have tested all the edges in the certificate of G. Obviously, $k \leq |E|$, and since π test the edges from low cost to high cost, we have $cost(\pi, G) \leq k \cdot cert(G) \leq |E| \cdot cert(G)$. □

Therefore, we consider the $O(|E|)$-approximation as baseline and aim to find algorithms with better performance guarantee. Next, we present our approximation algorithm that achieves $O(\ln |E|)$ approximation when the number of $s-t$ cuts and $s-t$ paths in $\mathcal{G}$ is polynomial to the number of edges $|E|$, which is a significant improvement from the previous $O(|E|)$ guarantee.

To begin with, we briefly introduce the main idea of the algorithm. Denote f as the function implicitly described by the $s-t$ connectivity in $\mathcal{G}$. Then, every $s-t$ path in $\mathcal{G}$ is a 1-certificate of f and every minimal $s-t$ cut is a 0-certificate of f.

Then, the determination of the connectivity of $\mathcal{G}$ can be interpreted as a finding a cover for the 1-certificates of f and 0-certificates of f. If an edge is tested to be exist, then it covers all the 0-certificates it lies in. If an edge is tested to be not exist, then it covers all the 1-certificates it lies in. When we have covered all the 1-certificates, then the function can be evaluated to 0 (i.e, we conclude that s and t are disconnected), and it holds similarly when finish covering all the 0-certificates. Hence, we can view the testing procedure as an adaptive covering process where the elements (edges) has two states (existing and non-existing). Denote the number of $s-t$ paths in $\mathcal{G}$ as Q_p and the number of minimal $s-t$ cuts in $\mathcal{G}$ as Q_c. Further, for a partial realization ψ, define $g_0(\psi)$ as the cuts covered by the tested edges in ψ and $g_1(\psi)$ as the paths covered by the tested edges in ψ. We construct the utility function g as proposed in [13]

$$g(\psi) = Q_pQ_c - (Q_p - g_1(\psi))(Q_c - g_0(\psi))$$

It is easy to verify that g is adaptive monotone and adaptive submodular, so we use the adaptive submodular optimization framework and adaptive greedy policy [12] to achieve an $O(\ln Q_pQ_c)$ approximation, which is in order $O(\ln |E|)$ when $\mathcal{G}$ has polynomial number of $s-t$ cuts and $s-t$ paths.

Specifically, under a partial realization ψ, we select the edge

$$e = \arg\max_{e \in E}\{\frac{p_e(g(\psi \cup \{e=1\}) - g(\psi) + (1-p_e)(g(\psi \cup \{e=0\} - g(\psi)))}{c_e}\}$$
$$= \arg\max_{e \in E}\{\frac{p_e(Q_p - g_1(\psi)(g_0(\psi \cup \{e=1\}) - g_0(\psi) + (1-p_e)(Q_c - g_0(\psi))(g_1(\psi \cup \{e=0\}) - g_1(\psi))}{c_e}\}$$

Note that the above greedy selection rule involves computing the number of $s-t$ cuts and $s-t$ paths an edge lies in, which are #P-complete in general [16]. To circumvent this, we can apply several proposed randomized fully polynomial time approximation scheme [17], [18] to get $(1+\epsilon)$ approximations in polynomial time, or use paths and cuts enumeration techniques [19] to compute the exact values in polynomial time when the number of cuts an paths is subexponential to the number of edges.

Now, we formally describe our approximation algorithm in pseudo-code. To ease the notations, we use $\Delta(e|\psi)$ to denote the greedy selection function under partial realization ψ as defined above.

```
Input: Random graph $\mathcal{G}(V,E)$, Probability vector $p$
       Cost function $C$, node $s$ and $t$
Output: An approximate sequential testing strategy
Initialize: Partial realization $\psi$, Set of tested edges $E_\pi$
as empty sets.
Repeat until $g(\psi) = Q_pQ_c$
$e = \arg\max_{e \in E \setminus E_\pi}\{\Delta(e|\psi)\}$.
Set $E_\pi$ as $E_\pi \cup \{e\}$, test $e$ and observe the outcome.
if $e = 1$ then
  | $\psi = \psi \cup (e,1)$
else
  | $\psi = \psi \cup (e,0)$
end
```

Algorithm 2: The Approximation Algorithm

B. Present A Case Where Our Approximation Algorithm Is Significantly Better Than Greedy

APPENDIX

Theorem A.1. *Deciding the optimal action of the initial state in our problem is NP-hard.*

Proof. The proof is done by reduction from the set cover problem, which is a classic NP-complete problem [5]

Given a set cover problem, we construct a random graph G as follow. It contains a s node, a t node, a set section and a node section. The nodes in the set section correspond to the sets in the set cover problem. Each edge between a node in set section and s, which is called set edge, has a probability of P_s and a cost of C_s. Similarly, the nodes in the node section correspond to the elements in the set cover problem. Each edge between a node in node section and t, which is called node edge, has a probability of P_n and a cost of C_n. If a set includes an element in the set cover problem, then there is an edge connecting the corresponding node in set section and the corresponding node in node section in G. These edges have a probability of 1. We also add a special set edge M connecting all the nodes in node section with probability of P_M and cost of C_M. To distinguish between these two kinds of set edges, we call them normal set edges and special set edge.

