Stochastic Graph Exploration with Limited Resources

Ilan Reuven Cohen(✉)

Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel
ilan-reuven.cohen@biu.ac.il

Abstract. In recent years, the explosion of research on large-scale networks has been fueled to a large extent by the increasing availability of large, detailed network data sets. Specifically, exploration of social networks constitutes a growing field of research, as they generate a huge amount of data on a daily basis and are the main tool for networking, communications, and content sharing. Exploring these networks is resource-consuming (time, money, energy, etc.). Moreover, uncertainty is a crucial aspect of graph exploration since links costs are unknown in advance, e.g., creating a positive influence between two people in social networks. One approach to model this problem is the stochastic graph exploration problem [4], where, given a graph and a source vertex, rewards on vertices, and distributions for the costs of the edges. The goal is to probe a subset of the edges, so the total cost of the edges is at most some prespecified budget, and the subgraph is connected, containing the source vertex, and maximizes the total reward of the spanned vertices. In this stochastic setting, an optimal probing strategy is likely to be adaptive, i.e., it may determine the next edge to probe based on the realized costs of the already probed edges. As computing such adaptive strategies is intractable [15], we focus on developing non-adaptive strategies, which fix a list of edges to probe in advance. A non-adaptive strategy would not be competitive versus the optimal adaptive one unless it uses a budget augmentation. The current results demand an augmentation factor, which depends logarithmically on the number of nodes. Such a factor is unrealistic in large-scale network scenarios. In this paper, we provide constant competitive non-adaptive strategies using only a constant budget augmentation for various scenarios.

Keywords: Stochastic optimization · Graph exploration · Non-adaptive strategies

1 Introduction

Network exploration is a fundamental paradigm for discovering information available at the nodes of a network. The rise of social networks increased the number of nodes and links dramatically, and exploring them demands many resources. A network exploration algorithm defines a probing strategy that decides at each stage of the process which edges to probe. For the exploration of social networks, the network structure is known in

This research was supported by the Israel Science Foundation (grant No. 1737/21).

advance, e.g., followers on Twitter, friends on Facebook, etc. However, the aggregation of information in this network is uncertain, e.g., whether tweets/posts of a user would be retweeted/shared by their followers. Most of the recent work [23,26,27] deals with real-world networks' exploration when a limited budget is available, but they do not provide a comprehensive theoretical study of these problems. In this work, we continue the theoretical study of exploring networks, initiated by [4].

The main difficulty in designing effective probing strategies, other than their enormous size, is that they must perform well in an uncertain environment, where the amount of resource (cost) associated with a specific link (edge) is unknown in advance. A common assumption, instead, is that the distributions of the edges' costs may be well-estimated by using various properties of the connecting nodes. Accordingly, a good probing strategy might have an adaptive nature; it may determine the future network portion to be explored according to the realized cost of the edges already explored. Unfortunately, finding such optimal adaptive strategies is often intractable [15]. Moreover, implementing an efficient adaptive strategy might be impossible. In various exploration process applications, many machines work in parallel, and the updated adaptive strategy must be communicated to the machines participating in the process. As a result, the communication cost required by an adaptive strategy may also be high. Therefore, we are interested in devising non-adaptive probing strategies that are simple and that define the sequence of probes in advance before the process is started. We continue the recent line of research [8,21,22] developing polynomial computable non-adaptive strategies that are competitive against the optimal adaptive strategy.

In this work, we extended the work in [4] where they considered exploring a network from a root node. The graph has deterministic rewards on nodes and costs on edges, where each edge cost is drawn independently from a known distribution. They mainly focused on designing non-adaptive strategies for exploring the graph, where they demonstrated that in order to achieve a reasonable approximation guaranteeing a budget augmentation is mandatory, so a crucial aspect is the augmentation factor used by the non-adaptive strategy. Unfortunately, they only provide algorithms that use budget augmentation, which depends on the logarithm of the number of nodes in the network (and the maximal revenue from a node). As the main motivation is exploring large-scale networks where the number of nodes is immense, those algorithms are, in fact, impractical. In this paper, we develop non-adaptive algorithms for strategies that use a considerably less amount of resources compared to the currently known algorithms.

Problem Definition. Given an instance $\mathcal{I} = (V, E, r, \pi, \mathcal{R}, B)$, the underlying graph is $G(V, E)$ and $r \in V$ is the source vertex, and $n = |V|$ is the number of vertices. The edge costs $C : E \to \mathbb{R}_{\geq 0}$ are drawn independently according to $\pi(e)$, for $e \in E$, and deterministic rewards of vertices $\mathcal{R} : V \to \mathbb{R}_{\geq 0}$. (The model can be easily extended to rewards distributed according to independent random variables.) A graph-exploration process constructs a set of edges $F \subseteq E$ that it probes. All vertices of the subgraph of G spanned by F must be connected to r via the edges of F. The process probes one by one the edges and adds them to F. The actual cost of an edge e, drawn independently from the distribution $\pi(e)$, is revealed only when the edge is probed. The objective is to maximize the expected total reward from the vertices spanned by the edge set F, while the total cost of the edges in F remains bounded by a prespecified budget B. As soon as the total cost of F exceeds B, the process terminates. Our goal is to design a polynomial computable non-adaptive strategy, where, given an instance $\mathcal{I}$, it computes an ordered list of the edges in advance, where every prefix of the list induces a subtree that contains r. The expected gain of a list strategy is the sum of the vertex's reward times the probability that the vertex is successfully added, i.e., that the total cost of the vertex's list prefix does not exceed the budget. We compare this gain to an *adaptive strategy* expected gain, which may decide on the next edge to be probed after the cost of all previously probed edges is revealed. Note that the adaptive algorithm does not know the realization of the edges' costs in advance, and once it probes an edge, it must be added to its set. As mentioned, there exist simple instances where the ratio between any non-adaptive expected reward to an adaptive reward is $\Omega(n)$. Therefore, we allow the non-adaptive one to use a limited amount of budget augmentation. Accordingly, we call an algorithm (α, β)-approximate if it computes a strategy which uses budget $\beta \cdot B$, and obtains an expected reward of at least $1/\alpha$ times the optimal reward (obtained by an adaptive algorithm). In this work, we will focus on algorithms that are $(O(1), O(1))$-approximate, i.e., algorithms for strategies that use a constant factor of budget augmentation and ensure a constant fraction of the optimal adaptive reward.

1.1 Summary of Results and Techniques

Our main contribution is developing non-adaptive competitive algorithms for the stochastic graph exploration problem, which uses only a constant amount of budget augmentations. We provide positive results for two important scenarios: spider graphs and bounded-weighted-depth trees.

Spider Graphs. We study a spider-tree graph where all the vertices except the root have an out-degree of at most 1. These graphs are natural extensions for job scheduling applications for the stochastic knapsack, introduced in [15]. In this application, the goal is to schedule a maximum-value subset of n jobs with uncertain duration within a fixed amount of time, and captures the reality of job scheduling, where we cannot go back in time, in case a scheduled job has taken too long to complete. The spider graph instances apply to scenarios where before processing a certain job, it must process some chain of jobs in advance, and the duration of each job in the chain is stochastic. Note that for scenarios where the reward is just when completing a complete job chain (i.e., the reward is zero on non-leafs vertices), the spider-tree formulation gives extra power to the adaptive strategies versus the stochastic knapsack formulation. This arises from the fact that in such instances, an adaptive strategy may abort a chain in the middle, while in the knapsack setting, the entire chain must be represented as a single edge (so if begun, it must be completed). On the other hand, in both formulations, the non-adaptive strategies must complete the entire chain of jobs once started.

Bounded Weighted Depth Trees. For a general tree structure, we achieved a constant approximation and augmentation for bounded weighted depth trees instances. In those instances, the expected cost of the path from the root to each node is bounded (with respect to the entire budget). The structure of the social network (as in *Six degrees of separation assumption* [30]) implies that this scenario is extremely practical.

Our Techniques. Given an instance $\mathcal{I}$, we bound the value of the optimal adaptive strategy using a linear program (LP) formulation $\Phi_{\mathcal{I}}$. This extends the LP formulation of [15] and captures the dependence between the probing probabilities of different edges in the graph. In addition, we define a relaxation of this LP, namely $\hat{\Phi}_{\mathcal{I}}$, and characterize the structure of an optimal solution for it. In order to solve *spider graphs* instances, we divide the set of nodes into two, the risky nodes, where the expected cost from the root to those nodes is more than half of the entire budget, and the rest of the nodes (which are a connected component that contains r). For the risky nodes, using the solution of the LP $\Phi_{\mathcal{I}}$, we prove that a non-adaptive strategy that probes a single leg and uses budget augmentation is a constant competitive. For the non-risky nodes, we use the solution's structure of $\hat{\Phi}_{\mathcal{I}}$ and present a competitive set strategy. Combining the two results implies a constant approximation algorithm that uses at most constant augmentation. Our algorithm for *bounded-weighted-depth trees* instances also uses the solution of $\hat{\Phi}_{\mathcal{I}}$. However, the resulting structure of the optimal solution is more complex, and the total expected cost of the edges in support of the solution might be much higher than the budget. To address this issue, we introduce the *Tree Decomposition* algorithm, which can decompose such a solution to a bounded number of subtrees, where each subtree contains r and has a bounded total expected cost. We prove that sampling one of those subtrees (entirely) is a competitive non-adaptive strategy. Finally, we prove the limitation of the proposed LP $\Phi_{\mathcal{I}}$, by presenting an instance where the gap between the LP and an adaptive solution (even with budget augmentation) is unbounded, which raises questions about future directions on how to bound a general instance of this problem.

1.2 Related Work

The first work on the adaptivity gap of stochastic problems has been studied for the knapsack problem [10,15]. Note that the stochastic knapsack problem is a special case of the problem we study, where the underlying graph is a star-graph. As mentioned, the model studied in this paper has been introduced in [4]. In [4], they show an $\Omega(n)$ adaptivity gap for this problem, and in order to circumvent the impossibility result, they allow a limited amount of resource augmentation. They presented polynomial time computable non-adaptive strategies in a graph of n vertices and where R is maximum reward of a vertex, which are $(O(1), O(\log nR))$-approximate for trees and $(O(\log(nR)), O(\log(nR)))$-approximate for general graphs.

Another set of problems related to the stochastic graph exploration problem are various *stochastic orienteering* problems [8,19,28]. In stochastic orienteering, the set of traversed edges must form a path in a metric graph with deterministic costs on the edges, while the time spent on a node is a random variable, which follows an a priori known distribution. In stochastic graph exploration, the random variables are the costs of the edges of the graph, but we cannot ensure that the costs on the edges form a metric since the random variables are independent.

The adaptivity gap has also been studied for budgeted multi-armed bandits [17,18,24] by resorting to suitable linear programming relaxation. Unlike previous work on budgeted multi-armed bandit problems, we consider the setting in which new arms appear after some arms are pulled. Stochastic probing problems have also been studied for matching [1,7,13], motivated by kidney exchange and for more general classes of matroid optimization problems [20,21]. Various other stochastic problems are recently studied, such as variations on the Pandora's box [5,9,11,12,25], stochastic k-TSP [22], scheduling [2,3,16], probing [14,21,29], stochastic vertex cover in (hyper-)graphs [6], etc.

2 Preliminaries

Notations. We assume without loss of generality that $B = 1$, and we focus of on tree instances. For instance, $\mathcal{I} = (V, E, r, \pi, \mathcal{R})$, where $r \in V$ is the root of the directed tree $G = (V, E)$, we assume that the edges are pointed away from the root. Accordingly, we denote $Parent(v)$ as the parent of a vertex v, the vertex connected to v on its path to the root, $\langle Parent(v), v\rangle \in E$, and let $e_v = \langle Parent(v), v\rangle \in E$. For ease of notation, we denote $C(v) = C(e_v)$ ($C(r) = 0$), and $\pi(v) = \pi(e_v)$. Let $Path(u, v)$ be the set of vertices on the path from u to v (including u, v), and let $D(v) = Path(r, v)$. For a subtree $F \subseteq V$, let $C(F)$ be the total realized cost of its edges, i.e., $C(F) = \sum_{e \in F} C(e)$ and let $\mathcal{R}(F)$ be the total reward of its vertices, i.e. $\mathcal{R}(F) = \sum_{v \in F} \mathcal{R}(v)$.

Types of Strategies. An *adaptive strategy* determines for any subtree $F \subseteq V$, $r \in F$, and remaining budget $1 - C(F)$ which adjacent edge to F to probe, (i.e., edge $\langle u, v\rangle$ such that $u \in F$ and $v \notin F$). Note that if $C(F) > 1$, the strategy must halt. A vertex v is *successfully added* if e_v is probed and $C(F) \leq 1$ immediately afterward ($\mathcal{R}(v)$ is added to the total gain of the strategy). For a fixed adaptive strategy, we denote x_v the probability that this strategy probes the edge e_v, and y_v the probability that this strategy successfully add the vertex v to its connected component. The expected gain of this strategy is:

$$\sum_{v \in V} y_v \cdot \mathcal{R}(v).$$

In Sect. 3, we bound the optimal solution value via a set of constraints on $\{x_v, y_v : v \in V\}$.

The two main non-adaptive strategies are: the *list strategy* and the *set strategy*. A *list strategy* specifies an ordered list of the vertices $L = (v_1 = r, \ldots, v_n)$, where for any k, $(v_1, \ldots, v_k)$ is a connected subtree. The expected gain of a list strategy L with a budget $\mathcal{B}$ is:

$$\sum_{k=1}^{n} \mathbf{Pr}\left[\sum_{i=1}^{k} C(v_i) \leq \mathcal{B}\right] \cdot \mathcal{R}(v_k).$$

A *set strategy* (all-or-nothing) specifies a subtree F ($r \in F$) in advance, and either gains the entire content of F, if the total cost of F did not overflow the budget, or gains nothing otherwise. The expected gain of a set strategy F with a budget $\mathcal{B}$ is:

$$\mathbf{Pr}[C(F) \leq \mathcal{B}] \cdot \mathcal{R}(F).$$

3 Bounding the Optimal Adaptive Policy

In this section, we bound the value of an adaptive strategy given an instance $\mathcal{I}$. Recall that for a fixed strategy, we denote x_v, the probability that the strategy probes the edge e_v, and y_v, the probability that the strategy successfully add vertex v to its connected component. We first note that we can bound the probability that a strategy probes an edge by the probability it probes its successor edge, i.e., for any $v \in V$ we have $x_v \leq x_{Parent(v)}$.

A simple Example 16, demonstrated that this condition is not sufficient. Instead, we bound y_v (the probability of successfully adding v) by the probability it probed one of its predecessors edge $e_u \in D(v)$, times the probability the cost of the path between u, v is less than the adaptive budget (1).

Lemma 1. *Given a tree-instance $\mathcal{I}$, and $v, u \in V$ s.t. $u \in D(v)$, then*

$$y_v \leq \mathbf{Pr}[C(Path(u,v)) \leq 1] \cdot x_u.$$

Proof. Given a pair of vertices v, u, where $u \in D(v)$, by the definition of a feasible adaptive strategy, it may probe the path $Path(u,v)$ only if it probes the edge e_u first, and the total cost of $Path(u,v)$ is independent of the probability of sampling e_u. Finally, in order that v would be successfully added to F, the total cost of this path must be at most 1, i.e., $C(Path(u,v)) \leq 1$.

Our second lemma bounds the set of all vertices that an adaptive policy tries to insert. We extend the lemma in [15] to deal with budget augmentation. The main idea is to exploit the property of irrevocable decisions, which forces the adaptive policy to keep a vertex even if its size turns out to be large. Let $\mu_{\mathcal{B}}(v) = \mathbb{E}[\min\{\mathcal{B}, C(v)\}]$ the truncated expected size with respect to the augmented budget $\mathcal{B}$, and $\mu_{\mathcal{B}}(F) = \sum_{v \in F} \mu_{\mathcal{B}}(v)$. For ease of notation, we denote $\mu(v) = \mu_1(v)$.