Suppose we have m nodes in set section and n nodes in node section. We wonder whether the original problem has a set cover of k. Then P_s, P_n, P_M, C_s, C_n, C_M take the following values:

$$P_s = \left(\frac{m}{m+1}\right)^{1/(2k+1)}$$

$$P_n = \frac{1}{2} - \frac{1}{2}max\left\{\left(1-\frac{1}{2}\left(1-P_s\right)^k\right)^{1/n}, \left(\frac{\left(1-{P_s}^k\right)m + {P_s}^k k}{\left(k+1\right){P_s}^{2k+1}}\right)^{1/n}\right\}$$

$$P_M = \frac{1}{2}$$

$$C_s = 1$$

$$C_n = \frac{2m+2C_s}{\frac{1}{2}(1-P_s)^m + (1-P_n)^n - 1}$$

$$C_M = \frac{1}{4}\left(\frac{{P_s}^{k+1}\left(k+1\right)\left(1-P_n\right)^n}{1-\left(1-P_s\right)^m} + \frac{\left(1-{P_s}^k\right)m + {P_s}^k k}{\left(k+1\right){P_s}^k}\right)$$

Now we are going to find out the optimal action to determine the connectivity between s and t. For a G with set edges and node edges, we choose an edge to test first. If the edge doesn't exists, we just delete that edge from G and denote the new graph G'. Since G' is of the same structure with G, the problem becomes find the connectivity between s and t in G', just like before. If the edge exists, we test all the edges in the other side which can form paths with this edge. The reason is stated as follow. If we intend to find out a cut to prove disconnectivity, the test of these edges are all necessary. If we intend to find a path to prove connectivity, no matter whether the first edge is a set edge or a node edge, the probability of the existence of a path is higher and the cost is lower when we continue to test all the edges in the other side. After testing every the possible path, we get a G' of the same structure as before. Then we perform similar trial in G'. We denote the process of G becoming G' a trial. So we wonder which edge to test first in every trial.

Firstly, we have to determine whether this edge is in set edges or node edges. There are three possibilities:

1) All the trials begin with set edge test. We denote the cost expectation C_1.
2) All the trials begin with node edge test. We denote the cost expectation C_2.
3) Some trials begin with set edge test and others begin with node edge test. We denote the cost expectation C_3.

We have upper bound of C_1

$$\begin{aligned} C_1 < & \left(1-P_s\right)^m\left(1-P_M\right)\left(C_M + mC_s\right) \\ & + \left(1-\left(1-P_s\right)^m\left(1-P_M\right)\right)\left(C_M + mC_s + nC_n\right) \end{aligned}$$

and lower bound of C_2

$$C_2 > \left(1-P_n\right)^n\left(nC_n\right)$$

Thus

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

After plug in P_s, C_s, P_n, C_n, we find

$$\left(1-P_s\right)^m\left(1-P_M\right) + \left(1-P_n\right)^n - 1 > 0$$

$$\begin{aligned} C_2 - C_1 > & C_n\left(\left(1-P_s\right)^m + \left(1-P_n\right)^n\left(1-P_M\right) - 1\right) \\ & - mC_s - C_M > 0 \end{aligned}$$

Obviously $C_1 < C_2$. We note that if P_s, C_s, P_n, C_n remain unchanged and we just reduce m or n or remove $1-P_M$ and C_M in $C_n\left((1-P_s)^m + (1-P_n)^n(1-P_M) - 1\right) - mC_s - C_M$, we still have $C_1 < C_2$. That is to say, for any subgraph of similar structure in G, we have $C_1 < C_2$. We find that every continuous trial beginning with node test in C_3 forms such a subgraph. These trials can be substituted with trials beginning with set test. Therefore the optimal solution in G is to do trials which begin with set edge tests until we find out the connectivity between s and t.

Secondly, we need to find out which edge in set edges will be tested in a trial. We denote total cost of set edges C_{set} and total cost of node edges C_{node}. There are also three possible strategies:

1) Always test the special set edge if it haven't been tested. That is, we first do the trail beginning with the special set edge and then do trials beginning with normal set edges. We denote C_{set} here as C_{set_1}.
2) Always test a normal set edge if some set edges still remain untested. That is, we first do the trail beginning with normal set edges and then do trials beginning with the special set edge. We denote C_{set} here as C_{set_2}.
3) Which edge to test first depends on the state of the graph. That is, we may first do a few trials beginning with normal set edges, then turn to do the trial beginning with the special set edge halfway. If the special set edge doesn't exist, continue to do a few trials beginning with normal set edges. We denote C_{set} here as C_{set_3}.