Lemma 2. *For $\mathcal{B} \geq 1$ we have $\sum_{v \in V} x_v \mu_{\mathcal{B}}(v) \leq 1 + \mathcal{B}$.*

Proof. Our proof extends lemma 2 in [15]. For an adaptive policy, let S_t denote the set of the first t vertices chosen by it. Once the size of S_t overflows, no further vertices are added to S_t, and S_t remains constant after this. Let $C_{\mathcal{B}}(v) = \min\{C(v), \mathcal{B}\}$, We define a series $X_t = \sum_{i \in S_t} C_{\mathcal{B}}(v_i) - \mu_{\mathcal{B}}(v_i)$. It is easy to verify X_t is a martingale, and since $X_0 = 0$, we have $\mathbb{E}[X_t] = 0$, which yields $\mathbb{E}\big[\sum_{i \in S_t} C_{\mathcal{B}}(v_i)\big] = \mathbb{E}\big[\sum_{i \in S_t} \mu_{\mathcal{B}}(v_i)\big]$. The process stops when $C(S_t) > 1$ or we have no more vertices left. Since $\sum_{i \in S_t} C_{\mathcal{B}}(v_i) \leq C(S_t)$ and each $C_{\mathcal{B}}(v_i) \leq \mathcal{B}$, we get $\sum_{i \in S_t} C_{\mathcal{B}}(v_i) \leq 1 + \mathcal{B}$ for any $t > 0$. The mean size of all the vertices inserted by the policy (including the first one which exceeds knapsack capacity) is $\mu(S) = \lim_{t \to \infty} \mu(S_t)$ and therefore $\mathbb{E}[\mu(S)] = \lim_{t \to \infty} \mathbb{E}[\mu(S_t)] \leq 1 + \mathcal{B}$.

Given an instance $\mathcal{I}$, we define $\Phi_{\mathcal{I},\mathcal{B}}(t)$ as the optimal solution value for the following linear program (Fig. 1):

Using Lemmas 1, 2, we have:

Theorem 3. *Given a tree-instance $\mathcal{I}$, the expected gain of any adaptive policy with budget $\mathcal{B}$ is at most $\Phi_{\mathcal{I},\mathcal{B}}(1 + \mathcal{B})$.*

$$\begin{aligned}
\max\ & \sum_v y_v \cdot \mathcal{R}(v) & & \\
s.t.:\ & \sum_v x_v \cdot \mu_{\mathcal{B}}(v) \le t & & \\
& y_v \le \mathbf{Pr}[C(Path(u,v)) \le 1] \cdot x_u & \text{for } u \in D(v) & \quad (1)\\
& 0 \le x_v, y_v \le 1 & \text{for } v \in V &
\end{aligned}$$

Fig. 1. $\Phi_{\mathcal{I},\mathcal{B}}(t)$ - the linear program for bounding the optimal adaptive policy.

Next, we provide several characterizations for the solution of $\Phi(t)$ (we omit the subscripts $\mathcal{I}, \mathcal{B}$ for a fixed set of parameters). First, we prove that $\Phi_{\mathcal{I},\mathcal{B}}(t)$ is a concave function.

Claim. For $\gamma \in [0,1]$ we have, $\Phi_{\mathcal{I},\mathcal{B}}(\gamma \cdot t) \ge \gamma \cdot \Phi_{\mathcal{I},\mathcal{B}}(t)$.

Proof. Given an optimal solution y_v, x_v for $\Phi(t)$, then $\gamma \cdot x_v, \gamma \cdot y_v$ is a feasible solution for $\Phi(\gamma \cdot t)$ and its values is $\gamma \cdot \Phi_{\mathcal{I},\mathcal{B}}(t)$.

The next claim states that the constraint $x_u \le x_{Parent(u)}$ for $u \in V \setminus \{r\}$ implied by the other constraints.

Claim. There exists an optimal solution of $\Phi_{\mathcal{I},\mathcal{B}}(t)$, such for $u \in D(v)$ that

$$x_v \le x_u \cdot \mathbf{Pr}[C(Path(u, Parent(v))) \le 1] .$$

Proof. Given a solution, and $v, u \in D(v)$ such that $x_v > x_u \cdot \mathbf{Pr}[C(Path(u, Parent(v))) \le 1]$, we show that x_v is not binding in any constraint and can be reduced. For a constraint v, w ($v \in D(w)$):

$$\begin{aligned}
x_v \cdot \mathbf{Pr}[C(Path(v,w)) \le 1] &> x_u \cdot \mathbf{Pr}[C(Path(u, Parent(v))) \le 1] \cdot \mathbf{Pr}[C(Path(v,w)) \le 1] \\
&\ge x_u \cdot \mathbf{Pr}[C(Path(u,w)) \le 1] \ge y_w,
\end{aligned}$$

where the first inequality is derived from our assumption, the second inequality arises from: $Path(u,w) = Path(u, Parent(v)) \cup Path(v,w)$, and the last inequality is a by the definition of $\Phi_{\mathcal{I}}$.

Corollary 4. *There exists an optimal solution of $\Phi_{\mathcal{I},\mathcal{B}}(t)$, such that, for $v \in V$ we have $x_v \le x_{Parent(v)}$.*

Next, we define a linear program $\hat{\Phi}_{\mathcal{I},\mathcal{B}}(t)$, a relaxation for the original linear program $\Phi_{\mathcal{I},\mathcal{B}}(t)$, and characterize an optimal solution for it.

Using Corollary 4, and the fact that in $\Phi_{\mathcal{I},\mathcal{B}}$, we have $y_v \le x_v$, we conclude:

Observation 5. *For any instance $\mathcal{I}$, and for any $t > 0$, we have $\hat{\Phi}_{\mathcal{I},\mathcal{B}}(t) \ge \Phi_{\mathcal{I},\mathcal{B}}(t)$.*

While $\hat{\Phi}_{\mathcal{I},\mathcal{B}}$ might not have a constant gap for general instances, see Example 16, we will use it for sub-instances where the probability of probing any vertex in this sub-instance is constant. As mentioned, the next lemma would characterize a possible optimal solution.

$$\max \sum_{v \in V} x_v \cdot \mathcal{R}(v)$$

$$s.t. : \sum_{v \in V} x_v \cdot \mu_{\mathcal{B}}(v) \leq t \qquad (2)$$

$$x_v \leq x_{Parent(v)} \qquad \text{for } v \in V \qquad (3)$$

$$0 \leq x_v \leq 1 \qquad \text{for } v \in V$$

Fig. 2. $\hat{\Phi}_{\mathcal{I},\mathcal{B}}$ - relaxation for the linear program for bounding the optimal adaptive policy.

Lemma 6. *There exists a solution for $\hat{\Phi}(t)$, where for each pair of vertices $v_1, v_2 \in V$, such that $0 < x_{v_1}, x_{v_2} < 1$, then for all vertices $w \in Path(v_1, v_2)$ we have $x_w = x_{v_2}$.*

Proof. We show a constructive proof that starts from an optimal solution x and computes a modified optimal solution x' until the condition holds.

We omit all vertices v and their corresponding edges such that $x_v = 0$. For $i \in \{1, 2\}$, let $v_i \in V$ such that $0 < x_{v_i} < 1$ and $T_i = \{w : x_u = x_{v_i} \text{ for all } u \in Path(w, v_i)\}$, such that $T_1 \neq T_2$ (if such a pair does not exist the condition on x holds). Note that T_i is a maximal connected subtree, which contains v_i and includes vertices u such that $x_{v_i} = x_u$. Assume w.l.o.g that v_1, v_2 are the roots of their corresponding subtrees (the closet vertex to r in their corresponding subtrees).

If $\mathcal{R}(T_1) = 0$ (or $\mathcal{R}(T_2) = 0$), we may set $x'_u = x_u - \epsilon$ for $u \in T_1$ and $x'_u = x_u$. Otherwise, without decreasing the value of the solution, for large enough ϵ, some vertices will join T_1 connected component, or the value will reach 0. Similarly, if $\mu(T_1) = 0$ (or $\mu(T_2) = 0$), we may set $x'_u = x_u + \epsilon$ for $u \in T_1$ and $x'_u = x_u$, otherwise x' is feasible without decreasing the value of the solution. For large enough ϵ, some vertices will join T_1 connected component, or this value will reach 1.

Let $g_1 = \frac{\mathcal{R}(T_1)}{\mu(T_1)}$ and $g_2 = \frac{\mathcal{R}(T_2)}{\mu(T_2)}$, we prove that in an optimal solution x, $g_1 = g_2$. We show that there exists $\epsilon' > 0$ such that for $\epsilon \in \{-\epsilon', \epsilon'\}$, the following modified solution x' is feasible.

$$x'_v = \begin{cases} x_v + \epsilon, & \text{for } v \in T_1 \\ x_v - \epsilon \cdot \frac{\mu(T_1)}{\mu(T_2)}, & \text{for } v \in T_2 \\ x_v, & \text{otherwise} \end{cases}$$

First, Constraint 2 holds since:

$$\sum_{v \in v} x'_v \mu(v) = \sum_{v \in V} x'_v \mu(v) + \epsilon \cdot \mu(T_1) - \epsilon \cdot \mu(T_2) \cdot \frac{\mu(T_1)}{\mu(T_2)} = \sum_{v \in V} x_v \mu(v).$$

Second, note that, by definition $x_{Parent}(v_i) > x_{v_i}$ and for all descendants of u of T_i, $x_u < x_{v_i}$, therefore there exists a small enough ϵ for which Constraint 3 will still hold. Finally, the objective function for x' equals to:

$$\sum_{\ell \in L} x'_{v_\ell} \cdot \mathcal{R}(v_\ell) - \sum_{\ell \in L} x_{v_\ell} \cdot \mathcal{R}(v_\ell) = \epsilon \cdot \mathcal{R}(T_1) - \epsilon \cdot \mathcal{R}(T_2) \cdot \frac{\mu(T_1)}{\mu(T_2)} = \epsilon \cdot \mu(T_1) \cdot (g_1 - g_2),$$

and if $g_1 \neq g_2$, the value of the objective for x' for $\epsilon = \epsilon'$ or $\epsilon = -\epsilon'$ is higher than the value of the objective for x, which contradicts its optimality. Therefore, by setting $\epsilon = \min\{x_{Parent(v_1)} - x_{v_1}, x_{v_2} \cdot \frac{\mu(T_2)}{\mu(T_1)}\}$, we have that x' is an optimal feasible solution and a progress has been made (either x_{v_1} is 1, x_{v_2} is 0, or another vertex added to T_1).

Using Lemma 6, we conclude:

Corollary 7. *There exists an optimal solution x for $\hat{\Phi}(t)$ where there exists sub-trees $T_1, T_2 \subseteq V$ and a value $\zeta \geq 0$ such that for $v \in T_1$, $x_v = 1$ and for $v \in T_2$, $x_v = \zeta$, and $x_v = 0$ otherwise. The value of this solution is $\mathcal{R}(T_1) + \zeta \cdot \mathcal{R}(T_2)$, and $\mu(T_1) + \zeta \cdot \mu(T_2) \leq t$.*

Finally, for a subtree F, we prove a lower bound on the probability of the realized cost of F to be less than (a fraction of) the budget as a function of the subtree truncated expected size.

Lemma 8. *For a subtree F, and $\gamma \in [0, 1]$, we have $\mathbf{Pr}[C(F) < \gamma \cdot \mathcal{B}] \geq 1 - \frac{\mu_{\mathcal{B}}(F)}{\gamma \cdot \mathcal{B}}$.*

Proof.

$$\mathbf{Pr}[C(F) \geq \gamma \cdot \mathcal{B}] = \mathbf{Pr}[\min\{C(F), \mathcal{B}\} \geq \gamma \cdot \mathcal{B}]$$
$$\leq \frac{\mathbb{E}[\min\{C(F), \mathcal{B}\}]}{\gamma \cdot \mathcal{B}} \leq \frac{\mathbb{E}\left[\sum_{v \in F} \min\{C(v), \mathcal{B}\}\right]}{\gamma \cdot \mathcal{B}} = \frac{\mu_{\mathcal{B}}(F)}{\gamma \cdot \mathcal{B}},$$

where the first equality is due to $\gamma \leq 1$, the first inequality is given by Markov's inequality, and the second equality arises from the definition of μ.

4 Spider Graphs

Our first objective is to to develop a constant approximation algorithm for *spider graphs* instances using a constant augmentation. In spider graphs, all the vertices except the root have an out-degree of at most 1. Let L be the number of legs in the graph, we denote in leg $i \in [L]$, $v_{i,j}$ as the level j vertex, i.e. a vertex in leg i where its distance from the root is j, note that this vertex is uniquely defined. Let $C_i(j, k) = C(Path(v_{i,j}, v_{i,k}))$. Example 14 (Lemma 1 in [4]) demonstrates that even for this simple graph structure a budget augmentation is necessary to achieve a constant competitiveness. In addition, Example 15 demonstrates that even with budget augmentation there might not be a competitive set-strategy. Nevertheless, we show that using a constant budget augmentation, the adaptivity gap is bounded, by proving there exists a suitable list strategy.

Algorithm SpiderNoAdaptive divides the vertices into two sets, one set contains the risky vertices, where the expected cost of the path to them is at least half the budget, and the other set contains the rest of the vertices. The algorithm computes a constant approximation non-adaptive strategy for each set, and the maximum of those two strategies yield a constant non-adaptive strategy for the entire instance. For the risky vertices, the algorithm outputs a single arm, we prove that a non-adaptive list strategy that probes this arm and uses a constant budget augmentation has a constant competitive ratio. For the non-risky vertices, the algorithm computes a solution to $\hat{\Phi}_{\mathcal{I},1}(0.5)$ according to the structure of Corollary 7. The algorithm uses this solution to compute a fixed set of vertices. We prove that a non-adaptive set strategy that probes this set has a constant competitive ratio.

4.1 Non-adaptive Algorithm

For a constant $0 < \epsilon < 1$, we show that using $(1 + \epsilon)$ budget augmentation, there exists non-adaptive strategy and its expected gain is $O(\epsilon)$ factor of the optimal adaptive gain. The algorithm is composed of two parts, the first part would address the "risky" vertices, vertices with expected cost of the path from the root to them is at least 0.5, and the second part will deal with the rest of the vertices. Formally, let $\mathit{Risky} = \{v_{i,j} : \mu(D(v_{i,j})) > 0.5\}$. Accordingly, to the spider graph's notation, let $x_{i,j}$ the probability that the optimal adaptive strategy probed the edge $(v_{i,j-1}, v_{i,j})$ and let $y_{i,j}$ the probability that $v_{i,j}$ is successfully added to the probed sub-tree. The LP bound for adaptive strategy $\Phi_{\mathcal{I},1}(t)$ reduced to:

$$\begin{aligned}
\max \ & \sum_{i,j} y_{i,j} \cdot \mathcal{R}(v_{i,j}) \\
s.t.: \ & \sum_{v} x_{i,j}\mu_1(v_{i,j}) \le t \\
& y_{i,j} \le \mathbf{Pr}[C_i(k,j) \le 1] \cdot x_{i,k} && \text{for } k \le j \\
& 0 \le x_{i,j}, y_{i,j} \le 1 && \text{for } v \in V
\end{aligned}$$

```
**Data:** Spider leg tree instance $\mathcal{I}(V, E, r, \pi, \mathcal{R})$
**Result:** Non-adaptive list strategy
**Procedure** Risky($\mathcal{I}$)
    $x, y \leftarrow$ Solve $\Phi_{\mathcal{I},1}(2)$
    $L_i \leftarrow \sum_j x_{i,j} \cdot \mu(v_{i,j})$, for all $i \in L$
    $L^* \leftarrow L_i$ with probability $L_i/2$
    **return** $L^*$
**Procedure** NonRisky($\mathcal{I}$)
    $x \leftarrow$ Solve $\hat{\Phi}_{\mathcal{I},1}(0.5)$ // A solution according to Corollary 7
    $T_1 \leftarrow \{(i,t) : x_{i,t} = 1\}, T_2 \leftarrow \{(i,t) : x_{i,t} = \zeta\}$
    **return** arg max$\{\mathcal{R}(T_1), \mathcal{R}(T_2)\}$
$\mathcal{I}_{\text{Risky}} \leftarrow \mathcal{I}, \mathcal{R}(v_{i,t}) = 0:$ for $v_{i,t} \notin \mathit{Risky}$
$\mathcal{I}_{\text{Non}} \leftarrow \mathcal{I}, \mathcal{R}(v_{i,t}) = 0:$ for $v_{i,t} \in \mathit{Risky}$
$T_r \leftarrow$ Risky($\mathcal{I}_{\mathit{Risky}}$)
$T_n \leftarrow$ NonRisky($\mathcal{I}_{\mathit{Non}}$)
**return** arg max$\{\mathit{NonAdpative}(T_r), \mathit{NonAdpative}(T_n)\}$
```

Algorithm SpiderNoAdaptive: Non-adaptive algorithm for spider trees

4.2 Risky Vertices

Lemma 9. *For instance $\mathcal{I}$ where $\mathcal{R}(v_{i,j}) = 0$, for $v_{i,j} \notin$ Risky, Procedure* `Risky(`$\mathcal{I}$`)` *outputs a list strategy which gains at least $\epsilon/8$ factor of the optimal adaptive policy using $(1+\epsilon)$ budget augmentation.*

Proof. First note that the procedure is well defined since $\sum_i L_i \leq 2$, by $\Phi(2)$'s definition. The probability of vertex $v_{i,t}$ to successfully being probed is $L_i/2 \cdot \mathbf{Pr}[C_i(1,t) \leq 1+\epsilon]$. For a leg i, let k the first index s.t. $\sum_{t=1}^{k} \mu(v_{i,t}) \geq \epsilon/2$.