In every distribution of the existence of set edges, all these three strategies are just different in the sequence when we test node edges which exists in equal possibilities, so C_{node} are the same and we just have to compare C_{set}. We denote C_{normal} as total cost of normal set edges when the special set edge is untested or doesn't exist. P_{cover} is the probability we can end the test just through testing normal set edges. Thus We have

$$\begin{aligned} C_{set_1} &= C_{normal} + \left(1-P_{cover}\right)C_M + C_{node} \\ C_{set_2} &= C_M + \left(1-P_M\right)C_{normal} + C_{node} \end{aligned}$$

Now comparing C_{set_1} and C_{set_2} equals to comparing $\frac{C_M}{P_M}$ and $\frac{C_{normal}}{P_{cover}}$. We have lower bound and upper bound of C_{normal} and P_{cover}

$${P_s}^k k\left(1-P_n\right)^n < C_{normal} < \left(1-{P_s}^k\right)m + {P_s}^k k$$

$${P_s}^k < P_{cover} < 1 - \left(1-P_s\right)^m$$

We have

$$\frac{C_{normal}}{P_{cover}} < \frac{\left(1-{P_s}^k\right)m + {P_s}^k k}{{P_s}^k} < \frac{C_M}{P_M}$$

and

$$\frac{C_{normal}}{P_{cover}} > \frac{{P_s}^k k\left(1-P_n\right)^n}{1-\left(1-P_s\right)^m} > \frac{C_M}{P_M}$$

That means if the original problem has a set cover of k, the second strategy is better than the first one and if not, the first strategy is better than the second one.

As for the third strategy, if the set cover is of size k, after a few tests in G', the rest cost of trials beginning with normal set edges test is less than before and switching to test M will cost more. If not, testing the special set edge directly avoids unnecessary tests of normal set edges at first.

In conclusion, if a set cover of size k exists, the optimal action is to test a normal set edge first. If not, the optimal action is to test the special set edge first. Since the set cover problem is NP-complete problem, deciding the optimal action is NP-hard.

A further remark is that the complexity of this problem is twofold. The first one is that there is an exponential number of feasible solutions. The second is that even evaluating the value of a solution is hard. □

References

[1] Puterman, Martin L. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[2] Bellman, Richard E., and Stuart E. Dreyfus. Applied dynamic programming. Princeton university press, 2015.

[3] Papadimitriou, Christos H., and Mihalis Yannakakis. "Shortest paths without a map." Theoretical Computer Science 84.1 (1991): 127-150.

[4] Garey, Michael R., and David S. Johnson. "A Guide to the Theory of NP-Completeness." WH Freemann, New York (1979).

[5] Karp, Richard M. Reducibility among combinatorial problems. springer US, 1972.

[6] Cox Jr, Louis Anthony, Q. I. U. Yuping, and Warren Kuehner. "Heuristic least-cost computation of discrete classification functions with uncertain argument values." Annals of Operations Research 21.1 (1989): 1-29.

[7] Charikar, Moses, et al. "Query strategies for priced information." Proceedings of the thirty-second annual ACM symposium on Theory of computing. ACM, 2000.

[8] Tang, Jie, et al. "Transfer link prediction across heterogeneous social networks." ACM TOIS (2015).

[9] Buccafurri, Francesco, et al. "Discovering links among social networks." Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg, 2012. 467-482.

[10] Newman, Mark EJ. "The structure of scientific collaboration networks." Proceedings of the National Academy of Sciences 98.2 (2001): 404-409.

[11] Zhu, Ying, et al. "A survey of social-based routing in delay tolerant networks: positive and negative social effects." Communications Surveys & Tutorials, IEEE 15.1 (2013): 387-401.

[12] Golovin, Daniel, and Andreas Krause. "Adaptive submodularity: Theory and applications in active learning and stochastic optimization." Journal of Artificial Intelligence Research (2011): 427-486.

[13] Deshpande, Amol, Lisa Hellerstein, and Devorah Kletenik. "Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover." Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, 2014.

[14] Allen, Sarah R., et al. "Evaluation of DNF formulas." arXiv preprint arXiv:1310.3673 (2013).

[15] Kaplan, Haim, Eyal Kushilevitz, and Yishay Mansour. "Learning with attribute costs." Proceedings of the thirty-seventh annual ACM symposium on Theory of computing. ACM, 2005.

[16] Valiant, Leslie G. "The complexity of enumeration and reliability problems." SIAM Journal on Computing 8.3 (1979): 410-421.

[17] Karger, David R. "A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem." SIAM review 43.3 (2001): 499-522.

[18] Karp, Richard M., Michael Luby, and Neal Madras. "Monte-Carlo approximation algorithms for enumeration problems." Journal of algorithms 10.3 (1989): 429-448.

[19] Vazirani, Vijay, and Mihalis Yannakakis. "Suboptimal cuts: Their enumeration, weight and number." Automata, languages and programming (1992): 366-377

[20] Fu, Luoyi, Xinbing Wang, and P. R. Kumar. "Optimal determination of source-destination connectivity in random graphs." Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing. ACM, 2014.