Note that, $\mathcal{R}(v_{i,k'}) = 0$ for for $k' < k$. Since by the definition of k, $\sum_{t=1}^{k'} \mu(v_{i,t}) < \epsilon/2 \leq 0.5$ and therefore, $v_{i,k'} \notin$ *Risky*, and $\mathcal{R}(v_{i,k'}) = 0$ by our assumption. Therefore, it is sufficient to bound the probability of vertices being successfully probed just for $v_{i,h}$ where $h \geq k$ and compare it to $y_{i,h}$ the corresponding adaptive probability. We first show that for a leg i, $L_i \geq \epsilon/2 \cdot x_{i,k}$.

$$L_i = \sum_{t=1} x_{i,t} \cdot \mu(v_{i,t}) \geq \sum_{t=1}^{k} x_{i,t} \cdot \mu(v_{i,t}) \geq x_{i,k} \sum_{t=1}^{k} \mu(v_{i,t}) \geq \epsilon/2 \cdot x_{i,k}, \qquad (4)$$

where the first inequality is by removing positive terms, the second inequality is by Corollary 4, and the third inequality is by k's definition.

Finally, the probability that $v_{i,t}$ (for $t \geq k$) is successfully probed:

$$\begin{aligned} \frac{L_i}{2} \cdot \mathbf{Pr}[C_i(1,t) \leq 1+\epsilon] &\geq \frac{L_i}{2} \cdot \mathbf{Pr}[C_i(1,k-1) \leq \epsilon] \cdot \mathbf{Pr}[C_i(k,t) \leq 1] \\ &\geq \frac{L_i}{4} \cdot \mathbf{Pr}[C_i(k,t) \leq 1] \geq \frac{x_{i,k} \cdot \epsilon}{8} \cdot \mathbf{Pr}[C_i(k,t) \leq 1] \geq \frac{y_{i,k} \cdot \epsilon}{8}, \end{aligned}$$

where the first inequality is by the decomposition of the path, the second inequality is by Lemma 8 ($\gamma = \epsilon$) $\mathbf{Pr}[C_i(1,k-1) \leq \epsilon] \geq 1/2$ since $\sum_{j=1}^{k-1} \mu_{i,j} < \epsilon/2$ by k's definition, the third inequality is by 4 and the last inequality is by (3) in the definition of $\Phi_{\mathcal{I}}$.

4.3 Non-risky Vertices

Lemma 10. *For instance $\mathcal{I}$ where $\mathcal{R}(v_{i,j}) = 0$, for $v_{i,j} \in$ Risky, Procedure* `NonRisky(`$\mathcal{I}$`)` *outputs a set strategy which gains at least a $1/16$ fraction of the optimal strategy gain without budget augmentation.*

Proof. Let x be a solution which fulfill the conditions of Corollary 7, and T_1, T_2 according to the algorithm's definition. Note that, since $x_r = 1$, therefore T_2 is a single arm. By Observation 5 and by Theorem 3, the optimal gain is bounded by $\Phi_{\mathcal{I},1}(2) \leq \Phi_{\mathcal{I},1}(0.5) \cdot 4 \leq (\mathcal{R}(T_1) + \mathcal{R}(T_2)) \cdot 4 \leq \max\{\mathcal{R}(T_1), \mathcal{R}(T_2)\} \cdot 8$. Note that, we have $\mu(T_1) \leq 0.5$, since for $v_{i,t} \in T_1$, $x_{i,t} = 1$ and $\sum x_{i,t}\mu(v_{i,t}) \leq 0.5$ by $\hat{\Phi}_{\mathcal{I},1}(0.5)$ definition. Second, note that $\mu(T_2) \leq 0.5$, since it's a single arm, and we omitted the risky vertices. Therefore $\mathbf{Pr}[C(T_i) \leq 1] \geq 0.5$ and the gain of the strategy is at least $0.5 \cdot \max\{\mathcal{R}(T_1), \mathcal{R}(T_2)\}$.

4.4 Putting Things Together

Theorem 11. *For spider graphs instances, and a constant $\epsilon < 1$, there exist a $(24/\epsilon, 1+\epsilon)$-approximate non-adaptive strategy.*

Proof. Clearly, half of the gain is from the risky vertices or from the non-risky vertices. If most of the gain is from the risky vertices, by Lemma 9, there exists a $\epsilon/8$ competitive list strategy using $1+\epsilon$ augmentation. In most of the gain is from non-risky vertices, then by Lemma 10, there exists $1/16$ competitive set strategy without augmentation. Therefore, if $\gamma \in [0,1]$ fraction of the optimal gain is out of non-risky vertices, the maximum reward out of these two strategies, is at least $\max\{\gamma/16, (1-\gamma)\cdot \epsilon/8)\}$ fraction, which is at least $24/\epsilon$ fraction for any γ.

5 Bounded Weighted Depth Trees Instances

We now focus on the practical scenario where the depth of the tree's instance is bounded. We define the weighted depth of a graph as the maximum over all vertices of the total expected cost of the path from the root to them. Formally, an instance is $(\beta, \mathcal{B})$ weighted depth bounded, if for all $v \in V$, $\mu_{\mathcal{B}}(D(v)) \le \beta$. We prove that for $(\mathcal{B}\cdot(1-\epsilon), \mathcal{B})$ bounded weighted depth instances, for some constants $\epsilon > 0$,$\mathcal{B} \ge 1$, there exists a constant competitive set strategy with a budget $\mathcal{B}$. Note that for $(1-\epsilon, 1)$ bounded weighted-depth instances, our result implies there exists a constant competitive set strategy without budget augmentation, i.e., $(O(1), 1)$-approximate strategy. We observe that, given a solution x for $\hat{\Phi}_{\mathcal{I},\mathcal{B}}(t)$, and let T be the support tree of x, i.e. $v \in T$ if $x_v > 0$, then unlike for spider graphs, $\mu(T)$ can be much higher than t); see Example 18. Nevertheless, we show that, given a tree T ($r \in T$), such that $\mu(D(v)) \le \beta$ (we omit the subscript $\mathcal{B}$ it is clear from the context) for $v \in T$. It is possible to decompose T to $\mathcal{S} = \{S_1, \ldots, S_k\}$, where for all $v \in T$ there exists $i \in [k]$ such that $v \in S_i$. For all $i \in [k]$, S_i is a subtree that contains r, and $\mu(S_i) \le \alpha$, and the number of subtrees is bounded by $k \le \lceil \frac{2\mu(T)}{\alpha-\beta} \rceil$. Given a subtree T', denote $S_{T'}(v) = \{u : v \in D(u)\} \cap T'$, the subtree of v with respect to T'.

Algorithm TreeDecompose works in iterations. At each iteration, it locates a proper set of subtrees, where the parent of the root of each of those subtrees is the same. It adds the subtree containing this set and its entire path to the root to the set of sub-trees, and removes this subset from the current tree. Specifically, it denotes T' as the current subtree. If $\mu(T') \le \alpha$, then T' is a proper sub-tree, and the algorithm adds it to the set of sub-trees and terminates. Otherwise, it locates a subset of vertices S', where S' contains several sub-trees with the same parent w, and $\mu(S') \ge \frac{\alpha-\beta}{2}$, $\mu(D(w) \cup S') \le \alpha$ holds. The algorithm adds $D(w) \cup S'$ to the set of sub-trees and omits S' from the graph. To locate such S', in each iteration, it starts from the root and iteratively proceeds to one of his children u with the maximum value of $\mu(S_{T'}(u))$, until it reaches a vertex v such that $\mu(D(v) \cup S_{T'}(v)) > \alpha$, the algorithm locates S' for the vertex w, the parent of the vertex v.

Lemma 12. *Algorithm TreeDecompose outputs $\mathcal{S} = \{S_1, \ldots, S_k\}$ a feasible tree decomposition and $k \le \lceil \frac{2\mu(T)}{\alpha-\beta} \rceil$.*

```
Input  : A $(\beta, 1)$ weighted depth bounded instance, edge weights $\mu$, and $\alpha > \beta$
Output: A tree decomposition $\mathcal{S} = \{S_1, \ldots, S_k\}$, where $\mu(S_i) \leq \alpha$ for $i \in [k]$
$\mathcal{S} \leftarrow \emptyset, T' \leftarrow T$
while $\mu(T') > \alpha$ do
    $v \leftarrow r$
    while $\mu(D(v) \cup S_{T'}(v)) > \alpha$ do
        $v \leftarrow \arg\max_{(v,u)\in E} \mu(S_{T'}(u))$
    end
    $w \leftarrow Parent(v)$
    Let $(u_1, \ldots, u_h)$ such that $(w, u_i) \in T'$ and $\mu(S_{T'}(u_i)) \leq \mu(S_{T'}(u_{i+1}))$ for
      $i \in [h-1]$
    $h^* \leftarrow \arg\min\{j \in [h] : \mu(D(w)) + \sum_{i=j}^{h} \mu(S_{T'}(u_i)) \leq \alpha\}$
    $S' \leftarrow \bigcup_{j=h^*}^{h} S_{T'}(u_j)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(D(w) \cup S')\}, T' \leftarrow T' \setminus S'$
end
return $\mathcal{S} \cup \{T'\}$
```

Algorithm TreeDecompose: Decomposition of a tree to bounded weight rooted subtrees.

Proof. First, we show that the algorithm is well-defined, and T' is a connected subtree of T and contains the root at any step, which follows from the fact that the algorithm only omits a vertex with its entire subtree from T'. Next, we observe that the inner loop is well-defined, and v would not be a leaf since for any leaf ℓ, we have $\mu(D(\ell)) + \mu(S_{T'}(\ell)) = \mu(D(\ell)) \leq \beta \leq \alpha$. Therefore, the inner loop always halts at vertex $v \neq r$ such that, for $w = Parent(v)$ then $v = u_h$ since $v = \arg\max_{(w,u)\in E} \mu(S_{T'}(u))$. Therefore, $\mu(D(w) \cup S_{T'}(w)) > \alpha$ and $\mu(D(u_h) \cup S_{T'}(u_h)) \leq \alpha$. Note that, h^* is well-defined since for $j = h$, we have: $\mu(D(w)) + \mu(S_{T'}(u_h)) = \mu(D(u_h) \cup S_{T'}(u_h)) \leq \alpha$. Next, we have that $h^* > 1$ since for $j = 1$ we have $\mu(D(w)) + \sum_{i=1}^{h} \mu(S_{T'}(e_i)) = \mu(D(w) \cup S_{T'}(w)) > \alpha$.

We observe that $\mathcal{S} = \{S_1, \ldots, S_k\}$ is the tree decomposition of T, since the algorithm terminates only after covering all the vertices. Additionally, each subtree added to $\mathcal{S}$ is $(D(w) \cup S')$, where $S' = \bigcup_{j=h^*}^{h} S_{T'}(u_j)$ is a collection of subtrees and their path to the root, and by the condition on h^*, we have $\mu(D(w) \cup S') \leq \alpha$.

In order to complete the proof, we need to show that $k \leq \frac{2\mu(T)}{\alpha-\beta}$. For any iteration where $\mu(T') > \alpha$, let w, v, h^* be the corresponding values of a main loop iteration, we have:

$$\alpha < \mu(D(w)) + \sum_{i=h^*-1}^{h} \mu(S_{T'}(u_i)) \leq \beta + \sum_{i=h^*-1}^{h} \mu(S_{T'}(u_i)) \leq \beta + 2\sum_{i=h^*}^{h} \mu(S_{T'}(u_i)),$$

where the first inequality is since $h^* > 1$, the second inequality is due to $\beta \geq D(w)$ for all $w \in T$, and the last inequality is a result of $\mu(S_{T'}(u_{h^*-1})) \leq \mu(S_{T'}(u_{h^*})) \leq \sum_{i=h^*}^{h} \mu(S_{T'}(u_i))$ since the vertices u_i are sorted accordingly. Therefore, the decrease in $\mu(T')$ in each such iteration is at least $\mu(S') \geq \frac{\alpha-\beta}{2}$, the number of iterations (until

$\mu(T') \leq \alpha)$ is at most $\lceil \frac{2(\mu(T)-\alpha)}{\alpha-\beta} \rceil$, and the number of subtrees is at most $\lceil \frac{2(\mu(T)-\alpha)}{\alpha-\beta} + 1 \rceil \leq \lceil \frac{2\mu(T)}{\alpha-\beta} \rceil$ as required.

Theorem 13. *For $(\mathcal{B} \cdot (1-\epsilon), \mathcal{B})$ bounded weighted depth instances, where $\epsilon > 0, \mathcal{B} \geq 1$, there exists $(\frac{\epsilon^2}{16 \cdot (\mathcal{B}+1)}, \mathcal{B})$-approximate non-adaptive strategy.*

Proof. Let x be a solution to $\Phi_{\mathcal{I},\mathcal{B}}(\mathcal{B}+1)$, which fulfills the conditions of Corollary 7, and let T_1, T_2, ζ be the corresponding integral and fractional trees and the fractional tree assignment value, respectively. Note that, since $x_r = 1$, we have $r \in T_1$. Let r_2 be the root of T_2, and let $T = T_2 \cup D(r_2)$. Assume w.l.o.g. that $\zeta \cdot \mathcal{R}(T) \geq \mathcal{R}(T_1)$. Note that by $\hat{\Phi}_{\mathcal{I}}$ definition, we have $\mu(T) \leq (\mathcal{B}+1)/\zeta$. By Lemma 12, and $\alpha = \mathcal{B} \cdot (1-\epsilon/2)$ we have a $\mathcal{S} = \{S_1, \ldots, S_k\}$, a feasible tree decomposition and $k \leq \lceil \frac{2\mu(T)}{\alpha-\beta} \rceil \leq \frac{4 \cdot (\mathcal{B}+1)}{\epsilon \cdot \zeta}$. Therefore, by choosing a uniform subtree S^* out of the k subtrees, the probability that a vertex $v \in T$ would be successfully added to the strategy is:

$$\mathbf{Pr}[v \in S^*] \cdot \mathbf{Pr}[C(S^*) \leq \mathcal{B}] \geq \frac{1}{k} \cdot (1 - \frac{\alpha}{\mathcal{B}}) \geq \frac{\epsilon \cdot \zeta}{4 \cdot (\mathcal{B}+1)} \cdot \frac{\epsilon}{2} = \zeta \cdot \frac{\epsilon^2}{8 \cdot (\mathcal{B}+1)}.$$

By summing over all the vertices, we find that the non-adaptive gain is at least $\zeta \cdot \mathcal{R}(T) \cdot \epsilon^2/(8 \cdot (\mathcal{B}+1))$, while the gain of the optimal adaptive policy is at most $2 \cdot \zeta \cdot \mathcal{R}(T)$.

6 Examples

In this section, we provide several of examples that demonstrate various tree graph instances' properties. Let $Be(s,p)$ denote a Bernoulli distribution, i.e., for a randomized variable $x \sim Be(s,p)$, then $x = s$ has a probability p, and $x = 0$ otherwise. Let 1^+ denote a large constant, i.e., $1^+ \gg \mathcal{B}$. Note that, for $e \sim Be(1^+, p)$, $\mu_{\mathcal{B}}(e) = p \cdot \mathcal{B}$.

First, for completeness, we state again the example in [4], which demonstrates that any competitive non-adaptive strategy must use budget augmentation even in spider graphs (Fig. 3).

Example 14. A spider graph with L legs, each leg i contains two vertices, $v_{i,1}$, and $v_{i,2}$, $\mathcal{R}(v_{i,1}) = 0$ and $\mathcal{R}(v_{i,2}) = 1$ for all $i \in [L]$, $\pi(v_{i,1}) \sim Be(2^{-i}, 1 - \frac{1}{L})$, and $\pi(v_{i,2}) = (1 - 2^{-i} + 2^{-L}, 1)$.

Claim (Lemma 1 in [4]**).** The adaptivity gap of stochastic graph exploration is $\Omega(n)$ (without budget augmentation) even in spider graphs.

Proof. Consider the spider graph in Example 14. Note that, for any $i \neq j \in [L]$, we have, $C(\{v_{i,2}, v_{j,2}\}) > 1$ with probability 1; therefore, any strategy would gain at most 1. Given a list strategy, let $v_{i,2}$ be the first second-level vertex in the list, the probability that this list strategy will gain is at most $\mathbf{Pr}[C(\{v_{i,1}, v_{i,2}\} \leq 1] = \frac{1}{L}$; therefore, the expected gain of any non-adaptive strategy is at most $1/L$.

On the other hand, an adaptive strategy probes $v_{i,1}$ sequentially from L to 1 until $C(v_{i,1}) = 0$ for some $i \in [L]$, and if such i exists, it probes $v_{i,2}$ and halts. Note that, in the case that such i exists, this strategy successfully probes $v_{i,2}$ (with probability 1). Therefore, the expected gain of this strategy is $(1 - 1/L)^L \approx 0.36$ and the gap is unbounded.

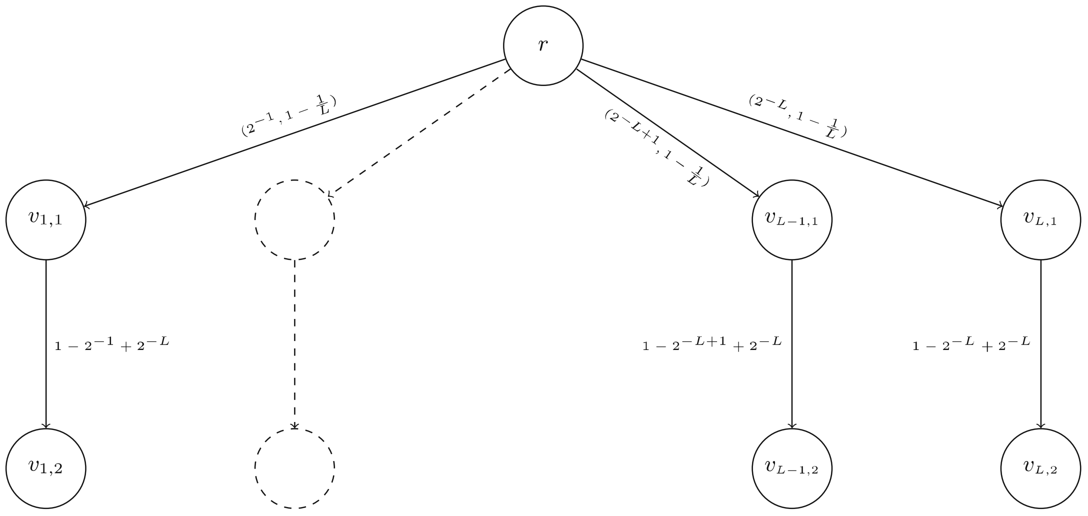

Fig. 3. Instance of spider graph demonstrating that the budget augmentation is necessary, see Example 14.

The next example demonstrates that even with budget augmentation, a *set* strategy is not competitive versus an *adaptive* strategy.

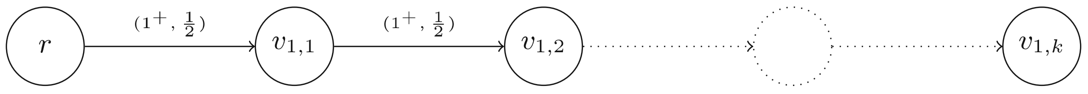

Fig. 4. An example demonstrating that a set strategy cannot approximate an adaptive strategy even with budget augmentation.

Example 15. A spider graph with 1 legs, the leg contains k vertices, $v_{1,j}$ for $j \in [k]$. $\mathcal{R}(v_{1,k}) = 2^k$, and $\pi(v_{i,1}) \sim Be(1^+, 1/2)$. See also Fig. 4.

Claim. The gap between a *set* strategy and an *adaptive* strategy is unbounded, even when the set strategy uses a constant budget augmentation.

Proof. Consider the graph in Example 15. The probability that a list strategy which probes the single leg will successfully probe vertex $v_{i,h}$ is 2^{-h}; therefore its expected gain is $\sum_{h=1}^{k} 2^{-h} \cdot 2^h = k$. While a set strategy which contains vertices with a prefix of $v_{1,j}$ for $j \in [k]$ would gain $\sum_{h=1}^{j} 2^h \leq 2^{j+1}$ and the probability it gains is 2^{-j}, there it's expected gain is at most $2^{j+1} \cdot 2^{-j} = 2$ and the gap is unbounded.

Example 16. A spider graph with 1 legs, the leg contains 2 vertices, $v_{1,1}$ for and $v_{1,2}$. $\mathcal{R}(v_{1,1}) = 0, \mathcal{R}(v_{1,2}) = 1, \pi(v_{1,1}) \sim Be(1^+, 1-1/k), \pi(v_{1,2}) \sim Be(0,1)$. The value of $\hat{\Phi}_{\mathcal{I},1}(1) = 1$, while the value of any adaptive policy (even with budget augmentation) is at most $1/k$.

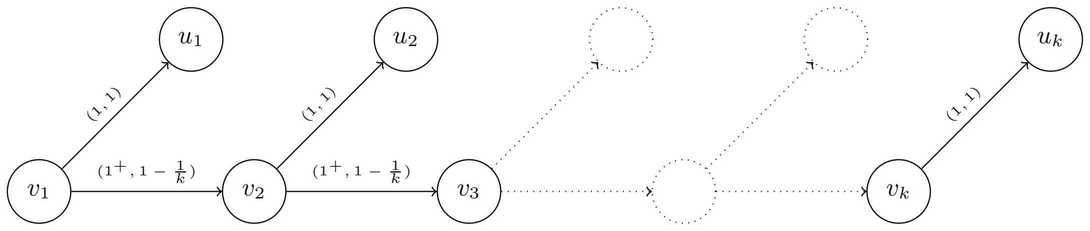

Fig. 5. An example demonstrates that for general instances, there is a non-constant gap between the value of the solution of Φ_I and the gain of an adaptive strategy, even with a constant budget augmentation.

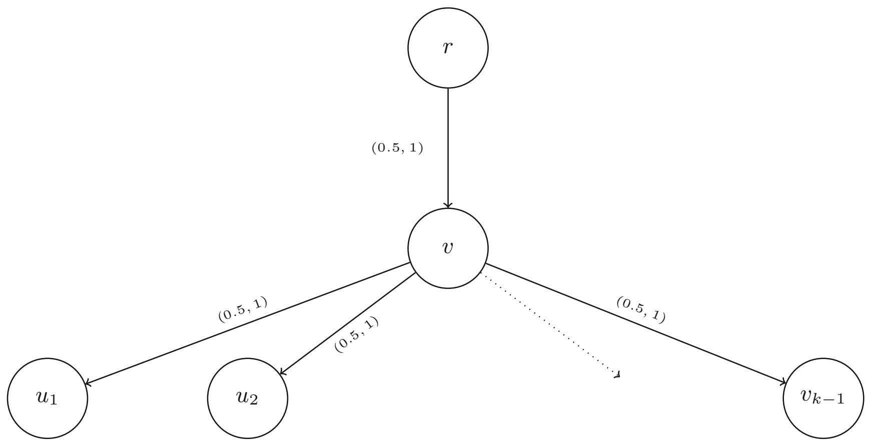

Fig. 6. An instance where $\hat{\Phi}_{\mathcal{I}}$ has an unbounded weight support tree.

Next, we demonstrate that for general instances, we cannot use the value of the solution of $\Phi_{\mathcal{I}}$ for the lower bound of an adaptive strategy, even with budget augmentation.

Example 17. Consider a graph with $2 \cdot k$ vertices, $v_1 = r, v_i, u_i$ for $i \in [k]$, $Parent(v_i) = v_{i-1})$, $Parent(u_i) = v_i$, and $\mathcal{R}(u_i) = k^{i-1}, \mathcal{R}(v_i) = 0$, $\pi(v_i) \sim Be(1^+, 1 - 1/k), \pi(u_i) \sim Be(1,1)$. See Fig. 5

Claim. There exists an instance $\mathcal{I}$, such that the gap between $\Phi_{\mathcal{I},\mathcal{B}}(2)$ and the value of any adaptive policy on $\mathcal{I}$ is unbounded, even with budget augmentation.

Proof. Consider the instance of Example 17; first note that for $i \in [k]$, $x_{v_i} = k^{-i+1}/2$ and $y_{u_i} = x_{u_i} = k^{-i}/2$ is a feasible solution and its value is $\sum_{i=1}^{k} y_{u_i} \cdot k^{i-1} = \sum_{i=1}^{k} k^{-i}/2 \cdot k^{i-1} = k/2$. Considering an adaptive strategy with budget $\mathcal{B}$, we can assume w.l.o.g that it is deterministic, since the only random variables are $C(v_i)$, and if $C(v_i) \neq 0$, the algorithm must halt. Let $h_1, \ldots, h_{\mathcal{B}}$, where the algorithm decides to probe u_{h_i} for $i \in \mathcal{B}$ (clearly there must be at most $\mathcal{B}$ since the cost of any of them is deterministically 1). The probability it gains, u_{h_j}, is at most k^j, and therefore it's expected gain is at most $\sum_{j=1}^{\mathcal{B}} k^{-j} \cdot k^j = \mathcal{B}$, while as we have shown $\Phi_{\mathcal{I},\mathcal{B}} \geq \Phi_{\mathcal{I},1} = k/2$.

Example 18. Consider a graph with $k+1$ vertices, r, v, u_i for $i \in [k-1]$, $Parent(v) = r$, $Parent(u_i) = v$, and $\mathcal{R}(u_i) = 1, \mathcal{R}(v) = 0$, $\pi(v_i) \sim Be(0.5, 1), \pi(u_i) \sim Be(1, 1)$. See Fig. 6. The optimal solution for $\hat{\Phi}_{\mathcal{I}}(2)$ will set $x_v = x_{u_i} = 4/k$, and therefore $\mu(T) = (k-1)/2$.

References

1. Adamczyk, M.: Improved analysis of the greedy algorithm for stochastic matching. Inf. Process. Lett. **111**(15), 731–737 (2011)
2. Albers, S., Eckl, A.: Explorable uncertainty in scheduling with non-uniform testing times. In: Kaklamanis, C., Levin, A. (eds.) WAOA 2020. LNCS, vol. 12806, pp. 127–142. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-80879-2_9
3. Albers, S., Eckl, A.: Scheduling with testing on multiple identical parallel machines. In: Lubiw, A., Salavatipour, M. (eds.) WADS 2021. LNCS, vol. 12808, pp. 29–42. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83508-8_3
4. Anagnostopoulos, A., Cohen, I.R., Leonardi, S., Lacki, J.: Stochastic graph exploration. In: 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2019)
5. Aouad, A., Ji, J., Shaposhnik, Y.: Pandora's box problem with sequential inspections. SSRN 3726167 (2020)
6. Bampis, E., Dürr, C., Erlebach, T., Santos de Lima, M., Megow, N., Schlöter, J.: Orienting (hyper)graphs under explorable stochastic uncertainty. In: Mutzel, P., Pagh, R., Herman, G. (eds.) 29th Annual European Symposium on Algorithms, ESA 2021, Lisbon, Portugal, 6–8 September 2021 (Virtual Conference). LIPIcs, vol. 204 pp. 10:1–10:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
7. Bansal, N., Gupta, A., Li, J., Mestre, J., Nagarajan, V., Rudra, A.: When LP is the cure for your matching woes: improved bounds for stochastic matchings. Algorithmica **63**(4), 733–762 (2012)
8. Bansal, N., Nagarajan, V.: On the adaptivity gap of stochastic orienteering. In: Lee, J., Vygen, J. (eds.) IPCO 2014. LNCS, vol. 8494, pp. 114–125. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07557-0_10
9. Beyhaghi, H., Kleinberg, R.: Pandora's problem with nonobligatory inspection. In: Proceedings of the 2019 ACM Conference on Economics and Computation, pp. 131–132 (2019)
10. Bhalgat, A., Goel, A., Khanna, S.: Improved approximation results for stochastic knapsack problems. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1647–1665. SIAM (2011)
11. Boodaghians, S., Fusco, F., Lazos, P., Leonardi, S.: Pandora's box problem with order constraints. In: Proceedings of the 21st ACM Conference on Economics and Computation, pp. 439–458 (2020)
12. Chawla, S., Gergatsouli, E., Teng, Y., Tzamos, C., Zhang, R.: Pandora's box with correlations: learning and approximation. In: 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pp. 1214–1225. IEEE (2020)
13. Chen, N., Immorlica, N., Karlin, A.R., Mahdian, M., Rudra, A.: Approximating matches made in heaven. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 266–278. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02927-1_23
14. Chugg, B., Maehara, T.: Submodular stochastic probing with prices. In: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 60–66. IEEE (2019)

15. Dean, B.C., Goemans, M.X., Vondrák, J.: Approximating the stochastic knapsack problem: the benefit of adaptivity. Math. Oper. Res. **33**(4), 945–964 (2008)
16. Dürr, C., Erlebach, T., Megow, N., Meißner, J.: An adversarial model for scheduling with testing. Algorithmica **82**(12), 3630–3675 (2020)
17. Guha, S., Munagala, K.: Approximation algorithms for budgeted learning problems. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, STOC 2007, San Diego, California, USA, pp. 104–113. ACM, New York (2007)
18. Gupta, A., Krishnaswamy, R., Molinaro, M., Ravi, R.: Approximation algorithms for correlated knapsacks and non-martingale bandits. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 827–836 (2011)
19. Gupta, A., Krishnaswamy, R., Nagarajan, V., Ravi, R.: Running errands in time: approximation algorithms for stochastic orienteering. Math. Oper. Res. **40**(1), 56–79 (2015)
20. Gupta, A., Nagarajan, V.: A stochastic probing problem with applications. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 205–216. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36694-9_18
21. Gupta, A., Nagarajan, V., Singla, S.: Algorithms and adaptivity gaps for stochastic probing. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, Virginia, pp. 1731–1747. Society for Industrial and Applied Mathematics, Philadelphia (2016)
22. Jiang, H., Li, J., Liu, D., Singla, S.: Algorithms and adaptivity gaps for stochastic k-TSP. arXiv preprint arXiv:1911.02506 (2019)
23. Laishram, R., Areekijseree, K., Soundarajan, S.: Predicted max degree sampling: sampling in directed networks to maximize node coverage through crawling. In: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 940–945 (2017)
24. Ma, W.: Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1154–1163. SIAM (2014)
25. Singla, S.: The price of information in combinatorial optimization. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2523–2532. SIAM (2018)
26. Soundarajan, S., Eliassi-Rad, T., Gallagher, B., Pinar, A.: MaxReach: reducing network incompleteness through node probes. In: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 152–157 (2016)
27. Soundarajan, S., Eliassi-Rad, T., Gallagher, B., Pinar, A.: ϵWGXX: adaptive edge probing for enhancing incomplete networks. In: Proceedings of the 2017 ACM on Web Science Conference, WebSci 2017, New York, NY, USA, pp. 161–170. ACM, New York (2017)
28. Thayer, T.C., Carpin, S.: An adaptive method for the stochastic orienteering problem. IEEE Robot. Autom. Lett. **6**(2), 4185–4192 (2021)
29. Wang, W., Gupta, A., Williams, J.: Probing to minimize. arXiv preprint arXiv:2111.01955 (2021)
30. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. Nature **393**(6684), 440–442 (1998)